ELSEVIER

Advances in Engineering Software 32 (2001) 477–487

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Problem solving environments in aerospace design

A.J. Keane*, P.B. Nair

Computational Engineering and Design Centre, School of Engineering Sciences, University of Southampton, Highfield, Southampton SO17 1BJ, UK

Received 5 April 2000; revised 31 July 2000; accepted 17 October 2000

Abstract

Recent developments in aerospace design systems are being driven by studies in a number of areas including new software methodologies, advanced approximation techniques, data archiving and fusion methods, artificial intelligence, and the natural biology and socio-economic behaviour of species together with the continuing developments in computational hardware. Advances in these areas are leading to interesting new ways of managing the design process when dealing with increasingly complex systems and also increasingly complex design organizations. This work covers topics as diverse as the formal optimisation of differential equation models, the management of workstation clusters in design offices and the re-use of linguistically formulated knowledge. Collectively, such studies allow the production of problem solving environments, where a wide range of approaches can be readily integrated by the design team to suit the problem in hand.

The ideas discussed in this paper have been formed by the authors' experiences gained in aero-engine, aircraft and satellite design optimisation. They are not meant to be exhaustive or prescriptive, but instead present a personal view of some of the challenges that lie ahead. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Design; Optimisation; Problem solving environment; Aerospace

1. Introduction

Over the last 30 years, the process of engineering design has been transformed by the introduction of massive computing power. There has been a move away from paper-based systems towards 3D solid models and computer simulations. This has allowed the development of increasingly sophisticated products such as the modern personal computer, the space telescope and large airliners. It remains the case, however, that we still desire to produce ever-more complex engineering systems [1]. Our ability to use computers to design and predict the performance of these systems is now being hampered by our methods for producing and managing the design process and associated computer software. Moreover, the software that we do produce does not easily integrate with existing software and is, additionally, prone to bugs and errors that are highly expensive to correct. It also tends to be integrated into systems by software specialists and delivered 'shrink wrapped' to designers, whereas designers would prefer to mix and match methods themselves as they tackle each new problem, but without recourse to writing software in the traditional way. Finally, many of the systems we do have are not human centred in nature and so do not allow design teams to give of their best. Essentially, we need a 'paradigm shift' in engineering design systems — a new approach to the problem.

Nature by contrast, finds little difficulty in producing incredibly sophisticated systems. Even the most simple plant or animal represents a triumph of design that has been achieved without any silicon based computational effort. Of course, the most complex systems we are aware of are the human brains that we all take so much for granted. The key feature of biological systems is their ability to adapt and show 'emergent' behaviour. The adaptation takes place on a range of time scales from the very short to the extremely long. Memory can be a very short-term process. Darwinian evolution can take aeons (although it is worth recalling that mankind, for example, has only existed for perhaps 2000 generations). Moreover, this adaptation may be in response to changes in the environment or to the presence and behaviour of other species or other members of the same species. In all cases, however, we see adaptation. The emergent properties of communities can be startling in their complexity. Study of a single insect in isolation would never lead one to expect the amazing structures that colonies of insects can produce. Equally, the processes by which humans experiment with sounds when learning language, although commonplace, are staggering in their

* Corresponding author. Tel.: +44-23-8059-2944; fax: +44-23-8059-3230.

E-mail address: ajk@soton.ac.uk (A.J. Keane).

final results. It is therefore natural to speculate that engineering systems, which also adapt and develop over time may overcome some of the limitations of our currently deterministic, pre-planned 'open loop' approach to design.

The idea of adaptation in engineering systems and in particular, engineering software is not a new one. However, some remarkable progress has recently been made in this area. The whole idea of artificial life is one that excites the imagination and offers very significant advances in the way we engineer the products we need. There are a number of key technologies that underpin the development of artificial life. They span genetic algorithms, neural nets, genetic programming, robotics, knowledge management, modern control systems design, expert systems, knowledge extraction, data mining, multimedia (MM) systems, computational resource management, etc.

This paper discusses some of these aspects and how they can be used to inspire the development of better paradigms for engineering design. These paradigms need to accommodate the behaviour of human designers but also the computer systems that they have access to. People cannot be directed in the same way as computers, and due allowance must be made for their hopes and aspirations. Equally, it is becoming clear that it is no longer sensible to assume that modern design computation can be rigorously policed and standards rigidly enforced. Systems need to be tolerant of a multiplicity of codes, standards, processor types, operating systems and less than perfect software development, networking and resilience. Only if we can integrate human aspirations and the explosion of modern computing technology in an adaptive and intelligent fashion, we will be able to design the systems we desire over the next millennium.

2. The traditional approach

The traditional approach to design has been to view the process as a sequence of events that are dealt with in series. Sometimes, this sequence is passed through only once, but more commonly it is cycled through a number of times. Traditionally, the design starts with some kind of concept decision-making. The concept design is then passed to embodiment design and detail design for refinement before finally going to manufacturing design and the product manufacturing process. On many occasions, embodiment or detailed design has a major influence on the original concept and it then needs refining and so on. This process can be illustrated by a design spiral in which a series of domains are investigated in turn, at ever increasing levels of detail until the final design has been produced, see Fig. 1. In whatever framework this process is set out, the key element is the transition of data from one level of the design process to the next in an orderly, pre-planned sequential fashion. Usually this requires standards for the transfer of data to be rigidly adhered to and also for software to be compatible with the approach. Moreover, in this process

