
Discussion on Robin Milner's First *Computer Journal* Lecture: Ubiquitous Computing: Shall We Understand It?

1. MORRIS SLOMAN

**Department of Computing, Imperial College London,
180 Queen's Gate, London SW7 2AZ, UK.**

Email: m.sloman@imperial.ac.uk

Although I agree with most of the points made in the paper, Robin indicates that he is tempted to consider a *global ubiquitous computer*. I think this would not work, as the scale of such a system would be so large, it would not be practical to apply design concepts, analyse or understand it. Software Engineering has taught us how to break large systems down into manageable components and similar concepts need to apply to ubiquitous computing. We advocate considering a Ubiquitous System (US) as multiple, interacting cells which we call self managed cells (SMCs) [1]. An SMC could correspond with a body area network which monitors (and in the future controls) the health of a person as well as providing the means to interact with other people. At another level an SMC could correspond to a meeting room in which personal SMCs interact with the other people in the meeting, as well as the local ubiquitous environment in terms of devices such as printers, displays, and audio systems supporting the meeting, devices controlling the air-conditioning and lighting etc. The meeting room SMC is part of a building SMC which manages the whole infrastructure of the building to optimize energy usage, environmental conditions, communication and even people movement via lifts etc. A large-scale service provider could also be considered an SMC providing communications services to both personal SMCs, vehicle SMCs and building SMCs.

An SMC is more complex than current software engineering concepts such as objects or components in that it is very dynamic. The components of the SMC may join or leave over short time-scales as people and vehicles are mobile.

An SMC provides a scope for designing autonomic functions such as self-configuration, self-healing, self-optimization, self-protection and context aware adaptation which are essential for ubiquitous computing systems. An SMC can provide a scope for applying theory for analysing, modelling and understanding. A personal SMC will also be the means to support multi-modal interactions with the ubiquitous environment and through the environment with other people's SMCs. Although a personal SMC may be a comparatively small-scale US, we also have to be able to consider

large-scale SMCs such as the communications service provider supporting millions of personal, vehicle and building SMCs.

From the above it can be seen that we have to consider the following forms of interaction between SMCs and ways of combining SMCs.

Peer-to-peer interactions support collaboration between people in a meeting or typical social interactions. However, peer-to-peer interactions also represent typical forms of collaborations between service providers e.g. for routing of communication; providing services to 'foreign' subscribers who are unable to access their own service provider as happens with mobile phone usage in a foreign country; or the collaboration between emergency services at the scene of an accident.

Hierarchical interactions are needed for supporting the applications which are built up from simpler lower level services, such as a location aware information service using a communication service, location tracking service and accessing various information providers for weather, traffic conditions or local restaurants.

Composition is needed to build a large SMC such as the Building SMC from room SMCs or an intensive care ward to manage the patient monitoring SMCs and supporting the actions of medical staff with their own SMCs.

In my view the concept of an SMC provides a basis for developing design methods, tools, models and theories for interactions which will cater for the very large scale global USs.

2. MARTYN THOMAS

Martyn Thomas Associates, UK.

Email: martyn@thomas-associates.co.uk

Robin Milner has drawn a visionary picture of ubiquitous computing and sketched the software science¹ that is needed if the vision is to be realized. The science of ubicomp is a Grand Challenge indeed, and one that may need rather more than 20 years before it is fully realized. The scientific challenges are exciting, and I hope they will inspire a generation of researchers, but I am deeply concerned about practical

¹I adopt Robin Milner's definition, in the hope that a useful term can be rescued.

issues of engineering. What will cause an industry to adopt the new science when building real USs?

The answer should be obvious: if these complex systems are built without the solid foundations of sound theory, they will fail: projects will overrun, costs will escalate, the systems will prove unreliable, insecure or unsafe in use and practical deployment will be patchy, falter and perhaps stall completely. That is a confident prediction because, as Robin Milner has shown clearly, the ubicomp community is targeting levels of complexity far beyond anything that has ever been built before, and requiring that their systems continue to function dependably whilst reconfiguring themselves to interact with other systems that had not even been considered in the original design.

That may seem a powerful argument for adopting software science, but it has not proven a compelling one for today's software industry, who continue to build complex and sometimes safety-critical systems without making appropriate use of the software science that exists today. As a consequence, most new systems cost far more than they should, take longer to develop than necessary and deliver fewer benefits than the customers expected. This problem is not unique to the UK: international surveys [2] and books [3–5] suggest that it is a widespread (and probably universal) problem.

Robin Milner has generously suggested that this is the inevitable result of a marketplace that is highly competitive and rapidly developing, but I believe the reverse to be the case. The marketplace for new complex systems is dominated by a small number of very large companies, and when customers increasingly outsource their entire IT, or seek to buy very large systems from single prime contractors, it is impossible for the smaller, innovative companies to compete. For example, there are a number of European companies whose software development methods, languages and tools are based on mathematically formal methods and sound computer science. They have shown that they are able to develop systems that consistently have 5% or fewer of the defects [6] that are routinely delivered by their larger competitors (with no increase in development costs) but these companies each have less than 200 staff: it is not possible for them to win projects of the scale of the National Identity Register, or the flight software for a new aircraft. So competition is limited and new science is only adopted slowly.

