Science for Global Ubiquitous Computing

A fifteen-year Grand Challenge for computing research

1 Challenge

Ubiquitous Computing entails large-scale networks of computing devices and agents. They are hardware or software; static, mobile or wearable; permanent or ephemeral; communicating, reflective and location-aware. They operate in highly distributed – even global– scenarios involving both processes and data, at low power and in a timely fashion, guaranteeing privacy and security, individually exhibiting high failure rate yet reliable and dependable as a whole.

For this Challenge we make no separation between the concepts of Ubiquitous Computing and Global Computing. They cover the Internet, together with the mobile physical devices linked to it and the software platforms built upon it; they also cover designed systems such as healthcare coordinated across a country, which involves highly distributed medical data, care-scheduling, mobile resources and emergency action. Furthermore they cover all possible collaborations among such systems, and between them and humans. We refer to this whole, which is part engineered and part natural phenomenon, as the *Global Ubiquitous Computer* (GUC).

As *engineered artifact*, the GUC is probably the largest in human history. Yet a rigorous understanding of it, and of how it might develop, is lacking. When we add devices and software to it, we do so with some understanding of these new parts, but no clear grasp of the whole onto which we graft them.

As *natural phenomenon*, the GUC is as complex as many others –physical, chemical, biological or ecological– that are objects of intense scientific study. Just as differential equations, Laplace and Fourier transforms, and numerical linear algebra serve as toolkits for physics and traditional engineering, so scientists must develop theories for understanding and building the GUC.

'Understanding' and 'building' are generic terms that cover a range of distinct activities. We may *adapt, analyse, combine, correct, design, diagnose, document, enhance, evaluate, expand, exploit, formalise, implement, instrument, refine, re-use, specify, test, validate, ...* systems. Pervading all these activities is *modelling*. The key to a science for the GUC is that the same models are used both in the analytic activity (the understanding) and in the synthetic activity (the building).

Considerable success has already been achieved over the past four decades in modelling many subtle features of computation. These models lead from highly developed theories of sequential computing and databases, to theories that are less developed –but already enjoy fair consensus– for concurrent interacting systems and distributed data. Here is a skeleton, roughly in order of discovery:

> universal machines, automata theory, formal language theory, functional calculi, database theory, automated logics, program semantics, logics for specification and verification, type theories, Petri nets and process calculi, temporal and modal logics, calculi for mobile systems, semi-structured data, game semantics.

Almost all of these have been augmented with automated tools for design and analysis, such as simulation, computer-assisted reasoning and model-checking.

This is a substantial science base. In a companion paper *Theories for ubiquitous processes and data*, here called the Platform Paper, we briefly survey the state of the art in these topics. The paper contains a large bibliography, and is available at `http://www.cl.cam.ac.uk/users/rm135/plat.pdf`. It gives ample evidence of progressive refinement of the science, and also of its influence on industrial practice.

Nonetheless, this influence has been incomplete and haphazard. Why? The explanation lies in the extraordinary pace of technological development, and the corresponding pace of change in market expectations. The science has been aimed at a moving target, attempting to underpin the ever more complex designs made possible by advances in hardware and networking technology. Moreover, theories typically remain far longer in gestation than opportunistic design practices. Two effects can be observed:

- The theories themselves are not yet complete or unified;
- Software houses have designed what the market required, rather than what has been subjected to all available theoretical analysis.

In other words, theories have not sufficiently informed software design. Often they have been retrofitted to it, revealing weaknesses too late to mend them. A classic example is the application of type theory to legacy code, revealing just where it was vulnerable to the Y2000 problem. There were no great disasters after the millennial date, but enormous expense was incurred *before* it, in anticipation of what might happen. Such lack of confidence would not arise with well-typed code. The necessary type theory had been researched and published at least two decades previously.

A second example[1], closer to the GUC, concerns the IEEE 802.11 standard for data confidentiality known as Wireless Equivalent Privacy (WEP), introduced in 1999. This was found in 2001 to be severely imperfect. Analysts showed how an attacker, using a few million encrypted packets, can deduce the shared key used by WEP. Several other attacks have subsequently been found. By then, millions of devices employing WEP had been sold worldwide.

