A Paper Presentation on Software Development Automation by Computer Aided Software Engineering (CASE)

Nishant Dubey

School of Computer and Electronics, IPS Academy
Indore, MP, PIN 452012, India

Abstract

Now a day, system developers are faced to produce complex, high quality software to support the demand for new and revised computer applications. This challenge is complicated by strict resource constraints, forcing management to deploy new technologies, methods and procedures to manage this increasingly complex environment. Often the methods, procedures and technologies are not integrated. Therefore, they achieve less than desired improvements in productivity, or force management to make tradeoff decisions between software quality and developer efficiency. Thus the production lines have to be developed faster, too. A very important role in this development is *Software Engineering* because many production processes are 'computer aided', so software has to be designed for this production system. It seems very important to do the software engineering right and fast.

Keywords: *CASE, Software Engineering, Tools, Process,*

1. Introduction

Software Engineering is still a relatively new area of engineering. Indeed the phrase itself gained widespread use after a 1968 NATO-sponsored conference. As the name suggests, Software Engineering deals with the exposing of the process of designing, creating and maintaining software. It is perhaps useful to bear in mind that a structured approach to a solution is in no way a barrier to creativity. Indeed, much modern architecture, although bound by rigid guidelines and specifications can be truly breathtaking.

Today everything has to go faster. Because of the increasing speed of changing market-demands new products replace old ones much earlier than before, so the development of new products has to go faster. Thus the production lines have to be developed faster, too In the past, software systems were build using traditional development techniques, which relied on hand-coding applications.

Computer-Aided Software Engineering (CASE) helps system developers meet their challenge by providing a new generation of integrated system development tools which provides an automated environment in which to design and implement system projects. CASE technology enables system developers to improve both quality and efficiency, resulting in a net improvement in maintenance and development productivity.

The objectives of this paper are to provide the reader with sufficient information about CASE technology to develop an evaluation and implementation strategy for utilizing CASE to improve systems development productivity.

2. Computer Aided Software Engineering

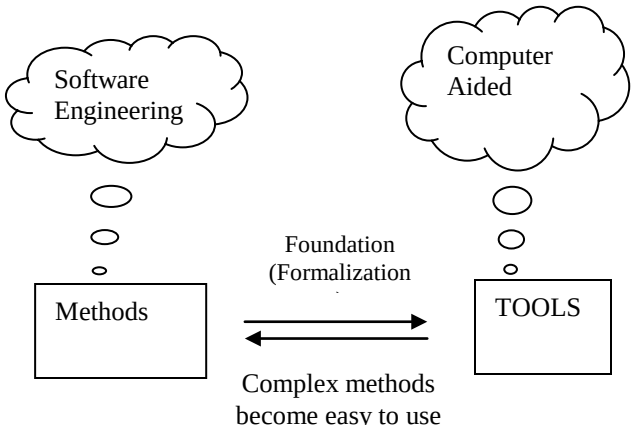

Fig 1 CASE

This term is used for a new generation of tools that applies engineering principles to the development and analysis of software specifications. Simply, computers develop software for other computers in a fast way by using specific tools. When implanted in a concurrent engineering environment, this process is taking place while some parts of the developed software are running already. It's a sort of on-line software engineering. There are a lot of problems with these kinds of systems because they are very complex, not easily maintainable and fragile.

The tools developed right now are evolutional products out of earlier tools. The first tools, developed in the mid '70s, were

mainly introduced to automate the production and to maintenance structured diagrams. When this automation covers the *complete life-cycle process*, it is called Integrated Computer Aided Software Engineering (I-CASE). When only one specific part of the life-cycle process is covered we called just- Computer Aided Software Engineering (CASE).

