**Computer-aided software engineering** (**CASE**) is the application of a set of tools and methods to a [software](http://en.wikipedia.org/wiki/Software) system with the desired end result of high-quality, defect-free, and maintainable software products.[[1]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-1) It also refers to methods for the development of [information systems](http://en.wikipedia.org/wiki/Information_system) together with automated tools that can be used in the [software development process](http://en.wikipedia.org/wiki/Software_development_process).[[2]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-2)

* [1 History](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#History)
* [2 Components](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Components)
* [3 Supporting software](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Supporting_software)
  + [3.1 Tools](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Tools)
  + [3.2 Workbenches](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Workbenches)
  + [3.3 Environments](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Environments)
* [4 Applications](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Applications)
* [5 Risks and associated controls](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Risks_and_associated_controls)
* [6 See also](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#See_also)
* [7 References](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#References)
* [8 External links](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#External_links)

History

The ISDOS project at the University of Michigan initiated a great deal of interest in the whole concept of using computer systems to help analysts in the very difficult process of analysing requirements and developing systems. Several papers by Daniel Teichroew fired a whole generation of enthusiasts with the potential of automated systems development. His PSL/PSA tool was a CASE tool although it predated the term.

Another major thread emerged as a logical extension to the DBMS directory. By extending the range of meta-data held, the attributes of an application could be held within a dictionary and used at runtime. This "active dictionary" became the precursor to the more modern "model driven execution" (MDE) capability. However, the active dictionary did not provide a graphical representation of any of the meta-data. It was the linking of the concept of a dictionary holding analysts' meta-data, as derived from the use of an integrated set of techniques, together with the graphical representation of such data that gave rise to the earlier versions of I-CASE.

The term CASE was originally coined by software company Nastec Corporation of Southfield, Michigan in 1982 with their original integrated graphics and text editor GraphiText, which also was the first microcomputer-based system to use hyperlinks to cross-reference text strings in documents—an early forerunner of today's web page link. GraphiText's successor product, DesignAid, was the first microprocessor-based tool to logically and semantically evaluate software and system design diagrams and build a data dictionary.

Under the direction of [Albert F. Case, Jr.](http://en.wikipedia.org/wiki/Albert_F._Case,_Jr.) vice president for product management and consulting, and Vaughn Frick, director of product management, the DesignAid product suite was expanded to support analysis of a wide range of [structured analysis and design methodologies](http://en.wikipedia.org/wiki/Structured_Analysis_and_Design_Technique), notably [Ed Yourdon](http://en.wikipedia.org/wiki/Ed_Yourdon) and [Tom DeMarco](http://en.wikipedia.org/wiki/Tom_DeMarco), Chris Gane & Trish Sarson, Ward-Mellor (real-time) SA/SD and [Warnier-Orr](http://en.wikipedia.org/wiki/Warnier-Orr" \o "Warnier-Orr) (data driven).

The next entrant into the market was Excelerator from Index Technology in Cambridge, Mass. While DesignAid ran on Convergent Technologies and later Burroughs Ngen networked microcomputers, Index launched Excelerator on the IBM PC/AT platform. While, at the time of launch, and for several years, the IBM platform did not support networking or a centralized database as did the Convergent Technologies or Burroughs machines, the allure of IBM was strong, and Excelerator came to prominence. Hot on the heels of Excelerator were a rash of offerings from companies such as Knowledgeware (James Martin, [Fran Tarkenton](http://en.wikipedia.org/wiki/Fran_Tarkenton) and Don Addington), Texas Instrument's [IEF](http://en.wikipedia.org/wiki/Information_Engineering_Facility) and Accenture's FOUNDATION toolset (METHOD/1, DESIGN/1, INSTALL/1, FCP).

CASE tools were at their peak in the early 1990s. At the time [IBM](http://en.wikipedia.org/wiki/IBM) had proposed [AD/Cycle](http://en.wikipedia.org/w/index.php?title=AD/Cycle&action=edit&redlink=1), which was an alliance of software vendors centered around IBM's [Software repository](http://en.wikipedia.org/wiki/Software_repository) using [IBM DB2](http://en.wikipedia.org/wiki/IBM_DB2) in [mainframe](http://en.wikipedia.org/wiki/Mainframe_computer) and [OS/2](http://en.wikipedia.org/wiki/OS/2):

*The application development tools can be from several sources: from IBM, from vendors, and from the customers themselves. IBM has entered into relationships with Bachman Information Systems, Index Technology Corporation, and Knowledgeware, Inc. wherein selected products from these vendors will be marketed through an IBM complementary marketing program to provide offerings that will help to achieve complete life-cycle coverage*.[[3]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-ADC_SAaA-3)

With the decline of the mainframe, AD/Cycle and the Big CASE tools died off, opening the market for the mainstream CASE tools of today. Nearly all of the leaders of the CASE market of the early 1990s ended up being purchased by [Computer Associates](http://en.wikipedia.org/wiki/Computer_Associates), including IEW, IEF, ADW, [Cayenne](http://en.wikipedia.org/w/index.php?title=Cayenne_(software_engineering)&action=edit&redlink=1), and Learmonth & Burchett Management Systems (LBMS).