The result is that industry still uses design notations with shallow semantics, programming languages that are ambiguous and impossible to analyse mechanically to any depth, and components that lack adequate specifications or warranties. The problems that were identified in the NATO software engineering conferences of 1968 [7] and 1969 [8] are still with us, and yet the software science to overcome them has been available for at least 35 years [9]. The two main international standards [10] for safety critical systems are currently being revised and, shockingly, neither of these revisions will require that all developers of safety-critical

systems specify the required safety properties with mathematical rigour.

This is a major barrier to practical ubicomp. Unless industry can be persuaded or compelled to adopt the relevant software science of the last 40 years, there is no hope that companies will be in a position to adopt the output from Robin Milner's Grand Challenge. And, as the paper shows so clearly, without the science, the applications will remain little more than academic dreams.

3. KAREN SPARCK-JONES

**Computer Laboratory, University of Cambridge, UK.
Email: Karen.Sparck-Jones@cl.cam.ac.uk**

Taking USs seriously means thinking about systems whose constituent subsystems can be properly described as autonomous agents, i.e. systems that do not just have functions, but have goals. However the presumption seems to be that constituent agent activity is motivated and bounded by the larger US. Subsystem agents are autonomous only in how they achieve their goals.

Robin Milner writes about designing USs in a way that assumes control over the whole enterprise: i.e. as if a US is a single teleological construct that can be understood, because of its 'singlemindedness', sufficiently to support a comprehensive, but meticulously grounded, and rigorous design. (Variant external user interests are negotiated into the design.)

This may seem reasonable. But USs will in general not be greenfield operations. They will be built on top of existing system(s). These existing systems, though viewed as US constituents, will have their own prior functions, i.e. goals.

US builders will need to factor in these existing goals. This will be problematic: first, because these goals will be heterogeneous and probably conflicting; second, because given the subsystems already exist, perhaps as antique legacy, their goals may be unobvious or inaccessible. However it will not, in general, be sufficient to model the subsystems from their observed behaviour. US designers will have to think, not just about *what* a subsystem does, but about *why* it does it.

As an analogy, the bees in a hive are a miniature US. The bees are autonomous agents, with class goals, e.g. get nectar, and individual goals, e.g. for some worker bee, pick up this blob of nectar, fly back to hive, or for a queen, lay eggs here, start off swarm now. There is separate behaviour, e.g. foraging, and collective behaviour, e.g. swarming, and communication. This is the bee view. The beekeeper, who did not create the bees but provided their hive, has his own goals, e.g. getting honey, making money. There are also users, the honey buyers, with their goals.

In designing his bee US, the beekeeper will not get a system that satisfies his goals unless his model factors in the bees' goals. Of course the bees' goals are inaccessible. But to get a nicely working hive producing honey he needs to think not just about what he can see the bees doing when they're foraging,

but about why they might be doing it the way they are. For example, they forage as they do so as to minimize journey time to the hive. That will help him locate the hive well. Similarly with swarming. If he does not accommodate the bees' inferred goals, he'll get badly stung. But because he can only infer the bees' goals, he'll get it wrong sometimes, so he needs to wear protective clothing. Even then, he'll get stung on occasion, part of the price to pay for the honey.

4. JON CROWCROFT

Cambridge University, UK.

Email: Jon.Crowcroft@cl.cam.ac.uk

The three towers (with apologies to J.R.R. Tolkien)

Robin Milner has presented a view of the challenge of ubiquitous computing from the perspective of software science. In the process, he has also made a strong plea for the value of the science of software, which I also subscribe to. The manifesto for the challenge incorporates three viewpoints, the other two covering the experiential and technological aspects of the problem space.

I am interested in the engineering principles that will be required to succeed in conquering the technological problems in UbiComp. I'll use two examples from the past to try to illustrate what I think we need to do in this space for the future.

In the manifesto we wrote that the principles will be visibly uncovered when we can see them being taught and applied in Masters courses, and in exemplar projects around 10–15 years from now.

What I would like to add to Robin Milner's discussion is this: we could envisage three towers of models, leaning together to make a tripod, a stable mutually supportive structure that contains experience, science and technology. I am not the person to address a tower of models for the human factors, interactive systems and experiential side of UbiComp. I will attempt to assert that there will be a tower of engineering design principles or rules, applicable in a similar way to the tower of scientific models. As with the debate at the lecture based on Robin Milner's paper, I agree that the word 'tower' is of course too restrictive—the application of models, or of design rules, may be made in some arbitrary graph of course, but the word tower is there and will stick.

I do not know precisely what the types of design rules and principles will be specifically for this Grand Challenge (of course, otherwise it would not be a challenge), but I would like to point out that the idea of design principles for computer systems has been successfully applied in at least two areas in the last 25 years. The design philosophy of the DARPA Internet was documented in the 1988 paper [11] by Dave Clark of MIT (about 12 years after the migration of the ARPANET from NCP to TCP/IP). An elegant expression of the set of rules of thumb used to design successful cryptographic protocols was documented by Abadi and Needham [12].

Other broader questions were discussed by Sally Floyd *et al.* [13], and an economic and policy perspective was brought to bear in [14]. These works represent a tip of the iceberg of practices today. They took around one to two decades to emerge as the underlying computing and communications systems were being researched and prototyped. They incorporate much material that is delivered in Masters programmes today in communications systems, and in security engineering (for example, we reference them in material taught in Cambridge). They are referenced in standard documents today.