Two observations lead to our Grand Challenge. The first is negative: unless we offer a soundly based methodology to supplant the practice of opportunist software creation there will be consequences of the kind we have illustrated, and a further mass of inscrutable legacy software. These consequences will be greatly more damaging than previously, because the GUC is pervasive, self-modifying and complex in the extreme.

The second observation is positive, and concerns the range of concepts that we must bring under control in understanding the GUC. This range is so impressive as to justify a science; it also ensures that the design of software and systems will undergo a revolution, during which entrenched practices may be abandoned and the science may properly inform all analysis and design, as indeed it does in every other engineering discipline. Our Grand Challenge is therefore:

[1] Reported in Communications of the ACM, May 2003.

- *To develop a coherent informatic science whose concepts, calculi, theories and automated tools allow descriptive and predictive analysis of the GUC at each level of abstraction;*

- *That every system and software construction – including languages – for the GUC shall employ only these concepts and calculi, and be analysed and justified by these theories and tools.*

We deliberately pose this as an *ideal* goal. It will never be fully achieved, but we pose it in this ideal form because we see no argument that limits the degree of attainable success. If at first it seems absurd, consider that other engineering disciplines approximate well to this goal, since –unlike software engineering– they are founded on a pre-existing science.

A criterion for success in meeting the Challenge is the extent to which, in one or more activities –existing or new– mounted on the GUC platform (e.g. distributed business processes, instrumented buildings, healthcare coordination), both the structure and the behavioural analysis of specific software systems are couched in terms of the new science. Thus the success can be incremental.

In the rest of this note we

- explore the GUC concepts that have hitherto been identified and list some achievements expected in the short term (Section 2);

- discuss a strategy for mounting an attack on the Challenge, by means of a network and experimental projects (Section 3);

- assess the Challenge in terms of the criteria for maturity of a Grand Challenge (Section 4).

2 Platform

Every long-term challenge must be approached by a mixture of bottom-up research and goal-directed navigation. Structures for navigation must be considered, and more is said about them in the following two sections. In this section we emphasise that we are not starting from scratch in addressing our Challenge; theoretical work over the past fifty years has created an impressive platform of concepts and structures relevant to it.

In the Platform Paper we briefly survey the state of the art in these topics, arguing that they provide a sound platform from which to launch an attack on the Challenge, and predicting advances we may expect within a few years. It describes research under eight headings:

> Space and mobility; Security; Boundaries, resources and trust; Distributed data; Game semantics; Hybrid systems; Stochastics; Model-checking.

This is neither a complete nor a coherent classification of relevant work; other topics will emerge, but these provide an initial foothold. The rest of this section is devoted to

a summary, under these eight headings, of the work described and advances predicted in the Platform Paper. Note that these are just *predictions*; they are not a research programme. In Section 3 we propose ways in which a research programme for our Challenge can be defined by a community of interested researchers.

Space and mobility It is now accepted that *locality* and *movement* of components (whether data, processes or devices) are properties to be represented directly, at least at one level of modelling a widely distributed system. At a more abstract level, for example in a behavioural specification, they may be omitted (e.g. left to the implementation). This illustrates the necessity of multi-level modelling, and of a theory that can validate a description at one level against a specification at another.

Calculi and logics are now emerging, some based upon existing process calculi, such as the π-calculus and Mobile Ambients, in which the physical space occupied by real systems is treated as one with the virtual space of software and data structures.

> *Expected advances:* Within five years we can expect at least one calculus or logic, with an associated programming language, to be installed for experiment in a prototypical system concerned with space and mobility, e.g. a sentient building. There should also be associated program design patterns and analytical tools. These experiments will reveal further challenges for progress towards the Grand Challenge.

Security and privacy Safe languages –typically with robust *typing* and *static analysis*– can demonstrably provide protection against security attacks such as buffer overruns. Currently, model-checkers and symbolic techniques based upon logical proof or bisimulation can routinely verify or falsify security protocols.

> *Expected advances:* Within five years we can expect designs of safe general-purpose languages close to C, to which existing codebases in C may be ported with minor adjustment, and which are immune to certain attacks. Such languages may also be treated as precursors of successor languages derived directly from theories. We can also expect both manual and computer-assisted verification of *implemented* protocols, not only their abstract *descriptions*.