Components

1. Diagrammatic Tool
2. Information Repository
3. Interface Generators
4. Management Tools

Supporting software

Alfonso Fuggetta classified CASE into 3 categories:[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering" \l "cite_note-AF_93-4)

1. *Tools* support only specific tasks in the software process.
2. *Workbenches* support only one or a few activities.
3. *Environments* support (a large part of) the software process.

Workbenches and environments are generally built as collections of tools. Tools can therefore be either stand alone products or components of workbenches and environments.

**Tools**

CASE tools are a class of [software](http://en.wikipedia.org/wiki/Software) that automate many of the activities involved in various [life cycle](http://en.wikipedia.org/wiki/Product_lifecycle_(engineering)) phases. For example, when establishing the [functional requirements](http://en.wikipedia.org/wiki/Functional_requirements) of a proposed application, [prototyping](http://en.wikipedia.org/wiki/Prototyping) tools can be used to develop graphic models of application screens to assist end users to visualize how an application will look after development. Subsequently, system designers can use automated design tools to transform the prototyped functional requirements into detailed design documents. Programmers can then use automated code generators to convert the design documents into code. Automated tools can be used collectively, as mentioned, or individually. For example, prototyping tools could be used to define application requirements that get passed to design technicians who convert the requirements into detailed designs in a traditional manner using [flowcharts](http://en.wikipedia.org/wiki/Flowchart) and narrative documents, without the assistance of automated design software.[[5]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-FFIEC08-5)

Types of tools are:

* Business process engineering tools.
* Process modeling and management tool
* Project planning tools.
* Risk analysis tools
* Project management tools
* Requirement tracing tools
* Metrics management tools
* Documentation tools
* System software tools
* Quality assurance tools
* Database management tools
* Software configuration management tools
* Analysis and design tools
* Interface design and development tools
* Prototyping tools
* Programming tools
* Web development tools
* Integration and testing tools
* Static analysis tools
* Dynamic analysis tools
* Test management tools
* Client/Server testing tools
* Re-engineering tools

Existing CASE tools can be classified along 4 different dimensions:

1. Life-cycle support
2. Integration dimension
3. Construction dimension
4. Knowledge-based CASE dimension[[6]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-6)

Let us take the meaning of these dimensions along with their examples one by one:

**Life-Cycle Based CASE Tools**

This dimension classifies CASE Tools on the basis of the activities they support in the information systems life cycle. They can be classified as Upper or Lower CASE tools.

* Upper CASE Tools support strategic planning and construction of concept-level products and ignore the design aspect. They support traditional diagrammatic languages such as [ER diagrams](http://en.wikipedia.org/wiki/ER_diagram), [Data flow diagram](http://en.wikipedia.org/wiki/Data_flow_diagram), [Structure charts](http://en.wikipedia.org/wiki/Structure_chart), [Decision Trees](http://en.wikipedia.org/wiki/Decision_Tree), [Decision tables](http://en.wikipedia.org/wiki/Decision_table), etc.,,
* Lower CASE Tools concentrate on the back end activities of the software life cycle, such as physical design, debugging, construction, testing, component integration, maintenance, re engineering and reverse engineering.

**Integration dimension**

Three main CASE Integration dimensions have been proposed:[[7]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering" \l "cite_note-7)

1. CASE Framework
2. ICASE Tools
3. Integrated Project Support Environment(IPSE)

**Workbenches**

Workbenches integrate several CASE tools into one application to support specific software-process activities. Hence they achieve:

* a homogeneous and consistent interface (presentation integration).
* easy invocation of tools and tool chains (control integration).

