Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Future Generation Computer Systems 19 (2003) 1299–1307

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/future

The synergistic integration of mathematics, software engineering, and user-centred design: exploring new trends in education

C. Angelov, R.V.N. Melnik*, J. Buur

University of Southern Denmark, Mads Clausen Institute, Grundtvigs Alle 150, Sonderborg DK-6400, Denmark

Abstract

There is an increasing recognition in the society that interdisciplinary challenges must be part of new educational practices. In this paper, we describe the key curriculum activities at the University of Southern Denmark that combine mathematical modelling, software engineering, and user-centred design courses. These three disciplines represent a core of our graduate program, aiming at educating the professionals that will be capable of not only using but also further developing new technologies, and therefore, will be capable of fostering further the progress in computational science and engineering. Finally, we show how the learning environment, with emphases on broadening the student experience by industrial links, affects the student career aspiration.

Keywords: Education; Computational science and engineering; Pervasive computing and mechatronics

1. Introduction

Computational science and engineering (CSE) has evolved into a discipline in its own right and its role in the society has continued to increase. Numerical simulation plays a key role in, and is an intrinsic part of CSE since it allows us to facilitate scientific discoveries and to make the process of engineering design more efficient than has ever been possible. However, designing technology and creating new products that would have an impact in years ahead is not about just performing computations based on the existing models or the models that are currently being developed. Rather, it is close continuous interactions in *the process of product design and technology development* between applied mathematicians, mechanical and electronic engineers, software and usability engineers, and hardware specialists that will make a difference in years ahead.

We see CSE as the synergistic integration of mathematical models, sciences, mechanical, electrical, and computer systems with human expertise and experience. The current economic forces and the market-driven university environment are not the same as they were several decades ago. They might not be very favourable in attracting the best minds in mathematics, physics and other core science curricula. However, it is clear that as the boundaries between the sciences and engineering become increasingly blurred, an increasing number of very strong students will look for a *multidisciplinary education*. Moreover, there is evidence to suggest that educational programs with an engineering design component will become increasingly popular [2]. Being an integral part of CSE education, the program we discuss in this paper is different from what has been already developed at other institutions of which we are aware (e.g. [9]). The difference

* Corresponding author. Tel.: +45-6550-1681; fax: +45-6550-1610.
E-mail address: rmelnik@mci.sdu.dk (R.V.N. Melnik).

0167-739X/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-739X(03)00088-8

is concerned primarily with the fact that the curriculum at the Mads Clausen Institute (MCI), University of Southern Denmark is developed with the major focus on teaching students how to apply mathematical models in conjunction with tools and methodologies of mechanical, electronic and computer engineering to *the design of industrial products*.

The MCI at the University of Southern Denmark is a centre for product innovation with research and development expertise in the areas of mathematical modelling of physical systems, in particular modelling of electromechanical systems and computational electronics, software design, and user-centred design (UCD). The MCI has a strong tradition of links with manufacturers of mechatronics devices and in applying mathematical modelling, software engineering, and UCD tools during the process of product development. In particular, we have close links with the largest industrial concern in Denmark, Danfoss.

Since CSE education is traditionally based on problem solving methodologies, it is generally agreed (e.g. [9]) that (a) CSE differs from classical education in mathematics and/or computer science because CSE by definition and by tools is based on collaborative efforts of people working in different disciplines, and (b) CSE differs from an often-used approach in engineering education where students learn how to use "canned" codes because CSE by definition and by tools is based on creativity. Nowadays CSE is considered as an important partner to theory and experimentation in the set of tools that advance science and engineering [5,10]. It is evident that higher education has to respond to an increasing demand in training people who are able to effectively use integrated knowledge and methodologies coming from a fusion of several different disciplines pertinent to CSE. We believe that a core subset of these disciplines should include mathematical modelling, software engineering, and UCD in a graduate program aiming at educating the professionals that will be capable of not only using but also further developing new technologies, and therefore, will be capable of further fostering the progress in CSE. At the MCI, University of Southern Denmark we strive to integrate the above three areas, and in this paper we show that the intended fusion is possible, and can bring along important benefits to students, not achievable otherwise.

