USE OF CASE TOOLS IN THE SOFTWARE ACQUISITION MANAGEMENT PROCESS

Aimee A. Murray, Barbara J. Pemberton,
Judy E. Walton, Douglas J. Classe
Naval Training Systems Center
Orlando, Florida

ABSTRACT

A new application of Computer Aided Software Engineering (CASE) tools to assist managers improves the training systems acquisition process. The perils of schedule slips and cost overruns are detectable early and offset by managers integrating CASE tools into the process for training systems acquisition. CASE tools automatically provide: 1) graphic representations of software design; 2) reports on quantitative measurements of software quality; and 3) control flow diagrams which graphically disclose complex software modules that increase time and effort during system integration and testing. Managers can more effectively and efficiently monitor the development of software-intensive training systems with the application of CASE tools to the software acquisition management process.

ABOUT THE AUTHORS

Aimee A. Murray is a Computer Engineer for the Department of the Navy at the Naval Training Systems Center. Ms. Murray is currently the Computer Engineer for the several Army trainers. Ms. Murray is investigating the effectiveness of CASE tools to assist in the management of trainer system software acquisition. Ms. Murray holds a Master of Science in Engineering and a Bachelor of Science in Engineering from the University of Central Florida.

Barbara J. Pemberton is a Computer Scientist for the Department of the Navy at the Naval Training Systems Center. Ms. Pemberton is currently the Principal Investigator for research projects dealing with automatic scenario generation and control using artificial intelligence; Ada on distributed micro-processors for Aircrew training devices; and assessment of software methods, Ada, and CASE Tools for training systems. Ms. Pemberton holds a Master of Science in Management from Rollins College and a Bachelor of Science in Mathematics from the University of Tennessee. Her previous experience includes training system software, specification, and development at General Electric and Martin Marietta.

Judy E. Walton is a Software Engineer for the Department of the Navy at the Naval Training Systems Center. Ms. Walton is currently the Software Engineer for an Army program, and for research projects in the areas of automatic scenario generation and control using experts systems, assessment of software methods, ada, and case tools for training systems. Ms. Walton holds a Bachelor of Science in Computer Engineering from the University of Central Florida.

Douglas J. Classe is an Electronic Engineer for the Department of the Navy at the Naval Training Systems Center. He is currently a member of the Software Acquisition Team involved with a Navy trainer. He holds a Bachelor of Science Degree in Electrical Engineering from the University of Central Florida.

USE OF CASE TOOLS IN THE SOFTWARE ACQUISITION MANAGEMENT PROCESS

Aimee A. Murray, Barbara J. Pemberton
Douglas J. Classe, Judy E. Walton
Naval Training Systems Center
Orlando, Florida

INTRODUCTION

A new application of Computer Aided Software Engineering (CASE) tools improves the software acquisition management process. CASE tools automatically provide: 1) graphic representations of software design; 2) quantitative measurements of software quality; and 3) control flow diagrams showing software complexity. Integrating CASE tools into the software acquisition process, managers can more effectively monitor their acquisitions.

The perils of schedule slips and cost overruns are detectable early and offset by management utilizing CASE tools. Omissions and oversights in the software design are identifiable using an ***analysis and design tool.*** Quantitative measurements of software reliability, maintainability, portability, and adherence to coding standards are reportable with a ***quality measurement tool.*** Complex "spaghetti code" is detectable early with a ***test and complexity tool.*** The cumulative use of these tools results in a savings of management time and resources.

OBJECTIVE

The objectives of this paper are:

- Report on the use of CASE tools for monitoring the development of trainer system software and
- Discuss when to use CASE tools in the software acquisition management process.

CASE TOOLS FOR MONITORING ACTIVITIES

Although Computer Aided Software Engineering (CASE) tools are used primarily by software developers, an investigation has shown that CASE tools can also be effectively used by monitoring

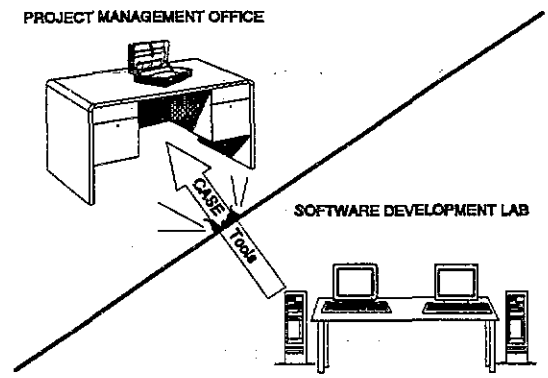

Figure 1. New Application of CASE Tools

activities. Three types of CASE tools are discussed:

- ***Analysis and Design Tool***
- ***Quality Measurement Tool***
- ***Test and Complexity Tool***

Multiple tools often come within one CASE product (e.g. the ***analysis and design tool*** that was studied consists of three separate tools). The CASE tools presented in this paper are considered representative of their specific areas and are currently available to project engineers in the Navy and Army. No attempt has been made to survey the entire CASE tool market.