They incorporate a web of other disciplines including control theory, optimization, agents and game theory, queueing systems, random and self-stabilizing algorithms, modularization, virtualization, layering, viewpoints transparencies (*viz.* ODP Affordance) as well as security principles.

However, there are some interesting meta-lessons from the success of both security and inter-networking engineering design. In both of these, I think the meta-lesson is that engineering and science exist in a subtle relationship which inform each other (and I am certain that one can make the same argument for the third element of the challenge too, although of course, we know the three body problem is a tad trickier!

In the Internet, we have succeeded in designing a system that is reliable (extraordinarily so compared to prior systems), despite the unreliability of its components. Indeed, we are starting to understand the performance and the behaviour of the Internet in great detail, and yet the component computers (hosts and routers) are a long way from being built out of scientific software (the majority of computers run Microsoft Windows, the majority of routers run Cisco IOS, and these contain 20-year-old code written in C, with scarce attention even to software engineering disciplines, let alone software science).

In that same Internet, we routinely carry out secure transactions. So secure are these that credit card and banking agencies would like to stop using legacy mechanisms for absolute cost reasons, and yet also are happy to underwrite the risk as it is lower, and, crucially quantifiable so, than in those legacy systems.

What this is supposed to illustrate is that one can envisage UbiComp meeting some desirable goals, without the necessity to apply a microscopic level of scientifically correct programming at every level. By analogy, I would point out that while we have a tower of models in physics (in some senses, all of natural science is a tower of models from quantum mechanics through chemistry, up through biology to ecosystems), we do not need to use it to build systems that are correct.

To spell it out, we do not use quantum mechanics in aeroplane design.

What emerges is that we can, through prototyping, develop empirical results which can lend us assurance that the necessary science could, in principle, be applied, but does not need

to be. In fact, some of the engineering techniques we use allow a more statistical approach to meeting desired behaviour—the correct operation of a very large ensemble of systems, despite faulty (even insecure and undermined) components is something we have developed methods to build (some mentioned in the list above). Recent theoretic results from the control theory world such as those reported by Doyle from Caltech [15], have started to explain this success. Thus, I suppose, a suitable challenge for the science of US could be to give us a more predictive model for the combination of multiple such systems. This is not in conflict with the tower of models Robin Milner proposes, but is a holistic view of that tower!

5. MARTA KWIATKOWSKA

University of Birmingham, UK.

Email: m.z.kwiatkowska@cs.bham.ac.uk

Robin Milner's lecture discusses ubiquitous computing and the risks associated with our failure to develop proper scientific foundation for ubiquity. We are already witnessing exemplars of systems that use sensor and mobile devices in safety- and business-critical applications such as medical monitoring and banking, but less so coordinated attempts to address the generic scientific principles that underpin the development of USs.

As my research interests have been close to Robin's, I strongly agree with his analysis and conclusion. Robin mentions examples of models and theories developed in the past that informed the practice of programming—data structures, types and concurrent processes. The recently espoused synergy between the pi-calculus and business process languages is indeed a very exciting prospect, with potential to influence the W3C choreography standard. The role of these models and theories goes far beyond informing the standards, however. The mathematical formulation of the problem allows for *formal reasoning* about system correctness, which is capable of handling systems of huge, perhaps even infinite, scale, via decomposition and mechanised theorem proving. Importantly also, it makes it possible to *automate* aspects of the processes of system validation and verification as embodied, for example, in *model checking* software tools [16]. Examples include the Static Device Verifier developed as part of the SLAM project [17] at Microsoft and used for model checking of compliance of device drivers to the specification.

The ubiquitous computing scenario, as aptly illustrated by Robin, presents new and daunting challenges for formal reasoning and model checking. We must consider stochastic models of mobility, be able to deal with context and adaptation, and address issues such as quality of service and security. Techniques such as *probabilistic model checking* [17, 18] can be of assistance, but currently can only handle static, finite system configurations.

One issue that I find particularly intriguing is the argument for scientific foundations that are not only descriptive, but also

predictive, and therefore more in line with established sciences. The issue of whether our subject (variously referred to as computer science, computing, computation, informatics) is a science in the conventional sense is hotly disputed right now. Personally, I agree with Alan Bundy [19] that strong scientific foundations are necessary and look forward to participating in the exciting future developments of our discipline.

Notwithstanding, I have one disagreement with Robin, and that is that I do not believe the tower of models is feasible or appropriate. Rather, a looser collection of theories, perhaps a graph, perhaps comprising subtowers, is what we should aim for.

