

Requirements for a Scalable Mobility Architecture

Draft 2:

We are looking for your thoughts and comments!!

Some changes based on comments are shown in the table of contents.

Text changes are shown in [blue](#).

Johannes Ernst and Joaquin Miller
NetMesh Inc.

Version 1.1 September 2004

To avoid spam, we created a special e-mail address:

comments+mobarch@netmesh.us

In the spirit of the Creative Commons-Attribution-ShareAlike-License,
we will give credit to all contributors.

This document is distributed under the Creative Commons Attribution-ShareAlike License. To paraphrase the license, this means you are permitted to copy, distribute, display, and perform this work. In return, you must give the author ("NetMesh, <http://www.netmesh.us/>") credit. You are also permitted to distribute derivative works, but only if you distribute them under the same license. For details, please refer to the exact license text at <http://creativecommons.org>. We would appreciate it if you let us know when you create a derivative work, but you are not required to do so under this license.

NetMesh, Situational, Situational Information Grid, One Life and the NetMesh logo are trademarks or registered trademarks of NetMesh Inc. No rights in these trademarks is granted.

Table of Contents

1	Introduction	4
2	Requirements for a Mobility Architecture	4
2.1	Basic technical requirements	4
2.1.1	Robust in the face of intermittent connectivity.....	4
2.1.2	Multi-network support	5
2.1.3	Support for a variety of devices	5
2.1.4	Power status / tradeoff support.....	5
2.1.5	Security.....	5
2.2	Requirements of system development.....	6
2.2.1	No major change in the corporate development process for corporate applications	6
2.2.2	Single code branch for thin and rich clients on a variety of devices.....	6
2.2.3	Centralized debugging, monitoring and upgrades.....	6
2.3	Requirements caused by mobile user behavior	6
2.3.1	One Life™: Seamlessness across many servers, business and private.....	6
2.3.2	Leverages traditionally unusual – but increasingly common – input devices	7
2.3.3	Situation-focused, not application-centric.....	7
2.4	Requirements for a vibrant mobile software ecosystem.....	8
2.4.1	Enables decentralized development, deployment and operation.....	8
2.4.2	Supports a large variety of types of software	8
2.4.3	Supports a variety of business models.....	8
2.4.4	Encourages innovation	8
3	Necessary Architectural Qualities and Features	9
3.1	Basic technical qualities and features.....	9
3.1.1	Operates while disconnected	9
3.1.2	Information consistency	9
3.1.3	Synchronizability.....	9
3.1.4	Roamability.....	9
3.1.5	Hand-offability...NEW	9
3.1.6	Location awareness...NEW	9
3.1.7	Updates from user...NEW	9
3.1.8	Push to user...NEW	9
3.1.9	Server initiated actions...NEW	9
3.1.10	Multi-Network.....	10
3.1.11	Non-IP protocol support	10
3.1.12	Frugality.....	10
3.1.13	Security.....	10
3.1.14	Privacy	10
3.1.15	Device and platform transparency.. SPLIT	10
3.1.16	Network transparency...NEW	10

3.1.17	Provider transparency	11
3.2	System development qualities and features.....	11
3.2.1	Stability.....	11
3.2.2	Process neutrality.....	11
3.2.3	Tool neutrality.....	11
3.2.4	Client-side flexibility	11
3.2.5	Development artifact conciseness.....	11
3.3	Operational qualities and features...NEW HEADING	11
3.3.1	Observability...CHANGED TEXT	11
3.3.2	Manageability...NEW	11
3.3.3	Maintainability.....	11
3.3.4	Deployment support...NEW	11
3.4	User behavior-related qualities and features...REORDERED.....	12
3.4.1	Situational awareness.....	12
3.4.2	Situational appropriateness.....	12
3.4.3	Proactivity	12
3.4.4	Semanticness.....	12
3.4.5	Timeliness	12
3.4.6	Sensitivity.....	12
3.4.7	Collaboration...NEW	12
3.4.8	Usability	12
3.4.9	Media transparency.....	12
3.4.10	Mode transparency.....	12
3.4.11	Coherence	12
3.5	Business ecosystem qualities and features	13
3.5.1	Business model neutrality.....	13
3.5.2	Applicability.....	13
3.5.3	Federatability.....	13
3.5.4	Resilience	13
3.5.5	Business opportunity.....	13
3.5.6	Incentive to innovate	13
4	Evaluation Matrix	14
5	Contributors ...NEW	16
6	References	16

