This item was submitted to [Loughborough's Research Repository](#) by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

CAE software in manufacturing engineering education

PLEASE CITE THE PUBLISHED VERSION

[http://dx.doi.org/10.1016/0360-1315\(90\)90158-4](http://dx.doi.org/10.1016/0360-1315(90)90158-4)

PUBLISHER

© Pergamon Press (now Elsevier)

VERSION

AM (Accepted Manuscript)

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Doyle, Robert, and Keith Case. 2019. "CAE Software in Manufacturing Engineering Education". figshare.
<https://hdl.handle.net/2134/14211>.

This item was submitted to Loughborough's Institutional Repository (<https://dspace.lboro.ac.uk/>) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

CAE SOFTWARE IN MANUFACTURING ENGINEERING EDUCATION

R. DOYLE and K.
CASE

Department of Manufacturing Engineering, Loughborough University of Technology, Loughborough,
Leicestershire LE11 3TU, England

Abstract—This paper examines the role of commercial computer-aided engineering (CAE) software in manufacturing engineering education. The proliferation of computing in industry demands that a significant proportion of students' timetables be devoted to CAE aspects, in order that they may gain an appreciation of the capabilities and limitations of such systems. However, it is also important to present this computing element so that it does not overwhelm other, equally important parts of the manufacturing engineering syllabus. The aim must be to integrate CAE courses into the curriculum, make real academic demands of students and to ensure a good understanding of CAE.

INTRODUCTION

Computer-aided engineering (CAE) has two roles to play, in manufacturing engineering education. The first is as a demonstrator of software that is commonly used in industry. The second is to illustrate and supplement engineering principles taught in the classroom.

Fundamentally, CAE is about the application of engineering principles, and as such it is important that students experience it at first hand rather than by demonstration. Many new graduates, who gain employment with companies using CAE systems, find themselves playing a major role in the use and development of CAE systems within their company. In terms of developing professional engineers it is important to establish an understanding of the potential and limitations of CAE systems in real life situations. At the same time it is also important to remember that an aim of manufacturing engineering courses is to educate students in CAE rather than produce graduates who are fully trained on specific CAE packages.

The second parallel role which CAE can play in manufacturing engineering is as a computer-aided learning (CAL) tool in two ways. Firstly, by illustrating the engineering principles which the software embodies, and secondly by offering an environment to produce CAL programs. There is some evidence of emerging CAE systems which are primarily designed for training and education (e.g. the TX system, BYG, 1989), and efforts have been made to understand which types of interface facilitate learning [1]. However, it is more usual to use existing CAE packages which offer a parametric programming facility to control the software which can also be used as a CAL programming tool. This approach is particularly useful where a powerful graphic capability is required. Generally, 2- and 3-dimensional design packages are more useful in this type of application. This approach also has the benefit of increasing the usage of the software making it more cost effective.

CHOICE OF SOFTWARE

The total range of CAE software currently available is large, which makes identifying a particular piece of software a difficult task. However, taking into account the limiting factors often reduces the list of potential software to a manageable number. The first factor must be to decide what functions are required of the software. CAE packages range from generalized design or analysis packages to very narrow programs which solve specific problems. Obviously generalized software offers greater scope in application, but, this does imply that more work will be required to present the software. The next major factor is the cost of the software. Good quality commercially produced software tends to be expensive since the companies investing in the software expect a return from the investment. The result is that the

cost of the software tends to be in proportion to the benefit gained. However, some software companies have realized the worth of advertising their software to students who will eventually have a significant influence on software purchase in industry after they have graduated. As a consequence there are currently some very good educational deals on software. The situation is further improved by the activities of bodies such as CHEST [2] who negotiate special educational discounts on a range of software. Other factors which appear to influence the choice of software are; personal preference, comparability with current courses and course goals, type and availability of equipment, and the degree of input required to make the software useful. Many manufacturing engineering departments already have a reasonable range of CAE software on site which is often under-utilized through a lack of development. In this situation it is worth considering applying resources to the development of the existing software, provided it has good potential, rather than to the purchase of new software.

In summary, the benefits of using commercial CAE software are as follows; exposure of students to industrial standard software, high standards of programming and support are associated with the software, a good base for student practical work and project work, case studies can be adapted for a range of courses, and, a potentially good environment for producing CAL programs.

PITFALLS IN THE APPLICATION OF THE SOFTWARE

Whilst the application of CAE software in teaching has many merits there are some significant barriers to overcome before the goal of using CAE software as an effective teaching tool can be achieved. Following are some of the main hurdles.