CASE workbenches can be further classified into following 8 classes:[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering" \l "cite_note-AF_93-4)

1. Business planning and modeling
2. Analysis and design
3. User-interface development
4. Programming
5. Verification and validation
6. Maintenance and reverse engineering
7. Configuration management
8. Project management
9. Design management

**Environments**

An environment is a collection of CASE tools and workbenches that supports the software process. CASE environments are classified based on the focus/basis of integration[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering" \l "cite_note-AF_93-4)

1. Toolkits
2. Language-centered
3. Integrated
4. Fourth generation
5. Process-centered

**Toolkits**

Toolkits are loosely integrated collections of products easily extended by aggregating different tools and workbenches. Typically, the support provided by a toolkit is limited to programming, configuration management and project management. And the toolkit itself is environments extended from basic sets of [operating system](http://en.wikipedia.org/wiki/Operating_system) tools, for example, the Unix Programmer's Work Bench and the VMS VAX Set. In addition, toolkits' loose integration requires user to activate tools by explicit invocation or simple control mechanisms. The resulting files are unstructured and could be in different format, therefore the access of file from different tools may require explicit file format conversion. However, since the only constraint for adding a new component is the formats of the files, toolkits can be easily and incrementally extended.[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-AF_93-4)

**Language-centered**

The environment itself is written in the programming language for which it was developed, thus enabling users to reuse, customize and extend the environment. Integration of code in different languages is a major issue for language-centered environments. Lack of process and data integration is also a problem. The strengths of these environments include good level of presentation and control integration. Interlisp, [Smalltalk](http://en.wikipedia.org/wiki/Smalltalk), Rational, and KEE are examples of language-centered environments.[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-AF_93-4)

**Integrated**

These environments achieve presentation integration by providing uniform, consistent, and coherent tool and workbench interfaces. Data integration is achieved through the *repository* concept: they have a specialized database managing all information produced and accessed in the environment. Examples of integrated environment are the ICL [CADES](http://en.wikipedia.org/wiki/CADES) system, IBM AD/Cycle and DEC Cohesion.[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-AF_93-4)

**Fourth-generation**

Fourth-generation environments were the first integrated environments. They are sets of tools and workbenches supporting the development of a specific class of program: electronic data processing and business-oriented applications. In general, they include programming tools, simple configuration management tools, document handling facilities and, sometimes, a code generator to produce code in lower level languages. Informix 4GL, and Focus fall into this category.[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-AF_93-4)

**Process-centered**

Environments in this category focus on process integration with other integration dimensions as starting points. A process-centered environment operates by interpreting a process model created by specialized tools. They usually consist of tools handling two functions:

* Process-model execution
* Process-model production

Examples are East, Enterprise II, Process Wise, Process Weaver, and Arcadia.[[4]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-AF_93-4)

Application

All aspects of the software development life cycle can be supported by software tools, and so the use of tools from across the spectrum can, arguably, be described as CASE; from [project management software](http://en.wikipedia.org/wiki/Project_management_software) through tools for business and functional analysis, system design, code storage, [compilers](http://en.wikipedia.org/wiki/Compiler), translation tools, test software, and so on.

However, tools that are concerned with analysis and design, and with using design information to create parts (or all) of the software product, are most frequently thought of as CASE tools. CASE applied, for instance, to a database software product, might normally involve:

* Modeling business / real-world processes and data flow
* Development of data models in the form of entity-relationship diagrams
* Development of process and function descriptions

Risks and associated control

Common CASE risks and associated controls include:

* *Inadequate standardization*: Linking CASE tools from different vendors (design tool from Company X, programming tool from Company Y) may be difficult if the products do not use standardized code structures and data classifications. File formats can be converted, but usually not economically. Controls include using tools from the same vendor, or using tools based on standard protocols and insisting on demonstrated compatibility. Additionally, if organizations obtain tools for only a portion of the development process, they should consider acquiring them from a vendor that has a full line of products to ensure future compatibility if they add more tools.[[5]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-FFIEC08-5)
* *Unrealistic expectations*: Organizations often implement CASE technologies to reduce development costs. Implementing CASE strategies usually involves high start-up costs. Generally, management must be willing to accept a long-term[payback period](http://en.wikipedia.org/wiki/Payback_period). Controls include requiring senior managers to define their purpose and strategies for implementing CASE technologies.[[5]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-FFIEC08-5)
* *Slow implementation*: Implementing CASE technologies can involve a significant change from traditional development environments. Typically, organizations should not use CASE tools the first time on critical projects or projects with short deadlines because of the lengthy training process. Additionally, organizations should consider using the tools on smaller, less complex projects and gradually implementing the tools to allow more training time.[[5]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-FFIEC08-5)
* *Weak repository controls*: Failure to adequately control access to CASE repositories may result in security breaches or damage to the work documents, system designs, or code modules stored in the repository. Controls include protecting the repositories with appropriate access, version, and backup controls.[[5]](http://en.wikipedia.org/wiki/Computer-aided_software_engineering#cite_note-FFIEC08-5)