The Role of Application Domains in Informatics Curricula

Tony Cowling

Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield, S1 4DP, England, A.Cowling@dcs.shef.ac.uk

The basic proposition of this paper is that undergraduate degree programmes in informatics will benefit from incorporating coverage of typical application systems. The paper begins by presenting arguments to support this proposition, and then goes on to examine the impact of it on curricula. It presents two classifications for the application domains from which such systems can be drawn, one derived from the ACM/IEEE model curriculum for software engineering and the other derived from the application of general systems theory to applications systems. The paper then uses these classifications, firstly to structure the material relating to application domains that needs to be studied, and secondly to guide the choice between different application domains. Finally, given the amount of material relating to an application domain that may need to be incorporated into a curriculum, the paper discusses the learning outcomes that are appropriate for such material and their relationship to the requirements of qualifications frameworks, such as the "Dublin descriptors" created within the Bologna process.

Keywords
Application systems, CS curricula, software development methods, systems requirements.

1. Introduction

Providers of undergraduate degree programmes in informatics face many challenges, including declining demand from potential students for such programmes, poor motivation or study skills in many of the actual students on them, and concerns from employers about how well graduates from such programmes are equipped to function effectively in the computing industry. These challenges apply to programmes in all of the disciplines that make up the field of computing, or informatics, viz computer science (CS), computer engineering (CE), software engineering (SE), information systems (IS) and information technology (IT) [1]. Since CS has historically been the dominant one of these disciplines, however, these challenges apply to it with particular force, and indeed they have led some to suggest that undergraduate degree programmes in CS should now be regarded as obsolete [2].

This is an extreme view, but it raises the question of what elements of informatics might need to be added to the conventional core of CS to meet some of these challenges. One way to answer this is to analyse each of the discipline-specific volumes of Computing Curricula 2001, to see what kinds of material they suggest needs to be added to this CS core. By comparison with the volume for CS [3] (CS2001 from now on), the one for SE [4] (SE2004) primarily emphasises software development processes, and within these the methods for ensuring that application systems will meet customer requirements, and for designing architectures for these systems that will achieve these goals. Similarly, the volume for CE [5] (CE2004) also focuses on development processes, but in the application areas of hardware and embedded systems, and so it emphasises too the economic aspects of producing such systems, and the co-design of hardware and software. Application systems are naturally one of the key concerns of the volume for IS [6] (IS2002) too, but here the main focus is on how such systems must support business processes, and how they must fit into the wider context of business and organisational structures. Finally, the volume for IT [7] (IT2005) also emphasises application systems, but from the perspective of how the users of these systems should be supported in acquiring, installing, configuring and actually using them.

Synthesising these various emphases identifies two areas where it appears that informatics curricula generally need to pay more attention, by comparison with the traditional concern for core CS. One area is that of application systems and how they relate to customer needs, and the other area is that of the processes of actually developing systems. The latter appears as a knowledge area in CS2001, but this was defined at a very early stage in the development of SE2004, and so does not provide a particularly well-structured treatment of this topic area, which is discussed in much more depth in SE2004.

By contrast, the area of application systems has received much less attention. This paper therefore explores the roles that this aspect should play within the informatics curriculum, in terms of what students ought to know about application systems, and what skills relating to them they should be expected to develop. For this purpose, of course, individual application systems are important primarily as examples of the application domains within which they exist, and so the paper is mainly concerned with these domains rather than with individual systems, even though it is the systems that the students will actually be directly concerned with during their studies. This exploration is therefore developed in three stages, beginning in the next section by developing a classification of application domains. Then, section 3 considers the kind of material that would need to be included in the curriculum in order to provide a proper basis for incorporating some study of application systems. Since these needs obviously create pressures on the curriculum, section 4 discusses the principles that can be applied in balancing such pressures. Finally, section 5 summarises the conclusions of the paper and outlines possible further work.

2. Application Domains

As the above analysis of the CC2001 models has indicated, the concept of application domains within the curriculum is not new, since IS and IT are essentially concerned with the domain of business systems for large organisations, while the CS2001 model has as one of its knowledge areas *Net-Centric Computing*, which is essentially an application domain. Also, degree programmes that focus on the domain of computer games systems are now well-established, and more recently programmes have been created that focus on forensic computing [8], and grid computing is emerging as another application area [9].