“Seamless ubiquitous access to a user’s uniquely customized personal environment is the holy grail of mobile computing.”

— Michael Kozuch, et al. ^[1]

1 Introduction

An effective architecture for mobile software:

- addresses the **technical challenges** of mobile computing. For example, it enables the mobile user to be productive even when temporarily disconnected from the network.
- offers a software **usage model** to the mobile user that is compelling and that leverages the unique advantages of the mobile device, while sidestepping or avoiding its disadvantages (e.g. the lack of a full-size keyboard, but the availability of location-related information through technologies such as GPS, Bluetooth, or RFID)
- encourages the **growth of a mobile software business ecosystem** consisting of infrastructure, application, content and service providers that can effectively address the mobile software market opportunity represented by 1.5 billion mobile devices in use already.

At the time of this writing, no universally accepted mobility architecture has yet emerged; however, the requirements for such an architecture are becoming fairly [well](#) known (for example, see Intel’s at [1]).

This document attempts to collect the requirements for such an architecture and identifies the features and qualities that it needs to have. We hope it is useful when evaluating various proposed mobility architectures and technologies (such as mobile software from the OS to middleware).

This document goes beyond the sometimes narrow technical focus on particular technical challenges that need to be addressed (e.g. synchronization between a mobile device and information on a server, PC or other mobile or stationary device). It also considers requirements arising from the development process employed to construct mobile systems and the operation of mobile systems.

Further, it considers technical requirements that arise from the increasingly recognized need by the participants in the mobile business ecosystem to gain agreement around a core architecture that catalyzes rapid growth and innovation.

2 Requirements for a Mobility Architecture

Requirements for a mobility architecture can be categorized into four groups: these are basic technical requirements, requirements arising from the needs of systems development and operation, those rooted in the mobile user’s behavior, and requirements that must be met in order to enable a vibrant mobile technology ecosystem.

2.1 Basic technical requirements

2.1.1 Robust in the face of intermittent connectivity

In most cases, it cannot be assumed that the mobile device is permanently connected to a reliable network. Since much mobile software interacts with, or even depends on software located at other network nodes (e.g. servers [and other mobile devices that come into range and leave again](#)), mobile software must be robust in the fact of intermittent connectivity.

Ideally, a suddenly disconnected device should not impact the operation of the mobile system at all; in practice, mobile software must approximate this ideal as much as possible through local data caching, error correction and other techniques.

2.1.2 Multi-network support

With the proliferation of wireless networking technologies and a growing variety of business models for mobile networks, a mobile device may, at any point in time or over time, be connected to the internet through more than one network. Different networks typically have differing cost and performance profiles, thus requiring often non-obvious tradeoffs.

For example, at some point in time, a mobile device may have simultaneous GPRS and WiFi access to the network. While the WiFi network may be preferable over GPRS when available due to its speed and (typically) lower per-megabyte cost, its coverage area is much smaller and thus mobile software may have to switch between networks when the user moves around.

Similarly, a mobile device may be able to communicate with nearby other mobile or stationary devices through more than mechanism (e.g. Bluetooth, a wide-area wireless network). As mesh networking technologies mature, which network link constitutes the “best” link to the internet becomes even more complex to determine, and more dynamic.

Changes in the network topology, and advances in the creating and management of networks (e.g. mesh networking) should not impact higher levels in the networking / software stack; if they did, application-level software would be much harder to develop. The architecture should take this into account (e.g. by providing a tunneling mechanism).