2. Challenges of CSE education in the era of pervasive computing and mechatronics

The basic structure of the relationship between CSE and the other disciplines is typically plotted as an intersection set between three major areas, applied mathematics, engineering/science, and computer science (e.g. [9]). In discussing educational issues in CSE, it is important to realise that the resulting set is impressively large, its growth is stimulated from each of the three disciplines mentioned, and each educational program can only cover a subset of what is represented by that intersection. There are, however, some basic *technological* trends which, we believe, are worthwhile discussing here since they will influence dramatically further enrichment of the whole community of CSE.

Recent years have witnessed an ever-increasing use of embedded systems and the advent of the so-called post-PC era. The latter can be characterised as the era of pervasive computing and pervasive mechatronics in which industry, infrastructure and everyday life will be dependent on the operation of billions of embedded control systems and components such as dedicated controllers, intelligent sensors and actuators, etc. [12]. The widespread use of embedded systems poses a serious challenge to software developers in view of diverse, severe and conflicting requirements, e.g. reduced costs and shorter time to market, software safety and predictability, support for in-site and on-line reconfiguration, etc.

The above requirements cannot be met by currently used technology, which is largely based on ad-hoc software design and manual coding techniques. Moreover, current practices are often considered to be some kind of art rather than computational and engineering science, or even sort of "medieval exercises in alchemy" [15].

The above situation is largely due to the fact that embedded systems have been ignored in the research and educational programmes of computer science departments [7], and now, when their importance is becoming obvious, there is lack of vision as to how software should be developed for this class of system. Instead, we are witnessing attempts to adapt existing (informal) software design methods for the use in the real-time embedded world. However, these methods have their origins in conventional computer science and related

human experience, as they have been originally developed for information processing systems, whereas *embedded systems are predominantly real-time control systems* that are closely coupled to the machines and processes being controlled [12]. This requires a truly engineering approach for embedded software development that is based on control engineering and systems science concepts, rather than on conventional software design methods. In particular, it is necessary to develop formal methods for embedded control systems analysis and design that will adequately specify systems structure and behaviour for a broad range of real-world applications. This will help create industrial software technology for embedded systems featuring formal models (frameworks) and reusable components. Accordingly, component-based design must be properly introduced in the curriculum, which implies close cooperation with other areas such as mathematical modelling and UCD.

When focusing on how users interact with embedded systems, there are similar conflicts with traditional Human–Computer Interaction theory and methods, which were developed primarily for designing user interfaces of desktop PCs and large information processing systems. The primary purpose of interacting with embedded systems is control and monitoring. Users, e.g. process operators in industrial plants that incorporate numerous intelligent sensors, actuators and dedicated controllers, foremost interact with the technology to make things happen. An information processing view of this interaction can only partly capture the work practice of such users. It is crucial for operators to maintain a physical sense of the processes, something they cannot obtain from the purely visual cues of a control room monitor. On the other hand, there is a major component of human reasoning in most plant operation. Intelligent components are potentially unstable, and control algorithms are often not flexible enough if they are designed for "normal" running conditions, a state that very seldom exists. Rather than planning to replace human operators, the sound way of thinking in design is to support human work practice.

Another challenge for the UCD field brought about by the advent of pervasive computing is to rethink theory and methods in order to cope with communities of users interacting with *networks* of products, rather than single users interacting with single products. In a world of numerous microprocessors surrounding us, it is unrealistic to expect that all should be operated through displays and keypads. Entirely new methods of interacting with *the web of technology* must be developed for the future of pervasive mechatronics.

The above observations have given us the insight needed to build the educational program to reflect the trends of the ongoing evolution of the engineering design paradigm in order to educate professionals that will be capable of further developing and using the future technologies.