What is needed is a systematic treatment of application domains, and the best place to begin this is with the SE2004 model. During the development of the body of knowledge for this model a number of topics were identified that appeared to be important, but which did not fit particularly well with the main structure for the body of knowledge. It was identified that a common feature of these topics was that they were related to the needs of different kinds of application systems, and so they were accommodated in the model by creating a knowledge area called *Systems and Application Specialties*. This identified 15 possible domains (with one knowledge unit and typically about 3 topics for each), and the model then specified that at least one of these domains (ie knowledge units) should be studied as part of any undergraduate degree programme in SE.

Given the process by which they were initially identified, it is not surprising that these 15 domains form a rather eclectic set. If the relationships between them are analysed, however, it becomes apparent that they can be roughly classified into three categories. One category arises from the purposes of the applications; a second arises from specific technologies that are distinctive of those applications; and the third category arises from specific properties that are required for the systems. Of course, there are some overlaps, since in some cases the purposes of the systems depend on particular technologies, or particular properties are important to these purposes, and indeed one of the domains comes in all three categories. This classification is presented in table 1, but omitting the codes used in the model to identify the knowledge units.

Table 1 Categories of application domains.

Category	**Application Domains**
Purposes of the systems	information systems and data processing, financial and e-commerce systems, embedded and real-time systems, bio-medical systems, scientific systems, telecommunications systems, avionics and vehicular systems, industrial process control systems, multimedia, game and entertainment systems, systems for small and mobile platforms.
Technologies needed for the systems	network-centric systems, embedded and real-time systems, telecommunications systems, industrial process control systems, multimedia, game and entertainment systems, systems for small and mobile platforms, agent-based systems.
Properties required for the systems	fault-tolerant and survivable systems, highly secure systems, safety-critical systems, avionics and vehicular systems, systems for small and mobile platforms.

From this table there are five key features of application domains that can be derived, and the significance of the overlaps in this table is that actually each of these features will apply to all domains, and not just to the ones for which the features are most characteristic, as identified in the table.

2.1. Features of Application Domains

The first feature is that an application domain will reflect particular purposes within kinds of organisations (eg accounting, data mining, entertainment, etc). The second is related to this, and it is that an application domain will reflect the kind of organisation (from individuals through to society at large) within which these purposes will arise, and hence within which the application systems will be used. The third feature is that application domains may depend on particular technologies (eg networking, databases, visualisation, etc), to such an extent that limitations or features of the technologies may significantly affect applications within the domain, although not all domains will have such dependencies. The fourth feature is that the context for an application domain (ie the combination of the organisations and their purposes) may require particular properties for application systems (eg information security, safety, fault-tolerance). The fifth feature is that, in order to achieve such properties for systems, and application domain may require the use of either particular specific product technologies (such as specialised hardware or software architectures) or particular process methods (eg fault tree analysis, formal verification, timing analysis, etc).

These five features provide a more systematic basis than the three categories in table 1 for classifying application domains, in the sense of trying to answer the question as to whether two application systems belong to the same domain. From this perspective on classification it appears that the purposes of the systems are the most important feature of a domain, so that they could almost be regarded as the key that identifies a domain. Of course, to some extent the purposes depend on the kinds of organisations that give rise to them, but this is not a functional dependency in either direction. Thus, any organisation can have a number of purposes, and very similar purposes can arise in different kinds of organisations: for instance, both large companies and small amateur organisations may need accounting systems, and the basic purpose will be the same for both, even though the requirements of scale may be very different. Hence, the key that identifies a domain does need to be the combination of its purposes and the kind of organisation from which these arise, and we will refer to this combination as a business class.

The other three features then all depend on this key, and so actually identify groups of domains rather than individual ones. Thus the characteristic of depending heavily on a particular technology, and perhaps pushing it to its limits, will typically cover a number of different purposes. For instance, the category of telecommunications systems covers a huge variety of applications, depending on what kind and quantity of data is to be communicated and between whom. Similarly, the characteristic of requiring particular properties, which again usually means having to put particular emphasis on these properties (for instance "highly secure systems", as compared with those that just need to be ordinarily secure) can apply to a whole range of different purposes, and hence domains. Consequently the same is also true of the need to apply particular process or product technologies, since these needs result directly from either the characteristics of a particular technology or a requirement to achieve particular properties.

