Is CASE Technology Still Alive?

Camilo Ocampo*[,1], Begoña Albizuri[†,2], Pere Botella*[,3]

**Departament de Llenguatges i Sistemes Informàtics*

Universitat Politècnica de Catalunya

Barcelona, Catalonia

[†]División Académica de Ingeniería

Instituto Tecnológico Autónomo de México (ITAM)

Mexico City, Mexico

Abstract

CASE Technology was thought as the solution of many problems of the Software Engineering. Time has shown although such technology is a condition to support Software Development Process, it is not the unique solution. In the last years, interest in the world of CASE Technology has been declining notably. This could be clearly seen in the decreasing number of research groups, magazines and conferences about the topic. Despite this fact, we think CASE Technology has not died, rather it has been applied on the automation and management of the software development phases. In this article we present how traditional CASE Technology has evolved to Process Technology, dealing with more issues like interoperation, human collaboration, project management, and so on. So CASE Technology has not disappeared, it has been principally applied on automating and managing the Software Process.

Keywords: *Software Engineering, CASE, Software Process*

1. Introduction

Probably, the term *Computer Aided/Assisted Software/System Engineering* (CASE) became popular after the publication of an article in the Wall Street Journal the 24 September 1986 [PD95], but the idea of using tools and environments to support different phases of Software Development Process[4] comes from late 1970's. For example, such kinds of early tools were compilers, or software to support

[1] Supported by CONACyT, Mexico. This work has been partially supported by the project OBJECTFLOW. E-mail: cocampo@lsi.upc.es

[2] Visiting Professor at Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya. E-mail: albizuri@lamport.rhon.itam.mx

[3] E-mail: botella@lsi.upc.es

the design and validation of data flow diagrams. Moreover, some authors [Fugg93] consider that the first CASE tools could be assemblers used in 1960's.

At the beginning of the 1990's the popularity and interest on CASE Technology[5] start decreasing notably. Some conferences on the topic changed the subject (e.g., the International Workshop on CASE has changed to become the Software Technology and Engineering Practice). Several authors ([NCR91], [GS94], and [Somm92a]) mention some of the reasons because CASE did not accomplish all user expectations. The aim of this article is to analyse the State of the Art of the CASE by the end of 1990's and mentioning our viewpoint of trends and challenges for the next years of CASE supporting the Software Process.

The structure of the article is as follows. At the Background Section we start reviewing briefly concepts and classifications about CASE. We are interested, in Section 2; in to know how this technology was before the Process Technology concept appeared. Such analysis will allow us understand the natural transition from CASE to Process Technology. This takes us, in Section 3, to see the state of art in Process Technology, i.e., modelling, assessment, improvement and management of Software Process. Finally, we expose our point of view about trends and new improvements related to CASE that should take place in next years.

2. Background

We can find several CASE definitions in the literature. McClure [McCl89] defines computer-aided Software Engineering simply as the automation of *Software Process*. A CASE definition based only into the Production-process Technology was given by Sodhi: "Computer-Aided Software Engineering (CASE) encompasses a collection of automated tools and methods that assist Software engineering in the phases of the software development life cycle" [Sodh91]. CASE definitions were changing because CASE has evolved. For example, Ghezzi et al. [GJM91], Norman et al. [NCR91], Sommerville [Somm95], and Fuggetta [Fugg93] agree that the CASE is associated with the computer-aided support offered to the entire Software Process.

The purpose of CASE is to support the different phases of Software Process development under an engineering approach by means of software. Traditionally, Software Engineering has put more attention on the product instead of the process itself. It is at the end of 1980's when some authors ([Broo87] or [Oste87]) start pointing at the necessity to be concerned about processes to achieve better quality in software products. The introduction of Process Technology could led to divide more

[4] In what follows we simply refer to Software Development Process as *Software Process*.
[5] In what follows, we use just the word *CASE* to refer to *CASE Technology*.

clearly the Software Engineering in two large areas: *Product Engineering* and *Process Engineering*. During the last decade several efforts have been done in developing a Process Technology [FKN94].

