GC4

Scalable Ubiquitous Computing Systems
or just
Ubiquitous *Systems*

A 15-year Grand Challenge for computer science

Draft of February 21, 2003, editor Jon Crowcroft (contributions from many)

1 Overview

This project is one of a set of three, and the second of two "exemplary" Grand Challenges. The first of these will be an exemplar focused upon one or more specific applications, e.g. "A sentient building" or "Healthcare across a city" and will represent a driving set of requirements. The third project is the "Theory for ubiquitous data and processes" nexus, which underpins the other two. This challenge is to create the required novel architectural structures for scalable ubiquitous computing systems[1]. It is clear that we lack techniques and algorithms to design and implement information processing artifacts both in the large and the small that have desirable properties associated with other ubiquitous systems (e.g. buildings, food, entertainment) such as safety, low cost, availability and integration.

There are clearly strong links with some parts of the strong software engineering grand challenge(s), and with the human computing interaction challenge(s). what's that

2 Challenging Properties of Ubiquitous Systems

The crucial key property of ubiquitous computing is that it may be invisible:- embedded in the environment and can interact with users. The medical exemplar captures some of this aspect.

A list of key unsolved issues (no specific ordering intended) follows:

Context Awareness This may include current geographical/spatial location, nearby neighbours, device capabilities relating to I/O, power, processing, activity of users etc. A ubiquitous system cannot assist a user without being able to accurately determine what they are doing. Sensor networks may need to be aware of what they are trying to measure or detect - human activity, cars, animals etc.

[1] OED: "A set or assemblage of things connected, associated, or interdependent, so as to form a complex unity; a whole composed of parts in an orderly arrangement according to some scheme or plan."

Trust, Security and Privacy Ubiquitous applications may involve collaborations between ad-hoc groups of devices, agents or infrastructure servers. It may require migration (mobile agents) or downloading of code. There are complex issues of knowing what resources, services or agents to trust. Do you trust your neighbour to send you packets for ad-hoc routing ? - it could be a denial of service attack to deplete your battery. From trust, and possibly recommendations, you may be able to derive authorisation policy as to what access you will permit to your resources or possibly what services you should refrain from using. You may need to validate credentials without access to network infrastructure and certification authorities. Most ubiquitous systems have the capability of tracking users and determining patterns of activity - this can be very dangerous if it falls into the wrong hands. Jon discusses some of these issues in a posting on this list.

Seamless Communication Ubiquitous systems need to interact with mobile users or without wired infrastructure. This generally involves some form of wireless communication - Bluetooth, wireless LAN, cellular, infra-red, reflecting lasers (see Dust Motes). Mobile users (e.g. for healthcare applications) may need to seamlessly switch between different modes of communication when they move form house to outdoors etc. Communication policies are needed to limit drain on limited power sources.

Low Powered Devices Power sources for devices is one of the biggest problems. If you have 1000s (possibly 100k) of devices per person, you cannot keep changing batteries. Use of solar cells, fuel cells, heat converters, motion converters may all be possible. The biggest challenge is to design very low powered devices, transmitters etc although in the past our emphasis has been on faster chips.

Self Configuration New approaches are needed to provide flexible and adaptable software and hardware both for mobile devices and the intelligent environment. The scale of these ubiquitous systems necessitates 'autonomic' (self-organising, self-managing, self healing) systems which can dynamically update software to cater for new services and applications. Design tools and theories may be needed to support large-scale autonomic computing for small devices - current work is focused on web servers, databases etc. A related issue is discovery of resources/ services that your require. (see IEEE computer Jan 2003 pp41-50 for a good discussion on Autonomic computing)

Information overload Vast numbers of sensors can potentially generate petabytes of data - this needs to filtered, aggregated, reduced etc. as close to source as possible e.g. using programmable networks, collaborative sensing strategies. Sensors may have to be programmed to look for specific events. How do you find required information - the information being published needs to be described using meta data, (not just time stamp, spatial location/origin) c.f. work on semantic web. Ins some cases it may be better to send queries to sensor networks rather than assume they send information to databases.

Information Provenance Authoritative source of information, how it has been modified or aggregated may be needed for some applications.

Support tools are almost non-existent For design, programming, analysis, reasoning.