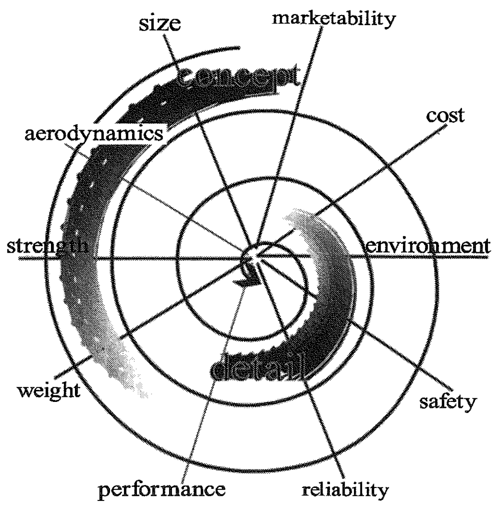

Fig. 1. An aerospace design spiral.

individual designers look to take control over only one particular stage and have little interest in the others. In addition, engineering specialists like structural engineers or materials scientists feel ownership of only very limited aspects of the process. Lastly, this approach does not recognize that design teams may be spread over different time zones and sites and over organisations that may use different computer systems for similar tasks (consider the many finite element or CAD packages currently in widespread use, even within single companies).

There are a number of key problems with this approach to design as systems become more and more complex. For example, it is not normally possible for the stages of design, shown in Fig. 1 to proceed in parallel without design-times becoming too long for current needs. Equally, there is usually a mismatch between the analysis methods used in one design stage and those in the next. It is quite common for the concept design engineer to use very crude methods for analysing the stresses in a product or the fluid flow around it while the detailed design team accesses far more sophisticated tools. Clearly, there is then scope for difficulties when designs are passed between these groups. Another aspect that limits the design process is the human factors that come into play when design teams are brought together. In a world of rapidly changing design and rapidly changing manpower, it is crucial that the design process is not disrupted by changes in personnel or in design methodology.

In fact, there are a whole host of improvements that could be sought in the design process and supporting technologies. The following, by no means exhaustive, list sets out some of those that have been identified in recent studies:

- improve quality control;
- facilitate team decision-making;

- improve design environments;
- create a seamless integration between design and analysis;
- understand the product realisation process;
- archival and re-use of design history;
- determine the impact of decisions;
- promote continuous learning;
- integration between analysis tools;
- enhance creativity and innovation;
- reduce development time by maximising parallelism;
- improve information infrastructure;
- produce globally optimized designs;
- manage complexity and risk;
- enhance critical thinking and evaluation methods;
- integrate product design data;
- improve communication of design specifications to remote sites and companies;
- integrate product and manufacturing process development;
- integrate large-scale systems.

It is clear that there is much to do and we must therefore seek radical new ways of attempting to equip, direct and structure the design process if we are to produce more complex designs in faster time-scales. Moreover, we must draw on a wide range of disciplines if we are to make real headway. Unfortunately, design research itself has by no means been as productive as one could wish. We must therefore also seek ways to better implement and deliver the ideas being formulated in the research community. We must also recognise, however, that any one vision may be mistaken and so we should aim to progress a number of visions simultaneously until it becomes clear which one will offer most benefit in any given situation. Thus, we must build system demonstrators and deliver demonstrations of numerous ways of working if we are to be sure of making steady progress. These must lead to feedback from practising design staff which is acknowledged and acted upon — delivering design systems to teams that have had no part in their development will almost certainly lead to their being ignored by the very people they are developed for.

3. Problem solving environments

Recently, much interest has been focussed on what are termed problem solving environments (PSEs). Note that although this term is relatively new, the idea behind it is not, see for example Ref. [2]. PSEs aim to assist engineers to help them solve the tasks they have to undertake. A design PSE must be able to handle all of the requirements of a design team and seamlessly integrate between the stages of the design process. Moreover, it must be configurable by the design team itself to suit each new problem as it is tackled. It is already possible to specify some of the component parts that such environments will need to encompass:

- the ability to capture knowledge about and used in the design process as it happens;
- the ability to provide advice and knowledge to the designers at appropriate times and at speeds/volumes the designer can cope with;
- the ability to undertake routine tasks without prompting, based on the current status of the design;
- the ability to support various workflow models that the team may adopt;
- the ability to deal with design systems integration and communication issues as well as mathematically based models (e.g. my code can use your code, my code can use your data, my program starts automatically when yours finishes, etc.);
- the ability to deal with multiple data sources, and complex and changing work flows;
- the use of swarms of communicating but small-scale, autonomous systems, i.e. intelligent agents;
- the aiding of individual and group decision taking in the face of uncertainty, i.e. the management of risk;
- human centred, individually customized computing and interfaces;
- enhanced methods of interacting with computers, e.g. virtual reality (VR), MM, etc.;
- tele-working and virtual teams using interfaces with common look and feel;
- interaction with extended enterprise business process systems;
- the ability to draw on large numbers of heterogeneous, widely dispersed computing nodes and their software which can me managed with limited overhead.

Those engaged in design research are focussing on these issues and how they can be taken forward. It seems clear, however, that a user-reconfigurable, computational framework encompassing such ideas on a global network will form the backbone of a good design PSE.

4. The structure of a problem solving environment

In order to place some of these ideas in context, it is helpful to sketch out the structure of a basic aerospace design PSE. We can begin this process by describing a number of elements that are familiar to all those involved in design and which we will wish to incorporate. First must come the design team itself: notice that we are here concerned with a group of people and not the traditional single 'user' often seen in many descriptions of software. Moreover, we accept from the outset that these people will have differing backgrounds and tasks and will probably not be collocated or working in the same time zones. They will also have differing levels of experience and career aspirations that must be catered for if we are to ensure that all can engage creatively in the design process. They may also disagree on how to proceed at some points in the process and such conflicts must be resolved in the best interests of the design and the team.