2.1 A taxonomy

Recent classifications of CASE tools of some authors, like Fuggetta [Fugg93] and Sommerville [Somm95], consider CASE in a more broad sense than earlier taxonomies were, as it can be noticed with the definitions provided in the previous section. Such CASE tool categories are:

1. **Tools**. They are the simplest types of CASE tools to support a specific activity of the Software Process, e.g., editing, programming, verification and validation, configuration management, and testing. They are used independently because they were designed with not integration assistance at all.
2. **Workbenches**. An extensive set of integrated software tools called the CASE Workbench supports specific Software Process activities. Such tools automate tasks as business planning and modelling, analysis and design, user interface development, programming, verification and validation, maintenance and reverse engineering, configuration, and project management.
3. **Environments**. They are collections of tools and workbenches to support the Software Process. Fuggetta considers five classes of environments depending on the integration degree among the component tools: toolkits, languages-centred, integrated, fourth generation, and process-centred.

2.2 CASE before Process Technology

Through the time, the idea of CASE has evolved from a product-centred point of view to a more general approach, covering both product and process support. Figure 1 shows the evolution suffered through the time by CASE.

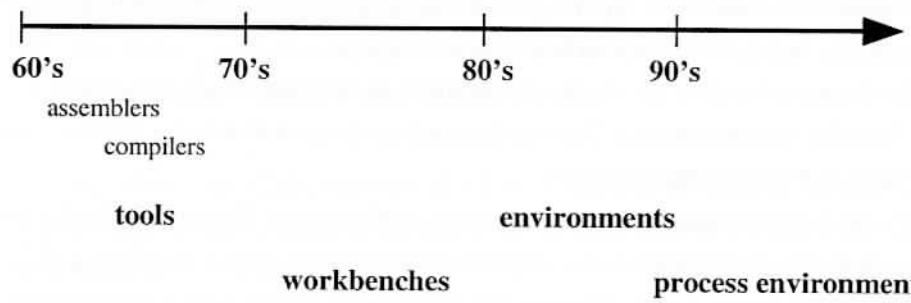

Figure 1. CASE evolution

In the early days of computing, Software Process consisted practically on writing code and no more else. Programmers had some tools, such as compilers, linkers, and loaders. After that years, the neces-

sity to produce more complex and large programs led to the development of methods and tools to support new phases of the Software Process, e.g., analysis and verification.

In the early 1970's, some of the main reasons to create integrated collections of tools, i.e., workbenches, were the increasing complexity of software, and that software practitioners realised the use of tools might have a positive effect on both productivity of people involved and the quality of products being developed. Tools might help in several activities such as, supporting checking conformance to standards, quantifying the degree of testing, supporting progress tracking, and so on.

Later, in the 1980's, the perception of software engineers was that productivity is improved with the use of automated tools, as Norman and Nunamaker reported it in their study [NN89]. Software engineers thought that, as better tools became available and, in particular, as better integration and mutual tuning of tools were achieved, significant productivity could be gained.

Environments were created to achieve better productivity and quality in the Software Process. Early environments supported all activities of the production process. Freeman [Free87] considers Software Process itself as a system. Under control of certain rules this system transforms some input (the user's requirements) into some output (an automated system). Within this development system, a number of elements work together in a reasonably harmonious way: goals, rules, procedures, people, tools and information. To make optimal use of environments, they have to fit the other system elements.

It is difficult to give precise and unambiguous environment taxonomy. Developments in this area rapidly follow each other, as is apparent from the literature from this topic. We will sketch developments in this area following taxonomy given by Dart et al. [DEFH87]. In this taxonomy four categories are distinguished, which based on trends that have a major impact on support environments:

1. Environments based on a specific programming language that contains tools specifically suited for the support of Software Process in that language.
2. Environments based on the structure of programming language contain tools aimed at manipulating program structures. These environments can be generated from a grammatical description of those program structures.
3. In toolkits we find tools that are generally not so well integrated. The support offered is independent of a specific programming language. A toolkit merely offers a set of useful building blocks. In particular, toolkits tend to contain tools that specifically support *programming-in-the-large*.

