ABSTRACT

CASE is an excellent acronym to be applied to all the managerial, technical, maintenance and support activities in building new systems and enhancing existing systems. This paper will describe the key technologies and trends affecting our ability to develop higher quality software more productively. The components of a complete CASE environment will be described along with the major technical and organizational issues in implementing CASE to help the reader become a smart shopper of CASE methods and tools.

What is CASE?

CASE can mean different things depending whether you're a vendor or a buyer. To the vendor, CASE means Computer-Aided Software Engineering, but a potential buyer, after listening to sales presentations and reading product literature, may think CASE is a Confusing Array of Software that does Everything. After closer inspection, CASE is not a confusing or complex concept, but a straightforward common sense approach to engineering quality software systems more productively.

A Definition of CASE

CASE was first used in 1984 to focus attention on tools which automated structured analysis and design. CASE, however, is an excellent term to apply to all methods, procedures, techniques and tools which can be applied to the job of engineering software. CASE is to the software engineer as CAD/CAM is to the mechanical engineer and CAE is to the electrical engineer. To this end, I offer the following definition:

CASE is the development and effective use of appropriate management approaches, systematic procedures and methods, and automated tools that permit teams of software engineers to produce software that 1) meets business and system requirements; 2) is completed with a predictable schedule; 3) is available within budget guidelines; 4) allows for easy maintenance and enhancement.

What the definition stresses is the integration of management approaches, methods, and procedures with the automated tools.

A Complete CASE Environment

If I wanted a complete CASE environment, I would want to have all the components depicted in figure 1.
Most companies already have all the components in some form. Manual techniques are legitimate alternatives to automation and need to be considered when assessing your current software engineering environment. CASE provides computer-aided alternatives to the components of figure 1. What is lacking with most tools is the integration of the environment. When evaluating a CASE tool, we must consider the new tools effect on current manual techniques and existing tools.

Some CASE vendors are developing products with a higher degree of integration offering support for more software engineering life cycle phases and job tasks. Figure 2 illustrates the dimensions of software engineering which with evolve into integrated CASE environments.

**CASE Implementation Issues**

Each year CASE Associates Inc. surveys users of CASE methods and tools to identify the key issues affecting the successful application of CASE. These issues will help new buyers better evaluate CASE tools and question vendors on their product strategy. No vendor has solved all these issues, nor may any solve them in 2 to 3 years. What is important to you as a buyer, is to feel comfortable with the vendor's understanding of your application problem, and the vendor's vision to evolve his products to meet your requirements.

**Organization Issues**

CASE tools change the way software engineers work. To successfully change behavior you must have top management commitment to the new approach and make a substantial investment in training. Establishing proper CASE tool evaluation and selection criteria is a direct result of auditing your current software engineering methods, practices, techniques and tools. The audit identifies areas of improvement and prioritizes the types of automation which will have the greatest impact on productivity and quality. The training is a direct result of modified methods and procedures which are a prerequisite to shopping for automated tools. When an organization is not properly prepared for CASE, the new tools could end up as "shelfware".

**Technology Issues**

With proper organization preparation, technology issues can be prioritized to meet your organization's requirements. You will be able to question vendors thoroughly to insure the tools you select complement your software engineering culture and meet your application development and maintenance needs.

---

**THE FOUR DIMENSIONS OF CASE**

- **The Design Dimension** → **Structured Analysis and Design** → **Front-end CASE**
- **The Development Dimension** → **Traditional Programming Tools** → **Back-end CASE**
- **The Management Dimensions** → **Project and Process Management Methods** → **IPSE's**
- **The Support Dimension** → **Re-Engineering, Maintenance and Documentation Tools** → **Basic CASE**
Techniques for Analysis and Design. Tools offer different analysis and design techniques. The alternatives for specifying processes, data, data flow, control flow and state transition need to be understood. Ask the vendor to describe 1) the alternative techniques offered; 2) the degree to which these techniques are integrated between themselves and other tools; 3) the potential effect the technique may have on your software engineering organization; 4) what training is required and who offers it.

Project Databases and Team Engineering. The project database (data repository) is the key enabler for team engineering activities. The repository provides the possibility for tool to tool data exchange, uniform tool integration, project-wide design analysis and consistency, checking feedback from code generation and testing to the more abstract front-end specifications, project-wide task and configuration management with requirements tracing, and project-wide security and access mechanisms. If a vendor's repository strategy or means of supporting team engineering is not compatible with your requirements, there's no need to give the vendor further consideration.

Automatic Code Generation. There are vendors with code generators claiming 100% code generation. Ask the vendor under what conditions 100% of the code is generated. There is sometimes a "gap" between high-level structured analysis and design and the low level representations required to generate 100% of the executable code. Usually for performance reasons, handwritten code is required. Some products do not detect the handwritten code so changes can be made to the design documentation. Without changing the design documentation, the source code and design are out of sync. Vendors are attempting to close the "gap" and provide back annotation from source code to design specifications. The three ways used for closing the "gap" are through expert systems, reusable software libraries and prototyping with step-wise refinement.

Reverse Engineering and Re-Engineering. The caring for existing systems has recently become a priority for CASE tool vendors. The objective is to create the "as built" design model for existing code so the new CASE design tools can remove requirements which no longer apply and add new requirements. In the 90's, CASE environments will have integrated reverse engineering, design engineering, and code generation.

Configuration Management. CASE tools make it easier to introduce changes into a software project putting stress on manual change control procedures. Controlling software configurations is an important step toward developing and maintaining quality systems. A thorough software configuration management tool must manage the interrelationships and evolution of software development, components such as requirement documents, structured specifications, designs, source code, executable modules, test procedures, test data and user documentation.

Requirements Tracing. With the emergence of object-oriented design repositories, it is becoming possible to trace the evolution of requirements from design through development to implementation. By showing the correspondence between a specific requirement and the project deliverables that satisfy the requirement, system verification and validation can be assured.

Standards for Tool Integration. Data Exchange and Open Architecture. There are three key issues facing every software engineering organization. First, how can I integrate the new tools with my existing tools. Second, how can data from a tool be passed to another tool. Third, how can I change the rules or tailor the tool to best meet my changing requirements. A vendor's position on these issues are critical to evaluate.

Software Testing. The automation of analysis and design has provided new testing capability for completely verifying and validating systems. Test cases can be generated from requirements and design specifications. Good designs minimize the chance of coding errors (built well) while test case generation verifies you built the right thing.

Path and coverage analyzers and complexity measurement tools are classes of testing tools to enforce coding standards, to insure the designs and resultant code satisfy good engineering principals and are easier to maintain.
Project Management Implications. Project management should be integrated with analysis and design tools and methods. Impacts of design changes need to trigger a new configuration version and also show impacts on the project plans. The project management system's integration with the CASE environment provides the valuable statistics needed for measuring productivity and quality improvements.

Summary

Becoming a smart shopper of CASE tools requires the patience to ask vendor's the right questions. Understanding the issues presented in this paper will help guide your investigation of CASE alternatives. Smart CASE Shopping means 1) the impact on the organization is understood and manageable; 2) the tools meets today's application requirements; 3) the vendor's product direction is consistent with your long term needs.