The databases that hold the description of the current manifestation of the product (or range of products) being worked on form another key building block. These should, however, be considered along with other associated, but informally structured data (hopefully held in digital format but also encompassing video, voice and pictorial data, i.e. MM). Again we do not assume a single database with a unified electronic product model. Although such a single model is often seen as desirable by company senior management it flies in the face of reality — design teams always hold multiple, overlapping views of the product(s) and their design databases often include personal and informal collections of information and knowledge which we should properly allow to be included in a good PSE. In fact the value in experienced staff often lies in their informal stores of knowledge and the associated understanding of where connected relevant data lies. To build a good, human centred, creativity enhancing PSE, we must accept these features ab initio and work with them, not assume they are difficulties that have to be ironed out in some brave new world.

Next comes a collection of computing nodes connected in some way via a network of variable and uncertain bandwidth. These nodes will be highly varied in type, capability, reliability and importance to the design process. Moreover, there will be very many such nodes that can potentially be called upon, i.e. all those owned by the team, their employing companies and their partners together with national or governmental facilities (i.e. certainly thousands and perhaps tens of thousands). This infrastructure is increasingly being referred to as 'the grid' and it is again far from the homogeneous well managed and policed system that management might desire. Thus, our PSE must be tolerant of such vagaries and be able to get the best from the system without depending on perfect reliability, uniform connectivity, zero latency, etc. In fact, it must be as tolerant of the vagaries of the grid as humans are of each other. This will partly come from more sophisticated software agents but also from duplication of processes that are of higher importance. (We already take this for granted in well managed local disk store with its RAID systems and robotic tape backup — we must move processing capabilities to similar levels of resilience by a variety of means, including migration or duplication of processes and communication streams.)

Design computing nodes are increasingly used to support a wide range of analysis codes that are used to predict the performance of products. Note that although these may be used to add to our knowledge of the product and its performance, their fundamental task is not to change the product, i.e. they are aids for designers rather than being design tools per se. Here we retain this appellation for tools that can be used to alter the design in some computer initiated way e.g. as can optimisation methods. Analysis codes do, however, play a central role in design and also consume considerable quantities of computing resource. One must not, however, mistake the computational model for the product itself in this process, nor assume that modelling is a goal in itself: rather analysis is used by designers to aid in taking decisions on how to change the design. If the designer is content with the current product and has no intention of changing it then analysis ceases to have any relevance to the design process (unless it is used offline to enhance the capabilities of the PSE in preparation for the next design). It is worth noting, as an aside, that by far the greatest effort of university science and engineering departments in developing computer codes has been in the pursuit of increasingly accurate analysis codes: relatively little effort has been expended by comparison on systems for modifying designs or for helping designers manage the design process. The current capabilities of modern contact mechanics finite element analysis (FEA) systems and Navier Stokes computational fluid dynamics (CFD) solvers would seem to suggest that enhancements to the design process will be more readily obtained in other areas, than in continued further refinement of already astonishingly accurate tools: the law of diminishing returns is beginning to apply in the field of engineering computational analysis.

Fig. 2 shows the traditional single user, CAD based version of these tools (using the commercial CAD systems of CADDS, Pro-E, I-DEAS, etc.) connected in a conventional form as

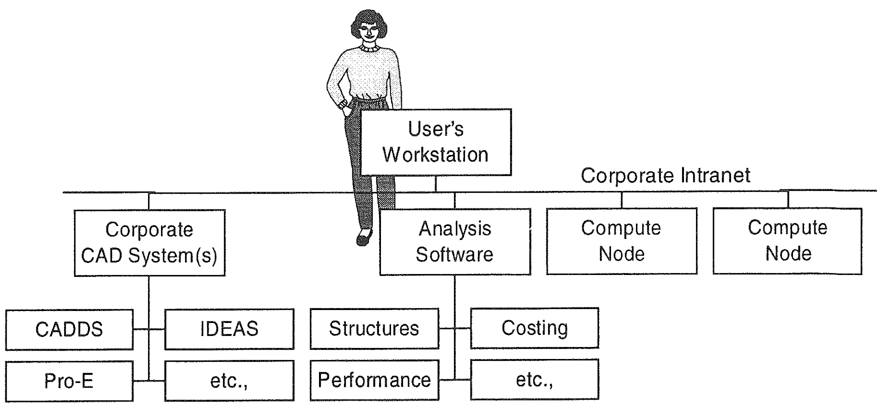

Fig. 2. A traditional (current technology) software environment.

commonly in use today. We may identify some of the weaknesses in such systems that make clearer the needs of a good PSE. In no particular order we may observe that:

- there is no system for deciding which member of the design team has access to which computing facilities;
- there is no rational for archiving the results of one design process to help another, whether concurrent or in the future;
- there is no system for ensuring that all members of the team can have visibility of the work of others;
- there are no aids that might suggest what changes the team might make to improve their design;
- the model implicitly assumes that data can be passed between machines and processes without interruption or difficulty in interpretation;
- the model implicitly assumes that the computing nodes are fast enough to support the analysis requirements of the designer.

To address some or all of these needs we require a further class of objects that do not fit into the previous categories of people, databases, computing resources and analysis codes. It is the central tenet of this article that the presence of such additional modules linked together in a *user reconfigurable* way is the hallmark of a PSE. This category of PSE tools includes, amongst others:

- load scheduling and resource management software;
- backbone software that is tolerant of heterogeneous hardware environments and that can work in a distributed environment;
- tools to enable individual users to work in personalised environments;
- tools to ensure resilience against computing resource limitations or failures;
- optimisation tools;
- computation check-pointing and restarting tools;
- approximation and data-fusion tools;
- knowledge capture and sharing tools;
- information archiving and re-use tools;
- MM capabilities;
- case-based and rule-based reasoning systems;
- tools that allow global and 24 h working;
- tools that allow the design team to reconfigure their PSE to suit the problem in hand without recourse to writing code (e.g. drag and drop linking).