4. Finally, environments may be based on certain techniques used in specific phases of the software life cycle, while integrated tool sets aim at supporting the full spectrum of the software life cycle in a co-ordinated fashion.

Above categories encompass both environments created manually around some given programming language, and environments generated from a grammatical description of the program structures being manipulated. In both cases, the support offered mostly concerns to the individual programmer.

We cannot identify in the time when exactly CASE appeared, because that depends on what we understand for it, i.e., which CASE definition we consider. As we have mentioned, last definitions and classifications tend to consider CASE in a more general sense, i.e., it is any technology that supports the Software Process. The acronym CASE appeared at the beginning of 1980's, in fact the term CASE was coined by J. Manley in 1981 who was the head of the Software Engineering Institute at Carnergie Mellon University [Russ89]. On the other hand, according Fuggetta's classification, CASE has existed since 1960's, with the creation of first assemblers.

By 1980's, several authors reported that some benefits of using CASE are helping to reduce the time used for the development systems, minimising the maintenance cost, and increasing the quality of the systems ([Perr88] and [Orli88]). Chikofsky [Chik87] thought CASE tools allow the analyst to create the documentation and the information system model from the requirements definition to the design and the implementation. Under Suydam [Suyd87] point of view, the documentation is an automatic approach from the effort of system development. They noticed that one of the greatest CASE advantages was the introduction of engineering as a discipline in the process of systems developments.

Although, CASE has provided several benefits, there were many organisations that had problems to implement them. Some obstacles found were their high cost, the resistance manifested by the systems designers to use it, and a learning curve few acceptable [Your89].

One common problem with CASE tools was that they did not provide guidance to the user about methods to be used to create models. For example, a user of a CASE tool had to know previously the technique in which it was based.

An important point we want to notice, is that before using structured methods and CASE it was necessary to have a previous training. In a study performed by Zagorsky [Zago90] was observed that the training for tools, e.g., code generators, data dictionary, and tools for screen reports prototyping, was rare times offered in the organisations.

Another important CASE problem was its focusing on a single reduced part of the Software Process, when the ideal situation would be that it included all activities of the process in an integral way [Suyd87]. Such problem lead to significant difficulties, e.g., when users try to link the different software life cycle phases. One solution proposed by Orlikowksi [Orli88] was to have CASE tools with a shared central data repository to make easier the transition between one phase to another.

In December 1990, at the 14th International Workshop on CASE, Norman et al. [NCR91] collected 450 issues about the state of art of CASE. In this paper they point out problems concerning CASE. The most important are:

- It was not used in research on modelling the technology transfer process.
- CASE was often understood as just tools, isolating it from process, environments, methods, management, cultural changes, technology, and people.
- It did not provide adequate support for software reuse in terms of classification, selection, understanding, modification, and adaptability.

In the same way, Gilles and Smith [GS94] report a survey to quantify the usage of CASE in several organisations in UK. They concluded that the main reasons of CASE usage problems were:

- Lack of multi-user facilities.
- Poor documentation.
- Poor user interface.
- Deficiencies on the complete support of methods.

Sommer [Somm92a] also reports some reasons of the problems at companies using CASE:

- Development of prototypes with fourth generation languages was very popular in the participant organisations at the Sommer's study. CASE did not enforce such kind of code generation.
- Few of them provided an integrated set of tools and some of them generated poor quality documentation.
- They did not reduce either time nor cost of system development and they did not help with overdue work.

3. Process Technology

Because CASE problems we mentioned above, and next ones we indicate, the CASE focus starts changing at the beginning of the 1990's. Then, the new way thinking about CASE led to the Process Technology birth.

Since market pressure has revolutionised the way companies make business; organisations have been viewed from a process perspective where agents co-operate to accomplish business objectives. Software is a product which construction can also be modelled as a process. Authors, like Brooks [Broo87], Osterweill [Oste87], and Ghezzi et al. [GJM91], have noticed the importance to model and manage the Software Process to get products of better quality, finished in projected schedules and done without over budget. From this perspective, we think Software Engineering could be divided in two large areas: *product engineering* and *process engineering*, as it happened in other engineering disciplines.