Human factors Human factors arise to cater for new modes of interaction and 'invisible' ambient technology which must be usable by non-technical people. There is a need to support transient organisations and dynamic, potentially mobile work arrangements including virtual teams, ad-hoc collaborations and virtual organisations.

Social Issues Human, social and organisational issues that arise in creating new forms of interaction based on ambient technologies and deploying those systems within everyday environments be they the home, the workplace, or more public arenas such as museums and galleries.

Business Models Ubiquitous applications often assume a support infrastructure which provides storage, processing, network connectivity and various services in the home, office, on planes, trains and even in the street. How is this paid for? What is the business model which will make sure this expensive infrastructure is available everywhere.

3 Systems Challenges from Ubiquity

There is a popular view amongst the public that the Internet, the Web, and the PC (hardware and operating systems) have solved the systems problems for large scale distributed computing. The PC appears to provide hardware that scales exponentially with time, up in performance (CPU, Memory, Storage), at the same time as scaling down in size, cost and power requirements. The Internet appears to support twice as many users at ever doubling performance each year. The World Wide Web supports an information sharing system that has evolved from a tool for a few research physicists, to becoming the replacement for physical libraries, shops and banks, for example, for many activities. Between Windows CE and XP, and Linux (for cluster computing and the GRID) it might be assumed that Operating Systems is a "solved" problem; even if these are not seen to address low end systems, then perhaps an appeal to Sun's JVM and Symbian's OS is perceived as satisfying the requirements.

However, over the next 15 years, we predict not just a quantitative expansion of computing, but qualitative change. Individual systems have to scale down as well as up to support ubiquity. Ubiquity implies accessibility in the broadest sense. In Marc Weiser's[2] vision, wearable computers will be the norm. However, in the decade since that original view, we now see we must add sensor networks, telemetry, and pervasive access to the infrastructure through a variety of wireless networks. There are inherent resource limitations in the technologies for processing, storage and communications (and power) in this context, and these lead to novel systems performance requirements. Two particular aspects of very large systems impact differently than of yore: *failure numbers* do not scale linearly with the overall number of computing nodes, thus traditional *models* for coping with failure will not suffice; the speed of light becomes the significant contribution to *latency* in long haul interactive applications. Many current systems attempt to mask these characteristics. We believe that future systems should expose, rather than merely statistically conceal them. This visibility will add to the human sensorium. This impacts fundamentally on privacy and other security concerns.

The requirements to scale further over several orders of magnitude on many performance axes are non-negotiable. However, the underpinning *systems* are very close to the limits of scale. We illustrate the problems with three *desiderata*, in no special order:

convergence (consistency)

> This is a chimera – lets take the example of global internet routing: Most of the algorithms designed to provide distributed routing were designed with a convergence goal in mind. This is to say that after a node or link failure, after some number of messages and some cost of computation, the system would be stable again. As the network has got larger, and more and more complex, the fact of the matter is that the system simply does not converge. Through some *luck*, it has a tendency to move towards convergence (i.e. routes are more correct than incorrect). It is entirely possible that we could have designed a system (especially if we had over optimized it in some particular direction) that got worse before it got better – there is evidence that one in-

[2]Weiser, M. "Some Computer Science Problems in Ubiquitous Computing", *Communications of the ACM*, July 1993.

stance of this, the Border Gateway Protocol (BGP) does exactly this. [3] Of course, at other levels of the system (e.g. in the final resultant computation state for many applications, in the HCAC exemplar) convergence to consistent state is probably very important. It is the ability to choose this range of consistency and to reason and manipulate convergence goals that will be novel in future ubiquitous systems.

relevance Achieving global temporal and spatial locality.

Ubiquity equals omnipresence. However, there are both policy and performance reasons why information tagged with a locality and a time should *not* be made available everywhere, at any time (cue *martini* music).

In computing systems we have been used to developing tools for localising access in a physical space as well as realtime. We now need to extend the tools to cope with parameters from the real world as well. From a performance perspective, we no longer merely demand information from a remote system: sensor networks deliver telemetry continuously. As information sources scale up past receivers ability to cope, and the networks' ability to provide, cooperative filtering is needed; this relies on location being a first class part of information and processing. From a policy perspective, we are used to the idea that we cannot see through a wall: a communicating computer in every interstice means that we could potentially *see* events in any physical place. From even the most naive perspective, this extends the set of subjects and objects of a security policy over a vastly more complex and difficult design space. It may even be that after study, we discover (and this has been proposed by some pundits) that privacy in the future will be a casualty of technology.