Fig. 3 shows a reworked version of Fig. 2, where a number of

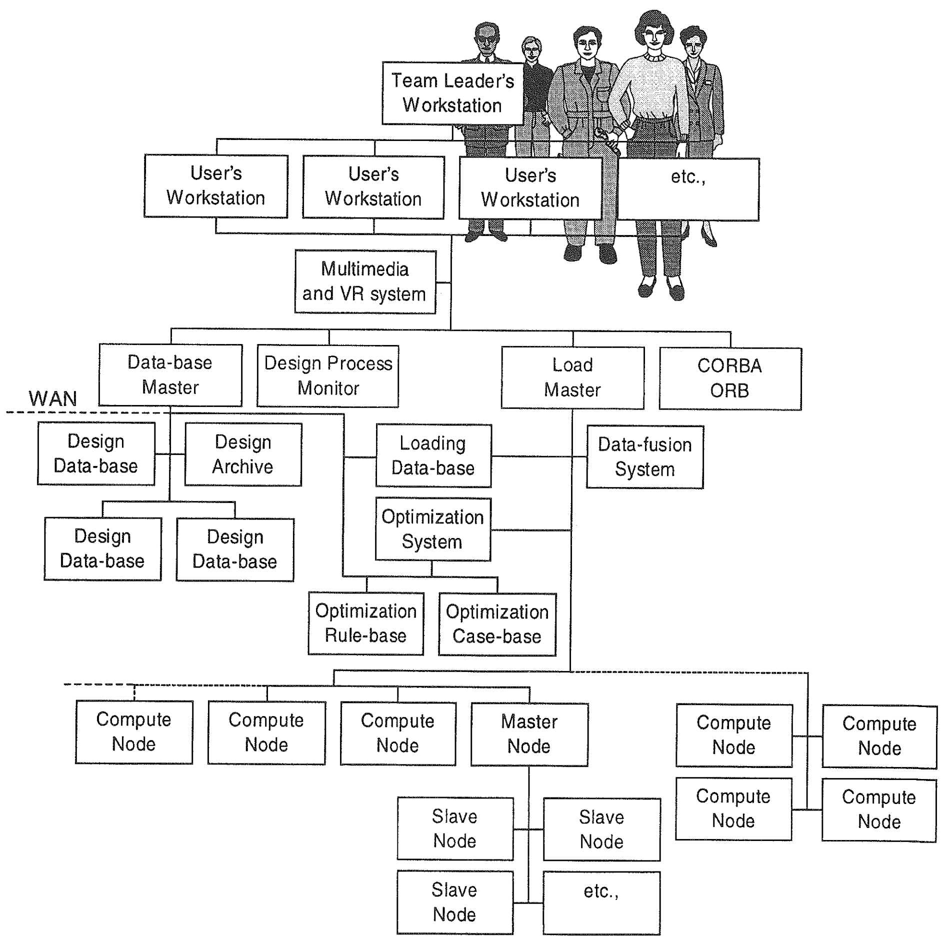

Fig. 3. An embryonic PSE.

more advanced tools have been added and wide area network (WAN) connectivity included, and which now offers some PSE capability. Moreover, it looks rather advanced when seen from the viewpoint of most aerospace design systems as they currently exist.

5. Existing software systems and their evolution

Study of the software currently available to design teams reveals an almost organic and often unstated evolution of traditional packages in the direction of PSE development. For example, the I-DEAS product marketed by SDRC [3] increasingly covers the whole range of mechanical engineering tools from 3D CAD through analysis to manufacture. Equally a number of specialist FEA and CFD codes are now offering design optimisation capabilities. Further there are a number of advanced optimisation packages currently available that focus on the development of engineering designs such as the iSIGHT package by the Engenious Software Co. [4]. Alongside these 'third party' products a number of the large international aerospace companies are producing in house systems for their design staffs. Although all of these offer some of the capabilities outlined above, none was designed at the outset to provide the kind of capabilities described here. They do not allow the kind of 'plug and play' capability for re-structuring the way the systems work that designers desire, nor do they allow ready integration of a variety of differing information sources or use across heterogeneous non-resilient computing environments.

The future development of such systems will no doubt continue apace but it seems clear that having some overview of what such systems should ultimately be capable of will help in this process [5]. Already it seems that adopting a client/server architecture with distributed and re-usable modules of the type adopted in the CORBA standards [6] looks set to be one promising architecture. Another is the adoption of autonomous agents that engaging in dialogues with each other before carrying out transactions of some kind [7]. It therefore makes sense to use these approaches within research groups that are already studying many of the other elements of engineering design software, even if they seem somewhat cumbersome at first sight. This is the approach being taken in the Computational Engineering and Design Centre at the University of Southampton.

This process may be illustrated by reference to the OPTIONS optimisation software tool developed by those working at Southampton [8]. This tool, which was originally written in the mid 1980s, is in use within several aerospace companies and has been used to underpin a number of research projects. A great deal of the underlying code in the system is, however, written in Fortran and this must be accommodated if the system is to be developed for future use. It is thus typical of many items of legacy engineering software from this and earlier periods which are still of great value to the engineering community — although it might be thought sensible to re-write such codes in modern languages, economic limitations usually prevent this. It must thus be deconstructed into usable chunks and wrapped in more modern clothes, retaining what is good and well tested from the past and only updating and modifying that where there will be significant benefit (the same can be said of many FEA, CFD and related numerical codes). This is more readily achieved if the initial codes were well structured to begin with — fortunately OPTIONS is highly modular in nature and uses a centralised database that allows this kind of development. Even so the work involved is not trivial.

