An Evaluation of CASE Tools as Pedagogical Aids in Software Development Courses

Francis Kofi Andoh-Baidoo
Department of Computer Information Systems and Quantitative Methods
University of Texas – Pan American
1201 W University Drive, Edinburg, TX 78539-2999
Tel. (956) 381-3397. Fax (956) 381-3367
e-mail: andohbaidoof@utpa.edu

K. Niki Kunene
Department of Computer Information Systems
College of Business, University of Louisville, KY 40292
Tel. (502) 852-5612. Fax (502) 852-4799
e-mail: niki.kunene@louisville.edu

Ross E. Walker
Xerox Corporation
1350 Jefferson Road, Mailstop 0801-31A
Rochester, NY 14623-3106
Tel. (585) 427-3408
e-mail: Ross.Walker@xerox.com

ABSTRACT

Computer-Aided Software Engineering (CASE) tools are important for systems/software development and implementation. CASE tools can be used to reduce the cost and time of system/software development while improving the quality of the product developed. In addition, CASE tools enable the user to present information and knowledge compactly and in a uniform manner for ease of communication. CASE tools may provide support for structured/traditional methods, object-oriented (OO) methodologies or both. Current research on CASE tool evaluation has increasingly focused on OO. Even though the industry is being led towards object-oriented approaches, the ability of a CASE tool to support both object-oriented and structured design methodologies can be especially beneficial for teaching systems development. Information systems (IS) university programs teach structured and/or OO methodologies. In this research, we evaluate the two CASE tools, Visible Analyst® and Rational® Rose which are two of the most widely used tools in IS programs. This paper explicates the features of Visible Analyst® and Rational® Rose which can be used by educators in the CASE tool selection process.

Keywords: *CASE tool, systems/software development, object-oriented and structured methodologies, Visible Analyst®, IBM Rational® Rose.*

INTRODUCTION

In recent years, computer-aided software engineering tools have emerged as one of the most important innovations in software development to manage the complexity of software development projects (1998). The benefits of using CASE tools include: increasing the speed with which software is developed and improving the quality of the developed system (Limayem, Khalifa, & Chin, 2004); reducing the cost of software/system development; and providing a uniform platform for software/system developers to present information and knowledge compactly for ease of communication (Banker & Kauffman, 1991; Church & Matthews, 1995; Orlikowski, 1989).

In addition to usage in industry by practitioners, CASE tools are employed by educators to teach students software/systems development skills. Evaluating and selecting a CASE tool for a specific systems development course may be difficult. CASE tools are complex because they offer a large array of options and support for software/systems development activities (Post et al., 1998). Current research on CASE tool evaluation has increasingly focused on OO approaches (Church & Matthews, 1995; Massacci, Mylopoulos, & Zannone, 2007; Post & Kagan, 2000, 2001; Post et al., 1998). Even though the industry is being led towards object-oriented approaches, the ability of a CASE tool to support both object-oriented and structured design methodologies can be especially beneficial for teaching systems development.

In this study, we evaluated two CASE tools: Visible Analyst® and Rational® Rose which are used widely for teaching systems development in IS programs. In selecting the criteria we reviewed the extant literature to identify the features that are necessary to facilitate the teaching of both structured/traditional and OO concepts.

The rest of the paper is organized as follows: section two presents the evaluation criteria; section three contrasts the two CASE tools' performance with respect to the criteria; section four is the discussion, and section five is the conclusion and future research considerations.

CRITERIA

A CASE tool should serve an important role in systems/software development as a methodology companion by providing methodology support. A tool that provides methodology support should generate an error or warning when the rules of the methodology are violated (Hatley, 1988) in other words provide methodology consistency checks. Vessey, Jarvenpaa and Tractinsky (1992) proposed the first approach to classifying CASE tools with respect to methodology support. They identified three categories: restrictive, guided, and flexible tools.

Vessey et. al (1992) described a restrictive CASE tool as one designed to direct the analyst to use it in a normative manner. A restrictive CASE tool's methodological consistency checks occur automatically when an item is being created, and so the analyst is required to remedy arising violations before proceeding. Whereas a guided CASE tool encourages the analyst to use it in a normative way, it does not however enforce said usage. Thus when a violation occurs during the creation of an item the analyst can proceed without attending to the violation. A flexible CASE tool according to Vessey et al. (1992) gives the analyst complete freedom; methodological consistency checks occur (after work on the item has been completed) when the analyst requests them. Jankowski (1995) argued that the differences between guided

and flexible tools were premised on semantics and that CASE tools can be sufficiently classified as restrictive and guided.