Similarly, event notification systems also allow one to deliver time-bounded data (e.g. stock market feeds) but to age stale data - these types of services are set to become critical building blocks in future ubiquitous systems. The underlying mechanisms such as asynchronous message passing, distributed semantic filters and expressions of interest management are far less well understood than classical remote procedure call and reliable message semantics tools.

correctness Assurance and availability

We need to know the authoritative source of information, we need to know information provenance, and we need to have authoritative meta-information.

[3]Even given this serendipity, eventual measured convergence times in the global internet today exceed 15 minutes. In a millennium when GHz processors are $500, this is truly surreal.

We have to have an audit trail or *provenance*, for assurances merely *about* information, such as its timeliness, relevance, relationship (relative timing) with other delivered information.

The latter is something we have just started to research. Systems of trust for e-commerce, for example, are beginning to be understood to the point where a principled approach to risk can begin to be outlined. On the former part of the problem space, one could look at cheating in online gaming or trading, to see that there are many problems that are not at all well understood, let alone any proposed systems for solving them.

The genericity of the systems for ubiquitous computing will be most evident when, having solved some problem for providing statistically correct behaviour in systems in the small (e.g. safe and accurate use of sensor networks for online patient monitoring in the home and street) we can show that the same technique provides massively improved performance in the same dimension in systems in the large (e.g. near perfect availability of national patient record databases).

There is as much ubiquitous processing as there is data. We need the same level of assurances about the programs and their actions and failures. This makes life for the distributed algorithms especially hard compared with traditional approaches.

4 The Research Challenge

Succinctly put, the challenge is "to scale *down* as well as *up*", without loss of generality.

To achieve this, we need a collection of principled techniques that apply to systems with many more nodes, smaller as well as larger ones, and a much larger range of capabilities per node, per cluster of nodes, all carrying out a much wider range and number of different tasks. Imagine the Internet repeated as many times as there are nodes on the Internet today, at a scale ranging from the human body, through buildings, on to global.

An central aim of this challenge for Computer Systems research is to make it feasible to program such systems without having to employ many orders of magnitude more programmers with greater skills than today's software engineers. From the model/architectural viewpoint, a goal should be to provide a hierarchy of abstractions that allow us to specify and understand these systems at many levels of detail From the pragmatic viewpoint, rather than a single system implementation, a classical computer science approach suggests itself - a generative approach to building systems, where the standards are the rules that the generator follows, but the systems that it constructs may be quite different for different pieces of the architecture, or different points in the scale.

Taking some of the example systems desiderata outlined in the previous section we can outline some steps.

For example, firstly this means that we need techniques for designing and implementing self stabilizing algorithms in presence of a large percentage of component failures, including unsignaled failures. We also need a set of subsidiary, information-theoretic, based approaches for distributing data, but also crucially, processing, over a number of components in the presence of partial disconnection. This is a departure from existing paradigms for parallel and distributed computing, at least for some applications. We envisage approximate solutions, with suitable annotations for human consumers, as being acceptable. To achieve these goals, we need to develop new techniques which help us form a category of algorithms and data structures for distributed applications that have convergence, consistency not as endgame, but a *tendency*.

Secondly, we will require techniques to enable applications to express interest in, and build data and processing for the relevant tasks to carry out relevant filtering and data transformation within the infrastructure (i.e. modifying the *end-to-end* model, although we can note that this is already being done in the Internet with proxies of various kinds becoming commonplace). There have been some steps along this path (e.g. the Active Networks and Programmable Networks research programs), but a great deal more needs to be understood.

Thirdly, we need to design mechanisms that have the right meta-data annotations so that information processing can take advantage of this to control where it goes, and report, correctly, its origins, despite any transforms and other intermediate stages, and without those intermediate processing stages compromising the meta-data inappropriately.

These are just parts of what will eventually form a new ubiquitous computer systems architecture. Architecting computing systems is recognised as being very hard. To some extent, this is why we build systems. The system properties are often emergent.[4] Simplicity is often touted as a great contributor to the success of the Internet architecture. We do need to maintain parsimony appropriately (KISS). For example, what is the appropriate model for the ubiquitous VM (surely not just the JVM?), and associated communications system?

