A New Look at Mobile Computing

Kimmo Raatikainen
University of Helsinki, Department of Computer Science
Helsinki Institute for Information Technology
Nokia Research Center
kimmo.raatikainen@{cs.helsinki.fi,hiit.fi,nokia.com}

Abstract— **In the recent years we have seen enormous change in information technology. The Internet has, at least in developed countries, become a commodity; mobile phones are used almost by everyone. The technology created for professionals 10–20 years ago is used by laymen—unfortunately with serious problems in usability, robustness and reliability. Now it is the time to go back to the basics and rethink the fundamentals. We ask three primary questions related to adequacy of current operating systems, programming models, and middleware. In each case the answer is no: existing solutions are not sufficient for tomorrow. Based on our observations and expectations, we have formulated six key research challenges that will enable to turn the current no-answers to yes-answers. These challenges include reconfigurable systems, context modeling, security-trust-privacy, software development, programming models, and efficient always-on connectivity.**

Index Terms— **Wireless Internet, reconfigurable systems, context modeling, security, trust, privacy, software development, programming models, and efficient always-on connectivity.**

I. Introduction

In recent years we have seen an enormous increase in Internet and mobile telephony. The next step will be a merging of these two technologies leading to the Wireless Internet. The Wireless Internet, however, will be much more than just internet access from mobile devices. Rather, the Wireless Internet will be almost invisible, as people will use mobile services and applications directly. Behind the scenes these services and applications will act as our agents, conducting searches and communicating with other services and applications to address our needs. The integration of mobile technology and the Internet paradigm will enable development of new context-aware applications. Besides traditional features such as user preferences, device characteristics, properties of connectivity, state of service (session) and usage history, the context includes features strictly related to user mobility such as current user's geospatial location (time and space).

The transition to the Wireless Internet will be much more demanding than the transition to mobile phones in voice services. The primary reason is the heterogeneous demands of various services and applications on the underlying computing and communications infrastructure. Direct use of existing Internet applications in a mobile environment has usually been unsatisfactory; services and applications need to take into account the specific characteristics of mobile environments.

Although Wireless Internet is already at its dawn, it is not clear what will be the final outcome. Various visions have been proposed. Mark Weiser spoke about invisible computing and ubiquitous computing [1], [2]. Leonard Kleinrock speaks about nomadic computing [3], [4]. Mahadev Satyanarayanan speaks about pervasive computing [5]. The European Commission speaks about ambient intelligence [6]. Wireless World Research Forum speaks about adaptable personalized ambient-aware services [7]. When these (and similar proposals) are carefully examined, the conclusion is that they are—at least—very close to each other, if not different names for the same thing. There may be some minor differences in emphasis but the core and major challenges are similar. In this paper we speak of the "thing" as the Wireless Internet.

Personal networking is still another notion that needs to be taken into account in the Wireless Internet. By personal networking we mean not only body and personal area networks but also protocol aspects of networking in the personal domain, digital home and other smart places like shopping centers, public and private transportation vehicles, ad-hoc communities, and networked (i.e., infrastructure provided) services (see Figure 1). In addition, the solutions should also work in a reasonable manner in non-smart places that do not have a high-bandwidth connectivity or any connectivity at all.

In order to bring Mark Weiser's dreams of invisible computing into reality for mass markets, that is for hundreds of millions of people, we still have a lot to do. In fact our claim is that we must go back to the fundamentals; to reconsider the foundations of mobile computing and communications.

The need of this reconsideration has its roots in the fundamental changes in usage patterns. Communication and computing devices move; users move and change their devices; (sub)networks in cars, trains and airplanes move; software moves from one execution environment to another. These changes can be characterized as personal networking domains. The objective is that the solution stacks for different domains are as similar as possible.

All of us hope that we can reuse existing software technology as far as possible. It saves us a lot of money now but may in the future turn out to be extremely expensive. Perhaps we should adapt the slogan used by the Queen Beatrice of the Netherlands: *"The natural resources are not a heritage from our parents but a loan from our children."* In mobile computing and communication the *natural resources* should be replaced by the *human-made artifacts*. The attitude will move the short-term focus from the current state we have to the future stage we leave behind us.

If we are not all the time ready for a revolution, we may

Manuscript updated August 16, 2004.

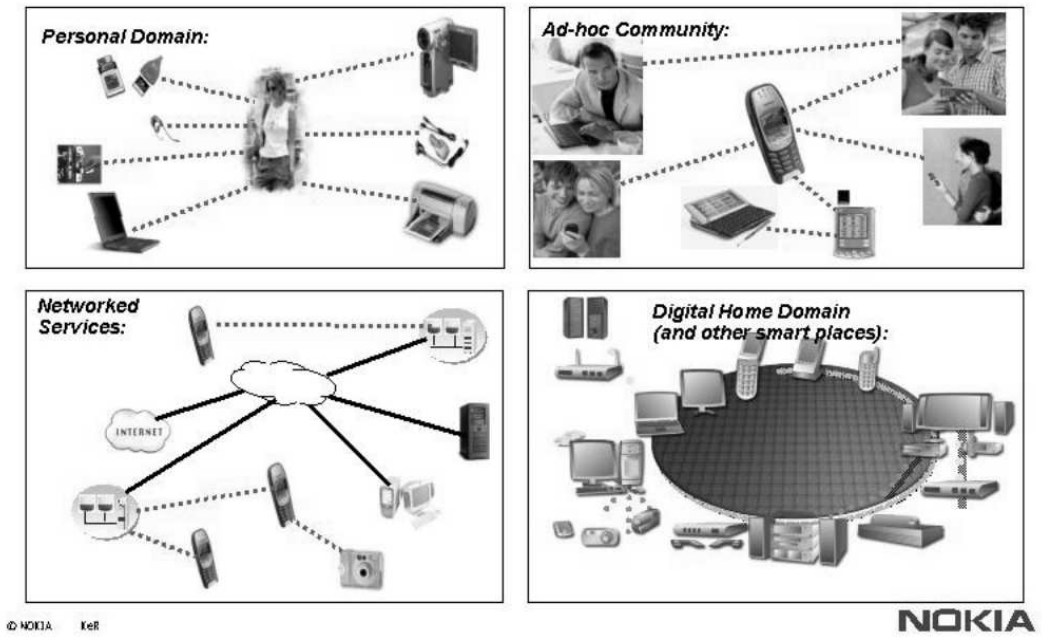


Fig. 1. Personal Networking Domains

miss the train and we may find ourselves at the trap of basing next releases of our products on existing legacy. We do not claim that today is the right time to forget all legacy systems. However, tomorrow it is even more costly to replace them. We should ask ourselves whether or not we want to produce pullovers for dinosaurs although the climate has already started to cool and sooner or later the dinosaurs will disappear.

In this paper we discuss challenges of the Wireless Internet. We start by stating our assumptions about future mobile applications. Based on the vision of adaptive, context-aware, personalized applications we have formulated three fundamental questions related to operating systems, programming models, and middleware. These questions are further elaborated into research challenges.