3. Core curriculum activities and their integration at the MCI, University of Southern Denmark

As the historical divisions between the various branches of science and engineering become less clearly defined [4], CSE can provide a roadmap for educating new professionals capable of fostering further the technological progress. The challenges described in the previous section led us to believe that the core structure of our graduate program should be based on mathematical modelling courses, courses in software engineering, and UCD courses. This combination reflects the importance of integrated modelling, design, and control in the industrial setting as well as the fruitful ground of challenging industrial mathematics and computational sciences problems upon which the educational program is being developed. All courses, while having their individual goals, interact between themselves, and this interaction is represented schematically in Fig. 1. In the present curriculum, all our students meet interaction between the research fields in at least three courses (total credit 16 ECTS) before their graduate thesis. In particular, Embedded Software Engineering 2 allows students to prototype user interface concepts developed in the UCD course; Advanced Control Theory constitutes a meeting point between mathematical modelling and embedded software; and the Mechatronic Project allows the students to work experimentally across two or all three disciplines. The distribution of credits between the three fields in the core curriculum activities is presented in Fig. 2, and includes Mathematical Modelling (28 ECTS), Embedded Software Engineering (16 ECTS), and UCD (12 ECTS).

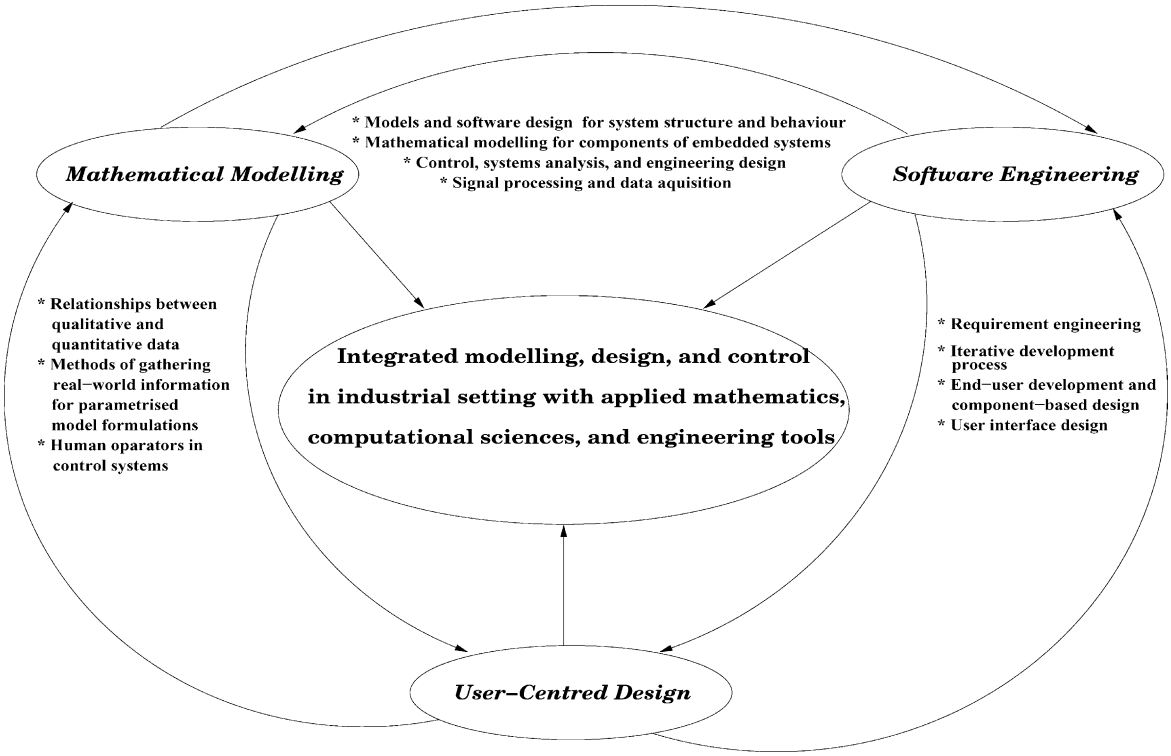


Fig. 1. Interactions between three core curriculum activities.

In what follows we describe in some detail the principles upon which the links between the core areas of our curriculum are being built.