Programmability is key. Fundamental to ubiquity is the notion of uniform nam-

[4]Some examples of this are "End-To-End Arguments in System Design, J.H. Saltzer, D.P.Reed, D.D.Clark, ACM TOCS, Vol 2, Number 4,November 1984, pp 277-288. and "The Architectural Principles of the Internet", Brian Carpenter, RFC 1958, 1996.

ing and access to resources – this implies common interfaces. These interfaces must capture much more complex and subtle facets of the underlying distributed components than existing distributed systems technologies allow. Thus another aspect of the ubiquitous system is the family of programming languages and APIs that let programmers build systems that work in the presence of failures, but with no safety net: The current world is one where the OS, the communications sub-system, and even the hardware conspire to mask faults at *lower layers*, and make life easier at higher layers – this layering abstraction and implementation is no longer so relevant. However, programming these systems is too hard at the moment for the anticipated set of possible applications one might wish to see deployed.[5] New ubiquitous system programming models are therefore required.

An aspect of the programming system challenge should be to find a minimal approach for the data structures and their semantics to allow decorative syntaxes which permit annotation of thread and communication with stochastic properties (e.g. HOT (theory)) Highly Optimized Tolerance this allows some interaction between intended and derived properties through introspection/reflection, as well as supporting the appropriate division and recombination of processing and data amongst individual under-resourced processing elements (e.g. in the sensor/telemetry end of the ubiquitous computing spectrum). Note that the reliability improvements achieved here apply equally well *in the large*, where a small-fold increase in redundant processing and storage would have massive pay-offs in availability, timeliness and, possibly, correctness.

Another important set of targets for the new programming systems is the improvement to design structures that yields intelligibility and (hence) maintenance and evolution.

The success of the approach at the architectural and implementation level, across the range of scales, will illustrate that the ubiquitous computing systems challenge is truly generic.

5 Some Demands from Exemplar Applications

An exemplar such as healthcare portrays a set of properties, including high availability across a scale with a level of trust.

[5]A senior engineer at Cisco, Dave Katz relates the typical experience of implementing the state-of-the art distributed routing standard, OSPF. A new engineer with a PhD can implement the standard (or similar ones) in around 3 months. It then takes 20 engineers 3 years to make it work reliably with other implementations. The correctness of the next level inter-domain protocol, BGP, is actually getting worse year by year – without more fundamental understanding, we could eventually reach a state where networks simply fail and cannot be fixed. As with Intel's processor road-map, the network routing road-map goes red, across the board, 3-5 years from now!

One way to capture this is to envisage a world where software and hardware carry no disclaimers. This does not mean that there are zero faults, but that we can model them, and, e.g., give refunds.

Not quite dependable systems evolution (see the strong S.E. GC), but strongly related.

Here we outline three possible applications of ubiquitous computing - these scenarios are chosen to illustrate different points in the problem space, and the consequential possible sources of Fear, Uncertainty and Doubt.

5.1 Always-on Healthcare

We envisage the entire population of the UK being permanently wired for various systematic monitoring of metrics such as heartbeat, skin conductivity, blood sugar, etc, and that this data is locally and remotely logged in a secure way via wireless communication. In the short term this is done mainly for people at risk. It might also be "phoned in" periodically rather than 24*7 telemetry.

In the longer term, for all. One obvious reason is for long term medical research. The next stage is to introduce intervention - firstly via remote diagnostics, and human advice; later via autonomic responses (e.g. triggering defibrillator's for cardiac patient undergoing unattended attack; appropriate other responses for epileptic (muscle relaxant?), diabetic, etc etc).

Third stage is to synchronise information (telemetry, long term data and local data) between patient, hospital and arriving emergency service/paramedic.