II. Future Mobile Applications

Since direct use of traditional Internet applications has not usually provided satisfactory end-user experience, we need to take a look at characteristics of future mobile applications. The characteristics that should be taken into account include communication, pocket size display as well as restricted memory size and computing power. We address communication and application composition. The issues of user interfaces are important but outside our expertise so we left them out.

A. Communication Characteristics

The most significant feature in communication needs of applications in the Wireless Internet Era will be diversity. All kinds of applications will be in use. Their Quality-of-Service requirements (bandwidth, latency, reliability) as well as communication patterns will be numerous. Some applications will also adjust their behavior according to the properties of connectivity. Future mobile terminals will have a few, say 3–6, applications simultaneously active. Some terminals will also be able to use different access technologies either simultaneously or one at a time. Therefore, the most important property of any communication system is its ability to handle a mixture of flows and traffic characteristics in a reasonable way.

When an application interacts with a human end-user, the interaction mode is a good classifier of applications. Most of the applications fall into one of the following three categories:

1) *Messaging.* These applications are non real-time. The underlying network can use store-and-forward, store-and-retrieve, or store-and-push mechanisms. The applications are not delay sensitive. The delays can be on the scale of seconds, minutes, or even hours. However, the delivery must usually be error-free. The message size varies from a few bytes to several megabytes. This category calls for efficient use of network resources.
2) *Interactive content retrieval.* These applications are nearly real-time. The users do not tolerate long delays. Some content formats, such as audio and video, are delay-sensitive but tolerate losses. In these applications playback buffers can be used to balance delay variation. Some other content formats, text in particular, are not delay-sensitive but require error-free transmission.
3) *Rich call.* These applications require real-time communication. An example from the present is voice call and that from the future is a multiparty videoconference.

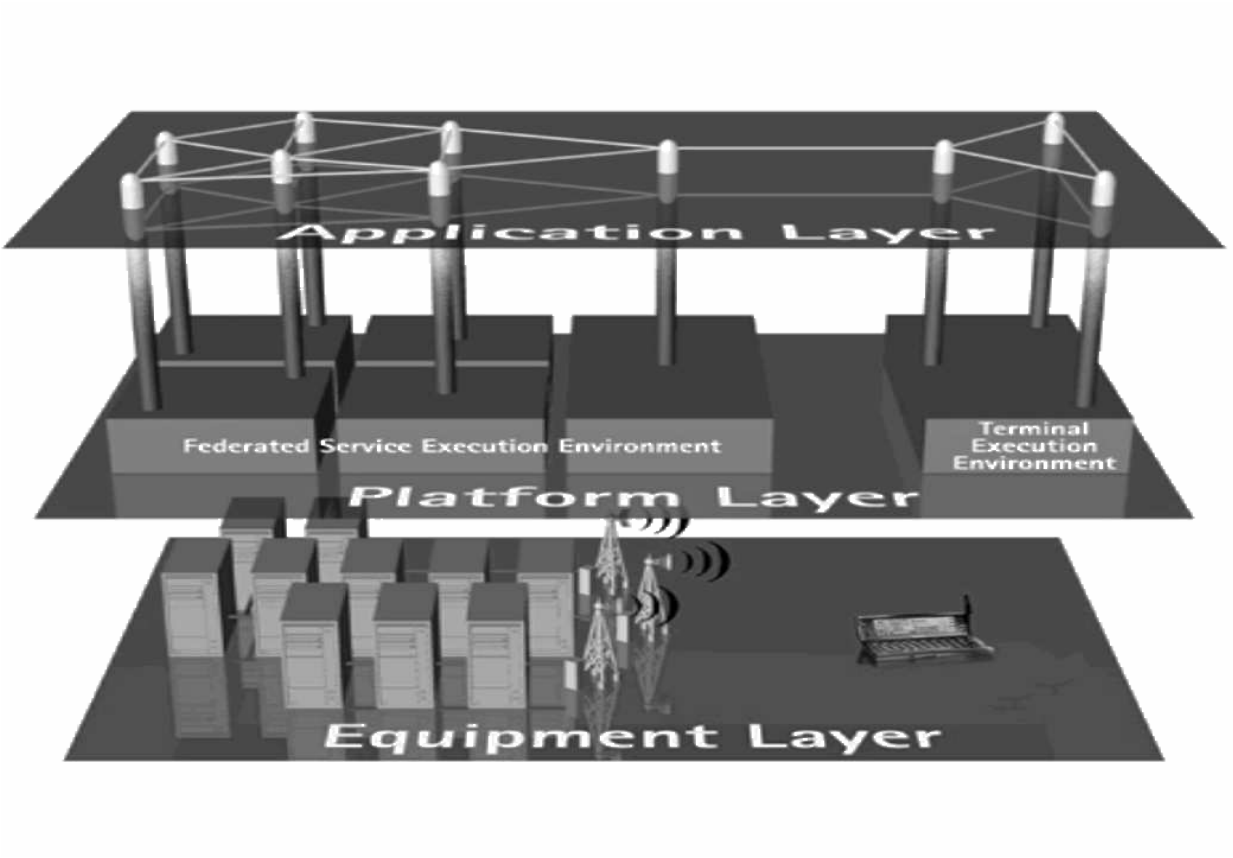


Fig. 2. Partitioning and Distribution of Application Logic

These calls will be annotated by drawings, text files, etc., like emails today have attachments.

When machine-to-machine applications, that is applications without human interactions, are also considered, then two main categories of applications can be identified: control and command applications and management applications. Their communication falls into the class of messaging. In control and command applications short messages (payload from 10 to 10,000 bytes) are typical. The message delivery must be timely, reliable and error-free. The delays can seldom be more than one second. The usual interaction pattern is either a simple request-reply or a single notification. In management applications the interaction patterns can be quite complicated involving a dialog of several messages. Management applications require reliable and error-free transmission but the time scale of delay bounds is in tens of seconds.

B. Application Composition

Fig. 2 depicts a highly abstracted vision of how an application is distributed among various application servers, network elements and terminal or end-user systems. It should be noted that, for simplicity, the figure only shows a single terminal device although multi-party applications will be much more important and challenging than one-party applications such as information browsing. In addition, we must also be ready to cope with end-user systems based on body area networks and home communication systems.

The execution environments or the platform layer consist of middleware, operating systems and protocol stacks that should support fast service development and deployment. The platforms should make it easy to divide the application logic into co-operating parts (someone may call them components), to distribute and configure these components as well as to redistribute and reconfigure them. Additional requirements for future mobile applications include adaptability to changes in the execution and communication capabilities, efficient use of available communication resources, dynamic configuration of end-user systems as well as ultimate robustness, high availability and stringent fault-tolerance.

The requirements for data accessed by these applications are quite similar. The execution environment should provide a consistent, efficiently accessible, reliable and highly available information base. This implies a distributed and replicated worldwide file system that also supports intelligent synchronization of data after disconnections.

C. Current Practice

The current trend in developing forthcoming telecommunication networks is to utilize Internet protocols. An immediate implication is that IP is the networking protocol, that is the layer 3 protocol of the ISO's OSI reference model. However, this is not sufficient. Other solutions—both above and below the IP protocol—are also needed.

