Building the Case for Global Computing

THE GC STRATEGY GROUP

March 26, 2003

(v0.99)

This document outlines a vision for GC2 as a FET pro-active initiative in the *Framework Programme 6* of the European Commission.

GC2 will build methods, theories, languages, software technologies, and infrastructures to underpin several heterogeneous visions of future computing and bring about their integration.

V. CAHILL, J.L. FIADEIRO, S. HARIDI, M. HERMENEGILDO, J. VAN LEEUWEN, U. MONTANARI, M. NIELSEN, D. SANNELLA, V. SASSONE, P. SPIRAKIS, J.-B. STEFANI, M. WIRSING

COORDINATED BY: **VLADIMIRO SASSONE** <vs@susx.ac.uk>

A post FET-GC Scenario

Jane's Health Insurance. It is 2010. Jane, like all citizens, owns and controls her personal health record which contains her genome along with a detailed medical history. The record is kept on one of Jane's personal devices and is updated continuously by data from microgadgets travelling through Jane's bloodstream and from other implanted sensors. Irregularities trigger alarms, and in extreme cases lead to medical assistance being summoned.

All of Jane's personal devices are continuously connected via wireless links to the ubiquitous computing infrastructure, and she permits limited access to her health record. For example, her personal trainer's PDA keeps track of Jane's heart rate in order to prepare a targeted training programme. Jane agreed to allow a certain level of access to her employer to enable him to verify she is telling the truth when she calls in sick; access to data from her genome is required when her doctor prescribes drugs. But she is not required to, nor does she wish to, provide complete access to her health record when applying for insurance.

One morning Jane decides to download a program for monitoring her genome against an ever-expanding catalogue of genetic diseases. The code is certified for the GRID and, indeed, it is correct for that framework. Jane feels safe to use it. Unfortunately, the GRID code is designed for a trusted environment and has weak security. The fact that Jane is at risk of a genetic disease is leaked to her insurance company, which applies a tenfold raise to her premium.

Joe's Genome-Monitoring Program. Joe is a programmer trained at the European Institute of the Disappeared Computer. He completed his doctorate brilliantly and went on to take up employment with UbiquitousHealth, a major multinational company spun off the Integrated Project '*Folding 500 Proteins by 2005*' and market leader in personalised ubiquitous health monitoring. Joe wrote the genome-monitoring code referred to above. As recommended by his manager, he reused legacy code previously developed for an application where privacy was not a concern: protein folding by massively parallel computation dispatching data and collecting results from several thousand computers on the GRID. Joe duly obtained certification for his code, and UbiquitousHealth correctly released it.

After Jane's incident, the code – now in wide use – is taken up by Privacy'Rus, a leading company in security which intends to market a secure upgrading. Unfortunately the program is written in `Grid-C++`, an ad-hoc language with a poor concurrency model and non-existent resource abstraction. After long and costly efforts and ever-increasing delays, the Executive Board of Privacy'Rus decide to abandon the project to avoid an even greater loss of money.

This is what might happen if the European Commission decides in 2003 not to promote research such as FET-GC2 that cuts across and integrates future visions of computing.

Vertical Visions of Computing in the Future

We are currently witnessing the advent of several suggestive and very promising visions for a near future in which information technology and computing, in their multiple

forms, will be driving factors to improve our quality of life. AMBIENT INTELLIGENCE (AMI) envisions future scenarios where we embed the best of technology in our daily routines. Making this a reality demands the success of a variety of parallel, coexisting visions for models of computing. A convenient way to look at them is by classification into computing environments.

- PAN (Personal Area Networks). This is the environment where devices are located on networks we wear on ourselves. Today's devices include PDAs, ebooks, cellular phones; tomorrow will bring personal communicators, digital alter-egos, and Jane's microgadgets and sensors. This, as well as HAN below, is part of the vision of the DISAPPEARING COMPUTER (DC).
- HAN (Home Area Networks). This is the environment where smart home appliances cooperate to make our life more comfortable. It integrates seamlessly, for instance, home theatre, cable services, smart mirrors and refrigerators, and more.
- VAN (Vehicle Area Networks). This is our car's environment, where devices such as GPS, navigation systems, traffic information services, engine and brake control are mounted on moving parts and interact on small local networks.
- SCIENTIFIC. This is the realm of the GRID, the environment of highly parallel, ultrafast scientific computing.
- PROFESSIONAL. This is our workplace's environment, and includes messaging systems, virtual communities, teleconferencing, eLEARNING, and much more.
- BUSINESS. This environment is dedicated to economic interactions, and includes stock exchange, auctions, transactions, trading and negotiation. It is the realm of eBUSINESS.
- SOCIETY. This is the computational environment for the citizen and includes, for instance, eHEALTH and eGOVERNMENT.