The software construction is not yet well understood and therefore a complete and universal process does not exist. Hence, each organisation has to be able to define and evolve its process model according to its needs, market and customers. Software industry has to track, record and learn about their previous software project development experiences. Such learning is indispensable to achieve better quality products and project success in time and costs. CASE is an enabling technology to automate those processes.

The early software life cycle models were the first attempts to describe process models, but they have shortcomings in such goal. One problem is that those models are concerned only with the product, but they do not deal with other issues of the process as *agents* or *roles*. Another problem is the granularity of their description is too much coarse-grained, so they are not useful to provide full computer based support to the process.

3.1 Software Process

During the 1990's, important research [DWK97] has been done in modelling, assessment, improvement and management of Software Process. Results of such work have been applied to the creation of several environments and they have improved CASE [FKN94].

A process can be described from different perspectives. Curtis et al. [CKO92] have mentioned most common are *functional*, *behavioural*, *organisational*, and *informational*. Functional perspective represents how agents perform the process and which information is needed for them. Behavioural perspective describes when the agents perform their work, fulfilling a process element; and it represents

the process flow. Organisational perspective shows where and which agents are assigned to each process element. Finally, informational perspective describes items produced or manipulated by agents in each process element. These perspectives have relationships among them, and it is difficult (if not impossible) to represent them with a single graphical language.

It was possible to create process models using CASE before Process Technology emerged. Although, such models did not cover all perspectives mentioned above, i.e., they were incomplete representations of the real world. Generally, different views of the process were created without an explicit connection among them. This situation provoked that changing a model presented many difficulties and it would be almost impossible to reflect modifications of one diagram in the others affected.

Process Technology was an improvement of CASE to fully support Software Process. Conradi et al. [CFF94] define Software Process as a partially ordered network of interacting activities carried out by human agents supported by tools, aiming at producing a software product.

Two sub-processes compose software Process: *Production Process and Metaprocess* [Conr92]. Production Process is related to all activities, methods, tools, and organisational structures; used in phases of analysis, design, implementation and delivering of software products. Metaprocess is the set of activities that have to be performed to carry out the production.

CASE was only focused on the Production Process without considering Metaprocess, since efforts in Software Engineering were also interested on the product, instead of improving the way software was developed.

Besides providing comprehension about Software Process, more important, Software Process models can be *enacted* in environments. *Process-Centred Software Engineering Environments* (PSEE) [FKN94] are such computer systems. PSEE support processes, enacting instances of a pre-defined process model, based in the modelled perspectives and the state of relevant parts of such process.

The enacted process model is active since its state changes to reflect modifications in the actual process being enacted. By the other hand, process evolves while is performed by participants in real world and PSEE should manage this evolution.

As we have mentioned before, Software Process models created by mean of CASE had shortcomings to reflect changes from a model view (e.g., a dataflow diagram) to other ones. Besides such Process Modelling issues, when traditional CASE tools have been used to support all the software develop-

ment activities (i.e., at the Software Process enactment), weakness also exist due the loosely integration of such tools. Productivity was not increased as it was expected because propagating a change from a process step to other ones was a tough matter, e.g., a change in a design item was hardly reflected to its corresponding pieces of code.

Earliest PSEE where constructed with the idea of containing in a single environment all the tools used in the Software Process, e.g., editors, debuggers, schedulers, and so on. This idea presents a significant problem: the complexity of those environments was tremendous and they finally could not to deal with all the required components. The later is because the new generation of PSEEs are being constructed like several autonomous components interoperating among them, and sharing information among them.

3.2 Process Technology Trends

It is necessary to do interdisciplinary research to face all problems around Process Technology, as it has been pointed by Sheth et al. [SGJ+96]. CASE experience could be applied to areas like simulation, prototyping, or monitoring. Such areas are no totally addressed in existent PSEE.