2.1.3 Support for a variety of devices

Mobile devices come in an extremely broad range of variations along many different dimensions (e.g. keyboard/not, display size, processing power, storage capacity, connectivity, peripherals). If mobile software had to be specifically developed or ported for each individual variation, this would mean economic infeasibility of most potential software solutions.

Thus, the mobility architecture should provide architectural support to reduce the amount of device-specific development and/or testing to the maximum extent possible. Further, the specific device used by a user should not impact the remainder of a (distributed) mobile system.

2.1.4 Power status / tradeoff support

Mobile devices' batteries contain limited energy, and the (at any point in time remaining) energy must be carefully managed.

Mobile software must not only be functionally correct but also power-conscious. If possible, mobile software should degrade gracefully if its power budget is being reduced, such as when batteries run low. The architecture should provide facilities that allow software to be aware of the power status and take appropriate action.

2.1.5 Security

As mobile devices and their networks cannot be secured through limited physical access, as is often possible for desktop computers, security poses a much greater challenge. Security requirements break down into three primary categories:

- The exchange of information between the mobile device and other nodes on the network (such as servers, other mobile devices, stationary devices in the proximity of the mobile device, etc.) must be protected against eavesdropping and other kinds of attacks (such as man-in-the-middle attacks).
- Information stored on the mobile device must be protected in a manner that renders it unusable for a non-authorized user (e.g. in case the device is lost or stolen).

- User identity and credential information stored on the mobile device for remote nodes (such as servers) must not be usable by a non-authorized user (e.g. in case the device is lost or stolen).

2.2 Requirements of system development

2.2.1 No major change in the corporate development process for corporate applications

Companies already have invested literally billions of dollars in their existing development processes, supporting development tools and technology stacks to effectively construct and deliver PC and enterprise software. If the requirements of mobility required a substantial change in those processes, tools and technology stacks, this would constitute a substantial impediment for the adoption of mobile software.

On the other hand, many new opportunities are being created through mobile technology, and some of those opportunities cannot adequately be addressed through existing value chains; where new value chains are coming into being, this requirement of mapping to existing corporate development processes may be less important.

2.2.2 Single code branch for thin and rich clients on a variety of devices

Corporate developers are well trained, by now, in technologies such as Java Server Pages (JSP), Apache Struts, ASP.NET etc. to deliver browser-based user interfaces. If a mobile architecture required those users to use an entirely different paradigm (say, Java AWT, midlets or the Series 60 widget set) to develop user interfaces for the mobile device, this would represent a substantial cost and impediment.

Further, most mobile enterprise applications today need to be accessible both from a mobile device and from browsers on PCs. It is largely unaffordable for organizations to develop two separate code streams for PC-based users and mobile users, and keep both of them maintained and in sync after the initial deployment.

The problem is compounded by the fact that most corporate JSP applications, for example, today run much or even all application logic on the thread that triggered the construction of a web page. No such thread is available in a typical mobile rich-client environment, requiring completely different software application architecture for the mobile client than the browser-based user interface.

The mobility architecture should make it cost-effective to deliver applications with both rich and thin client user interfaces, preferably from a single code stream.

2.2.3 Centralized debugging, monitoring and upgrades

Software bugs are a fact of life, and an effective mobile software architecture must enable developers to learn about, debug, and resolve those bugs throughout all phases of the software lifecycle.

Specifically, the architecture must provide the ability for developers to attach to their software while it is running in a production environment, to understand what it is doing, and to centrally deploy fixes.

2.3 Requirements caused by mobile user behavior

The behavior of a user of a mobile device is dramatically different from the behavior of a user of a stationary PC. This difference in behavior causes a difference in software requirements to optimally support the mobile user.