Each of these scenarios embodies a self-contained, apparently complete, global vision of the world's computational infrastructure. However, they are all based on *separate* compartments of human activity, on single computational scenarios which take no notice of each other. Taking the word with insight, we could say that, paradoxically, these global visions are in fact "local." In this document we will refer to these approaches as "*vertical visions*."

Were we to develop all these vertical visions to perfection, still we would not necessarily have realised a better quality of life for the citizen, unless we manage to make appropriate provisions for them to cooperate. My home network can be perfect, and so can my car's, but until I am able to pass from one to the other effortlessly, the computer *will not have disappeared* for me. Our AMBIENT INTELLIGENCE dreams may turn into nightmares when people are trapped in between vertical visions. The point to appreciate here is that such bad dreams have nothing to do with the undoubtedly high quality and internal coherence of the individual visions. Each of these makes certain basic assumptions – e.g. about the extent to which other parties can be trusted – and these differ from one vision to another. As our opening scenarios illustrate, although there is nothing wrong with Joe's certified code, nor with Jane's downloading it, problems arise when moving between vertical visions. Since they have developed in isolation,

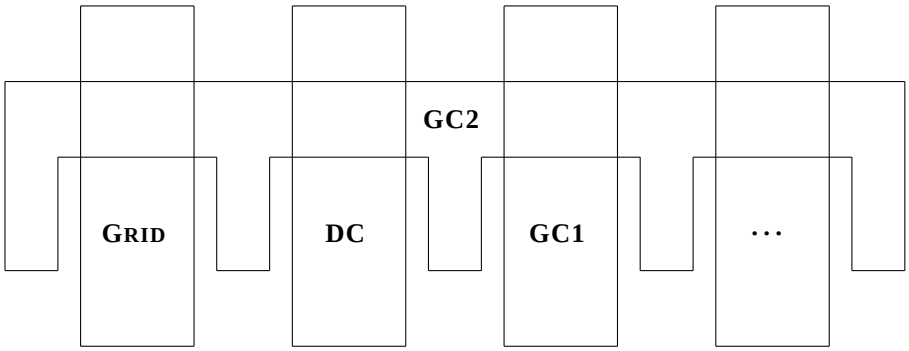

Figure : **VERTICAL VISIONS AND GC2**

nobody knows how they really relate to each other, or whether they do at all. From the technical point of view, as we will be arguing later, the vertical approach neglects the needs of Joe, who is naturally confronted with problems across their boundaries.

Inasmuch as it is wrong to decree that the different vertical visions cannot develop in parallel but must unite or dissolve, it is equally wrong to make no provisions towards their integration. We need to further each of them and, at the same time, focus our efforts on the innovative endeavour to build the horizontal "glue" between vertical visions. This is what FET-GC2 will be about.

Going Horizontal

The current incarnation of GLOBAL COMPUTING (GC1 in the following) is vertical too. It develops around a specific vision – we may call it CARDELLI'S MODEL – which states that on the global network it is ultimately impossible to abstract away from communication delays, network and node failures, requiring strategies for coping with these circumstances directly. As much as we can believe in its merits, it is fair to recognise that this is only *one* possible view of the future computational infrastructure, only *one* vertical vision. The GRID, for instance, represents a different model.

In order to be true to the adjective "*global*" in GC, we need to broaden our view and innovate by going horizontal. We need to build the foundations for the mutual integration of vertical models. For instance, our current work on security should open up to focus also on "*coordinated security*," whereby we consider security guarantees across the specific models of security (or insecurity) which apply to individual vertical visions. And rather than limiting ourselves to trust on the Internet, we should broaden our view and target a more general notion of "*trust negotiation*" while moving between vertical models. Besides defining the data model of the global net, we should also focus on a notion of "*third-party resource usage*" which spans across vertical visions.