6. PAUL GARNER

Head of Pervasive ICT Research Centre, BT Group Chief Technology Office, British Telecommunications plc, 81 Newgate Street, London EC1A 7AJ, UK.
Email: paul.2.garner@bt.com

Robin's paper discusses the lack of an underlying foundation of fundamental computer science which, if it existed, would allow us to more successfully model behaviour in large-scale computing systems and therefore design more correctly from the outset. In industrial research we aspire to develop a set of technologies that will allow the underlying components of USs to be assembled in real time on-demand, to provide useful, reliable and safe applications and services. In an attempt to fill the science gap we have embraced a nature-inspired approach to the creation of solutions to complexity, with many leading software scientists in this area being biologists by training. As set out in the IBM autonomic computing agenda, we aim to create distributed autonomous systems that are self* [20–22]. In order not to lose control over such fully distributed systems, future engineers will have to be able to make appropriate use of self-organization and 'infer' local rules capable of promoting the desired system behaviour from statistical, and not deterministic, predictions. Such self-managing solutions will play an important role in making any future ubiquitous environment 'invisible' so that users need not become full-time system administrators to their pervasive home entertainment, or healthcare systems. In support of this approach a new EU Integrated Project called Cascadas [23] has been formed to create an 'autonomic handbook' of engineering guidelines and decision support in how to build, understand and tune UBs. It is taking an approach which draws on complexity science and current principles of biological systems and their organizational attributes and behaviours, such as cooperation, coordination and communication.

Account also needs to be taken of future business and economic models as they will be the ultimate drivers of new applications. Detailed models need to be constructed that expose the potential value of USs and which reveal

how such value can be successfully delivered. We must engineer adaptive ambient user interfaces that will engender trust. USs that are adaptive and derive intelligence from our environment must produce services that are understandable and that have predictable behaviour in the eyes of the end user. These requirements need to be balanced with the desire for invisible technology, to avoid situations where end users are unable to ascribe specific outcomes to their interaction with USs.

This paper provides an excellent summary of the scientific challenges we face in trying to understand the underlying principles of viable ubiquitous computing applications for the future.

7. NICHOLAS R. JENNINGS

School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK.

Email: nrj@ecs.soton.ac.uk

On computational service economies

The effective design and development of complex computer systems is one of the biggest challenges facing computer scientists. Such systems are characterized by their open nature (components can come and go during the system's lifetime), their decentralized control regimes, the high degrees of dynamism and uncertainty that are endemic within them, and the fact that they contain components that represent the aims and interests of multiple distinct stakeholders [24]. Examples of these types of system include the ubiquitous computing systems discussed in Robin Milner's paper, as well as grid computing systems, peer-to-peer systems, the web, and autonomic systems. In all of these seemingly diverse types of system, however, I believe there is a common computational model that can be used as the underpinning for conceptualization, design and implementation. This is the service-oriented model. In this, the various components in the system are viewed as providing services to one another. This needs to be done in a flexible and responsive manner to cope with the dynamism and uncertainty that is present in the environment and so I will term the entities that produce and consume the services software agents. Now, these agents need to interact with one another in order to gain access to the services that others provide. Since the agents are autonomous and represent distinct stakeholders, the de facto form of interaction will be some form of negotiation. If a negotiation is successful it will result in some form of service agreement or contract that specifies the terms and conditions under which the service is provided.

For me, this service-oriented view is the right high-level model for UbiComp systems and so should form the top of Milner's tower of models. The other, more traditional,

computer science models mentioned in his paper are important and useful, but they are too fine-grained to be an effective point of departure for conceptualizing and designing the sorts of complex systems that are being discussed. Clearly, a mapping needs to be established to the more traditional models, but if we start with these then I believe we are doomed to fail since they are simply too low level a start point.

Given this standpoint, a natural source of concepts and models is provided by game theory. In this way, the complex system can be viewed as a computational service economy in which the various agents cooperate, coordinate and compete with one another in order to achieve their individual and collective aims and objectives. Game theory is compelling because it provides a series of concepts (such as preferences, equilibria, incentive compatibility) for both analysing what outcomes are likely in the system and how particular desirable characteristics (such as fairness or stability) can be attained. However, game theory as it currently stands is not a panacea. Traditionally, game theory has not considered issues associated with computability, and issues associated with dynamism and uncertainty have not been to the fore. Thus further work is needed to modify and adapt it to computer settings (see [25] for a more comprehensive discussion). Nevertheless, the successes of a number of projects that have used game theoretic techniques to analyse and build UbiComp systems (see, for example, [26, 27]) indicate that it has much offer in this space.

8. VLADIMIRO SASSONE

ECS, University of Southampton, UK.

Perhaps because I have been involved with some of the technical aspects covered in the paper, I was most struck by Robin's 'philosophical' remarks that our science is fundamentally different from the natural sciences, in that those are driven by the desire to understand, while we are by the need to build. I find his analysis of where this leads us particularly interesting. Robin's subsequent question, which underlies the entire paper, is shall we ever catch up? Given that advances and new technologies move the goalpost, shall we ever converge?

An observation hidden in Robin's argument is that *the universe works and will keep working relatively undisturbed whether or not we understand it; computer systems will not*. We have no comprehensive theory of the universe yet, but the laws of physics are not particularly perturbed by our lack of certainties about dark matter and energy. Not so of course for our ability to build artefacts, and I believe that pointing this out helps put in focus how seriously we should take our relative lack of foundational understanding of the emerging field of ubiquitous computing.

What seems important—and to me perhaps not so different between natural sciences and 'sciences of the artificial'—is that our wish for progress and advances must

be underpinned at the same time by adequate technological support, appropriate scientific understanding and suitable economic/social drive. To reach the moon we needed the models of Newtonian mechanics (the science), the thrust of suitable propulsion engines (the technology), and the large amount of money and drive of a cold-war competition to get there first (the socio-economics). While for travels in our solar system the science was at least 400 years ahead of the technology, for computing the technological support and the economic drive have a clear lead on the scientific understanding. Is this an intrinsic property of computing, or an 'accident' of history we will catch up with?