Boundaries, resources and trust Mobile agents in ubiquitous systems must expect to acquire resources as necessary from the environment that they visit. Access to resources can be controlled in terms of *boundary crossing* in a current spatial model such as Mobile Ambients, augmented by a type discipline. Methods based upon logics and types now exist to control *allocation* and *deallocation* of *resources*, including memory space. Logics and languages have been proposed for expressing the *trust* essential for resource allocation, in terms of notions such as *belief* and *authority*; this also allows a discipline for the *propagation* of trust to be formalised and implemented.

> *Expected advances:* Within five years we can expect the development of protocols, based upon present research, for mobile agents to acquire and to

manage resource. This involves *access negotiation* based upon an evaluation of *trust*; the latter will vary, under dynamically varying *knowledge* and *belief* about agents, and will depend crucially on how trust is propagated. Such protocols cannot be perfect; we can expect them to be submitted to intense investigation in experimental and simulated systems, and to intense stochastic analysis, to assess their degree of reliability.

Distributed data The need to query distributed data sources gave rise to an early practical example of mobile code. Several research problems arise from the increasingly evolutionary nature of databases. Adapting relational database theory and language design to semi-structured data is a non-trivial task, creating new research challenges. For example, the copying and mutation of data highlight two previously unproblematic aspects of data: *provenance* and *archiving*. Copying seldom preserves the provenance (i.e. the 'pedigree') of a datum; how do we then assess credibility? Frequent change of databases endangers the archival requirement; how can we ensure that the sources we cite remain inviolate? An associated issue is *annotation* – the overlaying of existing data with extra detail or comment. How does this affect the querying of a database, and how do we organise data to leave room for future annotation? The concept of *semi-structured* data begins to address the issue.

Expected advances: Within five years we expect strong advances in querying methods for highly distributed data. For example, this is an essential feature for a system for coordinating healthcare across a country. We expect progress in ubiquitous monitoring and integration of data. We also expect a merging of research on semi-structured data and process calculi to yield models, languages and tools to integrate the mobility of processes and data.

Game semantics 'Game semantics' refers to a theory, based upon mathematical games, that arose from a successful attempt to solve long-standing semantic problems in sequential computation. More recently, understanding the behaviour of an interactive agent to be a *game strategy*, it succeeds in classifying such agents by structural constraints upon their strategies. Looking further ahead, it is not unreasonable to represent the goals and intentions of *intelligent* agents by appropriately complex strategies.

Expected advances: We expect to explore the compositional power of strategies modelling agents –i.e. strategies for simpler parts compose into strategies representing larger assemblies. The semantics of languages for mobile and distributed systems will be addressed in terms of games and strategies. Foundational work will be done to make game semantics more mathematically tractable.

Hybrid systems A hybrid system is one with both discrete and continuous components. The last decade has seen considerable advance in models for hybrid systems, consisting of varieties of automata and process calculi; special attention has been given to *timed* automata and calculi, where typically time is the only continuous variable.

Many hybrid systems, e.g. those with continuous *space* will occur naturally in ubiquitous computing. Special model-checking techniques have been invented for timed automata, but are limited to small systems.

> *Expected advances:* Within five years we expect considerable advance in models and modular description languages for *mobile* hybrid systems. There will also be greater emphasis on the unification with control theory. Such results will facilitate practical application, e.g. to a sentient building. There will be further advances in both efficient algorithms and modular techniques for verification, in timed automata. The challenge of obtaining appropriate representations of state sets generated by complex continuous dynamics of hybrid automata will be addressed, developing ideas implemented in existing tools.

Stochastics The *performance evaluation* of stochastic systems, beginning with Erlang in the 1910s for capacity planning in telephone systems, has developed via stochastic versions of Petri nets and process calculi into general models, able to simulate, analyse and predict quantitative aspects of behaviour in arbitrary discrete systems, including ubiquitous ones. The models have dealt with a variety of probability distributions, and have combined them with non-determinism. *Probabilistic model-checking* is an active field; it encounters the usual problems of model-checking (see below), and also specific difficulties in compositional analysis.

> *Expected advances:* For stochastic modelling of spatial mobility, a strong feature of ubiquitous systems, we expect within five years considerable advance in extension of models based upon π-calculus, mobile ambients or hybrid automata. These will have to incorporate numerical methods, e.g. for differential equations. We expect increasing attention paid to model-checking for systems with (potentially) infinite state. We also expect an increase in the application of industrial-strength probabilistic model-checkers to relevant case studies, some at least in ubiquitous computing.