The main stream in the Internet Community seems to be

the IP-TCP-HTTP-XML protocol stack, in spite of the facts that

1) TCP behaves poorly on transmission paths containing both wireless and wired links [8]–[10],
2) HTTP 1.0 is one of the best examples of using TCP capabilities in an extremely inefficient manner, and
3) XML is verbose and expensive to process.

In TCP transport over a wireless leg, there have been significant enhancements due to the proposals from the IETF transport area (particularly from working groups tsvwg [11] and pilc [12]). The HTTP transport over a wireless link has not yet been addressed in IETF but practical proposals are available [13], [14]. For XML the situation is even worse. The WAP forum has proposed a binary XML format [15] but that addresses only the efficient presentation of tags. Recently[1] W3C held a workshop on XML Binary Infoset [16] to review needs of alternative wire formats.

III. Research Challenges in Wireless Internet

Based on the vision of adaptive, context-aware, personalized applications we have formulated three fundamental questions:

1) Do the current operating systems support transparent and seamless reconfigurability?
2) Are the current programming models and tools adequate for context aware applications?
3) Do the current middleware solutions support development of such applications?

Unfortunately the answer to each of the three questions above is negative. Our current legacy base does not support adaptive, context-aware, personalized applications.

The operating systems claim to support plug and play functionality. However, more than often you end up in rebooting your system that can take several minutes. If you execution context is changing more rapidly than you can reboot your system, then you have a useless system.

Current programming models assume that exceptions are rare and independent from each other. However, the usual situation in reconfigurable systems will be that the situations currently regarded as exceptions will be the usual case. Almost never you will have the "full system" available; some parts will be unavailable, some parts will be only partial available. In such cases the try-catch statement will be insufficient to capture the real-world situation to which the system tries to adapt itself.

In order to address the requirement of ever-faster service and application development and deployment, several service/application frameworks/platforms have been introduced, usually referred as middleware. The current mainstream of middleware is based on the object-oriented client/server paradigm. However, the emerging personal networking requires a different paradigm that takes event-based reflective middleware into account.

In order to get a roadmap for turning the three no-answers to yes-answers, we have identified six fundamental research areas: *reconfigurable systems*, *context modeling*, *security-trust-privacy*, *software development and maintenance*, *programming models*, and *efficient wireless connectivity*. This set is by no means the only possible division of the Wireless Internet research space. The selected areas are neither orthogonal in the sense that solutions in one area affect solutions in other areas. Same or similar research topics and issues appear in more than one research area. Other divisions of the Wireless Internet research space can be found in [1]–[7], [17]–[21].

A. Reconfigurable Systems

Recent developments in mobile communication and small computing devices have had a tremendous impact on society. They have brought the dream of ubiquitous or invisible computing and communication closer to reality. Already in the near future communication and computing devices will be in a state that technologically enables mass market scale ubiquitous services and applications. Therefore, the main challenge will be software that fulfils the needs of adaptive, context-aware, personalized services and solutions.

The end-user devices of today are primarily integrated units like PDAs, laptops, or mobile phones. However, the situation will change in the future. The successor of the current mobile phone, or at least the successor of its successor, will be quite different. It will not disappear or lose its importance but its role will be very different.

A personal trusted device, we call it as *FuturePhone*, will be the core of the personal networking system. It probes its surrounding looking for suitable peripheral devices such as displays, input devices, processors, fast access memories and access points to communication channels. It dynamically builds up the most appropriate end-user system that can be auto-configured. The *FuturePhone* also probes for other similar devices in order to establish suitable ad-hoc communities and different kinds of sensors in order to extract context information associated to the current smart place. The *FuturePhone* also tries to detect actuators provide the means to affect properties of the smart place.

In order to be able to construct an end-user system and ad-hoc communities according to user preferences several enabling technologies must be available. First of all the system needs to be self-aware or reflective. This implies that the system must have a conceptual model of its current state and configuration as well as that of user preferences; see also Section III-B.

Configuration management is one enabler of reconfigurability. The best-known configuration descriptions today for wireless devices are W3C's CC/PP [22] and OMA's UAProf [23]. Both of them concentrate on terminal capabilities and cannot easily handle dynamic changes. They are based on W3C's RDF [24], but in serialization CC/PP uses XML [25] and UAProf WBXML [15]. FIPA's Quality of Service Specification [26] provides definition of an ontology describing message transport. FIPA includes also Device Ontology Specification [27]. In addition, OMA has on-going work on Device Management [28] service enabler. In describing software configurations CORBA Component Model [29] and

[1]September 24–26, 2003, in Palo Alto, Calif.

J2EE [30] use XML format. The on-going work in W3C's Device Independence Activity [31] is a part of configuration management, although the work is just in its beginning[2].

Another part of configuration management is the maintenance of ad-hoc communities. This membership management is the key challenge in group communication that has a plethora of non-standard solution proposals from the research community since the 70s; see, e.g., [33]. Here the challenge is in combining recent developments in IP and middleware level multicast with light-weight conceptual model of dynamic configurations.

Detection mechanisms, that is finding new devices, detecting devices that have departed or changed their state, are mandatory enablers of reconfigurability. But we also need notification delivery with filtering. Filtering is a way we can implement rules for deciding which events are important enough to be notified to the platform and applications.

Research and development in detection has been intensive. There are also good solutions for asynchronous notifications. Available service discovery mechanisms include IETF's Service Location Protocol (SLPv2) [34], Jini [35], OMG's Trading Object Service [36], Universal Description, Discovery and Integration of Business for the Web (UDDI) [37], Salutation [38] and Universal Plug and Play (UpnP) [39]. In addition, Bluetooth provides its own Service Discovery Protocol [40]. The proposed approaches work fine in their own environments. Therefore, the challenge is to obtain interworking, not to specify a brand new unified solution.

The next step should be focused on decision rules for reconfiguration. When our system has detected a change in the system state, it must decide whether or not some adjustments are needed. Dynamic system control theory is quite advanced for continuous systems [41]. Discrete systems are harder to control. However, we need to be ready to tackle multidimensional mixed models.

One additional aspect of reconfiguration is software download and upload as well as on-line upgrades and rollbacks. Recent virus attacks have clearly demonstrated that reconfigurable systems need strong protection against faulty, malfunctioning, even hostile software. The security of a reconfigurable system must be addressed as a whole, from hardware support to application software; see Section III-C for details.

B. Context Modeling

Context-awareness is considered as a fundamental property of future mobile applications to provide rich and consistent user experience. Context in the sense of computing refers to the physical and social situation in which computational devices are embedded.

Almost any piece of information available at the time of interaction can be seen as context information: identity, spatial information (e.g., location, orientation, speed and acceleration), temporal information (e.g., time of the day, date, and season of the year), environmental information (e.g., temperature, air quality, and light or noise level), social situation (e.g., who are you with, and people that are nearby), resources that are nearby (e.g., accessible devices, and hosts), availability of resources (e.g., battery, display, network, and bandwidth), physiological measurements (e.g., blood pressure, hart rate, respiration rate, muscle activity, and tone of voice), activity (e.g., talking, walking, and running), schedules and agenda settings.