Resources

Resources are always a problem and there are specific resource demands which relate to the use of CAE software, above any initial capital expenditure. The issue of software and hardware maintenance cannot be avoided. Often it is this aspect which presents the main financial burden rather than the initial once-off capital investment. Hardware maintenance is required as hardware failures are not uncommon and repairs in house are not usually possible. As much software and the associated software licenses are linked directly to the hardware a software maintenance agreement is necessary. The availability of facilities is an important issue with the aim being to maximize the number of points of access to the facilities whilst maintaining a reasonable working speed for the users. For a given facility the demand, in terms of work required of the students, must match that facility. Where many different groups are using the facility, monitoring is required with appropriate responses should the system become overloaded. A method of increasing facilities, without an increase in the number of workstations, is to look at the facility opening hours. Many facilities are only open during normal office hours when students are committed to lectures and laboratory work. Extending the opening hours does imply an increase in running costs; mainly for security, heating, lighting etc. However, in terms of gaining effective use from the often large capital investment in hardware and software, the additional costs for extended hours are justified. Human resources are equally as important as any other resource. There must be a direct allocation of staff, in proportion to the volume of equipment in use, who are able to install, run, maintain and develop the equipment and software. Otherwise, relying on existing staff, the facility can only ever achieve a fraction of its full potential.

Customization

Many CAE software packages are designed with the intention that they should be customized. It is, therefore, rarely possible to use CAE packages effectively without some additional input. Systems often require data bases and macro programs to be produced according to the tasks that are to be executed and the particular application. An example of this would be a start-up macro which sets the initial conditions and default values for the software. This aspect is particularly important in allowing students to spend a significant proportion of their time dealing with the main issues at hand rather than with the functionality of the system. Often the documentation provided with software is not of a suitable format for student use. Reference manuals are too large and training manuals are too long for the time available. Also training manuals tend to be structured so that they deal with each function of the package with equal emphasis. Whereas the educational requirement is to demonstrate, in a short space of time, the fundamental features

of the CAE software. The solution to this is to produce in-house documentation which introduces the software, provides a structured path through the software and acts as a reference to the software.

STUDENT CATEGORIES

The above points illustrate that for CAE to be properly integrated into courses a considerable amount of effort is required coupled with matching resources. However, the benefits gained are commensurate with the input. The aim must be to align the CAE software with the aims of the course. This can be achieved by minimizing the problems associated with CAE software and identifying the areas of greatest usage for the effort involved. Clearly it is important here to identify groups of students and the nature of the coursework that is required. Three broad categories have been identified as suitable for the presentation of CAE software and are listed as follows:

Hands on demonstrations-category 1

Short, or even long demonstrations where students simply watch a computer terminal in normal use are not usually effective because information appears relatively slowly on screen and the view of the screen is often poor. Where a brief illustration of CAE software is required the best approach is to show quick fire snapshots of the end graphic output. This can be achieved through screen capture or by making use of a video camera. Where more time is available, between 1 and 3 h, the most effective method is to make the desired demonstration interactive. This requires a well-defined case study which takes the student through the demonstration in a tightly controlled manner via either handouts or parametric programs controlling the CAE software or a combination of both.

Introductory courses-category 2

These are fairly intensive courses which introduce students to a specific CAE package with approx. 8-10 h of tutored time associated with a similar amount of the students' own time. Generally it is expected that only one of these courses would be included in any one academic year. The courses aim to highlight the principal features of the software and are based on a handbook specifically written for the course. The handbook serves several purposes; it structures the course and allows each student to work at their own pace, acts as a quick reference and minimizes student demand on staff time. The handbook does take a deal of effort to produce but the standards obtained from its use are significantly higher when compared with other methods.

Long projects-category 3

Long CAE projects can potentially be very good but they do need setting up properly and also close monitoring is required during the project. Some of the main pitfalls are as follows: students tend to think they are working hard whilst they are sat at a computer terminal. However, the part of the project which contains the academic merit is more often than not carried out away from the terminal. It is important to emphasize that there are two phases to this type of project: learning to use the software followed by applying it to the problem. Most credit must be given to the latter. It also follows that adequate support must be given to prevent problems in operating the software from obscuring the goals of the project. Where software is being written it is important to stress the importance of achieving the goals of the project rather than producing a finished product. There should be an element in the marking scheme allocated to a presentation of the project on screen.