Model checking In analysing a process, expressed as communicating automata or in a process calculus, we typically wish to determine whether its behaviour satisfies a particular property – in turn expressed in a temporal or modal logic. *Model checking*, introduced over twenty years ago, is a form of automatic reasoning specialised to this task and has achieved great success. Recently attention has been devoted to *infinite-state* model checking, which will be important for ubiquitous systems. Other particular challenges pertinent to ubiquity are *probabilistic* systems, *mobile* scenarios, *scalable* and *compositional* methods, and application to real distibuted software.

> *Expected advances:* In the next five years we expect that our Challenge, and especially any Exemplar Challenges (e.g. a sentient building) that are mounted under it, will strongly influence the agenda of model-checking research. Because of the ramified structure of the GUC, we expect strong advances in compositional techniques. We hope to see model-checkers built into interactive tools for the analysis of components of the GUC.

3 Strategy

In the previous section a number of concepts were identified, each of which is already an established topic of research, and each essential to ubiquitous systems. The Challenge already entails a bewildering range of ideas, and more will emerge. To meet the Challenge, bottom-up research on each topic separately is not enough; they have to be studied in concert.

Here we identify three levels at which experimental projects can be defined without delay. We also propose a means by which the research community can generate a portfolio of such projects and train them towards the main Challenge. This will not supplant exploratory theoretical research; rather, it will enhance its value and provide incentive to undertake it.

Experimental applications The first kind of project will aim to achieve part of the goal of the Challenge for a particular application area; it will constitute an Exemplary application, probably defined and achieved in (say) three-year phases. The aim of such an Exemplar is primarily experimental, not practical; it will experiment with existing and new calculi, logics and associated tools to achieve a prototypical system in which the specification and programming are permeated by theoretical understanding. It success consists not in delivering the application for use in the field, but in exhibiting its formal structure and its analysis in the terms of an appropriate scientific model. Here are three possible topics for such project, all of which are currently researched:

- A sentient building[2];
- Health-care coordinated across a city or country;
- A platform for business processes[3].

For example, programming for the sentient building may be based upon a process calculus for space and mobility, expanded to accommodate continuous space and time; the database for the health-care application may illustrate a theory of mobile distributed semi-structured data; the business-process platform may illustrate a particular use of process calculus and logics for specification and implementation.

There is no reason why the studied application should be a new one; there is great scientific value in taking an existing system that works in the field and re-constructing it on a more explicitly scientific basis. Our Challenge aims to illustrate the extent to which theories can pervade the *construction* of a system, not merely be brought in to *analyse* it after construction. Comparison of such a theory-based design with one that is currently working is a valuable scientific experiment.

Experimental generic systems Experimental Exemplars such as the above will confront many conceptual problems. Many of these will be generic —i.e. we would expect

[2]See submission 5 under Panel B, on the Grand Challenges web site `http://umbriel.dcs.gla.ac.uk/NeSC/general/esi/events/Grand_Challenges/panelb/index.html`.

[3]See *Business Process Management: the Third Wave* by H. Smith and P. Fingar, amazon.com.

the same problem and solution in widely differing applications. Our sister Grand Challenge *Scalable Ubiquitous Computing Systems* (SCUS) seeks engineering principles for the GUC, and both Challenges will benefit from joint work on specific aspects of design. In each case we would expect to ask: How do theoretical models assist the structuring and analysis of certain aspects of a system? Three possible topics are:

- Stochastic models for reconfigurable systems;
- Resource allocation in an open distributed environment;
- Logic and language for reflectivity.

In the first topic, we aim for models that can predict the behaviour of reconfigurable systems –e.g. communications networks– that respond probabilistically to demands. We already have calculi for mobile distributed systems; we understand stochastic behaviour in non-mobile process calculi; we have experience in stochastic model-checking. The GUC provides the incentive to combine these three, in the attempt to establish design principles, and indeed to predict behaviour in existing systems such as internet.

In the second topic, one concern is how to represent disciplines for the allocation of resources –including processors, memory, and services– in a suitable calculus and associated programming language. Another concern is safety, in an open system where clients are not a priori trustworthy. This entails a logic of trust (e.g. if A trusts B and B spawns C, does A trust C?), and ways of verifying that a program implements a trust-discipline expressed in the logic.