Fig. 4 shows the original structure of the OPTIONS code. Notice that although there is a centralised database here it does not distinguish between product data and process data, i.e. things such as the length of a wing are stored in the same place as the size of the population used in a genetic algorithm search. Additionally, the original system has a parallel execution module that allows for load-balancing across clusters of machines using NFS and Unix socket requests. Both of these aspects are features we would choose to do without in a modern PSE, instead relying on separate distributed databases and CORBA modules, respectively, to provide this functionality.

On the positive side OPTIONS does provide access to a large collection of integrated search methods that can be deployed independently or in series, depending on the problem in hand, including response surface modelling [9] and multi-objective optimisation [10], alongside the more traditional single function based search. It also allows for loose or tight coupling with analysis codes but needs such interfaces to be hand-built for each new problem code. OPTIONS has been applied to a number of problems over

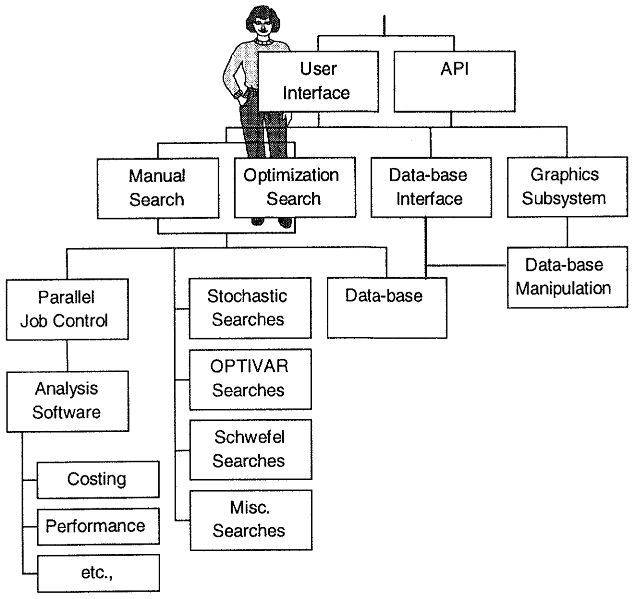

Fig. 4. The original structure of the OPTIONS code.

Table 1
Initial design parameters, constraint values and objective function values

Lower limit	Value	Upper limit	Quantity (units)
100	168	250	Wing area (m^2)
6	9.07	12	Aspect ratio
0.2	0.313	0.45	Kink position
25	27.1	45	Sweep angle (°)
0.4	0.598	0.7	Inboard taper ratio
0.2	0.506	0.6	Outboard taper ratio
0.1	0.150	0.18	Root t/c
0.06	0.122	0.14	Kink t/c
0.06	0.122	0.14	Tip t/c
4.0	4.5	5.0	Tip washout (°)
0.65	0.75	0.84	Kink washout fraction
	1,91,879	2,00,000	Wing weight (N)
40.0	42.35		Wing volume (m^3)
	4.179	5.4	Pitch up margin
2.5	2.693		Undercarriage bay length (m)
	3.12		D/q (m^2) — from empirical code
	2.85		D/q (m^2) — from CFD Euler code

the years — most recently it has been used to provide a multi-level or 'zoom' capability for concept design of transonic aircraft wings [11]. In this study a series of analysis codes offering different levels of fidelity were linked together so that the user may readily change the sophistication of their analyses as an optimisation run progresses or construct back-to-back data sets that allow meta-models to be built that highlight the differences between the codes in use.

The results from using this system to optimize a civil transport aircraft wing for operation at Mach 0.785 and a Reynolds number of 7.3 million illustrate its power. The objective was minimization of wing drag (D/q) as calculated by a commercial Euler CFD code but using an empirical (and much faster) alternative code to speed up the process. The target lift, wing weight, volume, pitchup margin and root triangle layout were chosen to be representative of a 220 seat wide body airliner. The initial design point for this example is summarized in Table 1, which details the starting values of the design parameters, the constraint details and the drag values. Notice that the drag values reported by the two codes differ by some 9%.

A number of optimisation runs were carried out on this problem, limited to at most 1000 full CFD evaluations. This might normally be considered rather a small number of evaluations for meaningful optimisation of such a problem, but is not unreasonable for a code taking nearly four hours to run. This leads to a total run time in serial processing of 24 weeks, or around three weeks using an eight-processor SGI power challenge machine in parallel.

To begin with, a single level optimisation was carried out using a genetic algorithm with a population size of 250 for 40 generations using the empirical code, see line 1 of Table 2. This shows that the empirical code allows a design to be produced which, when analysed by the CFD code, shows substantial improvements.

Next the best 100 members of the final population of the 10,000 step empirical search were used to form the initial population of a CFD based GA search, which consisted of 10 generations of 100 members each, see line 2 of Table 1. This approach results in further improvements, although they are slight over just using the empirical search. Either of these results is considerably better than those obtained from using the same effort on the CFD based search alone (here 10,000 empirical evaluations cost less than a single CFD evaluation), see line 3 of the table, although this final approach also significantly improves on the initial design.

It is clear that such multi-level approaches work well and that searches using more approximate methods should be carried out before transferring effort to more complex ones. Fig. 5 shows a CFD simulation created using the commercial Euler code linked into this system. Unfortunately, although powerful and representing a significant advance over conventional practise, this system does not offer all the flexibility that design teams would wish for, nor was it simple to construct.