STRATEGY FOR APPLICATION OF THE SOFTWARE

An important step at an early stage is to define a specification of the aims of a particular course and also plan to meet the specification. The specification should be regularly reviewed as the course develops and also after the course has been used. The following is a description of a typical route which might be followed during the evolution of a new course.

Firstly, the initial parameters are identified such as: the level of the course which is to be presented, the time available, the CAE software to be used, the resources available, etc. From the parameters a

specification of the course is produced which forms a plan to meet the set goals. A case study is then identified which meets the specification. In order that a good case study can be identified, usually an evaluation of the application of several case studies is carried out. This tends to be a time consuming exercise, however, the case study is often the key element on which the success or failure of the course depends. Having selected a case study it is then recorded in a reference document which fully describes the case study and how it is applied. The next stage is to write and program any required software, and also to produce an initial draft version of any required documentation. On completion an evaluation phase is carried out which determines whether the implementation meets the specification. The evaluation also identifies any bugs in the software and documentation. This is an essential stage since resolving problems caused by bugs, at a later stage, for large numbers of students can be very time consuming. When all the material is satisfactory the final document is produced and software documented. The course is then released to the specified group. Formal and informal feedback is encouraged and logged to facilitate the removal of any errors at a later date. Updating the material is usually carried out in parallel with updates to the CAE software on which the material is based.

THE HANDBOOK

An essential element for the effective presentation of category 2 style courses is the handbook. Figure 1 shows the overall structure of the handbook, describing a path through the document. The entire course is broken down into a series of “sessions” each of which contains a proportion of the concepts relating to the software. Each session is further broken down into elements each of which describes how a particular facility can be used and the relation between the facility and the software. Where possible, specific examples are given which the student must repeat using the software. At the end of each session the student goes to the long exercise which re-iterates the examples shown in the previous sessions. At the end of the relevant portion the student returns to the next session, and so on. Finally, the students must carry out some assessed work which again builds on the examples shown and also stretches the students’ understanding of the software.

The style of the document is equally as important as the content. The objective is to ensure the student is able to find his way around the handbook, absorb its contents and also use it as a reference guide. For the handbook to be successful it has to be written from the users’ point of view rather than the writer’s view. This approach does produce significant differences in presentation. The key problem for the users is having to deal with two media; the computer screen and the handbook. Since the main objective is based on the computer, student use of a handbook tends towards the minimal in order to cope with the difficulties of alternating between two media. This limited capacity in dealing with a number of incoming signals (from the screen and the handbook) has long been recognized as a factor in overall performance [3]. Card *et al.* [4] review the literature but it is clear that the handbook must be based on short clear statements describing the function being presented followed closely by the command to initiate the function. All other text in the document, whether intended or not, acts as an optional explanation system to the illustrated commands. Generally text in manuals is only read as a last resort, so the likelihood of the text being read is inversely proportional to the length of the text. It is suggested that three forms of text should exist within the handbook as follows:

- (1) Short direct comments immediately following the command.
- (2) Full descriptions, including diagrams, explaining the expected output from the expected input. This must appear just before or just after a block of commands. A high number of diagrams need to be included to illustrate the software (graphic software requires graphic manuals - a notion supported by the evidence of performance using iconic representation for commands [5]).
- (3) Where appropriate, appendices should be provided which describe the concepts and theory behind the functions being presented.

Fig. 1. Structure of a handbook.

Care must also be taken not to repeat any existing documentation which is provided with the software. The aim is to identify the pertinent points and condense them into a form which can be quickly absorbed.

Another important factor is the presentation of the handbook. The use of a range of text fonts, styles and sizes makes documents clear and easy to read. The readability of manuals can be greatly improved by the inclusion of symbols in the text. The purpose of the symbols is to graphically identify the function of each block of text. For example;

- would equate to 'this is a command'
- would equate to 'this is a comment'
- ▲ 2.1 would equate to 'refer to section 2.1'

Also as the user has to constantly move his attention between the screen and the handbook this type of symbolism is vital in allowing the user to identify and maintain his position on the page.

Inserting and compiling all of these different elements in a handbook is a complex task using conventional text processing facilities. The workload can be reduced considerably by making use of

desktop publishing package which allows easy interactive composition and editing of documents on screen.