Through instruments such as abstraction, virtualisation, modularity, encapsulation, software architectures, and abstract machines, Computer Science has succeeded in the past in getting to the core of issues and separating the essential from the incidental.

Such tools have brought fundamental advances to date, and we believe that they will be at the heart of the technologies enabling our vision. Complemented with research strategies developed specifically, and together with strength in system building, network technologies and foundational research, they will help us make an extremely challenging and speculative unifying vision into a focused and fruitful initiative.

GC2 will build *methods*, *theories*, *languages*, *software technologies*, and *infrastructures* to span across vertical visions, while furthering and realising each of them. In this respect, GC2 should actually be written as GC^2, and spelt out as "GLOBAL CONCEPTS FROM GLOBAL COMPUTING," where the first "global" is in the common sense of the term, the second in the sense of GC1.

An Alternative Post FET-GC Scenario

Jane's Health Insurance, revisited. It is 2010. Jane, like all citizens, owns and controls her personal health record which contains her genome along with a detailed medical history. One morning Jane decides to download a certified GRID code for monitoring her genome against an ever-expanding catalogue of genetic diseases. Fortunately, when Jane tries to access the code, the GC2-enabled global infrastructure kicks in and wraps the insecure genome-monitoring application inside a firewall which protects it against external access. Jane's privacy remains intact.

Joe's Genome-Monitoring Program, revisited. Joe, alerted by the GC2-enabled infrastructure, notifies his company UbiquitousHealth that the code needs to be improved. Since the code is written in a well-designed, resource-oriented, suitably multilevel abstract programming language, the project can be undertaken by Joe himself with excellent perspectives of success.

All this was enabled by the decision of the European Commission to fund FET-GC2.⋆

The Case for GC2

Overall, GC2 will complement and fill the gaps between vertical visions by focusing on delivering an integrated set of theories, languages, and infrastructures to work *on* and *between* vertical environments. Specific topics to be confronted include the following.

- RESOURCE USAGE. It is vital to monitor resource usage, as it is often essential to quantify the amount of resource being consumed in a transaction. Resource bounds must be built into programming languages and exposed as part of interfaces such as those of web services. The negotiation of resource bounds must be developed as part of entities' migration mechanisms, and these must be tested in experimental implementations.

⋆ The programme was such a success that the project officers responsible were presented with the Turing Award 2008 for "Research Vision." In 2009 the US Administration and the Central Bank of Japan applied to the EU for economic assistance, after the induced massive increase in European economic competitiveness brought about by GC2 led to most of their high-tech industry going bankrupt.

- SECURITY FRAMEWORKS. Besides classical language safety, this includes theories and systems of trust and authentication, as well as policies of access control and mechanisms for enforcing them. All this must be brought from a purely vertical to a convincingly broad horizontal dimension.
- MODELS OF INTERACTION. Access control policies and firewalls hide sites' activities from their environments and make interaction management a sensitive point. Issues include coordination, orchestration, reconfiguration, service discovery, open-endedness via connection/disconnection or via enhanced/reduced quality of service. The algorithmics of interaction should be developed, initially drawing inspiration from market and game-oriented mechanisms. Essential concerns are absence of deadlocks, mechanisms for assessing various forms of agreement and committed choice, different levels of description of interactions reflecting particular programming abstractions.
- ABSTRACTION MECHANISMS. To account for the enormous heterogeneity and variability of devices and resources, and yet have good design principles and tools, we need to conceptualise notions and extract suitably abstract models. This approach has proved to be a winner in many cases in the past. For instance, it is the abstract notion of "process" which differentiates Linux from the (classic) Apple OS; it is the abstract view of "file" which produces or avoids the effects of file fragmentation; it is the concept of "concurrency" which is now part of many languages and characterises, e.g., the passage from WIN95 to NT.
- PROGRAMMING LANGUAGE CONCEPTS. We need to cope with the interaction of similar yet subtly different concepts and issues arising on different platforms. This calls for new programming language mechanisms designed so as to deal successfully with abstractions such as those discussed above. This will empower the developers with the ability to design and implement correct programs to be deployed across the treacherous "gaps" between vertical visions. It is important here to go beyond the mere invention of new programming languages to develop novel and general language-level abstractions and concepts, that can then be applied to any language and impact real-world programming practice.
- COMPONENTS, MODULARITY, INTERFACES. We need to realise tomorrow's notion of compositionality. Component-based development and object-oriented methods fail hopelessly when the designer needs the ability to establish interconnections dynamically, "just-in-time," in a service-oriented manner. Interfaces, compositionality and reuse must now be understood in a dynamic model of configuration, if systems are to be agile enough to operate under the new economic rules. Tomorrow's interfaces will have to allow systems to navigate across different computing visions by dynamically subscribing services developed over a shared platform of global concerns.
- VALIDATION AND VERIFICATION TOOLS. Sharing tools among vertical visions will be an essential advantage for the designer at various steps of the design process, from requirement specification to system prototyping. The set of tools we need to develop includes test generators, symbolic interpreters, type checkers, finite state model checkers, theorem provers of various kinds.