Such a synchronisation might be optimistic - as might the interventions above - the failure modes should be derivable with transparent risk levels being part of the provable properties of the system, together with an automatically derived explanation when there is a failure ("it ran the de-fibrillation despite there being a xxthat this was not heart failure, because there was a .00xxthis doing harm and a yy

5.2 Gathering Evidence, Privacy and Surveillance

In this exemplar, we take the current world in which most people have mobile phones, and most public, and many private spaces are under video surveillance, and envisage how agencies might use this data in ways that do not intrude on privacy, but enhance safety and offer better (in the sense of accurate, provable) evidence for criminal charges, without creating incentives to undermine privacy, and create disincentives for criminal behaviour without intruding on rights.

Two real world examples suggest directions for this.

1. In recent test cases, mobile phone call records (which can establish the phone's location quite accurately, at least to 100meters) have been used in

defence as "evidence" of a person's absence from a scene. Rightly they cannot be used as evidence of a defendant presence at a scene by a prosecution, since there is no proof (short of voice recording too, which would require prior warrant to tap the call) to establish that the person had the phone, whereas in the opposite sense, a defendant has the right to the benefit of the doubt (automatically). If there is a human witness, though, the phone record can be used as corroboration. What if there is good video evidence? How do we combine information of this type appropriately?

2. The congestion charging scheme for London roads starts in February 2003. In the original design, cameras used to monitor vehicles only record registration plates+location+time if the database lists the car as unpaid for. In this sense, there is no record of people's (strictly, vehicles, since we don't actually know who is driving!) movements who obey the law. This has been modified in recent declarations of the use of the cameras and records under the data protection act to allow the police and other authorities routine access to location data of law-abiding users. This poses several threats. Given there are so many people with access to the police computers it is relatively likely that someone can be bribed or blackmailed into giving over data. Drivers may then be embarrassed (e.g. by journalists reporting them "at their lovers nest") or lose their jobs (at the match instead of home sick) or killed (terrorist learns MPs routes to constituency and parliament).

This is an inappropriate and unnecessary design.

In both 1 and 2, we would lodge all data with the user (virtually) by insisting that if it is logged, it is encrypted with a system and keys chosen (from a competitive range) by users. This would be declared under the data protection act as correct use together with a standard set of exported interfaces, which would require appropriate authorisation to call.

Queries on the data must go via the user and/or this interface. In some cases, the queries would be required to be a

- rate limited (how often can a question be asked or similar questions be streamed?).
- of limited semantics (e.g. not "where were you at time x?" but "do you have a location that is not Y?").

this safeguards (and more as they become understood or required) would be provable properties of the system open to inspection by the users or their appointed expert advisers.

5.3 Zero Road Fatalities

More young people die on roads than from any disease. It is amazing that this is socially acceptable. It should be possible to build a system that reduces death for pedestrians, cyclists and drivers to zero. One non technical solution is to put a 6 inch spike on the center of the steering wheel - this incentives drivers interestingly. Another is to require a man to walk in front of each car with a red flag.

Realistically, it is already possible to get cheaper insurance for sports car~~d~~s if the system is fitted with a rate limiter. It is only a small step from this, and the social acceptance of mandatory safety features for drivers such as seat belts, and airbags, to having external automatic control of the vehicle. Add to this sensors, and possible monitoring (telemetry see above) on humans in locale (e.g. people on sidewalk, or nearby and on a trajectory), a simple control system could be built to adjust the speed of vehicles to a safe limit whenever someone is near. Such a system can be introduced slowly, and incentives (as in insurance above) built in to make it deploy rapidly. It does not have to introduce ridiculously low speeds (in any case, other acceptable schemes such as congestion pricing for roads have been introduced where speeds are often under 10 miles an hour under load, so arguments in favour of 45Mph in town are fairly hard to swallow:-)

The interesting problems with such a system are clear when you consider what a driver would do - basically, you would put your foot to the floor at all times. What then if the system fails?

Secondly, it gets very interesting if combined with an in car navigation system with communication between vehicles, road traffic monitoring and so on - could we just speak a destination, put an elastic band on the accelerator, and sit back and read the paper...?

6 Support from Theory

The systems we envisage have frequent reconfiguration including migration. They also need to support introspection/reflection.

To reason about their performance, we need not just bounds- we need stochastic models that can deliver plausible results on the continuous behaviour as well as discrete.