Whatever the answer, the important point made by Robin's lecture is that there will not be a single comprehensive, all-explaining theory, even though that will always be our ultimate ambition (as it is for the natural sciences). We will have to have a tower, or perhaps a network, or even a patchwork of theories and models relating to the objects of study from different angles and perspectives. To develop all necessary levels of abstraction, so that all models interlace together properly is part of the present 'challenge'. It is a task that cannot be compartmentalized, e.g. in theory and engineering, exactly as theoretical speculation over nature cannot proceed isolated from observation and experimentation.

Robin suggests that at any one level of abstraction only a small number of concepts may be relevant. I suspend a final judgement on that, but I am happy to proceed with this working hypothesis. We certainly need to build confidence in the fact that theories can work seamlessly at different abstraction levels, whereby lower levels implement and validate assumptions made at higher levels which, in turn, will have to be feasible and in agreement with the observations. At an elementary level, Robin's example of a high-level axiomatic of trust can be mapped down to models based on temporal properties of system execution traces, as e.g. [28].

The forthcoming round of 'foothill projects' will have to build more sophisticated such mappings, and make them significant.

9. EAMONN O'NEILL

**Department of Computer Science, University of Bath,
Bath BA2 7AY, UK.
Email: eamonn@cs.bath.ac.uk**

Understanding ubiquitous computing: a view from HCI

A substantial body of research approaches ubiquitous computing from a Human-Computer Interaction (HCI) perspective. The goals of HCI as a discipline include concepts, theories, models, design principles, methods, tools and techniques. We may also agree on these goals for ubiquitous computing across a wide range of disciplines but can attach very different meanings to these terms.

Let us consider modelling. As noted by Wegner [29, 30] and others, interactive systems are very difficult to specify and to model. In HCI, we emphasize interactivity. From time to time, there are attempts to model more or less formally in HCI. Examples include the syndetic modelling of Barnard *et al.* [31]. This work, and HCI in general, reflects what Milner calls a 'holistic view' that considers both human and artefact. In contrast, Milner adopts a 'dualistic view' that focuses on modelling the artefactual parts of the system.

Milner [32] presents an example of the dualistic approach to modelling trust: 'A simple axiom for trust may be transitivity: if A trusts B and B trusts C then A trusts C'. Is this simple? Yes. Is it useful? Often it is not—because for *people* trust is not transitive. Of course, we can define trust to be transitive (or anything else that we would like it to be) but that does not help extend our modelling to cover human behaviour and experience. Milner [32] notes that in the dualistic approach, 'in the case of concepts like . . . trust, as they pertain to software agents, we shall not be concerned with the full human interpretation; instead we expect to use simpler pragmatic notions that are both definable and implementable'. This simplification is of course necessary in order to implement systems. But then people will come along and perform all kinds of unmodelled behaviours which subvert the simple model on which the system design is based. For example, we have empirical evidence that people will behave based on factors such as convenience, regardless of trust [33]. Whether within a dualistic or a holistic approach, we lack the ability effectively to model the human behaviours in the whole system.

For the Grand Challenge of ubiquitous computing, Milner [32] identifies three perspectives of ubiquitous computing: human experience (or HCI), design and science. He also proposes three goals that, he claims, map respectively to these three perspectives:

Methods and techniques that are sensitive both to the needs of individuals and society and to the impact of ubiquitous computing upon them;

Design principles that pertain to all aspects of ubiquitous computing and are agreed among both academic and professional engineers;

A coherent *informatic science* whose concepts, calculi, models, theories and tools allow descriptive and predictive analysis of ubiquitous computing.

But these goals do not map neatly to the three perspectives. For the HCI researcher, reducing the goal of HCI to the development of methods and techniques (goal 1) is anathema. Milner [32] asks if we can postulate criteria for success in achieving our vision for new human experience with the same clarity as criteria for success in the design and science perspectives. The answer is yes, if and only if our methods and techniques in HCI instantiate empirically tested design principles (goal 2) that are founded on the

concepts, models and theories of a coherent informatic science (goal 3). Hence, the goals of HCI as an approach to ubiquitous computing include concepts, theories, models, design principles, methods, tools and techniques.

10. MICHAEL WOOLDRIDGE

**Department of Computer Science, University of Liverpool,
L69 7ZF, UK.**

Email: mjw@liv.ac.uk

What is correctness in the age of ubiquitous computing?

Robin paints a compelling picture of the future of computing and clearly identifies the key issues that computer science must address in order to realize this vision. In this response, I want to comment on one particular issue that Robin's vision raises, which relates to one of the main underlying concerns of computer science—and *British* computer science in particular. This is the issue of *correctness*.

There is a well-established set of formalisms and associated technologies for investigating the correctness of computer systems of various types, of which model checking is perhaps the best-known and most successful contemporary example [16]. While these approaches differ in many ways, they all start from the assumption that there is some precise, formal specification of the desired properties and behaviour of the system under study; and the purpose of the verification exercise is to show that the system does (or does not) satisfy this specification. Typically, the specification is expressed in some form of logic—for example, temporal logic in the case of model checking. This formal system specification is, in AI terminology, a *goal*; and a key purpose of the development process is to develop a system that achieves this goal, i.e. is correct with respect to the specification.