The effect of these dependencies is to create a four stage process, where each stage gives rise to different groups of application domains. The first stage is that many business classes give rise to similar kinds of general requirements for systems, and so these general kinds of requirements form groups that we will call business domains. Thus, domains from table 1 such as information systems and data processing, or financial and e-commerce systems, are business domains in this sense, since they describe groups of application domains that have broadly similar purposes. Indeed, in some respects these groups may be too large, as for instance the business domain of data processing covers a wide variety of purposes, but they are important because they identify the common features of the application domains in them.

The second stage is that the specific properties which may be needed for a system, such as security, safety or real-time performance, then each define a group of application domains for which those properties are important. Thus, some of these groups, which we might call the property domains, are defined directly by domains from table 1, such as highly secure systems. By contrast, other properties may be significant for several of the property domains listed there, such as real-time performance, which is just as important for avionics and vehicular systems, or for industrial process control systems, as it is for the group called embedded and real-time systems.

The third stage is that the achievement of these properties may be constrained by particular technological features, and so these features give rise to another group of domains, which we may call the technology domains. For instance, several of the domains listed in table 1 are constrained by different aspects of networking and communications protocols, such as network-centric systems, telecommunications systems and systems for small and mobile platforms.

The final stage is that particular technologies and process methods (mainly the latter) are needed to overcome these limitations, or to achieve the required properties. This is the one stage in the process that does not give rise to a different grouping of domains, since typically these process methods (such as failure analysis or timing analysis) do depend just on the property or constraint at which they are aimed. Consequently, the use of them corresponds directly to what we are calling here the property and technology domains.

2.2. General and Specialised Domains

There is, though, a problem with trying to use business classes as the key to identifying individual application domains, and this is that there are a lot of them. In particular, within the EU the kinds of business organisations that can exist are described by the *Nomenclature générale des activités économiques dans les Communautés européennes*, or NACE [10]. This uses codes with a hierarchical structure of 4 digits to identify the different classes of industrial or commercial activity, and the newest version of it (Rev. 2.0) defines 615 classes. Even if one just takes the first two digits of the code, which form the top level of the hierarchy, there are 88 divisions (including one for *Activities of extra-territorial organisations and bodies*!), which are then grouped into 21 sections, although some of these are very broad. For instance, the section for transportation and storage covers land transport (road, rail and pipelines), water transport, air transport, warehousing, cargo handling and postal and courier services. Thus, for identifying the purposes of systems within application domains, one would need to go into more detail than just these 21 sections to obtain a suitable set of business classes.

Since Denning has pointed out [11] that the 30 core technologies of computing that are identified in the various CC2001 volumes are too many to form a basis for an effective model of curriculum structures, it is fairly obvious that 20+ business classes would represent too many different application domains, if they had to be considered individually in such models. Indeed, for such situations the number of application domains, or separate groups of them, should ideally not be more than the 15 listed in table 1, and fewer would be preferable. Since groups of application domains will, however, need to be treated individually within curriculum models, because particular topics will be needed to support the study of them, as with the examples in SE2004.

This therefore leads to a distinction between what we will call general application domains and specialised ones. Here the general domains are any where there is a need to study the requirements associated with the corresponding business, but beyond that they are not sufficiently distinctive that they need to be identified separately within curriculum models. For instance, the business domains identified in the SE2004 model mainly just require study of the application areas themselves, and identify relevant basic technologies that may need emphasising, and so arguably they do not really need to be included explicitly in such a model. By contrast, specialised application domains are identified with either property domains or technology domains, since it is these features of the specialised domains that impact on the curriculum models, by creating requirements for particular topics to be included in the curriculum, or studied in greater depth.

2.3. An Alternative Classification of Domains

These aspects of applications systems therefore lead to a second classification of domains, which is derived from general systems theory. This identifies three mutually exclusive categories of applications on the basis of the relationship between software systems and their context, where an application system and its context are viewed together as an information system. Each of these categories then corresponds to a set of related domains. One category consists of those applications that are self-contained, in that their contexts are effectively closed systems, as with embedded systems in discrete products. The second category consists of applications where the contexts are self-contained but open, meaning that they may also provide interfaces to enable their users to make connections with other related applications. Thus, almost any ordinary application system that is intended primarily to interact directly with a single user would come in this category, so that it will cover a variety of the domains described above, and particularly game systems and those for small and mobile platforms. The third category then consists of those systems where the context is open, because it has to form part of a larger information system in some organisation, so that the application and this larger system that forms its context must be developed in tandem, at least to some extent.