Reflectivity (the third topic) is the ability of a system to report on its own actions, and on its ability to fulfil its own intentions. What degree of reflectivity should be present at each level of a GUC system? The answer will be embodied in an engineering design principle, as sought by SCUS. The theoretical challenge is to define a calculus in which the reflectivity of a process is *modelled explicitly*, and to demonstrate that this reflectivity is *correctly implemented* in a lower-level calculus or language.

These three topics illustrate a rich vein of research challenges. They all explore the mutual influence between engineering principles and theoretical concepts. A pivotal component in all three is a programming language informed by the theory.

A theoretical hierarchy A distinctive feature of computational modelling is that models must exist at many levels. At a high level are logics; at a low level are assembly codes. Intermediate levels are already suggested by some of the above project topics. For example, at a certain level we may model trust but not locality; at a lower level, locality but not trust. Again, at a certain level we may model communications as instantaneous, but implement them at a lower level by complex protocols.

So, having seen some of the rich conceptual armoury required for the GUC, we can refine our Grand Challenge as follows:

- *To express theories for the GUC as a hierarchy of models and languages, assigning each relevant concept to a certain level in the hierarchy;*
- *To define, for each model M, how a system description described in M may be realised or implemented in models $M_1, \ldots, M_n$ lying below M;*

- *To devise methods and tools for reasoning both at each level and between levels.*

The second item is no mere theoretical exercise; it is essential, to justify the analysis of each attribute of a system at the level of description appropriate for that attribute.

We now begin to see how specific projects can be mounted to bridge the gap between the substantial platform of existing research and the achievement of the Challenge. Each such project can be seen as either developing a model for a limited range of concepts, or developing the realisation of such a model in terms of lower ones. For example:

- Extending an existing calculus for mobile distributed systems to incorporate continuous spatial variables and stochastic state transitions;

- A coordination calculus for systems that are heterogeneously modelled or programmed.

The first topic is of theoretical interest in its own right, but can be linked to the Exemplar study of a sentient building. It should naturally include a programming language as a sub-model.

The second topic acknowledges that, to meet the Challenge in a way that embraces applications (or parts of them) already in existence, one must accommodate systems implemented in arbitrary languages. Just as Corba (for example) coordinates the execution of heterogeneously programmed systems, so a coordination calculus must admit the analysis of such systems. A good example is provided by existing communications protocols; the way to accommodate them in the Challenge is to show –for each protocol in whatever language– that it behaves correctly according to a specification expressed in the coordination calculus itself.

A Network for GUC Theory? We have defined three categories of project that can be mounted on our existing theoretical platform (as defined in the Platform Paper), as first steps in attacking our Challenge. But this is not enough to get a concerted work programme going; the first step must be to enable a community to reach consensus upon the mode of attack. The Platform Paper identifies groups in the UK, and also groups in other countries with whom they interact, that would naturally form such a community. We propose that a Network be formed that will enable the community to convene in workshops, by mutual exploratory visits, via a web site, and by email, to discuss topics for projects (of which the above suggestions are merely tentative examples). An indication of maturity of the Challenge will be the emergence of a steering group to organise and administer the Network, on the assumption that appropriate funding can be found. [4] Here is a tentative remit for the Network:

- To discuss and refine a research strategy for attacking the Challenge;

[4] Already, in May 2003, a network called UK UbiNet has recently been initiated, funded by EPSRC. It has interests across a very wide spectrum in ubiquitous computing. The goals of this Challenge are primarily theoretical, but they can only be properly addressed in collaboration with engineering and applications research, as indicated above. We may therefore envisage a specialised network affiliated to UbiNet, or the formation within UbiNet of a special interest group for GUC theories.

- To identify a range of relevant research topics;
- To select certain topics for workshops, to be organised as early as practicable;
- To publish a record of these workshops;
- To manage a web site and email dialogue for its members;
- To continue these activities throughout the life of the Challenge.

The Network would give priority to promoting UK research, but international initiatives may serve a similar purpose more widely. One such is GC2, an FET pro-active initiative for *Framework Programme 6* of the European Commission. The GC2 Strategy Group has recently published a vision for GC2 entitled *Building the Case for Global Computing*; it can be found at `http://www.cogs.susx.ac.uk/users/vs/gc2/gc2.pdf` .