Context-awareness means that one is able to use context information. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use.

The grand challenge is to create a flexible context modeling framework. The objective is to have efficient means of presenting, maintaining, sharing, protecting, reasoning, and querying context information. It should be noted that different applications in use typically have different views of context information. Therefore, also distributed data management is an essential enabler for context-aware applications.

In system modeling UML [42] from OMG is the lingua franca. More recently OMG has introduced Model Driven Architecture (MDA) [43] to raise the abstraction level. In MDA, Platform Independent Models (PIMs) are used to provide a resource model including properties, capabilities and relations. See also Section III-D for additional discussion on MDA.

In W3C the modeling is addressed in Web Services (Web Services Description Language (WSDL) Version 1.2 [44]) and in Semantic Web (RDF [24] and OWL Web Ontology Language [45]). Some aspects are also addressed in Organization for the Advancement of Structured Information Standards (OASIS) [46], particularly in Universal Description, Discovery and Integration [37].

The challenge is how modeling approaches and tools, all of which have their merits and strong industrial support, can be used in a coherent way for the conceptual models needed in future systems. One may try to combine them, but that could end-up in an overwhelmingly complex and heavy-weight tool set. A more plausible solution can be based on suitable model transformations. An ultimate alternative is do develop a brand-new *"universal modeling language"*.

C. Security, Trust and Privacy

The success of the Wireless Internet will totally depend on consumers' trust. Current Internet is vulnerable to worms, viruses, spam and fraud. It clearly demonstrates that fix it later does not work. Security, trust and privacy must be addressed from the very beginning of system design and on all levels: hardware, operating system, protocols, middleware. Moreover, trust is not of type on-off. Different tasks need different level of trust. In addition, different persons require different level of trust for the same task.

Today game applications have direct access to display processors or accelerators. When these devices become direct memory access (DMA) enabled, applications can overwrite operating system kernel unless each memory access goes through a protected address translation. Similarly, any digital rights management and payment system is of little use if a display driver stores the uncrypted video signal.

[2]W3C Working Group Note on Device Independence Principles [32] was published in September 2003.

Trusted Computing Group [47] works on open industry standard hardware building block and software interface specifications. The group has published Trusted Platform Module (TPM) Specification and TCG Software Stack (TSS) Specification.

An identity infrastructure is also needed to assure the coupling of the FuturePhone with the user in a distributed fashion. Preferences, attributes and desires are characteristics that discern an individual and represent his or her identity. The future Internet will be a web of relationship between different identities that must be trusted and accepted. The Liberty Alliance consortium [48] is specifying an architecture to enable federated network identity management.

Research issues in the "Bermuda triangle" of security, trust and privacy include:

- protecting system against unauthorized modifications,
- program validation/verification (what an uploaded/downloaded piece of software really does),
- trust modeling,
- how fragments of information can be efficiently shared in a controlled manner,
- key/certificate management, and
- implications of ad-hoc communities (what can be done without trusted servers).

D. Software Development and Maintenance

Software crisis has been a repeating theme in software engineering literature since NATO workshops 1968 [49] and 1969 [50]. At least once in decade we have read that production of software artifacts has serious problems. Since the seminal work of Fred Brooks Jr. [51] the silver bullet[3] has been the magical target that will help us to increase the productivity in software development. In the wireless world we meet all the problems of traditional software engineering. In addition, we must take into account restricted resources—power consumption, in particular—of handheld devices.

Already in the early days of software business in the 60s and 70s the problems of programming were addressed by proposing use of high-level languages, structural programming, functional decomposition, information hiding, etc. By early 80s most problems of writing *one-man* programs were solved, but software problems were still there; now on the next level of complexity. This was primarily due to applying computing to solve problems magnitudes larger and harder than in the 70s.

During the years object-orientation, functional programming, aspect-orientation, rapid prototyping, among others, have been claimed to be the breakthrough in the software challenge. Through all the years modeling has been one of the most advocated candidates. The modeling proposals have varied a lot over the time. The most recent proposal is the Model-Driven Architecture or MDA [43] by OMG. In most solutions automation is considered as the key enabler. This is not surprising since automation is the essence of computer science.

[3]According to Bookrs Jr. *Silver Bullet* is a technique or methodology that improves software productivity by factor ten.

In many papers addressing the software challenge production of software is equated to industrial manufacturing process. I am quite sure that this metaphor is not a proper one. In manufacturing repetition is the core challenge. You have the form to create replicas and your challenge is how to produce a great number of similar artifacts. In software production the creation of replicas is not a problem. When you have the master copy, then it is trivial to create copies with almost zero cost. Therefore, traditional manufacturing as software development metaphor is fundamentally wrong. Software development is inherently a design activity with no aspect of construction or manufacturing. The diseconomy of scale we see in software development is inherent to its design nature. We know from other industries that economies of scale apply only to manufacturing processes but not to design tasks. Accoprding to Hans Buhrer [52] "In a world where construction and demolition are free, trial and error is the approach of choice, and basic research is for suckers."

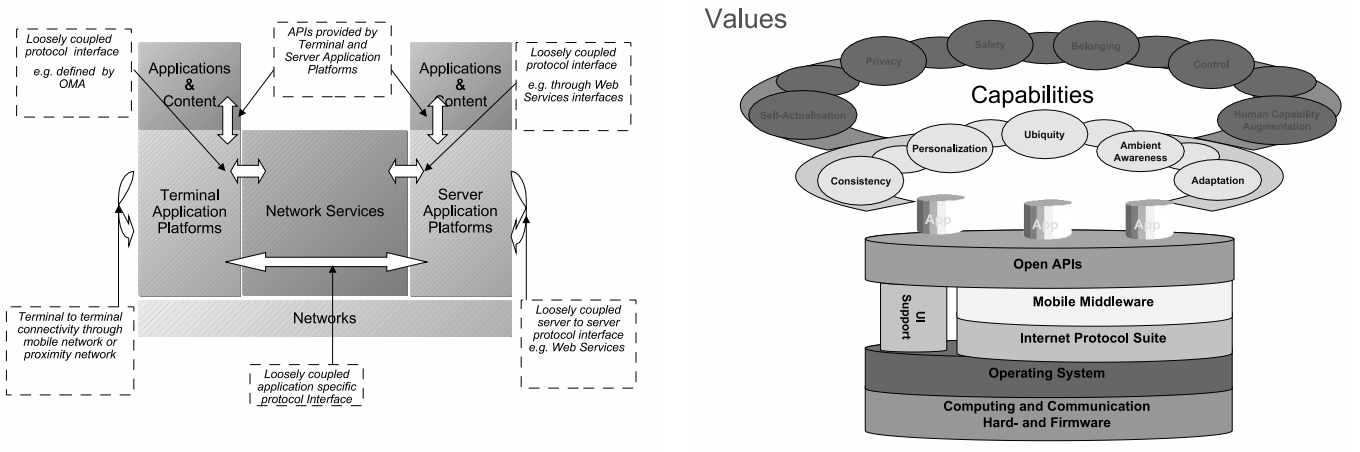
The promise of the Model-Driven Architecture is *"to move away from technology-specific code, helping to insulate business applications from technology specifics"*. [43] The MDA advocates promise easy software development through modeling. The only thing you need to do is to model your business processes. The rest is given "free" by MDA tools that generate the running code. This seems to be a similar *silver bullet* for software development as those proposed in the 70s, 80s, and 90s, but that did not work in practice[4]. However, we should not reject the MDA approach solely based on the unrealistic marketing hype. Behind the MDA there are many sound ideas and methodologies that need to be brought into a proper context, but we need to recognize its limitations. David Parnas' argumentation against SDI [54] is also valid against MDA. If we have never before done anything similar, we do not have any means to verify that we got it right at the first time. Therefore, MDA is useful but limited to specific domains where we have enough domain knowledge.