Analysis and Design Tool

The ***analysis and design tool*** used for this study is composed of three parts: 1) a graphical design editor, 2) an automatic Ada Code generator, and 3) a reverse engineering tool.

The first part of the ***analysis and design tool,*** the graphical design editor, shows the software

requirements in a picture or software design diagram. It also shows the overall software design in a picture, e.g., a Buhr diagram. A Buhr diagram shows the software components -- packages, procedures, tasks and data -- as labeled figures (e.g., rounded boxes, square boxes, parallelograms, and ovals). See Figure 2. This is similar in concept to a CAD generated wiring diagram for hardware. These software design diagrams are useful during software design reviews to quickly explain software to a diversely trained group of review attenders.

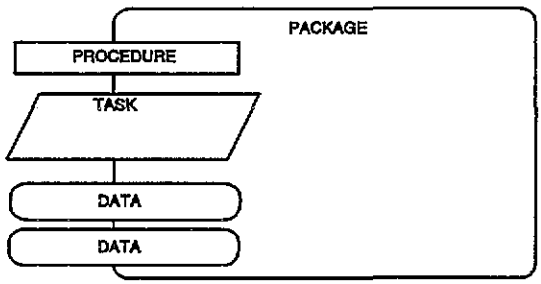

Figure 2. An *Analysis and Design Tool* Output Example (Buhr Diagram)

The second part of the ***analysis and design tool***, an automatic Ada code generator, creates the Ada Program Design Language, PDL, for a project from the software design diagram or Buhr diagram. This is most useful to the software developer. However, it assures the management staff there is consistency between the design pictured and the Ada PDL code produced.

The third part of the ***analysis and design tool***, reverse engineering, generates software design diagrams from compilable Ada PDL or Ada code. Software design diagrams show how the software changes as it progresses from the initial software design to the detailed software design.

Using the software design diagrams, the management staff quickly (i.e., minutes or hours rather than days) sees the answers to the following questions: (See Figure 3)

1) Requirements Visible?
   - Are all the required inputs/outputs visible in the software design diagrams?
   - Are all the major required capabilities visible in the software design diagrams?

2) Efficient Ada?
   - How and where is the software using Ada features such as tasking and packaging? The use of tasking may be appropriate for non-real time instructor station function, but may not necessarily be appropriate for other real-time functions.

3) Reusable/Consistent Design?
   - Is a structured, object-oriented or state based software system being developed? This is important to determine the potential reuse of the software and future ease of software modification.
   - Is the software design consistent? For example, do all packages and procedures interface in the same way with the system? This will impact the ability to develop, test, and maintain the software.

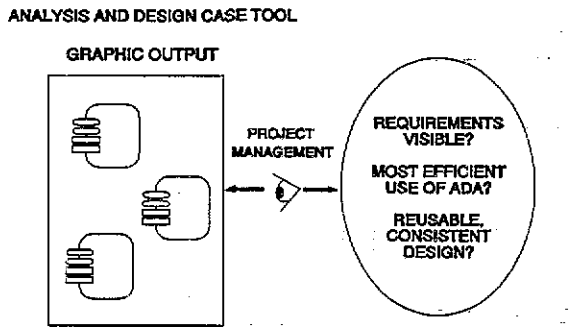

Figure 3. Using the *Analysis and Design Tool* Graphic Output

The above is just a sample of the information obtainable from the software design diagram. It should be noted, that software developers might also use the software design diagram as an internal automated quality assurance check.

This paragraph describes an application of reverse engineering to a contract. The contract required modification of existing trainer software. Prior to the Critical Design Review, baseline trainer software was analyzed using the reverse engineering tool. Comparing the graphic output of the reverse engineering tool against the new documentation submitted, two value-added benefits resulted. The first, a reduction in overall document review time from days to hours. The second, omissions of specification requirements were found.

Quality Measurement Tool

The ***quality measurement tool*** used in this study provides a measure of Ada code reliability, maintainability, and portability. One hundred fifty three characteristics of Ada code are examined automatically by the tool. The tool can be customized to measure how well software developer's software coding standards and guidelines are actually implemented in the Ada Code. Customizing the tool could also aid assessment of the code reusability and code testability.

