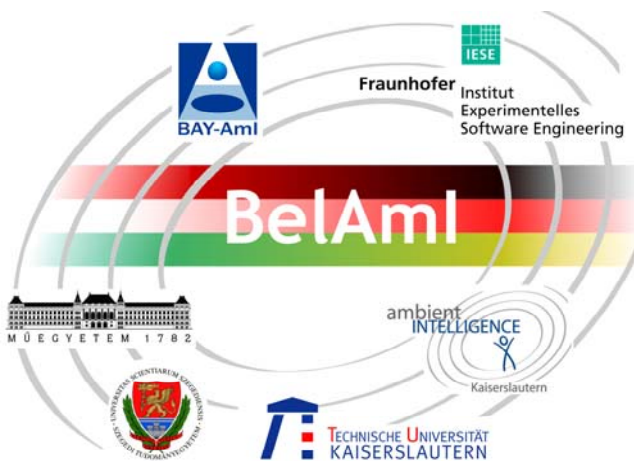




Fraunhofer Institut
Experimentelles
Software Engineering

Operating System Characteristics for Ambient Intelligent Systems



Authors:

Young-yeol Choo
Samir Amiry
Klaus Schmid
Martin Becker

BelAml-Report: No. 001.05/E
IESE-Report: No. 103.05/E
Date: 16 November 2005
Version: 1.0
Distribution: Public

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer-Gesellschaft.

The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach (Executive Director)
Prof. Dr. Peter Liggesmeyer (Director)
Sauerwiesen 6
67661 Kaiserslauterne

Abstract

Ambient Intelligence (Aml) includes a broad variety of new applications based on distributed, embedded, intelligent user interface, and mobile communication technologies. This new paradigm of information technologies requires new types of functional supports provided by Operating Systems (OSes). These include scalability, modularity, mobility support, various user interfaces, power management, hardware control, etc. Due to the variety of characteristics required in Aml systems, the selection of the appropriate OS is not a trivial task.

In this report, we present an overview of existing operating systems and relate their key characteristics to the typical requirements of Aml systems. In our analysis we focused on easily attainable operating systems and strived at supporting the everyday selection task that developers of Aml systems face. As no OS fully covers the range of characteristics required by Aml systems, the development of a new OS and a product line of OSes is still a challenging issue.

Keywords: Ambient Intelligence (Aml), Wireless Sensor Network, Ubiquitous Computing, Pervasive Computing, Operating System (OS).

Table of Contents

1	Introduction	1
1.1	Overview	2
1.2	Scopes	2
2	Ambient Intelligence Characteristics	4
2.1	Small and embedded	4
2.2	Distribution	4
2.2.1	Mobility	5
2.2.2	Heterogeneity	5
2.3	Real-time	5
2.4	Proactivity and context-awareness	6
2.4.1	Location	7
2.4.2	Time	7
2.4.3	State of system	7
2.4.4	Context management	7
2.5	Adaptivity	8
2.5.1	MMI	9
2.5.2	Communication	9
2.5.3	Computing	10
2.5.4	Spontaneity	10
2.6	Open Systems	10
2.7	Dependability	11
2.7.1	Accurate diagnostics	12
2.7.2	Fault tolerance	12
2.8	Scalability	12
2.9	Security	13
3	Operating Systems Properties	15
3.1	Real-time	16
3.2	Scalability	16
3.3	User interface	17
3.4	Modularity	17
3.5	Resources	17
3.6	Protection and security	18
3.7	Reliability	18
3.8	Open systems	19
3.8.1	Connectivity	19
3.8.2	Interoperability	19
3.8.3	Portability	19

3.9	POSIX compliant	19
3.10	Multi-tasking	20
3.11	Distribution	20
3.11.1	Distributed processing	20
3.11.2	Distributed file systems	20
3.11.3	Distributed shared memory	20
3.11.4	Distributed scheduling	21
4	Support of Aml Characteristics by Operating Systems: a Mapping Table	22
5	Conclusion	25

In the remainder of this section we give an idea what we consider an Aml system and outline the diversity of Aml applications