- SOFTWARE PRINCIPLES AND TOOLS. New software principles, processors and development tools must be deployed to allow the production of high-quality horizontal applications. This means to endow software developers with instruments to tame the complexity of dealing with all the intervening issues, from resource usage to new compositionality models. In order to support the process of "debugging" and ensuring that the resulting systems adhere to specifications, new forms of analysis and abstract testing technology will be required.

Vertical visions will multiply rapidly in time, and we need to develop and put in place appropriate foundations, both in terms of system production and analysis. One particular point of interest is system stability. Differently from the DISAPPEARING COMPUTER, which admits "unpredictability by design," we want to be able to guarantee controlled behaviour, even in the presence of unpredictable interactions.

Methodology. The methodology of GC2 will be to look at the vertical visions as points in a multidimensional space, and to focus on the comprehension and control of the entire space, and not just of the individual points. Integration of theory, language design, and system building will be characteristic of our approach. We need a suitably general conceptual understanding of infrastructures, computational models, and language mechanisms, and theory plays a decisive role in unifying notions and yielding flexible, adaptable mechanisms. Reciprocally, feedback from experiments is needed for the theory to be robust. In order to deliver integration between vertical visions, GC2 will promote cluster activities across communities and seek the participation of members of a number of the communities represented by the vertical visions. A common methodological problem here is the cross-dissemination of potentially complex new theories. This problem has appeared before in Computer Science and indeed in all engineering sciences. One proved solution is to induce learning by experimentation via computer-aided tools providing distinct innovative functionalities traceable directly back to the new theories.

GC2's horizontal vision bears a *promise of success*. Besides focusing on the computational infrastructure of choice and on the end-user of the final products, it upholds the role of the developers, who after all are the pivotal enabling elements in the "technology chain." In the end, no matter how skillful the researchers bringing the vertical visions to life may be, no ensemble of vertical models will work without the horizontal "glue," without focused initiatives that address global, cross-cutting concerns.

Our approach takes strength from the tradition of the classical Computer Science approach, which during the years has made a lasting impact on practitioners, changing forever the way we think about computing. This includes achievements such as garbage collection and language safety, which are now fundamental components of modern programming languages; compiler technologies, which gave notions such as "just-in-time" compilation; type systems and logics, which constitute the base for "bytecode verification" and typed assembly languages; abstract data types and encapsulation, which led to object orientation, the dominant programming paradigm. Our past is full of success stories such as, to cite one of the European cases, `Erlang`'s. Very complex software for a telephone switching network designed by Ericsson in the 1990s was only completed after the decision to switch from a very large group of programmers using `C++`

to a smaller group using the higher abstraction of the `Erlang` programming language, which incorporates numerous research results in concurrency, abstract machines, and compilation techniques developed in the context of logic and functional languages.