The service or software architecture is the fundamental challenge for the Wireless Internet. It is the key factor for the success of a software project. The inability to routinely create a good architecture is exactly what sets software development apart from established engineering disciplines. Architecture moves the focus of of a software developer from lines of code to components and their interconnections.

Without a conceptually coherent architecture it will be extremely hard or even impossible to achieve future-proof solutions. The hardest challenge is to have the right level of details: not too summary, not too detailed. If there are not enough detail, then we end up in an interoperability nightmare. On the other hand, if we go into too much details in the architecture, then we kill innovation and prevent progress.

The goal of the service architecture is to obtain a sustainable modular framework so that any module, at least in principle, can be replace without disturbing the others. Another goal—perhaps even more important—is to clarify thinking so that we do not mix apples and oranges. The crucial role of service architecture can be seen in the worldwide interest in service

[4]See Chapter 17 *"No Silver Bullet" Refired* in [53].



(a) OMA Architectural Framework

(b) Overall WWRF Reference Model

Fig. 3. Architectural Frameworks

architectures for the fourth generation systems or the systems beyond the third generation.

Service and software architectures are currently under wide development. Wireless World Research Forum (WWRF) [7] has come up with an architectural framework. In addition, architecture-related work has been carried out in Open Mobile Alliance (OMA) [55]. There are also regional activities like mITF [56] in Japan. Moreover, the industry—NTT DoKoMo [57] and Nokia [58], for example—has contributed its proposals to various forums.

The architectural framework of OMA (Fig. 3(a)) highlights the end-to-end view. The OMA framework is a logical architecture that does not propose any specific topology or physical location of servers. Furthermore, it does not suggest any hierarchy of protocol stacks to be used between domains. Network Infrastructure consists of the mobile network infrastructure that provides basic connectivity, transport and mobility support. Terminal and Server Application Platforms are the middleware solutions that are not application-specific. A platform contains functionality that can be shared by multiple applications running either in the terminal or on the server.

The service architecture developed in the Wireless World Research Forum is based on an I-centric approach. Future services will adapt to individual requirements implementing mass-market scale personalization. Fig. 3(b) outlines the WWRF architectural framework for I-centric communication. On top of the framework the *values plane* defines the interests of human end-users. The *capabilities plane* further elaborates functionalities needed in applications to fulfill those interests. The mobile middleware (Fig. 4(a)) provides a basic set of Generic Service Elements as well as Distributed Execution Environment. The Internet Protocol Suite (Fig. 4(b)) is the basis of an all IP-based communication system as the underlying communication infrastructure. It can reside on both fixed and wireless networks, where various access technologies can be used to attach different communication devices and end-systems to the wireless world.

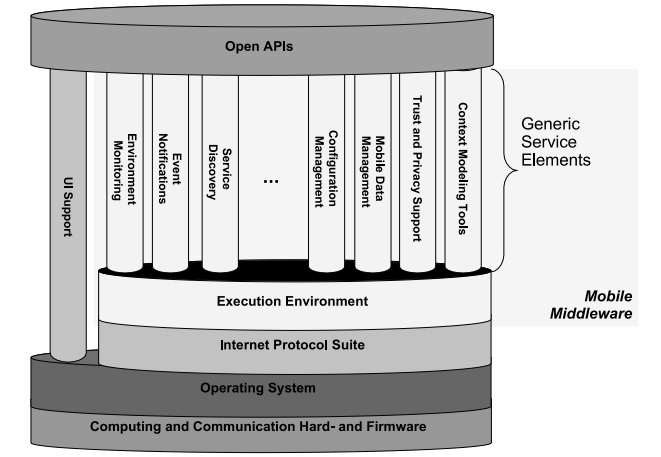
E. Programming Models

The current mainstream of middleware is based on the object-oriented client/server paradigm. The prime examples of object-oriented middleware (OOM) are OMG's CORBA [59], Sun's Java 2 Enterprise [30], Standard [60] and Micro [61] Editions, and Microsoft's .NET Framework [62].

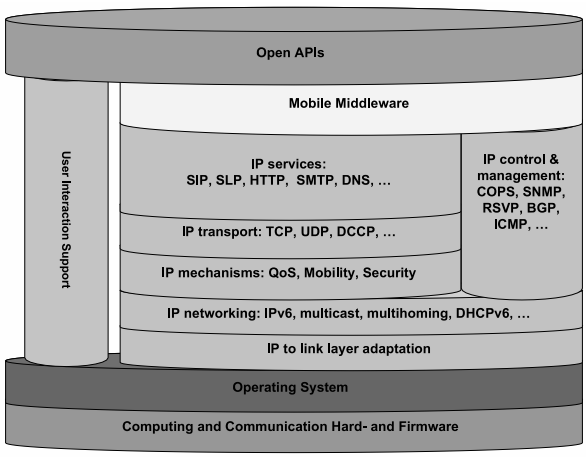
Another branch of middleware is based on message oriented middleware (MOM). Today, the difference between MOM and OOM is not so obvious. CORBA has one-way operations, Event Service [63], and Notification Service [64]. In the Java world Java Message Service (JMS) [65] and Sun Microsystems' Java System Message Queue [66] can be used for asynchronous messaging. The major MOM productions include IBM's WebSphere MQ [67], Microsoft Message Queuing (MSMQ) [68], Bea Systems' MessageQ [69], and TIBCO's ActiveEnterprise (including TIBCO Rendezvous) [70]. Open Source alternatives include ObjectWeb's JORAM [71] and xmlBlaster [72].

The emerging personal networking, however, requires a different approach that also takes into account paradigms of event-based and reflective middleware. Future mobile applications will be context-aware, adaptive and personalized. Therefore, the applications need to react to changes in execution environment. They must also modify their behaviour and/or configuration. Fig. 5 depicts a high-level structure of future mobile applications. The *world model* represents the state variables of interest. The *preferences/capabilities* defines the wishes of the user and the needs of the application. The *set of interests* defines the events that the application is interested in.