3.1. Mathematical modelling and software engineering

Technology is a key element in educating students in CSE. That is why many projects in the mathematical modelling courses focus on the key technological areas such as mathematical modelling of engineering systems, in particular electromechanical and thermoelectric devices, and modelling for microelectronics and nanotechnological applications.

A traditional link of mathematics courses with just mechanical and control engineering courses is not sufficient even for the current industry demands because an engineer has to deal with systems whose principles of operation are based on various *coupled effects* as it is the case for electromechanical systems and/or mechanical systems enhanced by electronic components. Moreover, what is termed now as the mechatronic design is far more than just a pure combination of such fields as mathematics and control, mechanical, electronic, and software engineering. It is a *process* which is a natural reflection of the society's needs and industry demands, and this process will persist into the future. In this process of mechatronic design, mathematical modelling plays a primary role. The description of physical systems with mathematical modelling tools is a key prerequisite to the optimisation and control stage where we have to optimise a design solution across such diverse fields as electronic, software, and mechanical design. Therefore, in training the mechatronic engineers, mathematical modellers, and computational scientists it is important to realise that we cannot rely any longer on a yesterday's view, where major emphases have been put on just the controller optimisation. The courses should be consistently structured with a broader view in mind where the optimisation of the system as a whole is being sought (see Fig. 3). In this sense, we view the mathematical

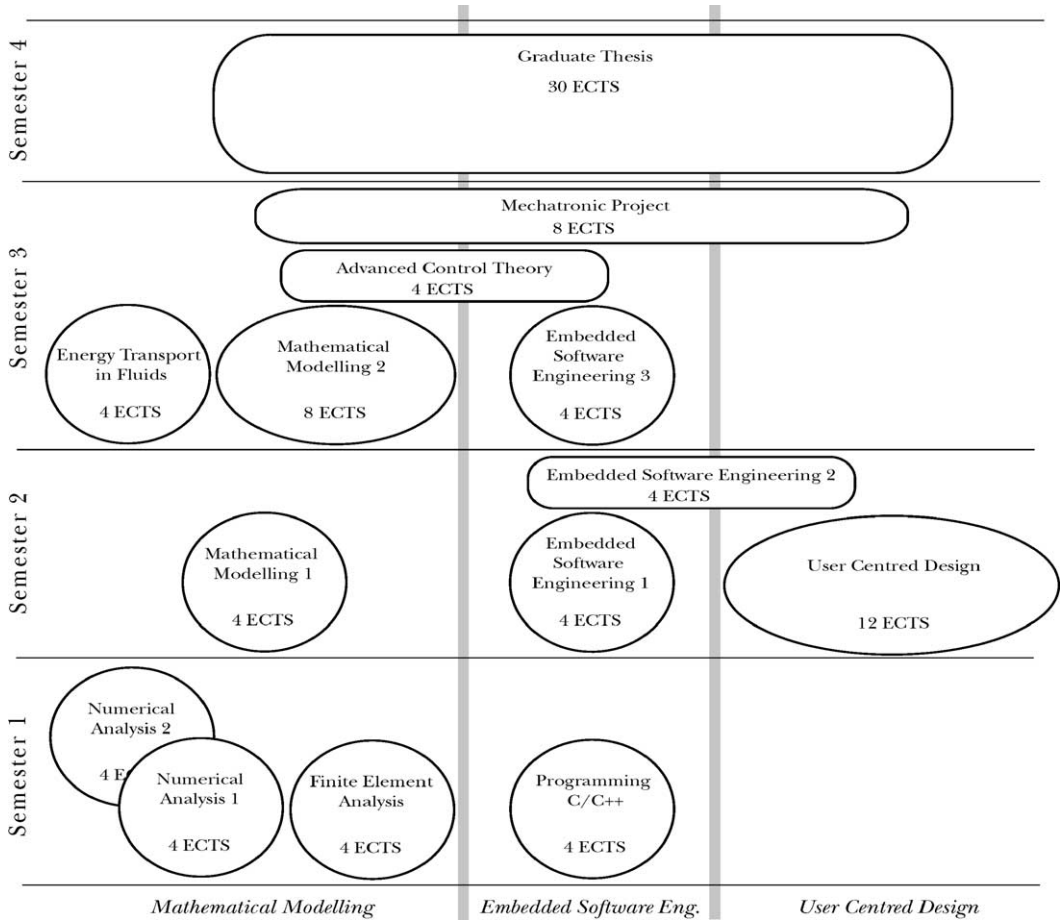


Fig. 2. Curriculum overview showing core courses within each discipline.