1.1 Overview

Ambient Intelligence (Aml) is a summary term for a large variety of different applications. They are characterized by the European Research Consortium for Informatics and Mathematics (ERCIM) [22] as follows:

- Intelligent User Interfaces
- Ubiquitous Communication
- Ubiquitous Computing

In correlation to this definition, Ambient Intelligence systems are supposed to provide access to information at any time and anywhere. Despite this characterization, there is no single commonality that holds for all Aml systems. Rather:

- **Applications:** There is a large range of widely different applications in domains like system maintenance, construction surveillance, inter-car communication, smart home, etc.
- **Contexts:** They can be applied in many different contexts: e.g., aircraft maintenance is highly regulated; some production shop environments are very difficult from a communication point of view; some contexts make it nearly impossible to work with traditional visually-oriented I/O; sometimes absolutely low-power operation is required, etc.

Thus, while each and every Aml application will be based on an operating system, the diversity described above will be reflected in a similar diversity of operating systems for Aml systems. The general purpose OSes cannot be used as is due to resource restrictions and lack of functionalities for Aml systems. In order to support a conscious choice of an operating system for any specific Aml application, this report describes typical characteristics of Aml systems and relates them to operating systems. Thus, the information described in this report can be used as a conscious basis for deciding on an operating system.

1.2 Scopes

Aml applications are based on various kinds of technologies. These technologies include distributed systems, wired/wireless communication, MEMS,

SoC (System on Chip) technology, artificial intelligence, MMI, Augmented Reality, etc. Moreover, Aml needs not only existing devices and technologies but also new ones. In Aml applications, a great diversity of Aml systems such as sensors, actuators, positioning systems, communication facilities, mobile computing devices, etc, are required to realize “smart environment” [18]. These systems are basically based on embedded system technologies. Therefore, we will mainly consider characteristics of Aml systems and operating systems in terms of embedded systems in this report. Embedded systems surely need the functional support of operating systems. Some parts of the applications require real-time services and others do not.

In the following, we list some scenarios that can be considered Aml applications (for details, see [23]):

- Assisted car driving – helps drivers to make driving more convenient and safe:
 - Speech generation for driver assistance
 - Brake temperature monitor (with MEMS technology)
 - Tire pressure monitor (with MEMS technology)
 - Ad-hoc cruise control
 - Pile-Up Avoidance- Intelligent Bus Stop
- Assisted training – training assistance in the sport domain
 - Optimizing training effects for a group
 - Testing different strategies to win a race
- Assisted living – facilitate people’s daily lives in their homes
 - Home help – e.g., cleaning, transport, maintenance, gardening
 - Monitoring and control
 - Entertainment and news – e.g., conversation, information
- Assisted working – “smart workshop” environment to support activities like transportation, detecting events, etc.
 - Automatic order-based prefabrication
 - Automated forwarding of the orders provided by the host to the production and warehouse environment as well as management of outdoor storage areas adapting to the changes in the production process and the workflow.
 - Intelligent transportation (i.e., navigation in the factory’s indoor environment, communication with the manufacturing tools)

The remainder of the report is organized as follows. In chapter 2, we describe the characteristics of Aml systems focusing on the distributed, embedded, mobile, and heterogeneous nature of Aml systems. The relevant operating system features to support Aml systems are presented in chapter 3. In chapter 4, the relation between Aml characteristics and OS attributes is addressed as a mapping table along with Aml characteristics. Chapter 5 contains concluding remarks.

2 Ambient Intelligence Characteristics

Reasoning about the necessary characteristics an OS has to provide in order to be applicable in a specific Aml context requires a deeper understanding of what the specific characteristics of Aml systems are. To this end, we identify and discuss such key characteristics in the following sections.

2.1 Small and embedded

Various types of devices such as cameras, head phones, data gloves, special sensors, smart phones, Augmented Reality glass, Active Badge [1], actuators, etc. are needed for the successful operation of Aml systems. These devices should have intelligence so that technologies are ultimately indistinguishable from our ambient condition. Inevitably, a large portion of Aml implementations comes in the form of embedded systems [10][19].