The tool was used by one managing office to compare multiple Ada programs. The results of the comparison indicated that translating Fortran code to Ada resulted in lower reliability and maintainability scores. Additionally, code written in the 90s that uses a more object oriented approach scored higher in reliability and maintainability than code written in the late 80s with a structured approach.

Table 1 is a table of project metrics from three trainer projects. The scores shown in the row titled "Quality Measurement Scores" are the software ratings for reliability, maintainability, portability and all criteria. (The numeric range on scores is 0 to 1, with 1 or 100% being the best possible.)

Table 1.
Evaluation and Assessment Table

Program Name		Navy SBIR Flight/Helo Simulator		Army Helicopter	Army Proced. Trainer
# Packages		49		338	43
# Procedures/Functions		360		1110	122
# Tasks		None		33	None
# Lines of Code (with comments)		53139		53756	7285
Host Development System		Sun		MicroVax II	Zenith '286
Target System		Motorola 68030		Motorola 68030	Zenith '286
Software Design/ Structure Design	Operating System	ASART (SBIR Product)		ARTX32	DOS
	Application	Translated FORTRAN	OOD	Structured Design/ Object-Based	Structured Design
Quality Measurement Scores	Category	ALL	ASART	ALL	ALL
	Reliability	.10	.77	.50	.37
	Maintainability	.17	.63	.60	.47
	Portability	.97	.90	.97	.89
	All Criteria	.69	.88	.88	.75
Ada Compiler		Verdix		Telesoft	Alsys
Hardware (# processors)		5		2 - IOS, 4 - Students	1
Development Stage		Done (delivered in '91)		Done (delivered in '91)	Done (delivered in '89)

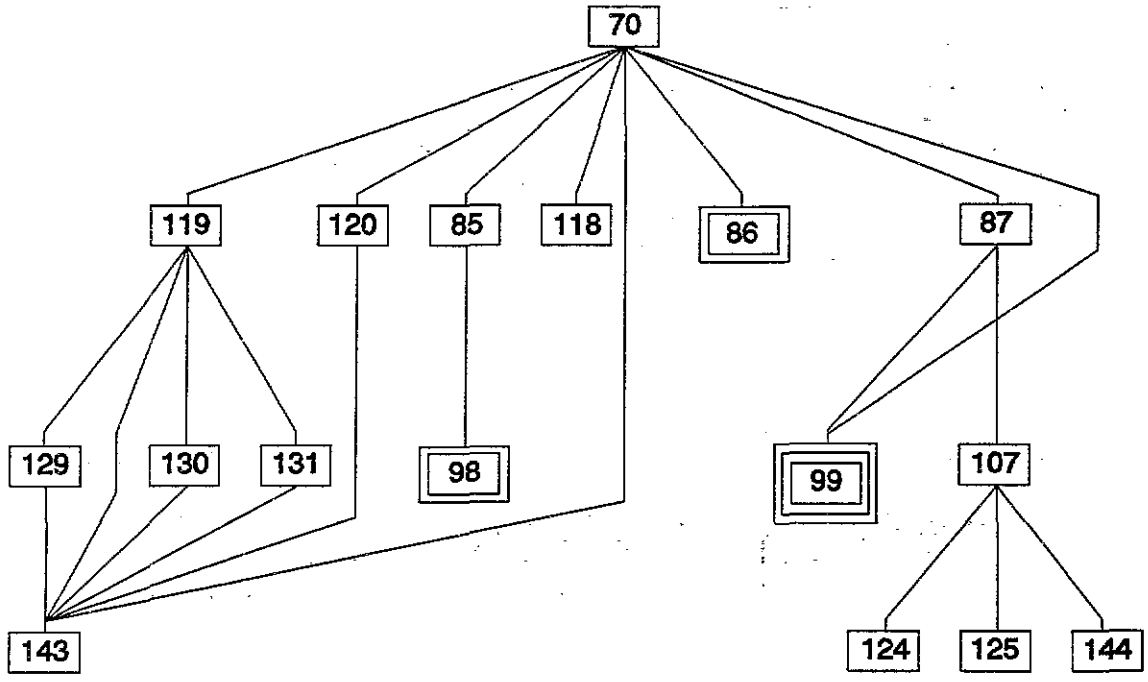

Figure 4. Test and Complexity Output Example

Test and Complexity Tool

The *test and complexity tool* is beneficial for detecting overly complex code. Historically, this type code is high risk to programs in terms of schedule slips and cost overruns. Highly complex software code is routinely error prone and increases the time and effort needed for testing and integration.

The *test and complexity tool* has the capability of producing control flow graphs depicting the control flow through software systems or software units. It also provides several McCabe complexity metrics on each software unit within a system.