This information systems approach also suggests a way of integrating some of the features from the SE2004 model into these categories, in that it makes a distinction between those features that apply to the context of the application system and those that apply to the system itself. Thus, the features concerned with the kinds of organisations and their purposes (such as banking, entertainment, scientific modelling, etc) relate primarily to the context of the application systems, as well as to the systems themselves. By contrast, features such as the way in which technology enables the classes of applications (leading to domains like e-commerce, or mobile platforms) relate primarily to the systems themselves. Hence, one could structure these into a two-level hierarchy, where the top-down nature of hierarchical structures would suggest that the top level of the hierarchy should identify the contextual features, namely the organisational purposes, and then the lower level should correspond to the features that relate to the systems themselves, namely the technologies on which they depend.

For the category of embedded systems, this second classification also identifies another feature that distinguishes different domains, namely the economics that underpin the requirements for the application systems. From these three main classes of embedded systems can be identified, based on the cost of the units in which the systems are embedded. One class consists of basic consumer products (eg entertainment systems or “white goods”), where the cost of the embedded system is a significant part of the cost of the whole product. The next class consists of larger products (eg cars or aeroplanes) where the embedded system cost is a much smaller part of the whole. The third class then contains those systems that are larger than single units of a product (eg railway signalling systems). Here, the basic embedded units will be replicated, so that a system will consist of a (possibly large) number of these units, which therefore have to interface to each other, rather than being designed to be independent.

These classifications of application domains have two important consequences. One is that some aspects of application domains are already reflected in the way in which the whole field of informatics is divided up into disciplines. This is most obvious with CE, which is mainly associated with the domains that relate to various forms of embedded systems. Similarly, IS and to some extent IT are mainly associated with the domains that relate to information systems for large organisations, such as data processing, finance, e-commerce, etc. SE by contrast is less specific, since its main focus is on those aspects of developing applications systems that occur once their purposes have been identified, but (as in SE2004) for systems within specialised application domains these aspects do also depend on the domain.

This leaves CS, which is not so clearly associated with any particular application domains. Hence the second consequence applies to it, namely that because application domains in general are defined by the purposes of the systems within them, every application system must belong to some domain. Consequently, every degree programme in informatics that actually requires students to construct some form of software system (which most do, even if only as part of their capstone project) will in practice therefore be associating itself with some general application domain, even if not necessarily with a specialised one. Of course, this domain may not be defined very clearly: indeed, in many CS programmes it may be best described as "computational problems that are interesting to the staff delivering the programme", but it will still be a domain, even if it is not one that the students would regard as very relevant. Hence, the challenge for all curriculum designers in informatics (and particularly for those in CS) is to make explicit this recognition of the related applications domains, and to ensure that they are properly integrated into the programmes.

3. Application Domains in Curriculum Structures

This therefore leads on to the question of how applications domains, whether general or specialised, affect the curriculum and should be integrated into it, and here both of the classifications of application domains described above identify relevant issues. From the first classification, each of its five features describes aspects of a domain that will need to be studied to some extent. Thus, the organisational purposes that give rise to a domain will need to be studied, and this study will need to be set in the context of the kinds of organisations that have these purposes. For general domains some of this study could, of course, be classed as general knowledge of the relevant sectors of commerce or industry, but even so one can not assume that all undergraduates will possess such knowledge. Furthermore, even for a general application domain those characteristics of the business domain that are significant, because they are implicit in the requirements for any application system within that domain, will often involve far more detail than could reasonably be described as general knowledge. This then applies much more strongly to any particular properties for systems that are relevant to a specialised application domain, since both the precise definitions of these properties and the nature of their relationships with the other requirements for applications systems will be quite advanced topics.

These features of an application domain involve a lot of material that is outside the traditional view of computing, but it is necessary in order to understand the other features of a domain that need to be studied. These are concerned with particular technologies on which the domain depends, and with product or process techniques or methods that must be used in developing applications within that domain. These topics are very clearly part of informatics, as well as being related to the application domain, so that in the SE2004 model most of the knowledge units associated with particular specialised domains are defined in terms of additional depth of study of topics such as networking, databases, real-time programming, computer security, etc. The domains themselves are also significant, though, in that the features of organisations, purposes and properties have to be studied as well if students are to understand why the technologies and methods are used in this way within them.