modelling courses as the fundamental background for understanding the entire design process. A set of mathematical modelling courses (see Fig. 1) is designed with the hands-on approach in mind to help the students understand how mathematical models can provide a deeper insight into the system behaviour (which on many occasions otherwise cannot be obtained due to the system complexity), how such models can be tested, and how they can provide a hierarchical structure in the entire design process allowing to treat concurrently separate components of the system and to estimate the influence of coupling effects. These courses are run in parallel with software engineering courses.

The goal of the MCI programme in Embedded Software Engineering is to provide students with necessary knowledge about relevant software engineering concepts (taking into account challenges outlined in Section 2), and to give them hands-on experience in two main areas: embedded systems software architecture (principles of operation and software modelling techniques) and advanced software technology for embedded systems. The first area presents the theoretical background of modern Software Engineering for embedded systems. Its purpose is to develop a systematic approach to software development for embedded real-time applications, using formal (mathematical) models specifying system structure and behaviour. This is accomplished by introducing modelling techniques of gradually increasing complexity, starting with basic task-level techniques such as state machines and signal flow

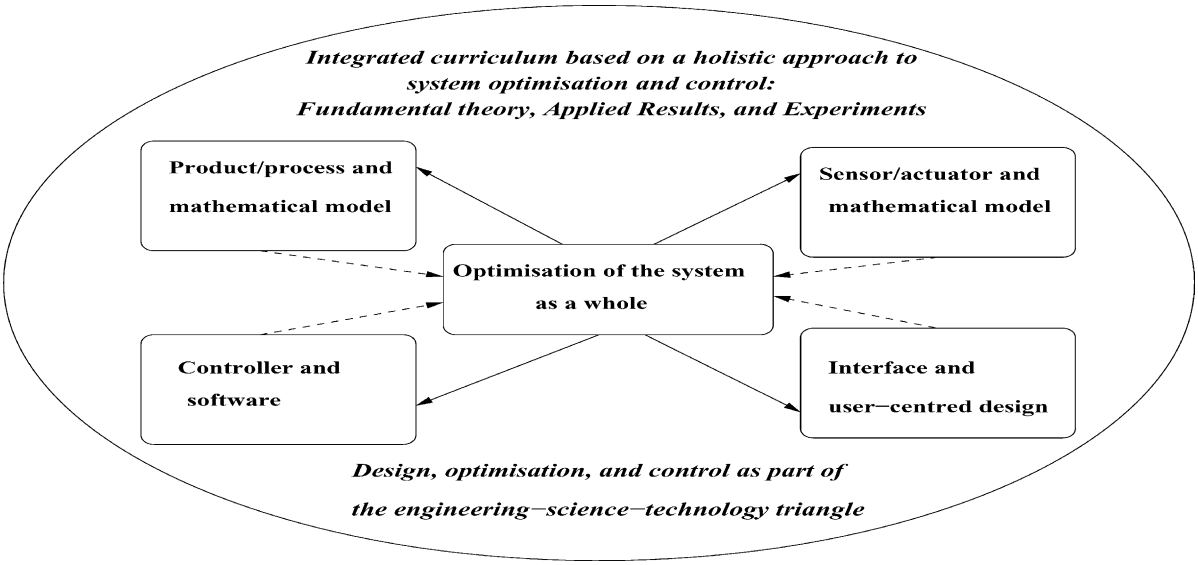


Fig. 3. The design and optimisation of novel industrial mechatronic products in integrating mathematical modelling, software engineering, and UCD disciplines in the MCI curriculum at the University of Southern Denmark.