4 Assessment

We now respond to a questionnaire that has been put to all Grand Challenge proposals, to assess their maturity and viability.

Scientific significance

Is it driven by curiosity about the foundations, nature or limits of basic science?

Global ubiquitous computing requires theories of computation and data management that go well beyond what we now have. Our challenge is driven partly by the need for such theories, to ensure that the GUC is robustly engineered. Beyond that, scientific curiosity drives the urge to understand the interplay between the many concepts involved (trust, delegation, locality, security, non-determinism, ...) in a way that is independent of any technology.

Shannon initiated a *quantitative* theory of information; there remains a huge intellectual challenge to model information *qualitatively*: not only how much information flows, but between which agents, with what purpose, and under what discipline.

The scientific drive is increased by initial indications that the theory can extend beyond artifacts to biological and other natural systems.

Are there clear criteria for the success or failure of the project after fifteen years?

The Challenge is to develop concepts, calculi, theories and automated tools that allow descriptive and predictive analysis of ubiquitous computing systems, so that *every system and software construction – including languages – shall employ only these concepts and calculi, and be analysed and justified by these theories and tools.* This is the ultimate criterion of success. We expect only approximate success; it can be measured by the extent to which each particular real system or subsystem meets the criterion.

Beyond this measurable test, there is an equally important test for which success is less easy to measure: to what extent has a stable qualitative theory of information been developed?

Does it promise a revolutionary shift in the accepted paradigm of thinking or practice?

Yes; and it requires such a shift, since the accepted paradigm of computer system development is ad hoc; currently, most initial designs and subsequent modifications lack scientific backing.

Does it avoid duplicating evolutionary development of commercial products?

The goals of the Challenge are in the interest of software and device producers; but no company can adopt more rigorous production practice until the theories are sufficiently advanced and tested upon experimental systems; nor can it afford to do so until the community demands greater rigour.

Impact on Practice

Will its promotion as a Grand Challenge contribute to the progress of science?

If the Challenge is adopted, the research community will be all the more encouraged to design research projects with its ultimate goals in mind. Progress towards these scientific goals, rather than shorter-term advancement of technology, will become an accepted measure of the success on such projects.

Does it have the enthusiastic support of the established scientific community?

Not yet. The wider scientific community has become used to thinking of computing systems as a technology that can be delivered in the required quantity and with the required performance, without recourse to science. It is largely unaware of the danger of imperfectly understood systems. It is also unaware of the scientific depth and complexity of the concepts involved in understanding the systems likely to be built for the GUC. By publicly adopting scientific challenges such as this one, computer scientists can make their own scientific aspirations endorsed by colleagues in other sciences.

Does it appeal to the imagination of the general public?

Not yet. The public seems unaware of the kinds of systems (e.g. memories for life, or computer-aided surgery) that are only a few years away. *A fortiori*, it is not aware of the science that should underpin them. However, the systems themselves can capture the imagination of lay people; by explaining the underlying ideas in non-technical terms, we can therefore hope to extend their interest to the science as well.

What long-term benefits to science, industry and society may be expected?

Systems that benefit society will continue to be built, as they are now, without help from this Challenge. But if the Challenge is not addressed we shall have postponed the answer to the question 'When shall we understand these systems conceptually, to ensure full control of them as they mutate and grow?'. Worse, by analogy with our current legacy software, global ubiquitous systems will stagnate, immune to correction

or amelioration. The analogy with E.M. Forster's *The Machine Stops* is not out of place.

So it is a clear benefit both to science and to society to understand these systems. Further, it is an *urgent need* for the industry that it be addressed without delay.

Scale

Does the project have international scope?

Many teams in Europe and in the USA, as indicated in the Platform Paper, have done fundamental research in this direction. As indicated above, there is an EU initiative on the topic. The UK is one of the theoretical leaders.

How does the project split into sub-phases with identifiable goals and criteria?

There is a holistic element to ubiquitous systems that will make the research very demanding. But in the Platform Paper, summarised in Section 2, we have identified several topics that present necessary and tractable subgoals, and more will follow. Building on this, in Section 3 we have given examples of several possible projects that can be mounted without delay. We have also suggested means –in particular a GUC Network– by which the community can set up a suitable research programme.

What calls does it make for collaboration of research teams with diverse skills?

