AutoCap: An Automatic Tool Encapsulator[1]

João Câmara *, Helena Sarmento **

R. Alves Redol, 9, 1000 LISBOA, Portugal

Phone +351 1 3100353 Fax: + 351 1525843 email jac@olivia.inesc.pt

* INESC ** IST / INESC

1. Abstract

CAD frameworks often have to be updated by incorporating new CAD tools. Encapsulation is the tools' incorporating technicality most often used. Once tools have been incorporated into a framework, the designer access to them may be unified and simplified.

The present work describes work recently developed, within the PACE framework, to address tool encapsulation and tool invocation.

2. Introduction

The dynamic nature of CAD systems implies the ability to easily modify them. These modifications often include adding new tools to the framework (or removing outdated ones).

Tools can be incorporated into a CAD system in two ways: integration and encapsulation. Integrating a tool means tightly connecting it to a CAD system by writing or modifying the tool to use framework services directly to get data and other information it needs to be executed. Encapsulation means loosely interfacing a tool to a CAD system trough a layer separated from the tool. Integrating a tool requires the tool's source code to be available and usually represents a substantial investment of time and expertise. On the other hand, encapsulating a tool is usually less time consuming and no modification of the tool's source code is required. This is

[1] Part of this work was done under ESPRIT project SPRITE.

why, when incorporating tools into a framework, encapsulation is more often used.

Nowadays CAD systems need to incorporate a large number of different tools, each one with its own particular, often unique, requirements for proper execution. The need to remember the correct tool location, command syntax and arguments necessary to invoke each tool often leads to errors and decreased productivity. The development of CAD frameworks, which are software infrastructures that provide a common operating environment for CAD tools, is very important since they offer a number of services that ease the task of managing and updating CAD systems. Due to the dynamic nature of CAD systems, the easiness to incorporate a tool into a framework is an important factor to be considered when evaluating a framework.

Once a tool has been incorporated into a framework, tool management, one of the services offered by CAD frameworks, frees the designer from the need to remember the correct location, command syntax and arguments necessary to invoke each tool.

This paper describes recent work developed within the PACE framework [Sarmento 90] to address two related problems: easiness of incorporating a tool into a CAD framework and easiness of invoking tools from a CAD framework. Section 3 refers related work developed within several CAD frameworks. Some aspects of tool encapsulation are mentioned in section 4. Section 5 briefly describes TES, the CFI standard for tool encapsulation information. Sections 6 and 7 describe AutoCap. Finally, some conclusions are presented in section 8.

3. Related work

The problem of incorporating tools into a framework has been addressed by different frameworks in different manners.

Both ULYSSES [Bushnell 85] and CADWELD [Daniell 89] frameworks are based on a *blackboard architecture*. The *blackboard* is global database containing references to all the files related with the tools incorporated in the system. In the ULYSSES framework the characterisation of a tool is part of the blackboard. In CADWELD, tool characterisation is obtained by building a software layer that allows the tool to interact with the blackboard. Both CADWELD and ULYSSES implement a name server as part of the blackboard, thus allowing an easier invocation of the tools.

Within FALCON [Mentor 90] framework tool characterisation is done by the association of a *qualification script* to each tool. This framework includes a *Design Manager* that permits the designer to easily invoke a tool with a graphical interface quite similar to the one used in Apple Macintosh [Apple 91] computers.

CADLAB [TIDL 90] framework offers an object oriented language, TIDL[2], to describe the design environment. The tool integrator uses this language to define the objects needed to encapsulate the tool. TIDL automatically generates de dialogue boxes that allow the designer to enter the tool's parameter list.

Within DESIGN FRAMEWORK II [DFII], the tool integrator writes a program, using the SKILL language, to wrap the tool thus permitting the management of the tool within the framework. Once encapsulated, tools are invoked by invoking the encapsulation function, which can be associated to the call back function of a menu item.

[2] Tool Integration and Description Language

4. Tool Encapsulation

As referred, the evolution of a CAD system often requires adding tools to the framework. The existence of tool encapsulation mechanisms is therefore of great importance.

Tool encapsulating requires both a tool description language to characterise the tool and a mechanism capable of interpreting a tool description. Although, theoretically, any description language may be used, the language should obviously be concise and have an adequate expressive power. It is also advantageous that the language has a human and computer readable form, since this allows the information contained in a tool description to be used in either a manual or automatic creation of a tool encapsulation.

Aiming at the implementation of a standard for a format for tool encapsulation information, CFI has developed TES (Tool Encapsulation Specification) [CFI 92], which we briefly describe in the next section.