At the language level, it appears that there is a serious gap between theory and practice today. despite great advances in theory, it cannot provide what we need in the world of real systems. In a world of ubiquitous computing, an important new form of failure is interoperability between an old and new version of software where the interface API has evolved. High-level programming languages still need to develop a long way before they neatly encapsulate all that is required. Today, it is still impossible to write a complete operating system in a high-level language that does not have to violate the type-safety of that language at some point or other, with the most notable points being the marshaling code used for RPC and the code generation phase at the output of a compiler. There appears to be a fundamental issue stemming from the Russell Paradox that must be encapsulated or buried somewhere in all real systems. The correct place to hide it is still far from clear in the envisaged world of first-class ubiquitous computing. The wide variety of fundamentally different ways of processing XML using Java is another illustration of this problem.

The theory of trust and provenance needs some overhauling. It may be that the linkage between human intuitions about trust needs to be included since we will make these aspects of data and processing first class elements of any program, and many programmers will not be mathematicians!

7 Assessment of the Challenge

We have ordered these: Impact, Timeliness, Significance then Scale. This is clearly quite a subjective prioritisation, and open to debate (as is the whole document!).

Impact

Is there a clear criterion for the success or failure of the project after fifteen years?

Yes, we will have a HCAC with assurance, and an underpinning theory to explain it. we will also have a generic system model with others will use (i.e. proof of genericity would be that e.g. road transportation, or e.g. disaster recovery would also use it...)

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

yes, we admit of failure but explain when and why we get it.

Does it avoid duplicating evolutionary development of commercial products?

Yes, most commercial products do not provide generic solutions, rather are an ad hoc collection of sometimes elegant hacks.

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

Systems built on the new paradigms will have massively broader application.

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

Ubiquitous computing already does.

Does it appeal to the imagination of other scientists and the general public?

The applications, systems and theory (in that order) are explicable to Jane Q Public.

What kind of benefits to science, industry, or society may be expected from the project, even if it is only partially successful?

Cost savings and insurance will be easier to achieve:-)

Timeliness

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

A - Marc Weiser stated his vision in around 1990...

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

integration, theory, and service needs are just now kicking in.

What are the first steps? What are the most likely reasons for failure?

First random walk steps have been made, but architectural principles need a lot of work. Possible reason for failure include industry shortcuts, and lack of research coherence. Outright failure is highly unlikely: rather we may end up with a set of more costly, and less satisfactory solutions.

Significance

Is it driven by curiosity about the foundations, applications or limits of basic Science?

Yes- it would sure be nice to work in the light instead of in the dark when building the next Internet!

Scale

Does it have international scope?

Obviously given Ubiquitous Computing already came from US, components are made in the far east as well as across Europe (esp. wireless wide area).

How does the project split into sub-tasks or sub-phases, with identifiable goals and criteria, say at five-year intervals?

There are several separate pieces outlined above already. The ordering is difficult to pin down, but it is clear that we can separate out sensor networks, and zero-power systems into longer term pieces in terms of having an integrated whole. In the near term, algorithms and data structures to manage scale of failure and latency and other performance problems seem to be critical, as do the privacy questions. A strawman plan might be to put the privacy and programming models first, the scale second, and single systems architecture in the third five-year plan for the 10-15 year epoch.

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

We need to have coordination between systems (networks and OS), programming language, security, human factors, and applications groups.

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

One could start top down, but also work bottom up (we already are in the systems world). The existence of an exemplar *and* a theory challenge also provides for creative competition for this part of the challenges.

8 Outcome from the Challenge

The UK CS research community has become risk averse. The community engages in small (3 year, 2 person) projects. Moore's law, and associated observed growth in memory, secondary storage, and networking performance in all dimensions require 10 year plans.

We need projects, programmes, centres (virtual and real, focused and multidisciplinary) which have a lifetime of more than a decade.

Ubiquitous computing is a step change in the growth rate of computing in the world. It has taken on the order of 20 years to build the Internet and the World Wide Web (arguably 25 years) and we are only just starting to understand it. If the systems envisaged in this document are to be successful 15-20 years from now, we need a coherent, principled, funded program of work.

What exists now? There are a number of projects in the UK notably in telemedicine, wireless networks, and human factors in mixed reality.

What is needed? Stability of funding to reduce competition, encourage collaboration, include other disciplines, and provide continuity.

What will it bring? Solutions to some of the problems mentioned in this document, discovery of many new unforeseen problems, and some solutions for them as well.