2.3.1 One Life™: Seamlessness across many servers, business and private

Surveys show that the vast majority of mobile devices are used for both personal and business purposes. Further, as a mobile user moves around, they often need to interact not only with the software and the networks that were provided specifically to them (say, by their employer) but also software and networks in other departments, facilities and often even other companies. In the course of a typical day, they may need to interact with dozens of different systems, all developed and maintained by many different and – from their perspective – often unrelated parties.

For the mobile user, interacting with many different stovepipe systems is frustrating and often too cumbersome to contemplate. Therefore, to optimally support their mobile lifestyle, users need all systems to appear to be integrated. After all, they have only One Life, not a separate life for each of the backend systems.

As a simple example, consider calendaring: from the perspective of the mobile user, it makes no sense to have separate calendar stovepipes for the user's business travel schedule, their private schedule and the obligations for a non-profit that they are involved in. After all, the problem users are trying to solve is to schedule all of their life – business and private – without conflicts. Instead, they need one single calendar application that can – and preferably automatically does – aggregate calendar data from a number of different, independently developed and maintained backend systems. The same is true for many other types of information. Note that this aggregation must be performed in a manner that is secure: neither the non-profit nor the user's employer should necessarily see the other schedule.

An effective mobility architecture needs to explain and support how mobile software developed, deployed and operated by different organizations – that often don't even know about each other – can interoperate in a manner that creates a seamless, integrated and secure experience, which we call One Life.

2.3.2 Leverages traditionally unusual – but increasingly common – input devices

Mobile devices increasingly tend to have a substantially larger and more diversified set of input devices than traditional desk-bound computing devices such as PCs. This includes functionality such as GPS, location-sensitive Bluetooth, still and video cameras, even motion sensors, a compass, RFID readers, barcode readers and others.

An effective mobility architecture needs to make it easy for mobile software to optimally take advantage of the presence of those devices.

2.3.3 Situation-focused, not application-centric

Business software has evolved from the mainframe to the PC. In both cases, using the idea of a “software application” as the primary way of looking at software made a lot of sense: users in either case approach the terminal or PC, log in, start the application they are interested in, work within the application for some time, then switch to another application or log out and move on to a different activity.

The mobile user's behavior is different. Conditioned by the familiar pattern of interacting with a mobile phone, mobile users expect mobile software to initiate contact when something important and relevant happens. They do not tend to think in terms of which applications to run; instead, they tend to carry their mobile device wherever they go, and expect it to support them in whatever situations they encounter. Most famously, Colly Myers, the founder and former CEO of Symbian (a leading mobile operating systems company) recently admitted in an interview [3] with The Register that he did not run a single Symbian application on his mobile device. That is because the traditional application paradigm does not fit a mobile user's requirements and usage model very well.

The mobile user needs the mobile device to support them in whichever situation they encounter in their mobile lives, and mobile software needs to follow the existing usage model of the mobile phone as much as possible: When a mobile phone rings, the user expects to see the caller ID and a choice to answer or ignore, **if** there is an incoming call. They also need to have the choices to hold and answer or to conference, **if** they are already on another call. But, instead, they want to see the message **if** one has arrived. Or to see the calendar alarm **if** that's what just happened. They want to see information about the business relationship, **if** the call is from someone they have interacted with previously.

It is the same with any mobile software. The information the user wants to see and the choice of actions the user wants to be offered depend on the situation.

In other words, while application-centricity is a good paradigm for mainframe and PC software, situation-centricity is a better paradigm for mobile software. Significant research confirms this, some of which we have collected at [4].

An effective mobility architecture must explain how server-based and mobile software and content offers this kind of situational support to the user.

2.4 Requirements for a vibrant mobile software ecosystem

A software architecture becomes most important when its existence allows an entire business ecosystem to emerge and grow rapidly. While this paper does not address the issue of how to establish this architecture in the market, it does address the technical table stakes that need to be met before such an architecture is broadly adoptable.