One of the fundamental problems is that of exceptions or try-catch statements in the current programming languages. In the future when systems and applications are adaptive, context-aware, and re-configurable, the usual situation will be that something exceptional is always going-on. Moreover, the situations currently covered by the catch-clause will not be independent. In other words, each try-statement will usually



(a) Mobile Middleware



(b) Internet Protocol Suite

Fig. 4. Detailed WWRF Reference Model

catch more than one exception. In these situations programs implemented using try-catch statements will be unreadable, "write-only" code.

Therefore, we need to go to the basics of programming models and languages. We need ways to express compensating actions and delayed actions. We may also need the condition-action programming model: to specify conditions under which each action is to be launched. This is nothing new for the students of computer science in late 70s and early 80s. We only need to bring back the basic ideas of Dijkstra [73] and Hoare [74] and to adjust them to current needs and technologies. The survey by Bal, Steiner and Tanenbaum [75] gives a good overview of programming languages proposed for distributed processing until late 80s.

In Dijkstra's Guarded Commands each `guarded list` is protected by a Boolean expression without side effects. For reactive systems we would like to generalize a program in simplified BNF:

```
<program> ::= <guarded action>+
<guarded action> ::=
              <event clause> -> <action>;
<action> ::= <statement> {; <statement>}*
```

The above looks very similar to Dijkstra's original proposal. However, we believe that the execution semantics needs to be relaxed a lot. The program runs until an explicit terminate statement is executed. Each action whose event clause is satisfied is executed in its private address space. When all the actions are completed, then the private address spaces are merged with the shared address space. Possible conflicts are resolved by the given conflict resolver for each data item. Finally, all the events generated by the actions are published in the underlying event system.

An alternative implementation could be as follows. All the statements belonging to actions whose event clause (condition) is satisfied are collected to an execution set. The execution set is examined for conflicts that are resolved before execution. The whole execution set (after conflict resolution) is executed as an atomic action.

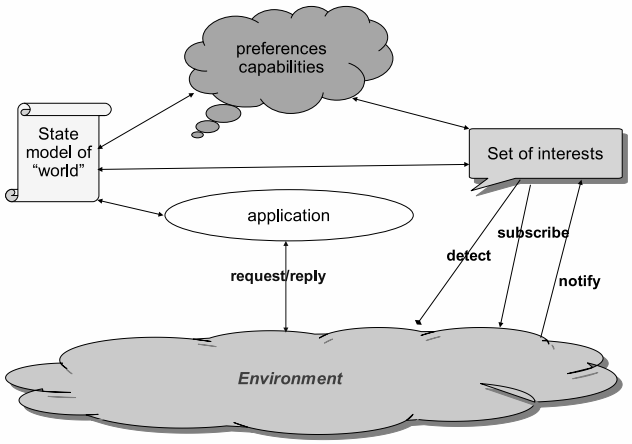


Fig. 5. High-level Structure of Future Mobile Applications.

The complexity of the execution semantics is due to the fact that we do not want to restrict the actions to be without side effects. That would significantly reduce the flexibility of the programming languages. In addition, we want that all actions fired by an event are executed instead of an arbitrary one as in Dijkstra's guarded commands and in its follow-ups such as CSP [76].

One could argue that event handlers in the current GUI systems are similar to our proposal. There are, however, significant differences. In GUI systems, there is only one subscriber for each event type. In addition, compound events lack support.

We have not yet addressed the event clauses. Typically they will include statements about a change in value of context variables. Below are three simple examples:

When I leave my office
When any of my buddies comes close
When my wife arrives our home

In other words, the event clauses are often personalized and utilize non-elementary semantics.

Applications based on the client-server paradigm are useful and widely used, but they are problematic in an ad-hoc community. In particular, most infrastructure services specified for 3G are based on the client-server paradigm. If we want future small wireless devices to work seamlessly in both environments, we need to find out how client-server applications are efficiently implemented for ad-hoc communities.

When infrastructure services are not available, then the server functionality must be distributed among the members of the community in order to achieve robustness against leaving members. Therefore, the "server" needs to be distributed among all members of the ad-hoc community. Here, an important challenge is the management of a distributed state.

In principle, the ad-hoc community could select one of its members as the server. In some cases this is a reasonable and easy solution but not a robust one. The feasibility of the "selected server" solution depends on recovery requirements. First of all, the application must have a soft state so that the new selected server can rebuild the application state from the scratch. Secondly, the timeliness requirement must be so loose that the application can wait until the soft state has been reconstructed.

Yet another research issue of fundamental importance is fault-tolerance. Replication, which is a commonly used method to achieve fault-tolerance in traditional distributed systems, is not alone sufficient. We cannot replicate everything onto a small handheld device. In principle, the baseline applications must remain operational, at least in a tolerable manner, even if some services of the underlying execution environment cannot be utilized. An alternative is that the application detects that it cannot make any progress and terminates or suspends itself. In any case the application cannot remain hanging and block all useful actions.

The programming model must have ways to express delayed and compensating actions to overcome the missing functionality in a partially available system. Disconnected operations and intelligent synchronization after reconnection are other important features to be supported. In addition, atomic transactions as used, for example, in the Argus system [77] may be needed in recovery from runtime failures.

To conclude, future context-aware, adaptive and personalized applications need an alternative programming model in addition to the currently dominating client/server paradigm. Essential features of such a programming model include condition-action style of programming, efficient access to context variables, lightweight state distribution among peers and a new approach to fault-tolerance.

F. Efficient Wireless Connectivity

As a communication channel, air is problematic. In the last ten years the progress in coding has significantly increased the capacity of wired channels. Unfortunately these fruits cannot fully be utilized on wireless channels. The applied coding is always a compromise between information density and redundancy providing robustness against interference. The basic problem with wireless links is instability in the sense that the level of interference varies in time and place, and according to environmental conditions.

It is very often said that the speed of 2 megabits per second, which is assumed to be available in the 3G, will be fast enough (or the tens of Mbits/s promised in 4G). However, in the history of computing, the spare capacity has never been left unused. In addition, one should notice that the capacity and speed of wired connections has increased much faster. For each magnitude of improvement in wireless communications there has been an improvement of 2-3 magnitudes in wired communications.

The problems of wireless links are not uniform. We have wireless LANs, satellite links, cellular networks, and short-range radio links. Each poses specific problems of its own. Therefore, wireless communications must be regarded as a polymorbid patient. Each of the access technologies mentioned above differs, at least in some aspect, from the others.

The support system of a nomadic user must be able to support communication links of different kinds. It must enforce the higher layers of communications to adapt to the situation at hand. However, the adaptation of communication is not sufficient, the behavior of applications also needs to be adapted.

Despite recent developments, we still have a lot to do in communication protocols:

- On the link layer we should examine adaptation to varying physical conditions.
- On the network layer we need seamless co-operation between mobility mechanisms, Quality-of-Service solutions, and security.
- On the transport layer we must find solutions for multiple simultaneous flows and different transport protocols.
- On the presentation layer we need bit-efficient wire formats.

In addition, we must also address cross-layer interferences and co-operation. In summary, we must have a reasonable solution on each layer since the performance can be destroyed on each layer. Anecdotically, when the protocol guys have improved one layer for mobile communications, the application guys have introduced two new layers that nullify the enhancements of the protocol guys. Java RMI is currently my favorite example of destroying the transport performance; for details, see [78], [79].