Stumbling blocks. Yet, we face several stumbling blocks. As hinted above, the main concern is the bag of tools for the *middleman*, the developer. Many of the enabling technologies are already in place for the engineers to use. The real problem is the integration of the individual components. We need to develop significant advances in system engineering based on rigorous methods that can deliver guaranteed reliability, guaranteed security, and guaranteed resource consumption even for such dynamic systems as networks of autonomous entities. Security is another fundamental obstacle to using GC in practice. Current defences against malicious or erroneous mobile code are rudimentary and simply not up to the demands. They are typically based on cryptographic certificates issued by a trusted authority that guarantees the origin of code but little if anything about its properties. Once the security problem is solved, we will be able to use mobile agent technology to develop useful and innovative solutions to existing problems exhibiting, e.g., dynamic adaptability and robust behaviour in heterogeneous environments. GC2 will have to address the issues of coordination, control, reliability and quality of autonomous entities. How can we integrate and coordinate mobile entities with each other and with non-mobile programming systems and platforms? How can we be sure of the behaviour of the autonomous entities, which guarantees can we give for their quality and reliability?

As illustrated before, GC2 is directly focused on such issues.

The principal *enabling technologies* we need to put in place will provide means for developing safe and secure systems horizontally, spanning several vertical visions. They include developing programming and architectural abstractions, to build actual systems, as well as reasoning and analysis techniques, to bridge the gap between programming and modelling and make it possible to provide guarantees of such systems' behaviour. Cornerstones of such approach will be the notions we have identified above (and in the course of GC1) as central to our view as, for instance, the notion of resource. The current technologies for resource-bound certification are in their infancy, and substantial work is needed to develop the theory and then engineer the technology. At the same time, suitable models and mechanisms must be identified for trust in resource access control, based on histories and past behaviours, and managed dynamically over the network. "Proof-carrying code" and "proof-carrying authentication" are promising starting blocks to deal with security/privacy/trust and resource issues.

Placing GC2 on the Map

GC2 is focused on a novel horizontal dimension which is not present elsewhere, and provides a fresh approach to old and new challenges. To reinforce our statement, let us remark that:

- GC2 is not the GRID, because the latter is a regular, well structured, highly coordinated network infrastructure which connects powerful nodes, a hypothesis contrary to the dynamic nature of other kinds of vertical visions covered by GC2;

- GC2 is not DEPENDABILITY, because the latter is just one of the aspects of interest for a GC system; on the other hand, DEPENDABILITY focuses on systems of various natures, e.g. an aeroplane engine or a data processing system, while our emphasis is on highly dynamic networks of interacting components;
- GC2 is not the DISAPPEARING COMPUTER, because the latter focuses mainly on architectures and challenging artifacts for UBIQUITOUS COMPUTING; GC2 deals with the movement of entities across networks and domains' boundaries, as well as with systems belonging to different vertical visions, e.g. eBUSINESS and GRID applications; also, a defining trait of GC2 is the interest in being able to predict whether or not a system will work;
- GC2 is not FORMAL METHODS (FM), because it integrates theory and system building and aims to apply theories to the very real systems it builds; also the most theoretical aspects of GC2 differ deeply from FM: they deal with foundational calculi to understand and abstract over novel notions to be integrated in programming languages and techniques, rather than with deriving mathematical theories; GC2 treats issues which are far more challenging than FM's, shifting the accent from "development" to "engineering and analysis" techniques;
- GC2 is not GC1, finally, because the latter is vertical.

As for its *expected results*, GC2 will have a defining effect on the research community, and will complete the process of focusing together a community which was initially (just over one year ago) dispersed in pursuit of hundreds of different, yet tightly related goals. Besides the advances on the specific action lines illustrated before, one overall final objective will be to produce usable, practical implementations in real programming languages of innovative ideas arising from such work. These will feature guarantees of resource bounds, trust, reliability, security, and more.

The scientific and technological advances will narrow the gap between Europe and the *international research* community. Regarding this, the US is definitely ahead on proof-carrying code and typed assembly languages, despite the fact that many of the basic ideas originally came from European work on formal methods and type systems. Europe is still ahead on the basic issues, but this may not last long nor bring the technological advances we expect, unless the ideas are coordinated at the European level and focused in a relevant manner.

The results that can be expected from GC2 are illustrated by our list below of a number of possible *Integrated Projects* (IP) that can be anticipated under GC2. We remark that at this stage the list is only indicative, as IPs can be merged and new ones can be formulated; this process will also serve to focus the community.