2.4.1 Enables decentralized development, deployment and operation

In order to make possible a vibrant mobile software ecosystem with many market participants, it is essential that the underlying mobility architecture allows industry consensus around it, that it supports the development, and then the deployment and operation of the parts of this architecture by many independent parties.

Unless we make sure that decentralized innovation can occur in all major subsystems, and that those innovations can be easily deployed by whomever made them to a broad user base with as few centralized obstacles or gatekeepers as possible, the ecosystem will be limited in its growth. In case of the then-new PC, anyone could develop and sell software for any PC, and an effective mobility architecture, at a minimum, must have the same characteristics in this respect as the PC has.

This requirement appears to be somewhat in conflict with the One Life requirement, as One Life obviously requires integration. However, without One Life interaction, usability of mobile software will continue to be low, which will not allow a strong business ecosystem to emerge either. An effective mobility architecture must be able to square this circle.

2.4.2 Supports a large variety of types of software

Software, in particular in the mobile area, now comes in ever increasing varieties. Traditional client-server applications are being joined by portals, applications on top of content management systems and collaboration systems, as well as peer-to-peer, social and messaging-driven applications. The availability of audio and video streams on mobile devices and their integration into more and more mobile applications will certainly further increase the variety in the types of mobile software.

An effective mobility architecture must provide the facilities to effectively support all of those different types as well as novel types that are still to emerge. Due to the One Life requirement, all of those must be able to harmoniously interact with each other and integrate, in order to optimally support users in the different situations that they encounter.

2.4.3 Supports a variety of business models

There are already many different license and business models for mobile software: from traditional enterprise software licensing and distribution to software downloaded from Handango and its competitors; from licensed software, through revenue-sharing agreements between software vendor and operators, to software distributed to serve a secondary business purpose (e.g. advertising, brand building, website traffic) and others.

An effective mobility architecture must not prohibit any of these and many other business models.

2.4.4 Encourages innovation

An effective mobility architecture provides abstractions that are both simple and powerful. By doing so, it enables innovators to innovate with a minimum amount of expense and a maximum amount of interoperability and integration.

3 Necessary Architectural Qualities and Features

Any system architecture consists of three parts:

- a high level set of requirements for the system, which is stable in the long term;
- a specification of the parts and connectors comprising the system, together with the rules for the interactions of the parts using the connectors, and
- a demonstration that this parts and connector specification meets the requirements.

By examining the qualities and features that the architecture provides or enables, one can evaluate the fitness of a given architecture for the requirements, and compare different architectures against each other.

This section lists important qualities for a mobility architecture.

3.1 Basic technical qualities and features

3.1.1 Operates while disconnected

The system continues to be useful even while the mobile device is disconnected.

When reconnecting after a disconnection, the system can easily resume operations that were interrupted by the disconnection.

3.1.2 Information consistency

The system presents a consistent view of shared information across all involved devices and servers at all times. (Due to disconnections, any architecture will never completely have this quality, but it can be approximated to the maximum extent possible.)

3.1.3 Synchronizability

When disconnected parts reconnect, the system efficiently restores consistency of data.

3.1.4 Roamability

The mobile device's connections can be switched between different networks without breaking them.

3.1.5 Hand-offability...NEW

The user can move from one machine to another and proceed on the second machine at the same point at which she left the first machine.

3.1.6 Location awareness...NEW

The system can determine where the device is located, whether by geo-positioning or proximity; the location may be physical, by co-ordinates, or logical, by name or type of place.

3.1.7 Updates from user...NEW

The user can update other systems from the mobile device.

3.1.8 Push to user...NEW

Other systems can push events and data to the applications on the mobile device.

3.1.9 Server initiated actions...NEW

Other systems can initiate actions of the applications on the mobile device.

3.1.10 Multi-Network

The mobile device can monitor the list of available networks, and dynamically select from it based on preference / cost criteria.

The system continues to operate with intermittent and unreliable network connections and when changing from one network or network protocol to another.