Without doubt the most powerful teaching tool in the presentation of CAE is the “handbook”. The fundamental concept behind the handbook is to take the student beyond the usual “sitting by Nelly approach”. The structure of the handbook encourages the user to retain a core understanding of the software. Observation of other courses using poorly structured documentation showed, that whilst students were able to follow a series of examples, they frequently showed little or no retention of the examples when they were asked to apply them. The handbook is able to counteract this by first illustrating a series of related examples and then asking the student to carry out a part of a long exercise which runs consistently throughout the whole handbook. The long exercise works by illustrating a series of “snapshots” through the exercise associated with a list of the commands which should be used to reach that point. If a student has not understood the previous examples he will be forced to review them in order to complete the long exercise.

SOME EXAMPLE COURSES

So far this paper has described a generalized approach to producing courses, however, it has to be remembered that every CAE package is different and that the proposed audiences cover a wide range. As such there is no clearly definable format for the evolution of these type of courses and therefore a pragmatic approach is required for their generation. As an illustration of the format which results from the application of the concepts described in this paper three examples follow of courses which have been developed in the Department of Manufacturing Engineering at Loughborough University of Technology:

The first course called Computer Integrated Manufacture is a category 1 style course, as described earlier, and is a part of the 1st year Engineering Applications course. The content has also formed the basis for the practical work in a final year Computer-Aided Design option. The course aims to present the generation and transfer of engineering information across a wide range of systems as follows; design through solid modelling, detailed design through 2-D drafting, Computer Numerical Control (CNC) programming, graphic NC verification and finally CNC machining.

The second course is called Introduction to DOGS, which is a category 2 style course, and is designed to be integrated into a 1st year course called Manufacturing Design. This course, through a handbook, introduces students to 2-D Computer-Aided Drafting (CAD) using the Pafec DOGS 2-D system [6]. The aim of the course is to make students self-sufficient in the use of the package and test their ability in the interpretation of engineering drawings.

The final example is of a course called Introduction to DUCT which is a category 2 style course and is a part of a 2nd year course called Engineering Computing. The course takes the students through various elements of a 3-D surface modelling package from Deltacam Systems called DUCT [7]. The aim is to show a design to manufacture approach where the data generated to describe the surface is also directly used to produce machining paths.

COMPUTER INTEGRATED MANUFACTURE

Figure 2 shows the form of overall concept which is presented in the course [8]. Essentially, data produced at the design stage is transferred through, and manipulated by, a series of CAE software packages to a point of manufacture. In comparison to the “real” industrial situation this concept is greatly simplified. There are two reasons for this; firstly, the course is presented to 1st year students, who have little or no background in this area in a short 4-h session, and secondly, the transfer of data between systems is problematic and simplifying the concept allowed a case study to be set up in reasonable time span.

The key element of the course is the case study which binds together all of the various CAE software packages. Had the wrong case study been chosen the course would not have worked and the objectives would not have been met. The main requirement in motivating the students is to present an easily recognizable object and not an amorphous item invented for the exercise. In this case, after a great deal

Fig. 2. Computer Integrated Manufacture

of searching, the casing of a simplified robot gripper was chosen (see Fig. 3). The original design was modified to ensure that the required features of the software were demonstrated in the course. At the same time a perfect case study was not presented in order that the fallibilities of CAE systems could be demonstrated alongside the assets.

INTRODUCTION TO DOGS

The handbook [6] for this course gives a good demonstration of the application of the “long exercise” technique to form the structure of the document. The handbook comprises four sessions each of which is associated with a 2 h tutorial. Each session describes and illustrates a range of topics in a structured manner so that new topics are introduced alongside topics already demonstrated. This has the effect of continually building on the student’s experience. Figure 4 shows some typical exercises. The experience of the student is further enhanced by the long exercise, part of which must be completed at the end of each of the sessions. Figures 5-8 show some extracts from the long exercise. Each of these diagrams show the end point that should be reached if the commands in the bottom right-hand corner are followed correctly. This long exercise also serves to show the student how a full drawing is approached in terms of making practical use of the more powerful facilities within the software.

As was the case for the Computer Integrated Manufacture course described above, the choice of the case study for the long exercise is critical to the success of the handbook. The hot air gun is easily recognizable by most students and certainly motivates the students whilst they are working on the smaller less interesting exercises.

A by-product of the handbook in the tutorial situation is that it allows each student to work at an appropriate pace and also reduces the number of tutors required for a given number of students. Allowing students to work at their own pace rather than dragging a whole group at a constant pace has illustrated

Fig. 4. A sample page from Introduction to DOGS [6].

Fig. 3. Case study, a robot gripper.

Fig. 5. Extract 1 from the long exercise

Fig. 6. Extract 2 from the long exercise.