The total diversity of required skills –in application domains, systems engineering and different theoretical domains– is large. In view of the variety of projects illustrated in Section 3, most projects will need only a subset of the skills. But there is a strong call for cross-relation between these projects, since the Challenge requires a consistent hierarchy of theories to be developed. This cross-relation must be achieved via a Network and a series of workshops and conferences.

How can it be promoted by competition between teams with diverse approaches?

Competitive research, leading to incremental consensus, is the only way for theoretical research to advance. At the same time, some research programmes will be needed that commit to a particular approach – in order perhaps to collaborate effectively with applications experts.

Timeliness

When was it first proposed as a challenge? Why has it been so difficult so far?

One such proposal as mentioned above (with a URL) is entitled *Building the Case for Global Computing*; it has been submitted to Framework Programme 6 of the European Commission. That initiative encompassed many general goals, including the building of theories, which is the main object of this Challenge. It arose from several meeting over the previous five years or so, including a brainstorming meeting in Edinburgh on September 2000.

The sharpened form of the present Challenge emphasises the importance not only of building theories, but also that software constructions for the GUC should use only the concepts and calculi of those theories. Previously, perhaps even now, most people would have dismissed this formulation as unnecessarily extreme. They would also have pronounced it too difficult, because our theories were not up to it. Our perception now is that it is still difficult, but that our theories are advancing fast; most importantly, the Challenge has become urgent because of the extraordinary implications of today's technology for pervasive, ubiquitous computing systems.

Why is it now expected to be feasible within a ten to fifteen year timescale?

The urgency *demands* that we consider it feasible; in other words, necessity must be the mother of our invention. Moreover, the state of the art in many aspects of relevant theory has advanced a lot in the last decade.

What are the first steps?

Projects of the kind that we have identified in Section 3 must be first steps. But this proposal cannot arrogate to itself the task of defining such projects in detail. We have therefore suggested that a GUC Network be set up to define an initial research programme, with the help of a series of workshops.

What are the likely reasons for failure?

The Challenge is built on the hypothesis that a suitable hierarchy of theories can be found. The first possible reason of failure is that the hypothesis is false. We do not believe that this is likely, because existing theories are already addressing parts of the Challenge. The most likely reason for failure is that, even with a coherent research community formed in response to the Challenge, the coordination of academic researchers with industry is not sufficient to mount the results of the Challenge in practical systems. This may occur if the demands of the market –as discussed in Section 1– continue to prevent industry from exploiting scientific advances.

Acknowledgements

Many ideas in position papers submitted to the Grand Challenges Workshop are relevant to this proposal, and have inspired us. Without any pretence to completeness, we would like to mention here some of the ideas that have particular relevance to theoretical modelling. [5]

David Bell draws attention to three concepts – *evidence, causality, and uncertainty* – that are not fully understood in computer science. They are certainly highly relevant to ubiquitous computation. He points out that an inevitable feature of ubiquitous systems is *ignorance* (by one component of the others); this entails evidence-based reasoning. A good example is in the evaluation of trust, in connection with resource allocation.

[5] These submissions can all be found on the website of the Grand Challenges Exercise, `http://umbriel.dcs.gla.ac.uk/NeSC/general/esi/events/Grand_Challenges/`.

Cornelia Boldyref advocates an emphasis on the modelling of *system interactions* and of *realized* operational systems. The latter is vital for the maintainability and evolution of systems; Muffy Calder echoes this point in emphasising the modelling of systems *as built*, not just *prescribing* or *specifying* them.

Christopher Simons advocates a *single system model* as opposed to a number of different, partial models. In relating computing systems to biological systems, Ronan Sleep advocates *computational models of the development of organisms*, which may then in return shed understanding on the systems we build.

Finally, Phil Trinder notes the need for *high-level coordination abstractions*, that may bridge the gap between process calculi and practical coordination mechanisms such as Corba. We regard this as essential to the incremental adoption of more scientific practice; indeed we have proposed a project to devise a coordination *calculus*, allowing existing applications to be combined reliably into larger wholes.

Contributors to this paper and its companion Platform Paper are: Samson Abramsky, Peter Buneman, Philippa Gardner, Andrew Gordon, Marta Kwiatkowska, Robin Milner, Vladimiro Sassone and Susan Stepney. Tony Hoare offered detailed comments.