This also applies to general applications domains, even though the characteristics of these domains do not require additional coverage of specific topics within informatics. Given the huge variety of organisations that exist, as described above, the approach that is suggested for structuring this material on organisations and purposes is to base it on abstractions of the underlying business processes. For instance, NACE has a section for wholesale and retail businesses, which covers 3 divisions and 91 classes, but the essential features of all of them are that they require just six basic business processes: maintaining a product catalogue, ordering from suppliers, paying suppliers, handling customer orders, handling payments from customers, and managing the arrival and despatch of items and the associated stock levels. While these processes are not sufficiently distinctive that they need to appear in a curriculum model, it might be useful to have some kind of standard domain model for general application domains. This could complement a curriculum model by describing briefly these and other common business processes and the application domains to which they are relevant, so as to provide support for the selection of the material that is needed to provide adequate coverage of any specific general application domain.

4. Application Domains and Curriculum Time

The other issue that arises from this need to include a lot of material from outside the core of informatics is that it creates pressures on curriculum time, which is a scarce resource and so needs to be allocated carefully. This means that it is not realistic to try to cover a number of application domains, even general ones, and so there is a need to be selective in the choice of domain. As already described, this selection is partly implicit in the division of informatics into different disciplines, and this corresponds roughly to the top level of the hierarchy produced by the second classification. Even within a discipline, though, one needs to be more selective, and so the second level of this hierarchy suggests that this choice should be based on the set of enabling technologies that characterise the chosen domains, and that must therefore be emphasised in the CS component of the curriculum. Then within this the second stage is to identify the relevant business domain (which might include other science or engineering disciplines), and this will determine the non-computing material that must be included in the curriculum in order to cover the organisational needs for that domain, the related economic aspects, and the system properties that follow from these needs.

Furthermore, when this selection has been made the amount of material from the domain that needs to be included will in many cases still be equivalent to a significant fraction of the core computing content of the degree programme. This therefore creates a problem in planning programmes, as to how far the breadth required to achieve adequate coverage of the chosen application domain should be balanced against the depth required within the computing content. This problem of balance has been particularly acute in the UK, where the National Qualifications Framework for England, Wales and Northern Ireland [12] specifies that a graduate with an honours bachelors degree must have demonstrated:

(i) a systematic understanding of key aspects of their field of study, including acquisition of coherent and detailed knowledge, at least some of which is at or informed by, the forefront of defined aspects of a discipline;

(ii) an ability to deploy accurately established techniques of analysis and enquiry within a discipline;

(iii) conceptual understanding that enables the student to devise and sustain arguments, and/or to solve problems, using ideas and techniques, some of which are at the forefront of a discipline; and to describe and comment upon particular aspects of current research, or equivalent advanced scholarship, in the discipline;

(iv) an appreciation of the uncertainty, ambiguity and limits of knowledge; and

(v) the ability to manage their own learning, and to make use of scholarly reviews and primary sources (eg ... appropriate to the discipline).

From this it can be seen that these requirements focus solely on the depth of study that is to be achieved, and make no provision for any compromise that might be necessary to achieve breadth of coverage as well. By contrast, the "Dublin descriptors" [13] for general learning outcomes, which have been developed through the Bologna process, do make at least some allowance for breadth. These require that graduates from programmes in the first cycle:

(i) have demonstrated knowledge and understanding in a field of study that builds upon and supersedes their general secondary education, and is typically at a level that, whilst supported by advanced textbooks, includes some aspects that will be informed by knowledge of the forefront of their field of study;

(ii) can apply their knowledge and understanding in a manner that indicates a professional approach to their work or vocation, and have competences typically demonstrated through devising and sustaining arguments and solving problems within their field of study;

(iii) have the ability to gather and interpret relevant data (usually within their field of study) to inform judgements that include reflection on relevant social, scientific or ethical issues;

(iv) can communicate information, ideas, problems and solutions to both specialist and non-specialist audiences; and

(v) have developed those learning skills that are necessary for them to continue to undertake further study with a high degree of autonomy.

Thus, while the first of these requirements (for knowledge and understanding) is expressed primarily in terms of the depth of coverage achieved, the second requirement (for the ability to apply that knowledge and understanding within work or a vocation) implies very clearly that such graduates must be able to operate within the context where this knowledge and understanding is to be applied. That is, the graduate must be able to operate within an application domain, whether general or specialised, and so for informatics programmes these descriptors do imply that the breadth of study necessary to provide adequate coverage of the requirements of the chosen application domain is just as important as the depth of coverage of core informatics material. Indeed, it is arguable that an undergraduate programme in informatics that did not attempt to provide any significant coverage of some suitable application domain would not be meeting properly the requirements for learning outcomes from the first cycle of the Bologna structures, as these are defined by the Dublin descriptors.