graphs for sequential, continuous and hybrid control systems. Implementation aspects are also discussed, emphasising high-level (programming-in-the-large) techniques and reusable components within conventional as well as advanced software development environments (e.g. VisualSTATE). Basic modelling techniques can be used to efficiently implement various types of embedded systems of low and medium complexity. However, they provide the foundation for complex system modelling techniques that can be used to fully specify all aspects of real-time systems such as system structure, sub-system and component behaviour and interactions. The second subject area builds on the theoretical foundation, which is provided by the preceding one, and proceeds with a detailed discussion of the implementation aspects of embedded systems. Here, the emphasis is placed on high-level solutions for complex embedded applications operating in a predictable system environment. The discussion outlines the architectural concepts and principles of operation of hard real-time systems, i.e. task and time management, event management, task interaction (synchronisation and communication) in the context of stand-alone and distributed embedded systems. It highlights advanced techniques and mathematical analysis methods for predictable real-time systems, e.g. rate monotonic scheduling theory. Theoretical and implementation aspects discussed in the above two areas are presented in the basic Software Engineering courses (Embedded Software Engineering 1–3), and they are further generalised in an elective course on distributed embedded systems.

3.2. Software engineering and UCD

It was earlier pointed out that existing software design methods are inadequate in the era of pervasive embedded systems. Therefore, a new approach towards software development is needed, i.e. component-based design which will allow for industrial production of software for embedded applications. Industrial software production methods are currently in the focus of major international programmes (e.g. [12]). Research is also going on at the MCI that is aimed at finding feasible solutions to the above problem. Guidelines formulated in Section 2 helped us also in developing *a prototype framework* and the associated method for component-based design of embedded software [1]. Certain aspects of this method have already been introduced in the Software Engineering courses, in particular in Embedded Software Engineering I. Component-based design of embedded

software requires an interdisciplinary research and development effort, including both mathematical modelling and UCD. Specifically, it emphasises end-user development by delegating design or programming capabilities to the users through easy to use software development mechanisms supporting software composability, scalability and reconfiguration. Implementation of transparent, user friendly interfaces is another aspect that will be possible through component-based design of software. That is a major meeting point between software engineering and UCD courses.

In more general terms, UCD activities aim at understanding users, establishing user collaboration, designing user interaction, and organising iterative design processes. UCD borrows from such diverse areas as ethnography, cognitive psychology, industrial design, drama, and the primary goal is to ensure that products are designed in such a way that they fit into the work practice and expectations of the people using them. It is the process of formulating design problems and establishing requirements for a design solution that requires a link between Software Engineering and UCD areas. In the traditional view of software "production", requirements are seen as the contract between the organisation in need of a system and the organisation developing the system. It is assumed that the customer organisation is able to envisage and specify the solution it needs, and that the development organisation is able to work against a full set of requirements. However, it has been shown that an understanding of the design process as an ongoing framing and re-framing of the problem is much more beneficial [11]. Requirements emerge not through an analytic activity preceding design, but throughout the initial stages of design by creating experimental design concepts, imposing them on the situation, and listening to the "backtalk" of the situation, e.g. the reaction of users, customers, stakeholders. The UCD courses at the MCI support this view of design by engaging students in projects with open problem settings in close collaboration with industry. Students learn to engage users in an ongoing dialog throughout the design process in a Scandinavian tradition of user participation [6]. They learn that iterative design cycles are imperative for achieving designs that correspond to human needs, and that these cycles can be investigated through user activities like user visits and user workshops [3]. In this process, the relationship between user field studies and requirements engineering need further attention. The UCD courses teach ethnographically inspired techniques for studying users in their work context [13]. Through exercises and project work, the students acquire the ability to reduce the richness of the actual situation to the strict formalia of, e.g. use-case descriptions for software development.