This model of correctness assumes that the specifier enjoys a privileged position, in the sense that the specifier is able to define the criteria by which the system is understood to be correct or otherwise. Of course, the specification will often have been derived from consultations with many stakeholders, but nevertheless, ultimately, there is a single overall position from which the correctness of the system is assessed. And of course, the specifier is assumed to have a *consistent* specification—the software cannot be required to satisfy logically inconsistent requirements.

But it is easy to see that this standard model of correctness simply does not map to Robin's ubiquitous computing world. It implies that somebody has unique ownership of the system, and that the system can be designed to satisfy their goals. But in a globally accessible network of software and hardware components, there *can be no privileged position* from which a unique standard of correctness is defined. The participants in USs will have different goals and agendas, and these goals—their specifications—will inevitably be mutually inconsistent.

So, what will replace this classical notion of correctness in a system containing multiple interacting computing elements, each seeking to achieve their mutually inconsistent goals? One approach we have been following is to adapt ideas developed in game theory—the mathematical theory of interacting self-interested agents. Central to game theory is the notion of an *equilibrium*: a standard of behaviour that is 'stable', under the assumption that agents within the system act rationally, i.e. in pursuit of their personal goals. Instead of asking whether the system is correct, we can then ask instead what are the equilibria of the system, and whether these equilibria are desirable. We can also try to engineer a system so that its equilibria are desirable from some point of view (e.g. so that any equilibrium of the system leads to an outcome that is 'fair' to all participants). Where the system involves a substantial legacy component, the metaphor of a *social contract* becomes useful: a social contract being an agreed standard of behaviour that agents within a system will abide by in order that the benefits of cooperation are not lost in conflicts [34, 35].

With respect to Robin's hierarchy of models, I believe that at one level in this hierarchy, we must recognize these multiple agents with their potentially conflicting goals; the task of linking these *micro-level* models to *macro* models characterizing system-level behaviours seems to me to be a key challenge.

11. CARSTEN MAPLE

**Institute for Research in Applicable Computing,
Department of Computing and Information Systems,
University of Luton, Park Square, Luton LU1 3JU, UK.
Email: Carsten.Maple@luton.ac.uk**

The aim of ubiquitous computing research and implementation is the embedding of systems into everyday life in a transparent manner: the user moves between locations and tasks, largely or perhaps completely unaware of the computing infrastructure [36]. Agent-based technologies have been developed to facilitate this transparency. A user interacts with some (software) agent or set of agents and this agent then acts on behalf of the user to achieve some goal. In achieving this goal, it is likely, certainly in a ubiquitous computing environment, that this agent will have to interact with other agents. This brings about issues concerning trust both within agents and between the user and the interaction agent. As He *et al.* note [37], users will need to become content to let a piece of software make decisions on their behalf. This will obviously require time and *will only occur as agents show what they are capable of*. Herein lies a major issue: if a user is to be content to allow some software agent to act on their behalf, then clearly, it is better if the user has some (even if it is quite basic) understanding of how that agent will attempt to achieve that goal. If the user is unaware of the interactions the agent will have to undertake with other agents, they will not be in a suitable position in which to make an assessment of risk. Each time that two agents interact, there is a possible

trust (and security) implication. A great deal of work exists in attempting to best formulate models of trust between agents, much of it building upon the work of McKnight *et al.* [38]. As the ubiquitous environment becomes more complex, in terms of numbers of agent-agent interactions, the user (at the point of interaction, and therefore often authentication) can soon become quite *distanced* from the point of the goal being achieved. As such the average user may have little real awareness of the power they are potentially devolving to an agent. It is possible that a rogue agent can use information gained from an honest agent, to act in some malicious way. The average user will have no idea of such possibilities: what can be seen as a great time- and effort-saving environment can be used to wreak unforeseen havoc if the appropriate risk assessment has not been undertaken. It is therefore imperative that the engineers of USs consider the distance between points of authentication and goal satisfaction. Should the maximum distance become too large, the user may feel a loss of control. Standard graph-theoretic techniques can be used to model USs and can provide a measure of distance between interaction and goal.

If a user has a mobile telephone that is left unlocked, most would be generally aware of the risks associated with someone finding the telephone and acting maliciously. This is largely due to the fact that goals achieved using the mobile telephone are relatively close to the interaction point in a graph representation of the system. Now consider mobile telephone that is configured to allow access to bank account details without the need for further authentication. Hence one interaction agent (the telephone interface) is used to access some other service. This brings about different issues of security. Due to the relatively simple system described most users can understand the associated risk, and so would likely ensure that there is a further need for authentication before accessing the bank account details. If we consider more interconnected devices and services in this ubiquitous environment: can a user be sure what access an attacker has, and what distress, inconvenience and cost malicious acts can result in? For a small proportion of users, the technologically literate, this might be easily answered; for those more innocent—users who do not enable security features or just leave the default setting [38]—they may be unaware of the danger, unable to assess the risks or prepare for the consequences.