Fig. 7. Extract 3 from the long exercise.

Fig. 8. Extract 4 from the long exercise.

that there is in fact a large range in learning curves. The majority of students complete the handbook in between 6 and 8 h. However, some students, who have had previous experience on similar systems, are able to complete the course in 4 h. A survey [9] has shown that these students constitute around 20% of the course, although this figure may vary from year to year. The handbook also leaves tutors free to concentrate on the main concepts behind the system rather than coping with the minor details of the basic operation of the system. Although the handbook was designed to be used in conjunction with the guidance of tutors it has still proved to be valuable where this help is not available. Typically this use of the handbook would be for final year students doing a project who required a refresher course in the use of DOGS. As DOGS is introduced in the 1st year several years may have elapsed since the student last used the software, which is also likely to have been updated in the meantime.

INTRODUCTION TO DUCT

As described in the previous examples this course is structured by the use of handbook [7] and so produces many of the same advantages. This software is particularly difficult to teach in the short space of time available because it is command language driven rather than menu driven. It has been established [10] that the recognition of a previously encountered menu item is very much easier than the recall of a

Fig. 9. Sample page from Introduction to DUCT [7].

command. The handbook must act as a ready reference to the software commands as well as acting as a guide to the course. The presentation of the handbook was particularly important as a large proportion of the communication between the user and the software is textual. A range of ten symbols were associated with the various forms of text in the document, such as input, output: Fig. 9 shows a typical page from the manual.

A feature of the command language is that it can be entered in either a long or short form. For example, entering the commands **draw duct** and **dr du** produce exactly the same response. To use the software effectively the short form should be used, on the other hand, the short form alone usually confuses new users. The command language was presented with the long form complete and the short form highlighted as follows; **dr aw du ct**. The implication of using this style of presentation is that it requires a high quality of reproduction with relatively high costs. Currently at Loughborough the handbooks, as described in this paper, are written in the Department of Manufacturing Engineering, published by the Computer Centre and made available to the University as a whole. Students are asked to pay a nominal charge of 50p to offset the printing costs. The handbooks have also been distributed to engineering departments in other universities and polytechnics, some of which are considering integrating the handbooks in courses for the next academic year.

CONCLUSIONS

Formal evaluation of the courses has been carried out in terms of issuing questionnaires to the students at the end of the courses and also by taking into account the assessed work. The results from both of these sources were encouraging. The questionnaire covers a range of topics but is mainly aimed at determining the students' attitude towards the presentation and quality of the course. In total 15 questions were asked each with a rating of 1-7, with 1 being the worst rating and 7 the best. The overall averaged score was 5.0 with a maximum average score, for any one question, of 5.4 and a minimum score of 3.6. Apart from the encouragement that the course appeared to be working as expected it was difficult to gain any feedback in terms of course improvements from these results. The results from the coursework also showed an improvement in standards over previous years but again no firm conclusions could be drawn from them.

However, a third form of evaluation was undertaken during the tutorial sessions. A master copy of the handbook was kept and any errors, misunderstandings and informal comments from the students were recorded in the handbook. This information was found to be an extremely valuable source of feedback to the consequent improvement of the handbook.

REFERENCES

1. Case K. and Acar B. S., Use of computer aided design systems for discrete part manufacture. *Behav. Inform. Technol.* In press.
2. Barker B. E. (Ed.), CHEST (Combined Higher Educational Software Team) Educational Software Scheme Catalogue. Bath University Computing Services (1988).
3. Broadbent D. E., *Perception and Communication*. Pergamon Press, New York (1958).
4. Card S. K., Moran T. P. and Newell A., *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, N.J. (1983).
5. Douglas S. A. and Moran T. P., Learning text editor semantics by analogy. In *Human Factors in Computer Systems* (Edited by Janda H.), pp. 207-221. Association for Computing Machinery, New York (1983).
6. Doyle R., Introduction to DOGS. Loughborough University of Technology, Computer Centre (1988).
7. Doyle R., Introduction to DUCT. Loughborough University of Technology, Computer Centre (1989).
8. Doyle R., Engineering Applications 1. Loughborough University of Technology, Department of Manufacturing Engineering (1989).
9. Doyle R., A survey of student opinion on the effectiveness the DOGS handbook. Loughborough University of Technology, Department of Manufacturing Engineering (1989).
10. Anderson J. R., *Cognitive Psychology and its Implications*. Freeman, San Francisco (1980).