5. TES

The current version of TES includes general tool information, description of all the tool arguments, description of all data associated with the tool function, description of the syntax for a valid command line to initiate the tool function and description of possible result codes. TES also includes a mechanism that permits to add arbitrary name/value pairs to a description construct, thus providing with an extension capability the current version of TES.

We have adopted TES as our description language and developed AutoCap (Automatic Encapsulator), a tool that interprets the information contained in a TES description and uses it to automate the encapsulation of the tool described.

6. AutoCap

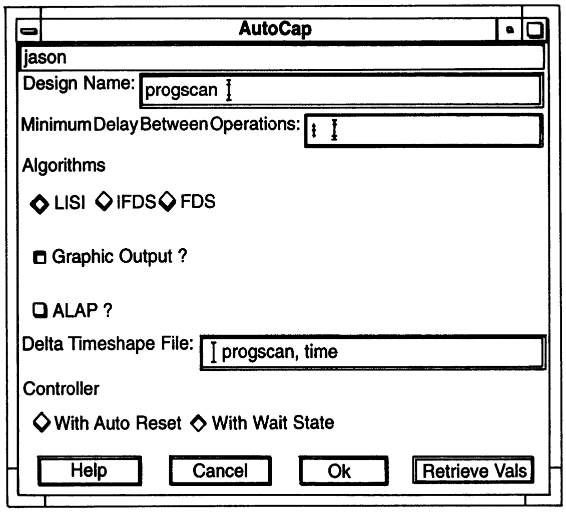
AutoCap reads a TES file and maps it to an internal representation structure. This structure is complex enough to deal with all types of information needed and general enough to be usable in the construction of any type of user interface (graphical, textual or other). The internal representation structure has to be of some complexity since AutoCap needs to represent all the information contained in the TES file internally. Alternatively AutoCap could deal separately with each information construct contained in the TES file, thus allowing the use of a simpler internal structure; however this approach would not allow the global treatment of the information, necessary in the construction of a graphical interface.

The information contained in a TES file may be used in two ways:

- read the TES file once, at the time the tool is registered with the framework, extracting its information and storing it in persistent objects which are then activated by the framework when the designer invokes the tool;
- read the TES file each time the tool in invoked using an interpreter-like program to extract the information and use it to invoke the tool directly.

AutoCap uses the second approach since it is simpler and since we have verified that the TES file interpretation may be done quite quickly. This approach also permits, when the framework invokes AutoCap, to pass to AutoCap parameters that are only defined at run time.

AutoCap automatically builds a user interface which is a graphical representation of the tool arguments described in the TES file. The designer is presented with a user friendly interface, where the values required are labelled and may be of different types: integer, real, string, multiple choice[3]. The required values may be entered either by pushing buttons or by typing then onto entry fields. Figure 1 shows an example of an AutoCap interface window.


AutoCap

jason

Design Name: progscan

Minimum Delay Between Operations:

Algorithms

◇ LISI ◇ IFDS ◇ FDS

☐ Graphic Output ?

☐ ALAP ?

Delta Timeshape File: progscan, time

Controller

◇ With Auto Reset ◇ With Wait State

Figure 1 - AutoCap interface window

The arguments the designer is required to enter do not necessarily correspond to the arguments used in the command line to initiate the tool function: some of them may be used to somehow evaluate other values that, in turn, are used in the command line. Similarly the values inputted by the user for some arguments may correspond to different values in the command line: for instance the value *true* of a boolean argument may correspond to the value *-g* in the command line.

Ranges of acceptable values for an argument may be specified in a TES file. AutoCap checks if entered values comply the restrictions specified in the TES file, if any. In case of an error, a *popup window* is used to notify the designer.

[3]Multiple choice arguments are represented either by radio buttons (exclusive multiple choice) or by toggle buttons (inclusive multiple choice).

A TES file may contain help lines. These lines are used to supply additional information to the user concerning the enclosing description constructs. On demand, AutoCap opens a window (see figure 2) presenting this information to the user.

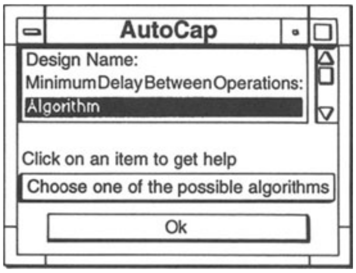

AutoCap
Design Name:
MinimumDelayBetweenOperations:
Algorithm
Click on an item to get help
Choose one of the possible algorithms
Ok

Figure 2 - Help window

When invoking a tool the designer often enters for each argument the same value he did last time the tool was invoked. This led to the existence of the *Retrieve Values* button (see figure 1) in the AutoCap interface window. By pressing this button the designer is able to retrieve the argument values he entered last time the tool was invoked.