At the software unit level, a test path flow graph and annotated test path listing are provided. This unit level information helps managers assess the completeness and extensiveness of software unit testing.

A system control flow graph produced by the *test and complexity tool* is shown in Figure 4. The numbered boxes represent software modules, or units. The numbers correlate to unit names which can also be shown. An indication of the risk associated with each software unit is shown as nested boxes. (e.g., three boxes - high risk, two boxes - marginal risk, one box - low risk). On color monitors the risk ratings are also shown in color -- red, yellow, and green.

Looking at Figure 4, managers should be concerned with modules 86 and 98. These units may involve additional effort to test and integrate. Module 99 shows high risk and requires attention. This module should be simplified or separated into less complex units. High risk units require additional time to test and integrate, and are less maintainable.

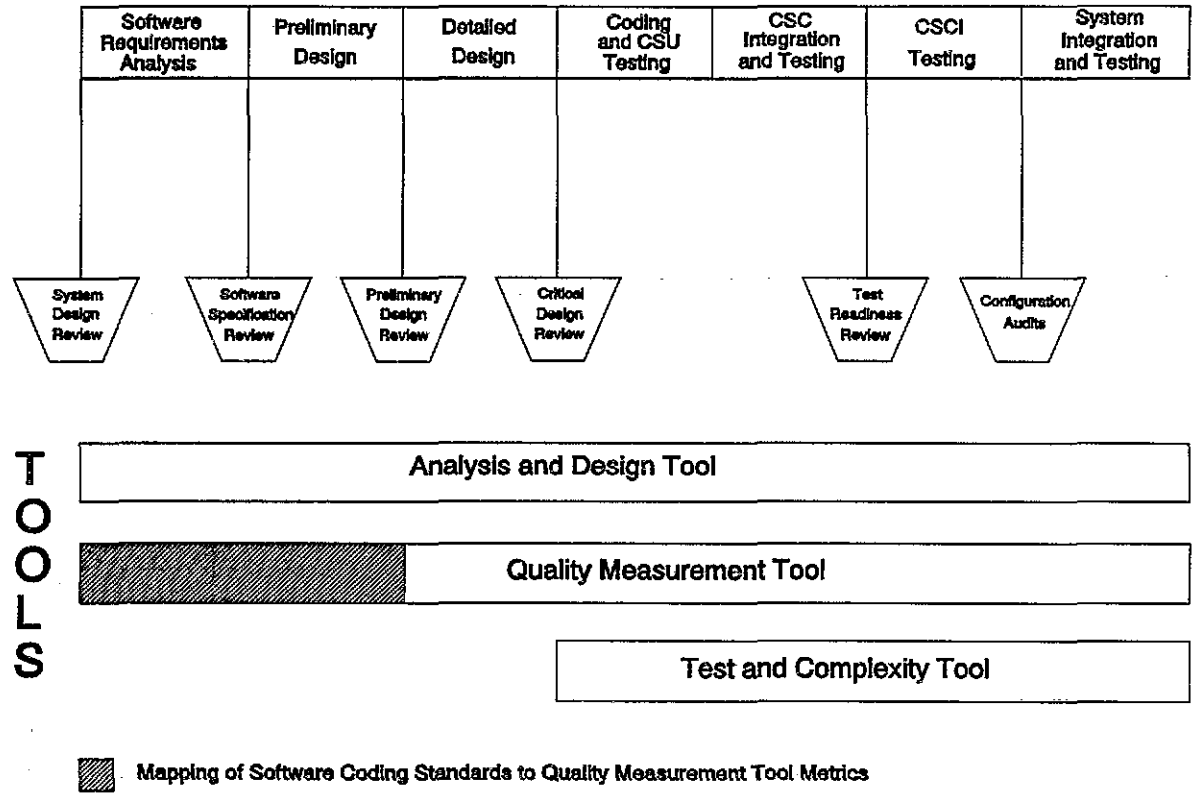

Figure 5. **The Mapping of Applicable CASE Tools to Milestone Reviews.**

APPLICATION OF THE CASE TOOLS TO THE MILESTONE REVIEWS

Most military program milestone reviews are in accordance with MIL-STD-2167A, Defense System Software Development and MIL-STD-1521B, Technical Reviews and Audits for Systems, Equipments, and Computer Software. These reviews include the system design review, the software specification review, the preliminary design review, the critical design review, test readiness review, configuration audits and acceptance testing. Figure 5 is a mapping of the CASE tools applicable to milestone reviews.

The following paragraphs recommend a beneficial application of CASE tools to assist in the software acquisition management process.