Additional research issues in (wireless) communications include:

- implications of ad-hoc networking,
- group communication,
- multicast and multihoming, as well as
- management of personal, session, and (sub)network mobility.

IV. Concluding Remarks

It is time to reconsider the fundamentals of computer science and how they are applied in systems and applications for Wireless Internet. We have identified six key research challenges: reconfigurable systems, context modeling, security-trust-privacy, software development, programming models,

and efficient always-on connectivity.. We do not claim that our division is the best one, but it is plausible. Alternative divisions of the mobile research space have been presented in various vision papers [1]–[7], [17]–[21]

Some important issues—although they are present as research areas in our research challenges—may have not received the attention they deserve. Such issues include mobile data management, and light-weight distributed management of state. In Open APIs we meet the cross-over problem of implementing same interfaces both in hardware and software. TO conclude, a lot, as identified in the research challenge section, is still to be done to make the "always on" vision of nomadicity, ubiquity, and pervasivity a reality.

ACKNOWLEDGMENT

During the last ten years I have had the priviledge to work with excellent colleagues and students as well as interesting partners in various European projects. Therefore, the list of persons I should acknowledge would be enormous. Instead, I only mention a few and apologize to the rest.

First of all I need to thank Timo Alanko for inviting me to the Mowgli-group and for many interesting and fruitful discussions during the years. Heimo Laamanen (now TeliaSonera; previously Sonera and Digital Equipment Corp.) and Heikki Saikkonen (Nokia Research Center) have been our reliable but demanding industrial mentors. My project managers—Markku Kojo, Oskari Koskimies, and Sasu Tarkoma—have contributed a lot, as well as my recent Ph.D.s: Heikki Helin, Stefano Campadello, Jukka Manner, and Andrei Gurtov. In addition, Mika Liljeberg, Jarkko Sevanto, Pauli Misikangas, Jaakko Kangasharju, and Pasi Sarolahti have had significant roles in our projects.

The international co-operation has had its impact. Randy Katz (UC Berkeley), Sebastiano Trigila (FUB, Italy), Sergio Palazzo (U. Catania, Italy), Dave Wisely (BT Exact, UK), Lazaros Merakos (U. Athens, Greece), and Hamid Aghvami (KCL, UK) have provided useful insight.

REFERENCES

[1] M. Weiser, *The Computer for the Twenty-First Century*, Scientific American, September 1991. pp. 94–104.
[2] M. Weiser, *Some Computer Science Issues in Ubiquitous Computing*, Communications of the ACM, July 1993. pp. 74–84.
[3] R. Bagrodia, W. W. Chu, and L. Kleinrock, *Vision, Issues, and Architecture for Nomadic Computing*, IEEE Personal Communications, December 1995. pp. 14–27.
[4] Cross-Industry Working Team, *Nomadicity in the NII*, Available at `http://www.lk.cs.ucla.edu/LK/lkxiwt/`.
[5] M. Satyanarayanan, *Pervasive Computing: Vision and Challenges*, IEEE Personal Communications, August 2001. pp. 10–17.
[6] K. Ducatel et al., *Scenarios for Ambient Intelligence in 2010*, Technical report, ISTAG, February 2001.
[7] Wireless World Research Forum, *Book of Visions 2001*, `http://www.wireless-world-research.org/`.
[8] T. Alanko, M. Kojo, H. Laamanen, M. Liljeberg, M. Moilanen, and K. Raatikainen, K., *Measured Performance of Data Transmission over Cellular Telephone Networks*, Computer Communications Review, October 1994. pp. 24–44.
[9] R. Cáceres, and L. Iftode, *Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments*, IEEE Journal on Selected Areas in Communications, June 1995. pp. 850–857.
[10] M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, and T. Alanko, *An Efficient Transport Service for Slow Wireless Telephone Links*, IEEE Journal on Selected Areas in Communications, September 1997. pp. 1337–1348.
[11] IETF TSVWG home page. `http://www.ietf.org/html.charters/tsvwg-charter.html`.
[12] IETF PILC WG home page. `http://pilc.grc.nasa.gov/pilc/`.
[13] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen, *Mowgli WWW software: Improved usability of WWW in mobile WAN environments*, Proceedings IEEE Global Internet 1996 Conference, 1996. pp. 33–37.
[14] B. Hausel, and D. B. Lindquist, *WebExpress: A System for Optimizing Web Browsing in a Wireless Environment*, Proceedings of MobiCom 1996. pp. 108–116.
[15] WAP Forum, *WAP Binary XML Content Format*, Document id WAP-192_105-WBXML-20011015-a.
[16] M. Leventhal, E. Lemoine, and S. Williams, *Binary Showdown*, XML Journal, Vol. 4, Issue 11, November 2003.
[17] A. J. Demers, *Research Issues in Ubiquitous Computing*, Proceedings of ACM PODC'94, August 1994. pp. 2–8.
[18] M. Satyanarayanan, *Fundamental Challenges in Mobile Computing*, Proceedings of ACM SigMobile, April 1997. pp. 1–7.
[19] G. Banavar et al., *Challenges: An Application Model for Pervasive Computing*, Proceedings of MobiCom 2000, August 2000. pp. 266–274.
[20] L. Kleinrock, *Breaking Loose*, Communications of the ACM, September 2001. pp. 41–45.
[21] R. Katz, *Adaptation and Mobility in Wireless Information Systems*, IEEE Personal Communications Magazine, Vol. 1, No. 1, 1st Quarter, 1994. pp. 6–17. Reprint in IEEE Communications Magazine, May 2002. pp. 102–114.
[22] W3C, *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*, W3C Working Draft, 25 March 2003.
[23] *OMA User Agent Profile Version 2.0*, `http://www.openmobilealliance.org/documents.html`.
[24] W3C, *Resource Description Framework (RDF)*, `http://www.w3.org/RDF/`.
[25] W3C, *XML Information Set*, W3C Recommendation, 24 October 2001.
[26] *FIPA Quality of Service Specification*, FIPA document number SC00094A, December 2002.
[27] *FIPA Device Ontology Specification*, FIPA document number SC00091E, December 2002.
[28] *OMA Device Management Version 1.1.2*, `http://www.openmobilealliance.org/documents.html`.
[29] *CORBA Component Model, v 3.0*, OMG document formal/2002-06-05, June 2002.
[30] Sun Microsystems, *Java 2 Platform, Enterprise Edition (J2EE)*, `http://java.sun.com/j2ee/`.
[31] W3C Device Independence Activity, `http://www.w3.org/2001/di/`.
[32] W3C, *Device Independence Principles*, W3C Working Group Note, 01 September 2003.
[33] G. V. Chockler, I. Heidar, and R. Vitenberg, *Group Communication Specifications: A Comprehensive Study*, ACM Computing Surveys, December 2001. pp. 427–469.
[34] *Service Location Protocol, Version 2*, IETF RFC 2608.
[35] Sun Microsystems, *Jini Network Technology*, `http://wwws.sun.com/software/jini/`.
[36] *Trading Object Service Specification*, OMG document formal/2000-06-27, June 2000.
[37] *OASIS UDDI Specification: Universal Description, Discovery and Integration of Business for the Web*, (UDDI Version 3), `http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm`.
[38] Salutation, `http://www.salutation.org/`.
[39] UPnP Forum, *Universal Plug and Play*, `http://www.upnp.org/`.
[40] *Bluetooth Service Discovery Protocol*, `https://www.bluetooth.org/spec/`.
[41] T. Glad and L. Ljung, *Control Theory, Multivariable and Nonlinear Methods*, Taylor and Francis, 2000.
[42] *Unified Modeling Language (UML) Specification 1.5*, OMG document formal/2003-03-01, March 2003.
[43] OMG, *Model Driven Architecture*, `http://www.omg.org/mda/`.
[44] W3C, *Web Services Description Language (WSDL) Version 2.0*, Two W3C Working Drafts, 10 November 2003.
[45] W3C, *OWL Web Ontology Language*, A set of W3C Candidate Recommendations, 18 August 2003.

