Theories for the Global Ubiquitous Computer

Robin Milner

University of Cambridge, The Computer Laboratory,
J J Thomson Avenue, Cambridge CB3 0FD, UK

Abstract. This paper describes an initiative to provide theories that can underlie the development of the *Global Ubiquitous Computer*, the network of ubiquitous computing devices that will pervade the civilised world in the course of the next few decades. We define the goals of the initiative and the criteria for judging whether they are achieved; we then propose a strategy for the exercise. It must combine a *bottom-up* development of theories in directions that are currently pursued with success, together with a *top-down* approach in the form of collaborative projects relating these theories to engineered systems that exist or are imminent.

A Grand Challenge for Computational Theories

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.

There is no doubt that over the next few decades we shall see ubiquitous computing pervade the civilised world. This paper describes an initiative to provide theories that will underlie this dramatic development, both to realise its full potential and to avoid the huge inconvenience and possible disaster that can be caused by the ad hoc engineering of such a pervasive network of artefacts. The exercise forms part of a Grand Challenge programme mounted by the UK Computing Research Committee, but is also intended to merge with international programmes with similar goals.

For this Challenge we make no separation between 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 have long

I. Walukiewicz (Ed.): FOSSACS 2004, LNCS 2987, pp. 5–11, 2004.

been the objects of scientific study. The parts we build form part of a whole which is not, and probably never will be, the realisation of a single design; to that extent it occurs 'naturally', and demands scientific understanding in the traditional sense.

Just as differential equations, Laplace and Fourier transforms, and numerical linear algebra serve as toolkits both for physical theories and for traditional engineering, so computer scientists must develop theories both for understanding and for 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 should be used both in the analytic activity (the understanding) and in the synthetic activity (the building).

Our Grand Challenge is therefore:

- *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 come close to achieving this goal, since –unlike software engineering– they are founded on a pre-existing science.

To be worthy of the name 'Grand Challenge', a goal must not only lie beyond the reach of existing concepts and technology, but must also admit clear criteria for achievement. Ours certainly meets the first requirement. For the second, we shall be able to declare success just to the extent to which, in one or more activities mounted on the GUC platform (e.g. distributed business processes, instrumented buildings, healthcare coordination), both the structure and the behavioural analysis of its specific software systems are couched fully in terms of the new science.

The full case for this Grand Challenge can be found on the UK website for Grand Challenges in Computing Research:
`http://www.nesc.ac.uk/esi/events/Grand_Challenges` .

The Existing Theoretical Platform

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. A companion paper to the Grand Challenge proposal, under the title *Theories for ubiquitous processes and data*, outlines the state of the art in these topics. This survey, referred to here as the 'Platform 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 engineers have designed what the market required, rather than what has been analysed even by currently available theories.

In other words, theories have not sufficiently informed software design. Often they have been retrofitted to it, revealing weaknesses too late to mend the design. 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.

There are two motivations for 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 motivation is positive, and concerns the range of concepts that we must bring under control in understanding the GUC. This range –as we briefly indicate below– 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 other engineering disciplines.

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

So what are the scientific concepts involved? We do not yet know them all, but we are not starting from scratch. Theoretical work over the past fifty years has created an impressive platform of concepts, structures and tools relevant to the GUC. The Platform Paper surveys several that have emerged most recently, 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. In all of these topics we can predict outcomes over the next few years that are certain to be important for the GUC. We can think of research in these directions as the *bottom-up* approach to a science for the GUC.

Strategy for Attacking the Challenge

To complement the essential bottom-up theoretical advances, a Grand Challenge must also be approached by goal-directed navigation; the *top-down* approach. What kinds of project provide this navigation?

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. These projects will will enhance the value of the bottom-up research and provide incentive to undertake it.

(1) Experimental Applications. The first kind of project aims to achieve part of the goal of the Challenge for a particular application area; it consists of 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 specification and design are permeated by theoretical understanding. Its success consists not in delivering the application for use in the field, but in exhibiting its formal structure and analysing its behaviour in terms of an appropriate scientific model. Here are three possible topics for such project, all of which are currently researched:

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

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, implementation and coordination.

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. The goal of our Challenge is that theories should pervade the *construction* of a system, not merely be brought in to *analyse* it after construction. To mount such a theory-based design and then compare it with one that is currently working is a valuable scientific experiment.

(2) 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 the same problem and solution in widely differing applications. This suggests that, besides underlying theories, universal engineering principles for ubiquitous systems are to be sought. A sister Grand Challenge, entitled *Scalable Ubiquitous Computing Systems* (SCUS), is being mounted as part of the UK exercise, with the purpose of eliciting these principles. The two 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 for collaboration 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 the 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 in each subsystem of the GUC? 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.

(3) A Theoretical Hierarchy. A distinctive feature of computational modelling is that models must exist at many levels. At a high level are specifications and 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.

With this in mind, models at many levels of abstraction were stipulated as part of the main goal of our Grand Challenge. Having seen some of the rich conceptual armoury required for the GUC, we can now see more clearly how these levels should be related, and can refine the main goal 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.*

We now begin to see how specific projects can be mounted to bridge the gap between the 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 existing applications, 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.

Mounting the Exercise

We have discussed theoretical topics to be developed bottom-up, and we have defined three categories of project that can be mounted on our existing theoretical platform (as defined in the Platform Paper), as first top-down steps in attacking our Challenge. But this is not enough to get a concerted work programme going; the various research communities need a means to converge upon specific initial projects. This is most likely to be achieved by networks and workshops organised for that purpose.

An example of a 'vertical' network —one that aims to link different topics of research relevant to ubiquity— is UK UbiNet, recently formed and already organising workshops for groups with different research skills (covering hardware, software and theory) to inform each other. The relevant website is `http://www-dse.doc.ic.ac.uk/Projects/UbiNet/` . In contrast, a European network focussing upon theories for global computing already exists; known as GC2, it is 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*, coordinated by Vladimiro Sassone; it can be found at `http://www.cogs.susx.ac.uk/users/vs/gc2/gc2.pdf` .

Conclusion

We can consider the Global Ubiquitous Computer as the ultimate distributed system. We have already responded to the exciting challenge of distributed systems; the result has been a new generation of computing theories. We now see that the technology of ubiquitous computing has extended this challenge still further; current theories of distributed and mobile computing systems can be seen as precursors of a still broader science. There is an opportunity, and an urgent need, to develop this science before methodologies for the GUC become established and hard to change. This can only be done by an ever closer collaboration between engineers, theorists and users.