System Design Review

The ***analysis and design tool*** is usable by management staff for preparing and presenting software information at a System Design Review. It provides a pictorial view of software requirements through diagrams. This enables visual correlation between specification requirements and software requirements.

Software Specification Review

The ***analysis and design tool*** is usable by management staff for timely reviews of software documents delivered in preparation for the Software Specification Review. The tool's graphic output of the software requirements provides the information to review the Requirements Traceability Matrix of the delivered documents (e.g. Software Requirements Specification) to ensure no omissions or oversights of requirements.

Preliminary Design Review

Two tools, ***analysis and design tool*** and ***quality measurement tool***, are usable by management staff preparing for a Preliminary Design Review. The tools' graphic output standardizes what information needs to be presented at the preliminary design review.

Using the ***analysis and design tool***, the contractually required program design language (PDL) specifications are reverse engineered into software design diagrams. These software design diagrams are used to assess the software design for completeness of requirements, efficient use of Ada, and reusable, consistent design. These diagrams are useful in review of software documents prior to the design review.

The compilable Ada specification can be checked for adherence to the agreed on software coding standards by using a ***quality measurement tool.***

Critical Design Review

Three tools, ***analysis and design tool, quality measurement tool***, and ***test and complexity tool*** are usable by management staff preparing for a Critical Design Review. The tools' graphic outputs standardize the content and format of information presented at the critical design review.

Using the ***analysis and design tool***, the contractually required Ada program design language (PDL) specifications and bodies are reverse engineered into software design diagrams. These diagrams are used to assess the detailed software design for completeness of requirements, efficient use of Ada, and reusable, consistent design. These diagrams are useful in review of software documents prior to the design review.

Adherence to the software coding standards within the specifications and bodies can be assessed with the ***quality measurement tool.***

A preliminary assessment of the test plans and descriptions can be made with the ***test and complexity tool.*** The ***test and complexity tool*** will produce diagrams depicting the control flow within the code and provide several quantitative complexity metrics. The tool also provides an identification of all test paths within a unit of code.

Test Readiness Review

Three tools, ***analysis and design tool,*** **quality measurement tool,** and ***test and complexity tool*** are usable by management staff preparing for a Test Readiness Review.

The tools applicable at this review are similar to the previous two design reviews. Additionally, the ***test and complexity tool*** graphically assists management to identify overly complex code. Historically complex code is error prone and increases the time and effort needed for testing and integration. Flagging complex code is an unbiased, automated capability of the ***test and complexity tool.***

Configuration Audits

Three tools, ***analysis and design tool, quality measurement tool,*** and ***test and complexity tool*** are usable by management staff preparing for configuration audits.

Reverse engineering of deliverable code using the ***analysis and design tool*** makes an up-to-date diagram of the deliverable software code. Concurrence between the code and documentation can be reviewed thoroughly.

A final assessment of the contractor's code with the ***quality measurement tool*** ensures that the contractor has maintained full compliance with the defined software coding standards and guidelines.

A final assessment of the contractor's code with the ***test and complexity tool*** provides diagrams of the control flow. These control flow diagrams are useful for maintenance efforts.

SUMMARY

The three tool types investigated proved to be effective tools to assist in monitoring the activities of software acquisitions. The ***analysis and design tool*** group produces descriptive figures which provide insight and comprehension of software design. The ***quality measurement tool*** produces reports on quantitative measurements of software quality. This tool is tailored to quantify compliance with software standards required by the contract. The ***test and complexity tool*** produces control flow diagrams to indicate complex, error-prone modules of the code.

CONCLUSIONS

Typically, Computer Aided Software Engineering (CASE) tools are used by software developers to increase efficiency, reduce human error, and automate mundane software development tasks. However, this paper concludes CASE tools can also be applied successfully for efficient and effective management of the software acquisition process.

RECOMMENDATIONS

Current information collected from the use of CASE tools are entered into a table of project metrics. The authors recommend: 1) expanding the table of project metrics to include more projects, 2) researching additional CASE tools applicable to the software acquisition management process, and 3) publishing of results for both Government and Industry.

REFERENCES

Dynamic Research Corporation. (1991). ADAMAT™ Version 1.1 [Computer Program]. Andover, Massachusetts.

Mark V Systems Limited. (1990). ADAGEN™ Version 1.7.3 [Computer Program]. Encino, CA.

McCabe & Associates, Inc. (1992). BATTLEMAP Analysis Tool™ Demonstration Disk [Computer Program]. Columbia, MD.

Software Technology Support Center. (1991). Toolbox/PC Disk System, Version 2.0 [Computer Program]. Hill Air Force Base, Utah.