3.3. Mathematical modelling and UCD

We believe that in educating top professionals in CSE the understanding of the role of human activities in complex technological systems is imperative. The traditional systems engineering view of "the human in the control loop", in which the human operator is regarded as an element comparable to the technical subsystems of the control system, must be abandoned in favour of a broader understanding leading to supervisory control concepts [8]. Humans act upon the clues of the situation rather than in a pre-planned way [14]. Studies of operators in, for instance, industrial plants show how humans act as supervisors of control systems, ensuring that the plant keeps running when situations occur that were not foreseen at the time of designing the system. Control systems need to be designed in a way that they support human operators in doing their work even in the situations where conditions change beyond the foreseeable. In this context, the UCD courses attempt to question the imperative of traditional engineering that technology can be designed to perform perfectly, even in an imperfect world.

In addition to supervisory control, another important meeting point between the mathematical modelling and UCD curricula is the methods of gathering information for the design process—the relationship between qualitative and quantitative data. Both are required, both are gathered through investigations and observations, but they play different roles. The qualitative information serves to propel the creative process of generating ideas, while the quantitative information serves to establish "boundary" conditions for the models the designers need to establish for the phenomenon in question. In our education, we emphasise both computational power of modern computers and the user's creativity in using computers and designing new products.

A new kind of CSE professionals should combine strong scientific/engineering background, knowledge

of technology, and a strong ethical and aesthetic sense in terms of product development. In a traditional setting, engineers are confined with design types where all the major parameters are already fixed and the requirements are set, either on the basis of financial reasons or due to using prior simplified evaluations. It is often beneficial to take a different point of view since the already imposed constrains might prohibit the optimisation of the system as a whole and might omit some interesting design options.

4. Benefits of the integrated approach and student career aspiration

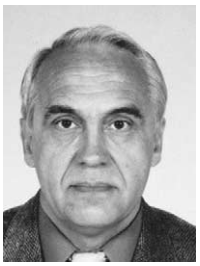
The subject of CSE encompasses many diverse areas. Therefore, it is important to create such a learning environment that would help our students to understand that solutions to many problems in CSE can be found in different domains, and only close collaborative links between experts in different fields, working as a team, can lead to the overall success. It is no longer appropriate to think of the engineering design as an optimisation problem requiring to find optimal controller parameters for a system allowing optimisation of some of its performance indexes [2]. We have to consider such a system as a whole, be able to integrate all the partial, typically sub-optimal, solutions obtained at different stages of the design process, and to take advantage of the features realised at these stages. This view on the design and optimisation leads to the integration of mathematical modelling, software engineering, and UCD courses in the MCI curriculum as a natural step, see Figs. 1 and 3.

Integration of the three major areas within the MCI is supported by an advanced learning environment stimulating interdisciplinary research and education. Within the MCI, we have all necessary infrastructure to develop and integrate challenging student projects, including a broad range of computing facilities and one of the largest cluster supercomputers in Europe. We believe that this type of integrated learning environment will strongly stimulate multidisciplinary education through hands-on experience and teamwork that will make it possible to cultivate a modern approach to solving complex engineering problems. The MCI runs both research and teaching activities in close collaboration with local industry. The recent 41st European Study Group with Industry (ESGI) meeting took place at the University of Southern Denmark and the MCI took an active part in its activities. The MCI will be the host of a 2003 ESGI meeting. Six industrial employees from Danfoss are co-located with university staff in the open office space. This provides an informal opportunity for researchers and our graduate students to join industrial product development activities (on a consultancy basis) to experiment with new technologies and methods, and for industrialists to join teaching to disseminate industrial practice. Naturally, students benefit from this mixed research/practice environment at the MCI, and become better adapted to industrial practice. Not only the MCI itself is sponsored by Danfoss. Each year, 25 students are awarded industrial scholarships from a range of mechatronic companies. This ensures an active interaction with about 30 medium and large sized industries through semester projects and graduate projects. Working in such an environment should help our students create a new vision of the design process emphasising computer-aided design and integration of all aspects of the design process—from UCD, through mathematical modelling and simulation, to embedded systems and real-time software engineering. On the other hand, students will develop practical skills and in-depth knowledge of modern computer technology in a broad range of activities and areas of applications. The combination of these major advantages should help in assisting our students in their career aspirations and will ultimately make them strongly competitive on the labour market.