- **LOGICS, LANGUAGES AND PROTOCOLS FOR RESOURCES.** It will deliver an integrated set of theoretical notions and practical protocols for the analysis and negotiation of resource bounds. It will provide logics to reason about qualitative and quantitative bounds, as well as programming languages for the respective applications. It will design and prototype infrastructures for third-party resource usage. Its impact will be major both on theory and applications, as it will put forward an explicitly *resource-oriented* programming paradigm.

- **METHODS AND INFRASTRUCTURES FOR SECURITY AND TRUST.** It will deliver a complete spectrum of security methods working resiliently across the boundaries of vertical visions and design the infrastructures needed to provide protection of hosts and agents alike. As sensitive decisions will be based on partial knowledge of the other party, trust will become central for security: notions such as behaviour history and negotiation of certified trust credentials will be developed and actualised on suitable *infrastructures*.

- **MODELS AND TOOLS FOR OPEN SYSTEMS.** It will provide a hierarchy of *interaction models* at different levels of abstraction, located between middleware and applications. The models will include both static and dynamic aspects, and will be supported by formal calculi, expressive graphical representations and analisys tools build on top of them. Focal points of the research will be the comparative analysis of such models' merits in relation to important computation environments, as well as a complete set of refinement and abstraction translations between different levels.

- **SYSTEM DESIGN PROCESS.** Industrial developers try to reuse as much as possible of their activity, from design to software and maintenance support, across different products and product lines. This IP will deliver a *design methodology* based on suitable software architecture styles, and will link it both to high level requirements and specifications and to existing object oriented programming systems. The methodology will be tested especially against design changes accommodating product developments across different vertical visions.

- **ALGORITHMS AND SOFTWARE PRINCIPLES FOR DYNAMIC SCENARIOS.** Collaborative infrastructures for future systems can only exist by virtue of realisable algorithmic principles for the interaction between system that are not centrally controlled. This IP will deliver suitable *algorithmics of interaction* for dynamic conglomerates of heterogeneous, (non-)cooperative systems and software principles for realising them reliably and verifiably in any application context. Concerning resources again, several situations will required us to characterise what can be computed efficiently in practice in the realm of dynamic multi-party interactions. Essential issues here are the vast amount of real-time interactions involving sharing of critical resources and essential nondeterminism in such interactions. The impact of the IP will be on the basic understanding of interactivity in dynamic environments, leading to principles for creating enriched local activity from global computing contexts.

GC2 will have a *Network of Excellence* (NoE) designed to bring together and integrate the broader community working in the area, and potentially open to members of projects adhering to one of the relevant vertical visions. It is envisioned that this will lead to a permanent European Research Institute for Global Computing with an annexed Doctoral School. Both the IPs and the NoE will have an open structure, with an initial kernel which will pro-actively seek to enlarge participation via specific calls, so as to build up the community in a bottom-up, competitive fashion.

The Legacy of GC1

Remarkably, after only one year of activity, FET-GC1 is producing a substantial defining and focusing effect on its research community. GC is already presented and introduced in courses and curricula, and has created new cooperation between researchers and projects. Different approaches begin to merge and, therefore, to yield better results and a deeper impact.

A few main notions are already emerging from GC1, and research starts to cluster and focus around them. These include certainly *resource control*, *distributed trust*, *security*, *design development*, as well as new *programming* and *analysis techniques* for GC systems. We believe that such themes will form the basis for a set of Integrated Projects in GC2, as illustrated above.

Concerning specific results to be expected for the end of GC1, we anticipate having new models and techniques, including functioning prototypes, for a variety of significant aspects spanning the entire arc of GC's interests. The specific list would indeed show that these results are the enabling elements for the research outlined in the list of Integrated Projects above. The result that realistically can and ought to be achieved in the next two years is to secure deep and solid theoretical underpinnings, as well as an encouraging experimental basis, on all such themes. We anticipate to be understanding the problems at their core, and to be approaching a good solution of certain problems, probably under some restrictions, while other problems will keep defying us. This will help us identify precise objectives for the Integrated Projects, and put us in the best starting situation for GC2. In order to consolidate and push through the advances generated by GC1, it is therefore appropriate to continue to work on the notions thereby identified and extend them "horizontally," so as to allow application in other vertical visions. The Integrated Projects of GC2 appear to be the best of contexts for that.