[46] Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org/home/index.php.
[47] Trusted Computing Group, https://www.trustedcomputinggroup.org.
[48] Liberty Alliance Project, http://www.projectliberty.org/.
[49] P. Naur and B. Randell, (Eds.) *Software Engineering*, Scientific Affairs Division, NATO, 1969. Available at http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF
[50] B. Randell and J.N. Buxton, (Eds.) *Software Engineering Techniques*, Scientific Affairs Division, NATO, 1970. Available at http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF
[51] F. P. Brooks, Jr., *No Silver Bullet: Essence and Accidents of Software Engineering*, In Information Processing 86, IFIP/Elsevier, 1986. pp. 1069–1076. Reprint in IEEE Computer Magazine, Vol. 20, No. 4, April 1987, pp. 10–19.
[52] H. K. Buhrer, *Software Development: What it is, What it should be, and How to get There*, ACM SIGSOFT Software Engineering Notes, March 2003. pp. 1–4.
[53] F. P. Brooks, Jr., *The Mythical Man-Month, 20 Anniversary Edition*, Addision-Wesley, 1995.
[54] D. L. Parnas, *Software Aspects of Strategic Defense Systems*, Communications of the ACM, December 1985. pp. 1326–1335.
[55] Open Mobile Alliance, http://www.openmobilealliance.org/.
[56] Mobile IT Forum, http://www.mitf.org/index_e.html.
[57] H. Yumiba, K. Imai, and M. Yabusaki, *IP-Based IMT Platform*, IEEE Personal Communications, October 2001. pp. 18–23.
[58] Nokia, *Mobile Internet Technical Architecture, Parts 1-3*, ISBN 951-826-671-9, IT Press, 2002.
[59] *Common Object Request Broker Architecture (CORBA/IIOP)*, OMG document formal/2002-12-06, December 2002.
[60] Sun Microsystems, *Java 2 Platform, Standard Edition (J2SE)*, http://java.sun.com/j2se/.
[61] Sun Microsystems, *Java 2 Platform, Micro Edition (J2ME)*, http://java.sun.com/j2me/.
[62] Microsoft, *.NET Framework*, http://www.microsoft.com/net/.
[63] OMG, *Event Service Specification*, OMG document formal/2001-03-01, March 2001.
[64] OMG, *Notification Service Specification*, OMG document formal/2002-08-04, August 2002.
[65] Sun Microsystems, *Java Message Service*, (JMS) http://java.sun.com/products/jms/.
[66] Sun Microsystems, *Sun Java System Message Queue*, http://wwws.sun.com/software/products/message_queue/index.html.
[67] IBM, *WebSphere MQ*, http://www.ibm.com/software/integration/wmq/.
[68] *Microsoft Message Queuing*, (MSMQ) http://www.microsoft.com/windows2000/technologies/communications/msmq/default.asp.
[69] Bea Systems, *MessageQ*, http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/more/messageq/.
[70] TIBCO, *ActiveEnterprise*, http://www.tibco.com/solutions/products/default.jsp.
[71] ObjectWeb Consortium, *JORAM*, http://joram.objectweb.org/.
[72] xmlBlaster.org, *Open Source for MOM*, http://www.xmlblaster.org/.
[73] E. W. Dijkstra, *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*, Communications of the ACM, August 1975. pp. 453–457.
[74] C. A. R. Hoare, *Communicating Sequential Processes*, Communications of the ACM, August 1978. pp. 666–677.
[75] H. E. Bal, J. G. Steiner and A. S. Tanenbaum, *Programming Languages for Distributed Computing Systems*, ACM Computing Surveys, September 1989. pp. 261–322.
[76] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.
[77] B. Liskov, *Distributed programming in Argus*, Communications of the ACM, March 1988. pp. 300–312.
[78] S. Campadello, H. Helin, O. Koskimies, and K. Raatikainen, *Performance Enhancing Proxies for Java2 RMI over Slow Wireless Links*, Proceedings of the Second International Conference and Exhibition on The Practical Application of Java (PA JAVA2000), Manchester, UK, 12-14 April 2000.
[79] S. Campadello, H. Helin, O. Koskimies, and K. Raatikainen, *Wireless Java RMI*, Proceedings of the 4th International Enterprise Distributed Object Computing (EDOC2000), September 2000.

Kimmo Raatikainen Prof. Kimmo Raatikainen has M.Sc. (1983) and Ph.D. (1990) degrees in computer science (University of Helsinki). He was a system manager at Finnish State Computer Centre, Division of University Support, 1981–1985. From 1986 he has been employed by Helsinki University Computer Science Department as a research and teaching assistant, as an Assistant Professor, as an Associated Professor and as a Full Professor (from 1998). From January 2000 he as been part-time Principal Scientist (and Research Fellow from April 2004) in Nokia Research Center and from January 2002 also part-time Principal Scientist leading the Mobile Computing research area in the Helsinki Institute for Information Technology, which is a joint research institute of the University of Helsinki and the Helsinki University of Technology. Prof. Raatikainen has participated in a leading role several European projects including DOLMEN (AC036), MONTAGE (AC325), PRIME (AC370), HPGIN (ESPRIT 29737), BRAIN (IST–1999–10050), CRUMPET (IST–1999–20147), MIND (IST–2000–28584). He has also led several national research projects (funded by National Technology Agency of Finland—TEKES—and industry) on mobile computing, wireless communication, middleware for mobile computing and on telecommunications software architectures. His current research interests include wireless communication, middleware for mobile distributed systems, and operating systems. He has over 100 scientific publications in these areas. He has supervised 7 Ph.D. thesis and c. 90 M.Sc thesis. Currently 19 Ph.D. students are conducting their thesis under his supervision. Prof. Raatikainen has actively participated in OMG since 1997 leading in the Telecom DTF the specification of Wireless Access and Terminal Mobility in CORBA. He is also the representative of the university at W3C advisory board and national delegate in the ESF MiNEMA project.