Recently, CASE tools have entered a third phase: the introduction of new methodologies based on capabilities of I-CASE tools. These new methodologies utilize Rapid Prototyping techniques to develop applications faster, at lower cost and higher quality. By using Rapid Prototyping a prototype can be made fast, so the developed system can be tested more often in between the development-phases because it doesn't cost much time to create a prototype. Mistakes can be detected and corrected earlier this way. The earlier this can be done, the better because correcting these mistakes gets harder and more expensive when the system is developed further. So a lot of time and money can be saved using Rapid Prototyping.

As said above, a new set of tools is necessary. These tools should automate each phase of the life-cycle process and tie the application development more closely to the strategic operations of the business. A lot of different tools have been developed over the years and are being developed right now. There are so many tools, that we could easily get confused. To survey all these CASE tools we divide them in the following categories:

i. *Business Process Engineering Tools* – The primary objectives of these tools is to represent business data objects, their relationships and how these data objects flow between different business areas within a company.
ii. *Requirement Tracing Tools* – The objective of these tools is to provide a systematic approach to the isolation of requirements that begins with the proposal or specification of customer's request. The typical requirement of these tools includes human interactive text evaluation with the database management system which stores and categorize every system need that is parse from the original specification.
iii. *Software Configuration Management Tools* – it is lies in the kernel of every CASE environment. These tools help in identification, version control, change control, auditing and status accounting.
iv. *Process Modeling and Management Tools* – Process Modeling Tools represent the key elements of a process such that it can be better understood. Process management tools provide the links to other tools for providing support defining process activities.
v. *PRO/SIM Tools* – Prototyping and Simulation tools provide the software engineer with the ability for predicting the behavior of a real time system prior to the time that it is made, and these tools enable for developing mock ups of real time systems.
vi. *Project Planning Tools* – These tools focus on Software Project Effort and Cost Estimation and Project Scheduling.
vii. *Matrix and Management Tools* – Software metrics improve the ability of manager to control and coordinate the process of software engineering. Management tools capture project specific metrics which provide the overall indication of productivity or quality.
viii. *Analysis and Design Tools* – These tools enable to create the models of system to be made. This model has a representation of data, function and behavior and characterization of data architectural, component-level and design interface.
ix. *Project Management Tool* – On a continuing basis the project plan and schedule must be tracked and monitored. These tools are extension for project planning tools.
x. *Risk Analysis Tools* – These tools enable a manager to make a risk table by providing detailed guidance in the risk identification and risk analysis.
xi. *Documentation Tools* – These tools provide an important opportunity to improve productivity. Document production tools support every aspect of software engineering and represent a substantial leverage opportunity for all developers.
xii. *Prototyping Tools* – Prototyping tools enables the creating of data design, coupled with both screen and report layouts.
xiii. *Quality Assurance Tools* – CASE tools majority claims on majority claim on quality assurance to focus are the metrics tools for auditing source code to determine compliance with language standard. For building the quality of software other tools extract technical metrics in an effort to the project.
xiv. *Programming Tools* – These tools encompass the compilers, editors and debuggers to support most conventional programming languages. In this category OOP environment, application generators and database query language resides.
xv. *Interface Design & Development Tools* – These tools are software component tool kit which contained menus, buttons, window, icon, device drivers etc. These tool kits replaced by prototyping tools.
xvi. *System Software Tools* – CASE is workstation technology and high quality network system software, distributed component support, bulletin board, email, object management services and other capabilities of communication.
xvii. *Database Management Tools* – These tools are include on the emphasis of configuration object from RDBMS to Object Oriented DBMS for CASE.
xviii. *Integration and Testing Tools* – in this the following testing tools are categorized
    a. Data Acquisition – These acquire the data that is to be used during testing.
    b. Statement Management – These analyze the source code without executing the test cases.
    c. Dynamic Measurement – These analyze the source code during execution.
    d. Simulation – These simulate the function of hardware or other externals.
    e. Text Management – These help in the planning development and the control of testing.
    f. Cross Functional Tools – These cross the bounds of the preceding categories.

xix. *Web Development Tool* – These tools help in the generation of text, graphics forms, scripts, applets and other elements of a webpage.