3.1.11 Non-IP protocol support

The mobile device can connect not only using the IP protocol, but also non-IP protocols such as Bluetooth, messaging protocols (e-mail, IM), pager networks etc. Higher-level functionality needs to be as unaffected about lower-level protocols as possible.

3.1.12 Frugality

The system is designed to actively conserve resources. It is aware of and able to make tradeoffs in power consumption, resource usage, and performance and between network throughput and latency and cost.

Sub-categories of this category include:

- Memory consumption
- Computational demand
- Network throughput
- Latency tolerance
- Latency optimization: has the ability to allocate throughput and computation resources to meet user-perceived latency requirements
- Power consumption

3.1.13 Security

The system is protected against unauthorized use and other threats. Security comprises a number of qualities, including:

- Authentication
- Authorization
- Integrity
- Confidentiality
- Non-repudiation

3.1.14 Privacy

Information about the user and their activities is under the control of the user. This may include identity privacy, location privacy, activity privacy and other privacy qualities. These qualities are distinct from security qualities as usually conceived.

3.1.15 Device and platform transparency...SPLIT

The behavior of the system is independent of the device and software platform used by any user.

This may include device and platform transparency from the perspective of the user (e.g. regardless which device is used, the user experiences the same functionality), from the perspective of a technology provider (e.g. a software vendor can deploy the same software to many different devices) or from the perspective of a technology component (e.g. the application software can use the same software interfaces, regardless of the used device, platform).

3.1.16 Network transparency...NEW

The behavior of the system is independent of the network used by any user.

This may include network transparency from the perspective of the user (e.g. regardless which network is used, the user experiences the same functionality), from the perspective of a technology provider (e.g. a software vendor can deploy the same software to devices over many different networks) or from the perspective of a technology component (e.g. the application software can use the same software interfaces, regardless of the network use to communicate with other components).

3.1.17 Provider transparency

Parts of the system do not need to know the provider of other parts (including software and hardware manufacturers and service providers).

3.2 System development qualities and features

3.2.1 Stability

The architecture, and its parts, are stable over time and in the face of changing technologies and requirements.

3.2.2 Process neutrality

The architecture, and its parts, can be used with a variety of different development processes and methods without requiring substantial changes or additions.

3.2.3 Tool neutrality

Developers are able to create parts of the architecture using a tool chain of their choice. If additional tools are required, these tools integrate into the tool chain the developer uses already.

3.2.4 Client-side flexibility

Client-side software parts must be able to use and take advantage of a variety of client-side technologies, such as a rich client user interface and a thin client user interface with a variety of screen sizes and resolutions etc.

3.2.5 Development artifact conciseness

In spite of the many types of mobile devices and their different characteristics, the volume and size of artifacts created during the development process (such as source code) is small and grows more slowly than the number of different devices and characteristics.

3.3 Operational qualities and features...NEW HEADING

3.3.1 Observability...CHANGED TEXT

When the system is **being tested** or in operation, the developer has the ability to observe its behavior in order to make sure it works as intended, or determine which errors occur where and under which circumstances.

3.3.2 Manageability...NEW

Once the system is deployed, it provides capabilities to manage its operation.

3.3.3 Maintainability

Once the system is deployed, the developer has the ability to maintain it through additions, subtractions or changes.

3.3.4 Deployment support...NEW

The system provides capabilities to deployed it and to deploy changes.

3.4 User behavior-related qualities and features...REORDERED

3.4.1 Situational awareness

The device and the software on it is aware of the user's current situation, in concepts that are the same as the user would use to describe their situation, without the user having to tell the device or software.

3.4.2 Situational appropriateness

The information and the functionality presented to the user are appropriate for the user's current situation.

3.4.3 Proactivity

The mobile device and software proactively support the user without the user having to take the initiative to ask the device or software.

3.4.4 Semanticness

Information and functionality is presented to the user as concepts that make sense to the user in the current situation.

3.4.5 Timeliness

The information and functionality presented to the user are up-to-date in real time.