Fig. 6 shows how we might choose to break down a code like OPTIONS into CORBA wrapped modules accessing databases that can be used on a more 'plug and play' basis along with the necessary additional modules required to provide the parallel execution and load balancing that has been found so useful in the past. Clearly this system is still far from complete and we are currently developing a number of further modules to provide enhanced functionality such as data archiving, retrieval and fusion plus case based and expert system based reasoning (here just dealing with search methodology). It does, however, conform to the standards being adopted by NASA in their Computational Aerosciences program [5].

Table 2
Results of the wing optimisations

No.	Search and analysis method	Empirical (D/q in m^2)	CFD Euler (D/q in m^2)
1	GA search, 10,000 empirical code evaluations	2.75	2.38
2	GA search, 10000 empirical code evaluations followed by 1000 CFD evaluations	2.75	2.37
3	GA search, 1000 CFD evaluations	3.02	2.54

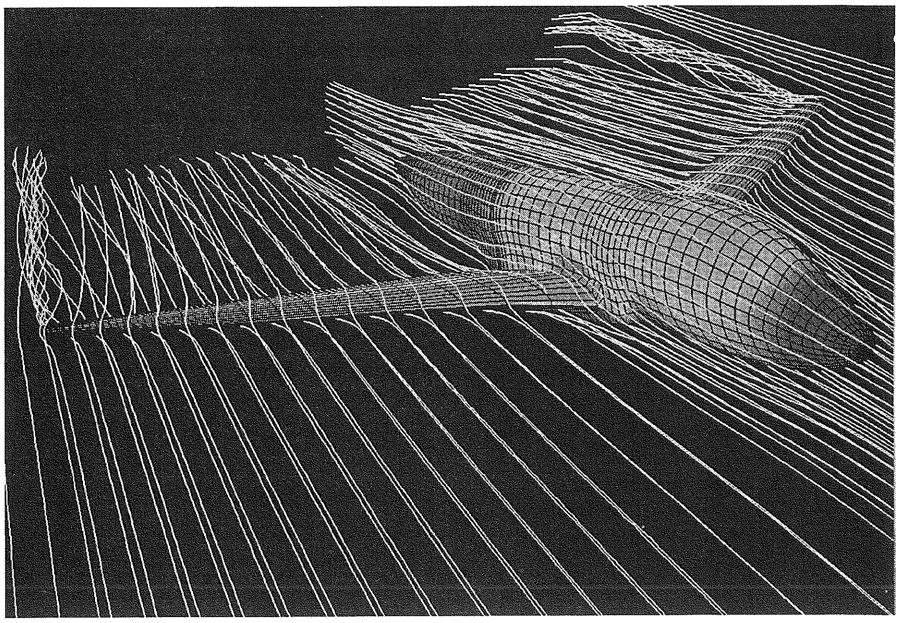
Fig. 5. The flow-field past a transonic wing body flying at Mach 0.75 computed using the Euler boundary coupled flow solver MGAERO.

6. Future plug-ins for problem solving environments

Having described the basic structure of a PSE it is finally worth discussing in more detail a number of the areas of technology that are currently developing that will enhance PSE capabilities and that are shown schematically in Fig. 3. These tend to span a wide range of areas and only a limited number can be outlined here.

6.1. Model approximation and data-fusion

One of the central problems with advanced, physics based

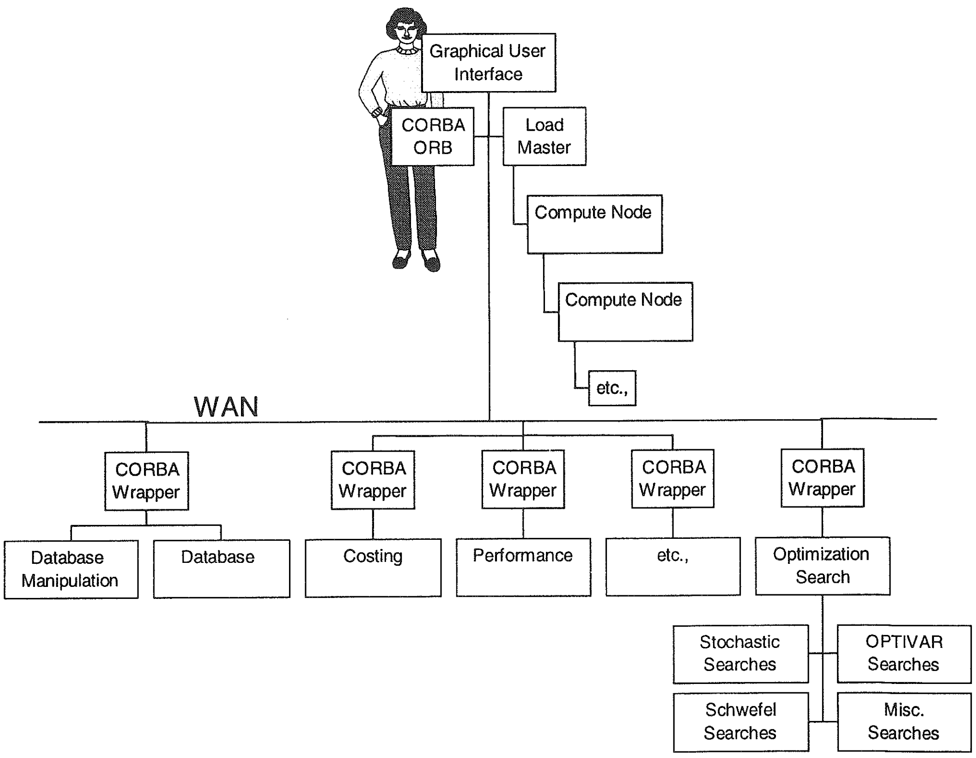

Fig. 6. The modules of OPTIONS broken apart and CORBA wrapped on a WAN.