Agent technology holds out the possibility of connecting multiple, seemingly (to the user) unrelated agents. In this scenario new risks appear notwithstanding inter-agent negotiation [39]: anyone who has access to one password or, say, an open phone may now have access to a wealth of personal information. In a world in which technologists offer up wearable computing, pervasive access control sensors and single sign-on, users stand to gain much in terms of immediate task fulfilment, yet may be rendered unable to conduct usability-versus-security risk assessments. Thus, embedding security into USs and at the same time engaging

users in understanding and employing security measures remains a significant challenge.

12. GORGE COULOURIS

**Digital Technology Group, Computer Laboratory,
Cambridge University, UK.**

Email: george.coulouris@cl.cam.ac.uk

My concern about the Ubiquitous Computing Grand Challenge is that it lacks an industrial perspective and has received little input from the potential producers and end-users. Fifteen years after Weiser's original vision statement, ubiquitous and pervasive computing remain predominantly research programmes with little associated industrial activity. The need for further research is perfectly understandable: the emergence of new technologies follows no set timescales and the more complex and generic the goals, the longer the required research activities needed to yield the necessary understanding. That is one of the important messages of Robin Milner's excellent paper. But he also identifies experience-led activities as one of three domains within which understanding must be advanced and the Grand Challenge manifesto elaborates on that theme. I take this to refer to the design of systems that enhance human activity and the experience of using information systems. (Weiser expected the ubiquitous computer to be 'invisible', but he presumably also expected an enhanced experience to result from it). Reflection on the history of the major developments in interactive computing—time sharing, interactive graphical systems and the personal computer—shows that these developments took place largely in research environments but were all stimulated and focussed by a day-to-day awareness of needs in industry and more widely. Although improved methods for the evaluation of HCI have emerged from recent research, it remains the case that the constraints of real-world needs and market forces are an essential context for design. (Eamonn O'Neill points out that the problem may be less prevalent for Japanese and other Asian researchers.)

13. DAN CHALMERS

Department of Informatics, University of Sussex, UK.

Email: D.Chalmers@sussex.ac.uk

The idea of user intention and training was raised, however, we should remain aware that pervasive computing will be subject to use by those who are curious, half-asleep, taking drugs, violent, suicidal or otherwise hard to plan for. It may be that there is no 'intention', or that intentions will not be those anticipated by the designers.

The component devices of pervasive computing will be mass-produced and subject to day-to-day life: chewed by babies; modified for cosmetic effect and to alter their performance (cf. over-clocking of PC CPUs); fitted or worked round by DIY enthusiasts; and taken to harsh environments such as the beach, the garden or the kitchen. To be viable, without

causing an ecological disaster, these multitudes of devices must be self-sufficient: not requiring regular servicing, remaining embedded in, and decaying with, their environment. Any careful calibration will soon be lost, so we must consider that their data about the world will be noisy or wrong and that individual devices will become unreliable and fail.

So, in addition to being wary of assuming we understand our users, we should be wary of assuming we understand the context of operation. In real deployments we are frequently reminded of this, but when forming theories it is easy to start simplifying the world into a clean, precise, logical place. Instead, we should endeavour to embed the idea of uncertainty in our models, as a connecting structure within and between the various layers of Robin's 'tower of models'.

Finally, in specifying correct behaviour the idea of the unknown arises again. We are already building and deploying the first components of the vision, based on current practice, varying quality specifications and a drive to market. However good our specifications they cannot incorporate the unanticipated, but we should expect fluid adaptation not obsolescence and restriction. We should be able to use models, of computers and of people, to examine from various viewpoints what may happen in emerging situations. A sound theoretical basis to our understanding will let us adapt our creations, in rather the same way as new uses for old buildings can be analysed; modifications, limits and anticipated properties specified; but their use evolved in ways which the original architects never dreamt of. As pointed out in the lecture, this requires the grand challenge's combination of experience, engineering and theory coming together to develop both real applications and scientific principles.