In Aml systems [2-9], a large number of devices will be deployed in our environment in such a way that a human may not know how to operate them or even will not perceive their existence. The environment may be peppered with a variety of networked sensors. Hence, these devices will adopt MEMS or nano-technology so that they take small space.

In the perspective of communication in Aml, devices will collaborate with each other using wireless communication network such as IEEE 802.11b, IEEE 802.15.4 ZigBee, Bluetooth, and sensor networks. In some cases, frequent service for Aml devices may not be possible since they will be placed in remote locations, which causes constraints in power consumption. In these cases, the life time of each Aml system may be determined by the battery life [20]. Consequently, they should be implemented in the form of small size devices using SoC or Nano technology. Generally, embedded system support is a basic requirement for the development of Aml systems [10][19].

2.2 Distribution

Inherently, Aml systems are working on the premise that all participating nodes are distributed in the environment and collaborate with others through networks, especially mobile networks. The distribution involves not only hardware and software, but also information and service. Concerning the distributed nature of Aml systems, the following features should be consid-

ered along with support for various distribution models such as peer-to-peer, client-server, and producer-consumer models.

2.2.1 Mobility

The nodes in Aml applications are usually mobile in their operation [6-10]. Even communication networks for nodes are of high *mobility*. Hence, mobility support is a must for Aml systems. When an Aml system moves to be faced with a new ambience, it joins ad-hoc networks and/or wireless sensor networks and exchanges information and service with the nodes of the ambience. Hence, self-configuration and re-configuration functions are important features to permit seamless addition and deletion of a node in a network as well as to cope with its vulnerability to failure.

2.2.2 Heterogeneity

Aml nodes include numerous types of devices, which pervade the “smart environment” [18]. They may be from various vendors, be based on various platforms including different hardware, operating systems, communication protocols, and application software, and provide different interfaces. Thus, the heterogeneity of Aml nodes should be worked out in all aspects.

2.3 Real-time

The applications for real-time computing include process control, intelligent vehicle highway system, telemedicine systems, telematics, multimedia system, etc. All of the above applications involve Aml implementation. Intensive research has been done on real-time computing [11-13]. Real-time systems can be defined as follows:

1) A real-time system is a system in which the correctness of the computations depends not only upon the logical correctness of the computation but also on the time at which the result is produced [13].

2) In POSIX standard 1003.1, real-time in terms of operating systems is defined as follows:

“Real-time in operating system: the ability of operating system to provide a required level of service in a bounded response time [14].”

Depending on the role of timing constraints, i.e., deadlines, real-time systems are classified as hard real-time or soft real-time. A deadline is said to be hard if the consequences of not meeting it cause system failure. The examples of hard real-time systems include industrial process control systems, vehicle highway systems, flight control systems, military sensor networks,

and so on. A deadline is said to be soft, on the other hand, if the consequences of not meeting it do not cause a system failure, although it is desirable to meet it. Transactions in database systems and multimedia communication belong to this category.

Aml applications have various real-time properties to be supported by the system level and the application level. In the case of manufacturing application of Aml, real-time processing is necessary for the following fields:

- Real-time control
- Alarm and event processing
- Real-time communication between control devices.

Because they are core functions in manufacturing and process control applications, meeting their respective deadline is the precondition for Aml applications in industries.

In traffic monitoring and control, auto cruising, and telemedicine system, some tasks such as anti-breaking system, anti-crash system, and teleoperation are critical for the safety of humans. This application requires hard real-time support not only on the system level, but also on the application level, along with proactivity support.

Depending on the characteristics of real-time Aml tasks, timing constraints range from millisecond order to several seconds. To support the real-time characteristics of Aml, an operating system provides features such as multi-tasking service, priority mechanism, pre-emption, predictable interrupt latency, and clock synchronization.

Some hard real-time applications require global clock in a limited area. In distributed hard real-time systems, synchronized clocks enable the total ordering of tasks in every node, which provides optimal task scheduling and facilitates the design of distributed real-time systems. Time Triggered Architecture (TTA) [12] is an example providing clock synchronization by wired communication. The application area are safety-critical hard real-time systems such as automotive control systems, aerospace systems, and so on.