analysis codes is the great computational expense they involve when used to model even modestly complex systems. It has therefore been natural ever since computational modelling began to develop a range of models of varying degrees of fidelity. These then enable the user to vary the degree of effort expended on a calculation by the appropriate choice of model. Placed in a PSE context it is possible to re-cast this process. Now the users decide how important a calculation is by specifying how much of the available time budget can be spared in this area and some measure of desired precision. The PSE then samples the available computing resources and solution methods and constructs a meta-model of its world so as to suggest a viable strategy to the users. This strategy may involve the use of an expensive code in a parallel fashion, it might use response surface methods together with model updating to yield an approximate solution or it might aim to fuse the data coming from a range of modelling tools of varying complexity. All of these would be backed up by searches in the available data-structures to see if existing data might help or even supplant the need for fresh calculations. The data sources may also provide information on the likely fidelity to be achieved by the various methods given previous results.

Preliminary studies in this area by the authors and co-workers, see for example Refs. [12–14], have indicated that approximating expensive functions by using knowledge of the underlying physics is to be preferred but that when this is not possible 'black-box' models can still be very effective. Such models essentially provide the means to interpolate in previously calculated data (a form of response surface modelling) but additionally allow for data from various computational models to be fused, along with any prior information about the confidence a designer has in any particular set of computations. Currently, this model building and data fusion process can require some experience to give best results and it would clearly be desirable for any PSE system to have embedded intelligence either via rule-bases or case-bases.

6.2. Automated design archiving

During the design process a great deal of computational effort is expended in various directions. Often the results from these computations are discarded as the design evolves and more refined models are adopted. This process can be extremely wasteful as much of the discarded material can be used to inform subsequent designs or subsequent design processes, i.e. it may be that results can be used to seed new calculations, used to improve 'black box' interpolators of models or allow more informed choices about appropriate design strategies. One effective way of doing this is to archive all results as they are produced and then to post-process these archives in such a way that useful information can be extracted from them.

The archiving and retreival process is by no means simple if the resulting systems are to generate more than just an enormous unstructured collection of data. Schemes need to be put in place to allow such data to be sensibly tagged as it is created and then to be managed and phased out over time once it has been collected, collated and used. A key to the successful implementation of such strategies is that the tagging must not require human intervention at create time – experience shows that design staff are always too pressed by their main functions to be prepared to expend this effort, even though it may save them time later.

Subsequent use of archived data then requires an intelligent query system since designers are not expert in the use of things such as SQL and so 'query by example' type approaches become important as does the ability of any system to learn the kind of queries that typical users may desire. Fortunately, a great deal of progress in this field is being driven by the explosion in internet use and browser technology. It is thus sensible to interface the data being archived to some form of advanced mark-up language and internet browser technology. Preliminary work in this field using XML has shown great promise, see for example Wason et al. [15].

Lastly, strategies must be adopted for the gradual compaction and removal of data over time. This process cannot simply equate age with lack of utility — rather as data ages so it must be boiled down and consolidated so that important key material that has been heavily used is retained along with information that is particularly difficult to recreate. Essentially, an electronic version of the old workbook system that most aerospace companies used in the 50s, 60s and 70s must be put in place, especially for the designs of products that may have long lifetimes and for which design audit trails must be maintained (40 year lifetimes are not uncommon for aircraft designs).

6.3. Load mastering and massively parallel computation

It is increasingly obvious that the only way that really significant increases in computational speed for design calculations will be achieved is via the mechanism of massively parallel systems [1]. Computational facilities with thousands of compute nodes will require powerful load mastering software to be developed. Such tools will need to go far beyond the products currently in use such as LSF [16]. Such schemes will need to adopt a client/server approach (and so might be CORBA based) and will need to hold internal representations of both the available hardware (including details about interconnection and latency issues) and of sensible strategies to make use of these. Moreover, any large system will obviously need multiple load mastering facilities and these will need to co-operate with each other using agent-based approaches [7]. Additionally, it will not be possible to assume that such systems will have very high reliability and so multiple, redundant calculations will be needed, i.e. redundant arrays of inexpensive CPU's (RAIC).

6.4. Multimedia and virtual reality systems

More complex ways of interacting with design systems are continuously emerging and it seems clear that both MM and VR systems will play an important role in future PSEs. Such systems should act as unifying agents that serve the designer/design team by pulling together all the various PSE resources and presenting information in a more meaningful way. They should also hide complex PSE system details from the designers thus allowing them to focus more on real engineering problems. Any MM or VR software will need natural language processing capabilities to support interactive Q and A sessions with the users: moreover it is desirable that such systems ask intelligent questions of the users. It is clear that advanced systems that support user interaction should have direct access to all design and process databases (local as well as distributed). They should be capable of intelligent retrieval of previous designs, related problems, issues, etc., and weave then into a 'design story' before presenting information to the user. For this purpose they might use a case-based retrieval and reasoning system (see for example Ref. [17]). They must also allow co-viewing of design documents (even as they are being modified) through the use of shared databases. At the same time they should allow the designer/design team to work on different parts of the design object as well as at different levels of complexity. The implementation of such an MM-VR system would typically require use of 'immersive' technology and availability of high bandwidth links for real time support.