An important role of a CASE tool in software development is to serve as a methodology companion by providing methodology support. A tool that provides methodology support should generate an error or warning when the rules of the methodology are violated (Hatley, 1988) Vessey, Jarvenpaa & Tractinsky (1992) proposed the first approach to classifying CASE tools with respect to methodology support. They identified three categories: restrictive, guided, and flexible tools; in this paper we use the categories put forward by Jankowski (1995) namely flexible and guided because as argued by Jankowski there is little distinction between restrictive and guided.

The nature of methodology support and the associated consistency checks is an important criterion for the evaluation of CASE tools. In addition, Church and Matthews (1995) from a review of practitioner and research literature proposed a set of criteria for evaluating tools that support both OO and structured methodologies. The criteria selected were: code generation, ease of use, document generation, and methodology support and consistency checking. Below, Table 1 gives summary descriptions of the criteria which serve as a guide for tool evaluation.

Criterion	**Description**
Support for structured and OO methodologies	Whether the tool can produce the construct diagramming found in both structured and OO approaches.
Automatic code generation	CASE tools can facilitate the design process allowing for some rudimentary code to be built and exported from the diagrams.
Document generation	CASE tools should support document generation by allowing the analyst to write scripts that can export components of the model into a regular word processing files.
Nature of methodology support and Consistency checking	Flexible and guided tools should check for methodological consistency.
Ease of learning	Dennis, Wixom and Roth (2006) define ease of learning as the ability of *a new user* to quickly learn and be able to use the tool effectively.
Ease of use	Ease of use pertains to the *expert user's* ability to get work done efficiently (Church & Matthews, 1995).

Table 1: Criteria for CASE tool evaluation

TOOL EVALUATION

The evaluation was performed in 2007 using the basic versions of Visible Analyst® and Rational Enterprise. To evaluate the tools we first created logical design artifacts. For both Visible Analyst® and Rational® Rose we created UML diagrams (use case, class, and

sequence). In addition we created structured design artifacts (a context diagram, data flow diagrams, and entity relationship diagrams) for Visible Analyst®. These logical design artifacts were created to model a simple personnel placement problem.

Using the criteria shown in Table 1 above, we evaluated both Rational® Rose and Visible Analyst®. First, we determined whether each tool supported one or both of structured and object-oriented (OO) methodologies. We then examined whether each tool has the features that allow for automatic code generation and document generation. We also looked at the nature of methodological support provided by each tool. Finally based on the definitional descriptions presented in Table 1 we assessed each tool for *ease of learning* and *ease of use* with respect to the application of the tools in creating the case study logical design artifacts.

RESULTS AND DISCUSSION

Table 2 summarizes the results of our evaluation.

Criteria	Visible Analyst®	Rational® Rose
Support for structured and OO methodologies	Supports structured and object-oriented design (UML)	Supports solely object-oriented design (UML)
Automatic Code Generation	Supports code generation for pre .Net versions of Visual Basic, C and COBOL. Will build and reverse engineer database schemas for SQL and Oracle	Supports code generation for pre .Net versions of Visual Basic, C++, Visual C++, and Java. Will build and reverse engineer database schemas for SQL and Oracle. Provides good integration with Java, and incorporates common packages into class diagrams and decompositions through classes.
Document Generation	Does not support printing of project documents. Can export and print all design diagrams.	Can export and print all design diagrams. With SoDa (the automated documentation application) documents and reports can be created and maintained.
Nature of methodology support and Consistency checking	Type: Flexible Consistency checking is user prompted.	Type: Flexible Consistency checking is user prompted.
Ease of Learning	Visible Analyst Tutorial is a vendor developed text that is provided with the software. A pdf version is also available. A new user can follow	There is no vendor provided tutorial. By searching the Internet a new user could find various tutorials developed by third-party users of Rational Rose.

		this step-by-step approach for learning how to use the software New users may find complex concepts that are not presented in the tutorial more difficult to learn.		The user must experiment to find a tutorial suited for his/her particular needs. Learning complex concepts may be more difficult.
Ease of Use		The interfaces are generally easy to use for the user, even though some modules are not well integrated for the use of interrelated diagrams.		Basic diagrams are easy to create (with sufficient OO conceptual foundations). The abundance of different modules makes the tool difficult to use.

Table 2: Summary of tool evaluation results

In our evaluation we found some similarities between the two tools. First both tools follow a flexible philosophy with respect to the nature of methodology support where consistency checks are user-prompted.

Based on the logical design artifacts created for our case study problem found that both tools enable the user to create professional looking diagrams and models with application tools that have simple drag and drop user interfaces. Visible Analyst® appears relatively easier to use and learn than Rational® Rose although additional research is needed to ascertain the validity of this proposition.