xx. *Client Server Testing Tools* – The client/server environment demands the testing tools that exercise the graphical user interface and the network communication requirements for client and server.

xxi. *Test management Tools* – These tools are used to control and coordinate software testing. These manage and coordinate regression testing, platform comparison and conduct the batch testing of programs with human/computer interfaces interactively.

xxii. *Reengineering Tools* – In these tools for legacy software address a set of maintenance activities that absorb significance percentage of all software related effort.

xxiii. *Static Analysis Tool* – Static testing tools help in deriving test cases. In this code based testing tools takes source code and perform number of analyses that results in test case generation. Specialized testing language enable to write detailed test specifications that describe each test case and logistics for its execution. Requirement based testing tools isolate specific user requirements and suggests test cases for exercising the requirement.

xxiv. *Dynamic Analysis Tools* – Dynamic tools can be either intrusive or non intrusive. Intrusive tools change the software that is to be tested by inserting extra instruction that perform the activities. Non intrusive testing tools use a separate hardware processor to contain the program being tested.

3. CASE Building Blocks

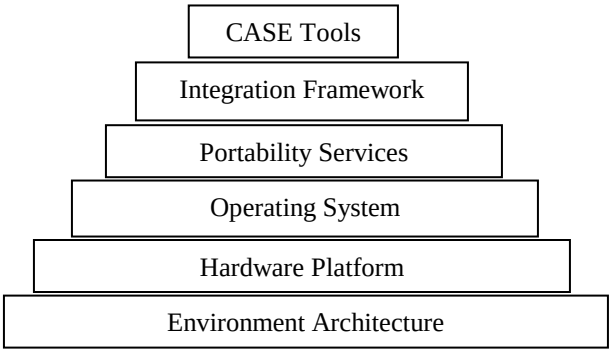

FIG 2 CASE BUILDING BLOCKS

Each building block forms a foundation for the next, with tools sitting at the top of the heap. The environment architecture composed of hardware platform and operating system lays the ground work for the CASE. The operating system includes the networking software, database management and object management services. Portability services provide a bridge between CASE tools and their integration framework and the environment architecture. Integration framework is a collection of specialized programs that enables individual CASE tools to communicate with one another, to create a project database and to exhibit the same look and feel to the software engineer (end user).

4. Conclusion

CASE technology is now available for most routine activities in the software process. This has led to some improvements in software quality and productivity although these have been less than predicted by early advocates of CASE.

Various companies offer CASE software capable of supporting some or all of these activities. While many CASE systems provide special support for object-oriented programming, the term CASE can apply to any type of software development environment.

5. References

1. Sommerville, Ian (2004). *Software Engineering*, Second. Pearson Education. ISBN 0321210263.
2. Pressman, Roger S. (2005). *Software Engineering: A Practitioner's Approach*, Sixth. McGraw-Hill. ISBN 0072853182.
3. Jones C.B., Software Development – A Rigorous Approach, London, Prentice Hall
4. Fairly R,(2000) Software Engineering Concept, Tata Mc Graw Hill Pub.
5. Pfleeger S.L.,(1988) Software Engineering, Prentice Hall International Inc.
6. Caine S & E Gordon, (1975), A tool for Software Design, Vol 47,AFIPS Press, Montvale, NJ
7. Halstead M. H.,(1977) Elements of Software Science, New York, Elsevier North Holland
8. Bennett K ,(1991), Software Maintenance, Butterworth – Heinemann Ltd, Ox Ford
9. Sage A and Palmer J.D.,(1990), Software Engineering, John Willey & Sons
10. Capers Jones(2009), Software Engineering Best Practices, Edition-1, ISBN: 9780070683594
11. David Gustafson,(2003), Schaum's Outline of Software Engineering, Edition: 1, ISBN: 9780071377942
12. Carma L. McClure,(1988), Prentice Hall Publication, ISBN : 9780131193307