6.5. Intelligent search and optimisation

Increasingly designers prefer to specify product goals rather than product features. Thus a natural way of working is to use goal-oriented software to help manipulate a design to produce a better one. Optimisation methods do this and have been available for very many years but have, until recently, not been heavily used by the design community. This situation is rapidly changing because computing power now allows increasingly accurate analysis codes to be deployed in this way, see for example the work reported by Jameson [18] in a recent theme issue dedicated to optimisation. Even so, it remains difficult for everyday designers to make use of this technology. First, they often find it difficult to connect together all the elements they need to correctly create a model, where changes in design decisions can be automatically reflected in computed goals. Secondly, even when they do they often cannot muster sufficient computational resources without recourse to significant efforts in connecting together the available computing power. Thirdly, even if these first two hurdles are overcome they often lack the required expertise in the use of search methods such as genetic algorithms, simulated annealing, etc., and therefore have little confidence that extensive computational runs will produce worthwhile results as opposed to just burning up compute cycles.

When seen from a PSE perspective the aspiration must be for the designer to be able to specify the things that might be changed, potential goals (often more than one) and constraints together with a time or resources budget. The PSE then should respond by suggesting strategies that may solve this problem and analysis modules that may be used to link together the goals and free variables. Moreover the selection of appropriate modelling techniques should be dealt with as described above, i.e. an appropriate mix of fused and re-used data with sensible and sparing use of new computational runs. The design team could then probe the PSE as to why the strategies being suggested have been chosen — probably the PSE would use a mix of case-based reasoning and expert system knowledge to do this and could then play back previous examples or expert opinion as to the choices made. The team's analysis experts might then wish to refine the exact choices of analysis methods being adopted before authorizing the chosen strategy. Having started a process the PSE should monitor its own progress by comparison with previous similar work, using its case-base, and adjust its own strategy on the fly, either with or without consulting the team. Such ideas may seem somewhat far-fetched but they do not require any technology not already deployed somewhere in the world of computing today. What is needed is the effort to integrate them into usable design systems.

7. Conclusions

In this paper, an attempt has been made to set out some of the ways in which the evolution of software paradigms is leading to the development of a new generation of systems to aid design engineers. These systems are termed PSE and their aim is to provide a flexible means whereby the engineer spends more time thinking about and solving the current and future design problems and less on moving data between disparate and disconnected analysis codes with few ideas on how to improve a product. Moreover, such systems will need to place a strong emphasis on a human centred team-working approach that allows the people in the team to give of their best. Such systems will perhaps be best measured by the demise of internet jokes which give spoof dialogues of the difficulties people have in dealing with their PC's! The goal must be to have systems that people hardly notice rather than ones that they have to fight and manipulate with cunning dodges to achieve their goals.

References

[1] Bierdon RT, Mehrotra P, Nelson ML, Preston FS, Rehder JJ, Rogers JL, Rudy DH, Sobieski J, Storaasli OO. Compute as Fast as the

Engineers Can Think! — Ultrafast Computing Team Final Report, NASA/TM-1999-209715, 1999.

[2] Cody WJ, Rouse WB. Automated Information Management for Designers: Functional Requirements for Computer Based Associates that Support Access and Use of Technical Information in Design. AL/CF-TR-1993-0069, Air Force Materiel Command, Wright-Patterson Air Force Base, Ohio, 1993.

[3] Igusa T, I-DEAS Master Series Reference Manual. Structural Dynamics Research Corp., 1994.

[4] iSIGHT — http://www.engineous.com.

[5] NASA Computational Aerosciences program — http://hpcc.grc.nasa.gov/hpcc2/welcome.shtml.

[6] Orfali R, Harkey D. Client/Server Programming with Java and CORBA. second. New York: Wiley, 1998.

[7] Jennings NR. Commitments and conventions: the foundations of coordination in multi-agent systems. Knowledge Engng Rev 1993;8(3):223–50.

[8] Keane AJ. OPTIONS Design Exploration System User Manual. B2.2. http://www.soton.ac.uk/~ajk/options.ps1999.

[9] Jones DR, Schonlau M, Welch WJ. Efficient global optimisation of expensive black-box functions. J Global Optim 1998;13(4):455–92.

[10] Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multi-objective Optimisation. Evol Comput 1995;3(1):1–16.

[11] Keane AJ, Petruzzelli N. Aircraft Wing Design Using GA-based Multi-level Strategies. 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, Long Beach, 2000.

[12] El-Beltagy MA, Nair PB, Keane AJ. Metamodelling techniques for evolutionary optimisation of computationally expensive problems: promises and limitations. In: Banshaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M, Smith RE, editors. Proc Genet Evolut Comput Conf (GECCO99). Orlando: Morgan Kaufman, 1999. p. 196–203. ISBN 1-55860-611-4.

[13] El-Beltagy MA. A Natural Approach to Multilevel Optimisation in EngineeringDesign. PhD Thesis, Southampton University, 1999.

[14] Nair PB. Design Optimisation of Flexible Space Structures for Passive Vibration Suppression, PhD Thesis, Southampton University, 2000.

[15] Wason JL, Cox SJ, Keane AJ. Flexible knowledge repositories for problem solving environments. In: Allan R, Kleese K, editors. Proc Int Workshop Adv Data Storage/Mgmt Tech High Perform Comput. Daresbury: CLRC, 2000. p. 199–205. ISSN 1362-0223.

[16] LSF load sharing software, http://www.platform.com/.

[17] Domeshek EA, Kolodner JL. The design of a tool kit for case-based design aids. In: Gero JS, editor. Proc. of Artificial Intelligence in Design '94. Netherlands: Kluwer Academic, 1994.

[18] Jameson A. Re-engineering the design process through computation. J Aircraft 1999;36(1):36–50.