5. Summary and Conclusions

This analysis of application domains has therefore led to five main results. Firstly, it has identified five features that characterise an application domain, namely the kinds of organisations that use systems from within this domain, the organisational purposes for these systems, the dependence of these systems on the limits of particular technologies, the characteristic properties that are required for these systems, and the particular product technologies or process methods that must be used in developing these systems. Secondly, from the relationships between these features three different groups of domains have been identified, namely business domains, property domains and technology domains. Then the distinction has been created between general and specialised application domains, where specialised domains may need to be treated individually within curriculum models (as in SE2004), whereas for general application domains it is only the common characteristics rather than the individual domains that must be considered in such models.

Thirdly, an alternative classification of domains has been derived from general systems theory, which provides a hierarchical structure that shows how most of the disciplines within informatics apart from CS are related to different classes of application domains, and also provides a structure within which the choice of application domains for a particular degree programme can be organised. Fourthly, from these the kind of application-oriented material that needs to be included in the curriculum has been identified, and it has been proposed that some kind of standard domain model needs to be created to identify the basic business processes that are associated with different domains. Fifthly, from considering the pressures on curriculum time that would result from trying to include this material in informatics degree programmes, it has been shown that coverage of this kind of material is necessary in order to meet fully the general requirements for the learning outcomes of programmes that have been developed within the Bologna process.

References

1 ACM/IEEE, The Joint Task Force for Computing Curricula 2005. Computing Curricula 2005: The Overview Report. ACM/IEEE, 2005. Also at <http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf>.

2 McBride N. Erase old programme and launch new version. Times Higher Education Supplement; 9 February 2007: 14.

3 ACM/IEEE, The Joint Task Force on Computing Curricula. Computing Curricula 2001 Computer Science. ACM/IEEE, 2001. Also at <http://acm.org/education/curric_vols/cc2001.pdf>.

4 ACM/IEEE, The Joint Task Force on Computing Curricula. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. ACM/IEEE, 2004. Also at <http://sites.computer.org/ccse/SE2004Volume.pdf>.

5 ACM/IEEE, The Joint Task Force on Computing Curricula. Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, ACM/IEEE, 2004. Also at <http://www.acm.org/education/CE-Final%20Report.pdf>.

6 Gorgone J T, Davis G B, Valacich J S, Topi H, Feinstein D L & Longenecker, Jr. H E. IS 2002: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems, ACM/AIS/AITP, 2002. Also at <http://www.acm.org/education/is2002.pdf>.

7 ACM SIGITE, Curriculum Committee, Computing Curricula: Information Technology Volume (Draft dated October 20, 2005), ACM, 2005. Also at <http://www.acm.org/education/curric_vols/IT_October_2005.pdf>.

8 Irons A. Computer Forensics - a Case Study on the Impact of New Technology on Informatics Education (Extended Abstract). In: The Higher Education Academy, Information and Computer Sciences; Programme for Informatics Education Europe, Montpellier. At <http://www.ics.heacademy.ac.uk/education_europe/Session_1/25_Alistair_Irons.doc>.

9 Ferguson D. Grid Computing the Challenge for Education and Training (Extended Abstract). In: The Higher Education Academy, Information and Computer Sciences; Programme for Informatics Education Europe, Montpellier. At <http://www.ics.heacademy.ac.uk/education_europe/Session_1/22_David_Ferguson.doc>.

10 Eurostat. Revision of NACE and CPA. At <http://circa.europa.eu/irc/dsis/nacecpacon/info/data/en/index.htm>.

11 Denning P J. Great Principles of Computing, *Communications of the A*CM 2003; **46**.11, 15–20.

12 Quality Assurance Agency. The framework for higher education qualifications in England, Wales and Northern Ireland - January 2001. At <http://www.qaa.ac.uk/academicinfrastructure/FHEQ/EWNI/default.asp>.

13 Joint Quality Initiative. Shared Dublin descriptors for the Bachelor's, Master's and Doctoral awards. At <http://www.jointquality.nl/content/ierland/Result%20Draft%20Dublin%20Descriptors%203%20cycles.doc>.