REFERENCES

- [1] Dulay, N. *et al.* (2005) AMUSE: autonomic management of (AHM 2005). September 19-22 2005, Available at <http://www.allhands.org.uk/2005/proceedings/papers/434.pdf>.
- [2] Standish Group. The Chaos Reports. Available at http://www.standishgroup.com/sample_research/chaos_1994_1.php and subsequent surveys; see also the 2001 BCS Review, p. 62.
- [3] Ewusi-Mensah, K. (2003) *Software Development Failures*, MIT Press, Cambridge, MA.
- [4] Smith, J.M. (2001) *Troubled IT Projects*. IEE, London, UK.
- [5] Yardley, D. (2002) *Successful IT Project Delivery: Learning the Lessons of Project Failure*. Addison-Wesley, London, UK.
- [6] Pfleeger, S. and Hatton, L. (1997) Investigating the influence of formal methods. *IEEE Comput.*, **30**, 33–42.
- [7] Naur, P. and Randell, B. (1969) *Software Engineering*, NATO, Brussels, Belgium.
- [8] Buxton, J. and Randell, B. (1970) *Software Engineering Techniques*. NATO, Brussels, Belgium.
- [9] Dijkstra, E.W. (1972) The Humble Programmer, Turing Award Lecture. *Commun. ACM*, **15**, 859–866.
- [10] International standard IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems, International Electrotechnical Commission, Geneva, Switzerland; and Avionics standard DO-178B: Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics, Inc, Washington, DC.
- [11] Clark, D. (1988) The design philosophy of the DARPA Internet protocols MIT, ACM SIGCOMM, pp. 106–114.
- [12] Abai, M. and Needham, R. (1996) Prudent engineering practise for cryptographic protocols. *IEEE Trans. Softw. Eng.*, **22**, 6–15.
- [13] Floyd, S. (ed.) (2002) *General Architectural and Policy Considerations Request for Comments: 3426*. Network Working Group, Internet Architecture Board. RFC Editor at <http://www.rfc-editor.org/>
- [14] Clark, D.D., Wroclawski, J., Sollins, K.R. and Braden, R. (2002) Tussle in cyberspace: defining tomorrow's Internet. In *Proc. ACM SIGCOMM '02*, Pittsburgh, PA, August. pp. 347–356. ACM Press, New York.
- [15] Doyle, J. (2005) Highly Optimized Tolerance. Available at <http://www.physics.ucsb.edu/~complex>.
- [16] Clarke, E., Grumberg, O. and Peled, D. (2000) *Model Checking*, MIT Press.
- [17] SLAM project website, Available at <http://research.microsoft.com/slam/>.
- [18] Rutten, J., Kwiatkowska, M., Norman, G. and Parker, D. (2004) *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*. CRM Monograph Series, vol. 23, AMS.
- [19] Bundy, A. *The Need for Hypotheses in Informatics*, Available at <http://homepages.inf.ed.ac.uk/bundy/seminars/Hypotheses.pdf>.
- [20] IBM (2003). *IBM and Autonomic Computing: An Architectural Blueprint for Autonomic Computing*. IBM Publication. Available at <http://www.ibm.com/autonomic/pdfs/ACwpFinal.pdf>.
- [21] Chase, N. (2004) An autonomic computing roadmap. IBM DeveloperWorks. Available at <http://www-106.ibm.com/developerworks/library/ac-roadmap>.
- [22] IBM. IBM's Autonomic Computing Website. Available at <http://www.research.ibm.com/autonomic>.
- [23] IP CASCADAS. Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services. Available at <http://www.cascadas-project.org>.
- [24] Jennings, N.R. (2001) An agent-based approach for building complex software systems. *Commun. ACM*, **44**, 35–41.
- [25] Dash, R.K., Parkes, D.C. and Jennings, N.R. (2003) Computational mechanism design: a call to arms. *IEEE Intell. Syst.*, **18**, 40–47.
- [26] Padhy, P., Dash, R.K., Martinez, K. and Jennings, N.R. (2006) A utility-based sensing and communication model for a glacial sensor network. In *Proc. 5th Int. Conf. Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan.
- [27] Rogers, A., David, E. and Jennings, N.R. (2005) Self organised routing for wireless micro-sensor networks. *IEEE Trans. Syst., Man Cybern. A*, **35**, 349–359.
- [28] Krukow, K., Nielsen, M. and Sassone, V. (2005) A framework for concrete reputation systems with applications to history-based access control. In *Proc. 12th ACM Conf. Computer and Communication Security, CCS 2005*, 7–11 November, Alexandria, VA, pp. 260–269. ACM Press.

- [29] Wegner, P. (1997) Why interaction is more powerful than algorithms. *Commun. ACM*, **40**, 80–91.
- [30] Wegner, P. and Eberbach, E. (2004) New models of computation. *Comput. J.*, **47**, 4–9.
- [31] Barnard, P., May, J., Duke, D. and Duce, D. (2000) Systems, interactions, and macrotheory. *ACM Trans. Comput. Human Interact.*, **7**, 222–262.
- [32] Milner, R. (2006) Ubiquitous computing: shall we understand it? *Comput. J.*, **49**, 383–389.
- [33] Kindberg, T., Sellen, A. and Geelhoed, E. (2004) Security and trust in mobile interactions: a study of users' perceptions and reasoning. In *Proc. Ubicomp*, pp. 196–213.
- [34] Binmore, K. (1994) *Game Theory and the Social Contract, Volume 1: Playing Fair*. MIT Press.
- [35] Wooldridge, M. and van der Hoek, W. (2005) On obligations and normative ability: towards a logical analysis of the social contract. *J. Appl. Logic*, **3**, 396–420.
- [36] He, M., Jennings, X. and Leung, H.-F. (2003) On agent-mediated electronic commerce. *IEEE Trans. Knowl. Data Eng.*, **15**, 985–1003.
- [37] Klein, D. (1990) Foiling the cracker: a survey of, and improvements to, password security. In *Proc. 2nd USENIX Unix Security Workshop*, Oakland, CA, August, pp. 5–14.
- [38] McKnight, D.H., Choudhury, V. and Kacmar, C. (2002) Developing and validating trust measures for e-commerce: an integrative typology. *Inform. Syst. Res.*, **13**, 334–359.
- [39] Parsons, S. and Jennings, N. (1996) Negotiation through argumentation—a preliminary report. In *Proc. Second Int. Conf. Multi-Agent Systems (ICMAS 96)*, Kyoto, Japan, pp. 267–274.
- [40] Weiser, M. (1991) The computer for the 21st century. *Sci. Am.*, **265**, 94–104.
- [41] Amey, P. Available at <http://www.sparkada.com/downloads/Mar2002Amey.pdf>