There are however some important differences between the tools. Visible Analyst® supports both the structured and object-oriented design methodologies whereas Rational® Rose only supports the object-oriented design using UML techniques. Visible Analyst® builds shell code with basic class structures in C. Rational® Rose on the other hand builds complete code in several languages, e.g. Java, C++ and Visual Basic. Additionally, when developing classes, Rational® Rose provides the ability to implement and extend classes from Java packages as well as the ability to use classes and data types such as array lists and vectors all of which cannot be done in Visible Analyst®.

Rational® Rose includes an add-on automated document module, SoDA, which enables the automated generation of project lifecycle documents and reports, whereas we found that project documentation generation is comparatively limited in Visible Analyst®.

CONCLUSION

From our study, we identified some similarities and differences between Visible Analyst® and Rational® Rose that can be used in case tool selection decisions in academic settings. Specifically, we found that Visible Analyst® can be used to support the teaching of systems/software development skills in both structured and object-oriented design approaches, whereas Rational® Rose supports the OO approach only. Rational® Rose provides more comprehensive support for automatic code and document generation than Visible Analyst®. Hence, Rational® Rose may be more appropriate for instructing large scale and complex development projects where documentation of code and activities are more cumbersome.

Visible Analyst® appears relatively easier to use and learn than Rational® Rose although additional research is needed to ascertain the validity of this proposition.

Future research will look at a more comprehensive classification scheme that includes the categorization of CASE tools in the UPPER and LOWER CASE nomenclature which represent support for the analysis and development phases of the systems/software development lifecycle respectively. We will also look at the impact of methodology support on student learning CASE tool adoption.

It appears that for educational purposes it may be important to separate the instruction of the systems design approach and specific methodologies from the instruction of CASE tool usage. Educators could incorporate a laboratory component to the course where students are instructed on how to use the tool using some methodology (e.g. Yourdon, Gane-Sarson or SSADM for the structured approached, and UML for the OO approach). Visible Analyst® has a vendor supplied tutorial that provides progressive lessons that students can follow to learn the tool in this regard while Rational® Rose does not have a vendor provided tutorial, although there are third party tutorials including educator developed materials that attempt to offer similar value. The presence of a vendor provided tutorial is advantageous in that it ameliorates the time constraints academics face when instructing CASE tool usage.

This study is limited in that it is an exploratory study, where the evaluation was performed by a couple of expert users. For future studies we intend to use our standardized artifacts on experimental groups and/or focus groups and controlling for user expertise thereby contributing to the generalizeability of the research findings.

REFERENCES

Banker, R. D., & Kauffman, R. J. (1991). Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study. *MIS Quarterly, 15*(3), 375-401.

Church, T., & Matthews, P. (1995, July 10-14). *An Evaluation of Object-Oriented CASE Tools: The Newbridge Experience.* Paper presented at the The 7th International Workshop on Computer Aided Software Engineering, Toronto.

Dennis, A., Wixom, B. H., & Roth, R. M. (2006). *Systems Analysis and Design* (3 ed.): Wiley.

Hatley, D. J. (1988). *CASE tool evaluation: A real-time example.* Paper presented at the Computer Aided Software Engineering (CASE '88), Boston, MA.

Jankowsi, D. J. (1995). CASE tool selection: using methodology support to choose the right tool for the job. *Journal of Systems Management, 46*(4), 20-26.

Limayem, M., Khalifa, M., & Chin, W. W. (2004). CASE Tools Usage and Impact on System Development Performance *Journal of Organizational Computing and Electronic Commerce, 14*(3), 153-174.

Massacci, F., Mylopoulos, J., & Zannone, N. (2007). Computer-aided Support for Secure Tropos. *Automated Software Engineering, 14*(3), 341 - 364

Orlikowski, W. J. (1989). *Division among the Ranks: The Social Implications of CASE Tools for System Developers* Paper presented at the The Tenth International Conference on Information.

Post, G. V., & Kagan, A. (2000). OO-CASE tools: an evaluation of Rose. *Information and Software Technology, 42*, 383-388.

Post, G. V., & Kagan, A. (2001). User Requirements for OO CASE Tools. *Information and Software Technology, 43*, 509-517.

Post, G. V., Kagan, A., & Keim, R. T. (1998). A comparative evaluation of CASE tools. *The Journal of Systems and Software, 4*, 87-96.

Vessey, I., Jarvenpaa, S. L., & Tractinsky, N. (1992). Evaluation of vendor products: CASE tools as methodology companions. . *Communications of the ACM, 35*(4), 90-104. .