3.4.6 Sensitivity

The system employs all input devices and sensors available on the mobile device.

3.4.7 Collaboration...NEW

The system supports collaboration between users.

3.4.8 Usability

The system is easy for users to learn and use. The criteria for usability on a mobile device are quite different from those for a desktop or laptop device. For example, the system must provide a consistent user experience on different devices, across backend data sources, across different networks and whether on-line or disconnected.

3.4.9 Media transparency

As many parts of the system as possible do not need to know the media currently in use by a user (e.g. text, graphics, audio, video).

3.4.10 Mode transparency

As many parts of the system as possible do not need to know the communications mode currently in use by a user (e.g. browser, messaging, voice).

3.4.11 Coherence

The information, the functionality, and their presentation to the user "hang together", in order to present the One Life quality, even if elements of it were obtained from multiple, independent backend systems.

3.5 Business ecosystem qualities and features

3.5.1 Business model neutrality

The architecture supports and encourages a large variety of different business models, and is extensible to not foreseen ones. It is neutral with respect to device manufacturer, software builder and usage business models.

3.5.2 Applicability

The architecture can be applied to a large variety of different types of software, from enterprise to consumer, from applications and portals to document management and file sharing, from individual and enterprise to group productivity solutions etc.

3.5.3 Federatability

Many parts of the architecture can be developed, deployed, operated and maintained by many different individuals and organizations. This is the opposite of requiring a centralized entity that is responsible for all aspects of the system.

3.5.4 Resilience

The system continues to function even if some of its parts fail. This quality is particularly important if parts are provided by many different market participants who not all are aware of each other.

3.5.5 Business opportunity

Adding additional parts to the system is an attractive proposition from a business perspective.

3.5.6 Incentive to innovate

The architecture encourages new ideas and inventions that can be easily tried out, modified and applied.

4 Evaluation Matrix

Quality or feature	Architecture 1:	Architecture 2:	Architecture 3:
Basic technical qualities and features			
Operates while disconnected			
Information Consistency			
Synchronizability			
Roamability			
Hand-offability			
Location awareness			
Updates from user			
Push to user			
Server initiated actions			
Multi-network			
Non-IP protocol support			
Frugality			
Security			
Privacy			
Device and platform transparency			
Network transparency			
Provider transparency			
System development qualities and features			
Stability			
Process neutrality			
Tool neutrality			
Client-side flexibility			
Development artifact conciseness			
Operational qualities and features			
Observability			
Manageability			
Maintainability			
Deployment Support			

Quality or feature	Architecture 1:	Architecture 2:	Architecture 3:
User behavior-related qualities and features			
Situational awareness			
Situational appropriateness			
Proactivity			
Semanticness			
Timeliness			
Sensitivity			
Collaboration			
Usability			
Media transparency			
Mode transparency			
Coherence			
Business ecosystem qualities and features			
Business model neutrality			
Applicability			
Federatability			
Resilience			
Business opportunity			
Incentive to innovate			

5 Contributors...NEW

We very much appreciate the feedback and contributions from many individuals, including: Adrian Blakey, Andrew McMeikan, Axel Peter Mustad, and Ihor Kuz.

6 References

- [1] M Kozuch, M Satyanarayanan, T Bressoud, C Helfrich, and S Sinnamohideen. Seamless Computing on Fixed Infrastructure. IEEE Computer, July 2004.
<http://csdl.computer.org/comp/mags/co/2004/07/r7065abs.htm>
- [2] Intel Mobile Application Architecture Guide. Intel 2003.
<http://www.intel.com/cd/ids/developer/asmo-na/eng/61193.htm>
- [3] A Orlowski. Symbian Founder on Mobile Past, Present and Future. The Register, July 2004.
http://www.theregister.co.uk/2004/07/21/colly_myers_interview/
- [4] NetMesh and partners. Situational Software Blog. <http://netmesh.info/blog>