Besides Software Process, Process Modelling and Automation, has been treated by other disciplines like Workflow Management (WM) [SGJ+96] and Transaction Management [WS97]. Similar research work has been done in all of these areas; e.g., Process Modelling Languages are present in environments to automate processes. Workflow Management Systems (WMS) are used to model and automate Business Processes, as well as PSEEs are used in Software Process. Unlike PSEEs, WMS have existed as commercial environments since the late 1980's. Nowadays there are more than two hundred of such software environments. They have a lot of shortcomings, but also they have generated a lot of experience and successful applications. For these reasons, we think Workflow Technology could be used to improve Software Process Technology and vice versa.

Recent developments (e.g., the widespread use of the Internet Technology) have increased the pressure on software engineers to deliver their products faster. This requirement lead to larger development teams that often will be globally distributed. The distributed development of software in (virtual) enterprises is a great challenge for project management, and requires new techniques for project co-ordination, document management and communication.

The number of tools involved in Software Process is enormous. It is quite difficult for one vendor to produce an environment providing all these tools; so, it is more natural to think in different tools in-

teroperating in a common framework. Such frameworks and standards of interoperability are necessary to facilitate the interoperation of tools from several vendors.

4. Conclusions

At the beginning, Software Process activities were not well defined and therefore tools that supported them were simpler. The increasing detail in the definition of activities produced that only one tool became insufficient to support them, and set of tools were constructed, and later integrated.

More attention and research have been put on process modelling than other areas of Software Process. We consider this area has to deal more than only process modelling, such as simulation, enactment, deviation, and so on.

Main motivations to create CASE was the necessity to increase the productivity of large systems development, to master the inherent complexity of software, to integrate tools and activities in common frameworks easier to manage and administrate. We think CASE Technology has been applied principally in Process Modelling, but could be also applied to other areas of Software Process. Hence, the advances achieved over the last years in CASE Technology has not been lost. We have shown in this paper the way in which such advances have been applied to different phases of the Software Process, and how they are the basis for a new generation of tools.

We want to mention that introducing new technology (like CASE) in the organisation it does not ensure successful projects or better performance achieving processes. It is necessary to take into account another issues, like having an *Information Technology Strategy*. Such strategy provides a way to analyse and to plan the introduction of new technology and it prepares the organisation for future business changes. More information about specific products and different kinds of CASE software can be found, for instance, at *http://www.qucis.queensu.ca/Software-Engineering/tools.html*.

Ideas expressed in this paper are result of the experience acquired by the authors from the OBJECTFLOW project [OB97], which consisted in the development of a Workflow Management Tool. Such kind of software is used to automate and manage Business Processes, another kind of Process Technology. Also, experience has been acquired from the research in other issues of Software Process [FR97].

5. Acknowledgements

We want to thank Mario Piattini, from the University of Castilla-La Mancha, and Wilhelm Schäefer, from the University of Paderborn, for their constructive and invaluable comments on a draft version of this paper.

6. References

[Broo87] F. Brooks. *No Silver Bullets: essence and accidents of Software engineering*, IEEE Computer Magazines, pp. 10-19, 1987.

[CFF94] R. Conradi, C. Fernström, and A. Fuggetta. *Concepts for Evolving Software Processes*, In [FKN94], Chapter 2.

[Chik87] E. J. Chikofsky. *Reliability Engineering for Information Systems: The Emerging CASE* Technology, Index Technology Corporation, Cambridge, Massachusetts, 1987.

[CKO92] B. Curtis, M. I. Kellner, and J. Over. *Process Modeling*, Communications of the ACM. 35(9): 75-90. September 1992.

[Conr92] R. Conradi et al. *Towards a Reference Framework of Process Concepts*, In Proceedings of Second European Workshop on Software Process Technology. Springer Verlag, Berlin, 1992.

[DEFH87] S. A. Dart, R. J. Ellison, P. H. Feiller, and A. N. Habermann. *Software development environments*, IEEE Computer 20, 11, 1987, pp. 18-28.

[DWK97] J. C. Derniane, B. Warboys, and A. B. Kaba. *Software Process: Principles, Methodology*, Technology. Provisional version, PROMOTER Group, to appear, 1997.