References

[1] C.K. Angelov, I.E. Ivanov, A. Burns, HARTEX—a safe real-time kernel for distributed computer control systems, Softw.: Pract. Exper. 32 (2002) 209–232.
[2] R.H. Bishop (Ed.), The Mechatronics Handbook, CRC Press, Boca Raton, FL, 2002.
[3] J. Buur, K. Bagger, Product design based on user dialogue: replacing usability testing, Commun. ACM 42 (1999) 4.
[4] A.R. Damasio, et al., Unity of Knowledge, The Convergence of Natural and Human Science, The New York Academy of Sciences, New York, 2001.
[5] G.C. Fox, From computational science to Internetics: integration of science with computer science, Math. Comput. Simul. 54 (2000) 295–306.
[6] J. Greenbaum, M. Kyng, Design at Work Cooperative Design of Computer Systems, Lawrence Erlbaum, London, 1991.

[7] E.A. Lee, What's ahead for embedded software? IEEE Comput. 9 (2000) 18–26.
[8] R.V.N. Melnik, On consistent regularities of control and value functions, Numer. Func. Anal. Optim. 18 (1997) 401–426.
[9] L. Petzold, et al., Graduate education in computational science and engineering, SIAM Rev. 43 (2001) 163–177.
[10] B. Schnabel, Education, SIAM Rev. 43 (2001) 161.
[11] D.A. Schön, The Reflective Practitioner, Basic Books, New York, 1983.
[12] Software technologies, embedded systems and distributed systems in FP6, in: An opening presentation to the Workshop on Software Technologies, Embedded Systems and Distributed Systems in the Sixth Framework Programme for EU Research, Brussels, Belgium, May 2, 2002.
[13] W. Sperschneider, K. Bagger, Ethnographic Fieldwork under Industrial Constraints: Towards Design-in-Context, NordiCHI 2000 Proceedings, STIMDI, Stockholm, 2000.
[14] L. Suchman, Plans and Situated Actions: The Problem of Human–Machine Communication, Cambridge University Press, New York, 1987.
[15] K. Rubin, Is real-time ready for prime-time? A white paper on what is missing from real-time development tools, Real-time Mag. 3 (1999) 13–19.



Christo Angelov received his MSc and PhD degrees in Computer Engineering in 1971 and 1984, respectively, from the Technical University of Sofia (TUS). He was associate professor of Computer Control Systems in the Department of Systems and Control, and Head of the Advanced Control Systems Laboratory, Faculty of Automation. Over the years, Christo Angelov has been involved in teaching and research in the area of computer control systems, with a focus on systems architecture, operational and application software for industrial computer systems. He has managed research projects and has published a number of papers and patents in that area. In 2001 he joined the Mads Clausen Institute for Product Innovation, University of Southern Denmark, as a professor of Software Engineering.

Roderick Melnik is Full Professor in Mathematical Modelling at the Mads Clausen Institute, University of Southern Denmark (USD). He received his MSc degree in Applied Mathematics and PhD degree in Computational Mathematics from Kiev State University in 1985 and 1989, respectively. Since 1989 he held academic tenures in Europe and Australia. Before joining USD in 2000, Dr Melnik held the position of senior mathematician at the Commonwealth Scientific and Industrial Research Organisation in Sydney, Australia. Professor Melnik has published extensively in the field of applied mathematics and mathematical modelling in science and engineering with his colleagues from Europe, USA, and Australia.

Jacob Buur graduated with a MSc in Mechatronic Engineering from the Technical University of Denmark in 1984. He spent two years in Japan studying product development practice in industry and completed his PhD on design methods for Mechatronics. After 10 years as a manager of the User Centred Design Group of Danfoss A/S, he joined the Mads Clausen Institute for Product Innovation, University of Southern Denmark, as a professor in 2000.