7. Implementation

AutoCap was developed using C++[4], and it implements a set of classes that allow the storage of the tool encapsulation information. These classes are associated with the different entities the information refers to, such as tool, argument, argument list, value, concatenated value and others. Figure 3 depicts the class hierarchy, using the notation proposed in [Rumbaugh 91].

AutoCap graphical interface was built using the services of GHOST [Santos 90], a graphical server that greatly simplifies the programming effort necessary to implement the interface.

[4] Approximately 6000 lines of code were written.

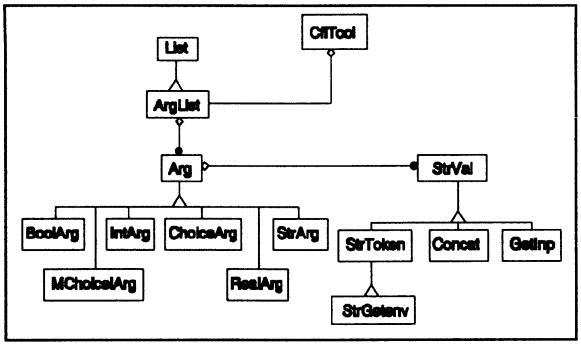


Figure 3 - AutoCap classes hierarchy

To deal with a TES file AutoCap uses the tools LEX and YACC. The use of these tools showed to be extremely useful since TES allows great flexibility in the order and number of elements that constitute each of its description constructs.

8. Conclusions

AutoCap is presently being used by DESKTOP [Martins 93], a part of the PACE framework, to invoke the encapsulated tools.

AutoCap was used to encapsulate several tools, such as ARCHITECT (a schematic diagrams visualisation tool developed at INESC) and JASON (a scheduler tool of the PHIDEO compiler developed at PHILIPS Natlab). Although there were no TES files available for these tools, these files where very easily written by the tool integrator, once he was told about the tool arguments.

AutoCap greatly simplifies the work of a tool integrator, since it only requires the existence of a TES file describing the tool. This file is supposed to be supplied by the tool vendor, so all the information the tool integrator has to add is the tool location. So, unlike the methods used by other frameworks referred in section 3, the tool integrator's work is reduced to an absolute minimum. By using a common user friendly interface, AutoCap also simplifies the designers work as far as tool invocation is concerned.

The development of AutoCap was greatly simplified by the use of GHOST, another service provided by the PACE framework. On the other hand, the existence of AutoCap was very useful in the development of BALANCE, an automatic dynamic load balancer recently offered by the PACE framework. By capturing information about all data associated with the tool function and possible result codes, AutoCap may also be very useful for flow management.

References

[Bushnell 85] M. L. Bushnell, S. W. Director, *ULYSSES - An Expert-System based VLSI Design Environment*, International Symposium on Circuits and Systems, June 1985.

[CFI 92 - a] TCC Approved Draft Proposal - CFI, *Toll Encapsulation Specification*, CAD Framework Initiative pilot release document CFI-92-P-10

[CFI 92 - b] CAD Framework Initiative - CFI, *Standards Release*, Notifications of CFI 2.0 Pilot Program, 1992.

[Daniell 89] J. Daniell and S. Director, *An Object oriented approach to CAD Tool Control Within a Design Framework*, 26th ACM/IEEE Design Automation Conference, 1989

[Martins 93] J. F. Martins, Final Prototype of a Desktop, technical report, INESC, SPRITE Esprit project, November 1993.

[Mentor 90] Mentor Graphics, *The Falcon Framework Technical Papers,* June 1990

[Rumbaugh 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, *Object-oriented Modelling and Design,*. Prentice-Hall International Editions, 1991.

[Santos 90] P. Santos, H. Sarmento and L. Vidigal, *Ghost/Spook User Interface and Process Management in the PACE Framework,* European Design Automation Conference, March 1990.

[Sarmento 90] H. Sarmento and P. Santos, *A Framework for Electronic Design Automation,* IFIP Workshop on Electronic Design Automation Frameworks, North-Holland, November 1990.

[TIDL 90] Cadlab, CADLAB Tool Integration Description Language Release 3.0 TIDL - Overview, 1990