[FKN94] A. Finkelstein, J. Kramer, and B. Nuseibeh, editors. *Software Process Modelling and Technology*, Research Studies Press Ltd., Taunton, Somerset, UK, 1994.

[Free87] P. Freeman. *Software Perspectives*, Addison-Wesley, 1987.

[FR97] X. Franch, and J. M. Ribó. *Software Process Modelling as Relationships between Tasks*. In Proceedings of the 23rd. Euromicro, Budapest, September 1997.

[Fugg93] A. Fuggeta. *A Classification of CASE Technology*, IEEE Computer, December, 1993.

[GJM91] C. Ghezzi, M. Jazayeri, and D. Mandriolo. *Fundamentals of Software Engineering*, Prentice-Hall, Englewood Cliffs, N. J., 1991.

[GS94] A. C. Gillies, and P. Smith. *Managing Software Engineering. Case studies and solutions*, Chapman & Hall, London, 1994.

[McCl89] C. McClure. *CASE is Software Automation*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[NCR91] R. J. Norman, E. J. Chikosfky, and B. L. Rubenstein. *CASE at the Start of the 1990's*, Proceedings of the 14th International Conference on Software Engineering, IEEE, pp. 128-139, Austin, Texas, May 13-17, 1991.

[NN89] R. J. Norman, and J. F. Nunamaker, Jr. *CASE productivity perceptions of Software engineering professionals*, Communications of the ACM, 32, 9, pp. 1102-1108, 1989.

[OB97] C. Ocampo and P. Botella. OBJECTFLOW: A Modular Workflow Management System. Internal Report LSI-97-5-R, Department of Software, Technical University of Catalonia, January 1997.

[Orli88] W. J. Orlikowksi. *CASE Tools and the IS Workplace*, In Proceedings of the 1998 ACM SIGCPR Conference on the Management of Information Systems Personnel, pp. 88-97, College Park, MD, April 7-8, 1988.

[Oste87] L. J. Osterweil. *Software processes are Software too*. In Proceedings of the Ninth International Conference on Software Engineering. IEEE Computer Society, pp. 2-13, Washington, DC, 1987.

[PD95] M. G. Piattini and S. N. Daryanani, editors. *Elementos y herramientas en el desarrollo de sistemas de información: una visión actual de la tecnología CASE* (in Spanish), RA-MA editorial, Madrid, Spain, January, 1995.

[Perr88] G. Perrone. *Primary Product in the Development Life Cycle*, Software Magazine, pp. 35-41, 1988.

[Russ89] F. Russell. *The Case of CASE*. ICL Technical Journal, Vol. 6, issue 3, May, pp. 479-495, 1989.

[SGJ+96] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf. *Report from the NSF Workshop on Workflow and Process Automation in Information Systems*. Computer Science Department Technical Report, UGA-CS-TR-96-003, University of Georgia, October 1996.

[Sodh91] J. Sodhi. *Software Engineering: Methods, Management, and CASE Tools*, McGraw-Hill, Blue Ridge Summit, Pa., 1991.

[Somm92a] M. Sommer. *The Impact of Computer-Assisted Engineering on Systems Development.* IFIP Transactions, K. E. Kendall et al., editors, p.p. 43-60, 1992.

[Somm95] I. Sommerville. *Software Engineering*, Addison-Wesley, Reading, Mass., November, 1995.

[Suyd87] W. Suydam. *CASE Makes Strides Toward Automated Software Development*, Computer Design, 1987.

[WS97] D. Worah and A. Sheth. *Transactions in Transactional Workflows*. In S. Jajodia and L. Kerschberg, editors, Advanced Transaction Models and Architectures. Kluwer Academic Publishers, 1997 (to appear).
Available in *http://lsdis.cs.uga.edu/publications/pub_ALL.html*

[Your89] E. Yourdon. *Serious CASE in the 90's: What Do We Do When the Novelty Wears Off?*, Show CASE Conference IV, 1989.

[Zago90] C. Zagorsky. *CASE Study: Managing the Change to CASE*, Journal of Information Systems Management, 1990.