2.4 Proactivity and context-awareness

Proactivity is one of the criteria that differentiate the integration of conventional computer technologies and Aml systems. Proactivity is the ability to act in anticipation of a future problem or need of a user by combining knowledge in system and environment. To support this characteristic, a certain level of artificial intelligence is inevitably required. To make a proper decision

based on proactivity, an Aml system continuously tracks human intent and has sufficient knowledge of the context related to the current circumstances (internal state + context information). Therefore, proactivity considers context-awareness as well as human intent.

Context is defined as follows:

“Context is the set of environmental states and settings that either determines an application’s behavior or in which an application event occurs and is interesting to the user [16].”

A user’s context may consist of the following attributes:

2.4.1 Location

Because Aml systems are highly mobile, reliable tracking of location is important for Aml applications. For instance, GPS (Global Positioning System) technology can be used to offer location information to an application when a system operates outdoors. On the other hand, GPS may not properly work indoors. Hence, another method, which is unobtrusive and cheap, should be devised for fine-grained spatial information. In this category, the location of neighborhoods and other objects should be considered at the same time.

2.4.2 Time

Time is one of the very natural and important contexts. It can include the time of a day, the month, season of the year, etc. as well as relative times.

2.4.3 State of system

Contexts such as network bandwidth, battery power, memory space, service schedule of processes, CPU load, etc. are important for system behavior in terms of service quality to be delivered to users.

2.4.4 Context management

Once the current context is seized by an Aml system, it is used to control the environment or adjust the system to the circumstance. According to the detection of context change, an Aml system may behave as follows:

- The Aml system automatically adapts to the new context by changing its activity.

- It reports the new context to the user by various MMI devices or stores the context for later retrieval.

Besides the contexts mentioned above, personal history, behavior pattern, and physiological state such as body temperature, heart rate, etc. could be contexts requested by a specific application. These might be collected by appropriate peripheral devices.

Examples of proactivity and context-awareness include:

- In the application of “Bicycle Team Training [6],” the Aml system suggests optimal speed, formation pattern, and the position of each cyclist at each timing section based on track profile, speed and direction of wind, physical condition of all cyclists, and so on.
- In the scenario of inter-car communication [8], to guide a user to his/her destination quickly and safely, the Aml system needs to collect and combine data for proper suggestion of the route. The data may be gathered from communication with nearby car as well as traffic information servers and filtered to prevent false suggestions.

2.5 Adaptivity

An Aml system is prone to failing to communicate reliably and operate properly due to its highly mobile operation pattern and depletion of battery capacity. The information and the services that an Aml system can get from environmental nodes are unstable regarding both quantitative and qualitative aspects. Mobile and distributed Aml systems should be designed to operate adaptively in unstable circumstances and when resources are scarce. That is, adaptation is needed not only when resources are insufficient for the proper working of an Aml system, but also when a decision on system activity is required due to the change of circumstances. The adaptation activity in the Aml application must happen without human intervention.

The resources can be bandwidth in wired/wireless communication, battery power, memory space, CPU capacity, and so on. To decide resource allocation or future system behavior, the following three strategies can be considered [17]:

- 1) Aml entities guide the application to use less of a scarce resource.
- 2) Aml entities can ask the environment to guarantee a certain level of resources.
- 3) Aml entities suggest a corrective action to the user when a good decision is not possible or the result of a bad decision entails high cost.

In this feature, the quality of service (QoS) is an important feature to consider. Although an application faces a new circumstance, it is not desirable that the service quality provided to a user is degraded drastically. Aml system should maintain the minimum QoS required to fulfill the mission of the application or reduce gracefully the service quality even in the worst case. QoS is specific to respective application. Therefore, to meet QoS of an application needs to be considered in all levels of Aml systems including hardware, OS, middleware, application, etc.

Adaptation issues are presented in the next subsections.

2.5.1 MMI

Aml systems, just like conventional systems, involve some sort of interface schemes. Perhaps, the traditional interfaces such as PDAs and mobile phones might be still used. However, Aml interfaces often differ widely from traditional interfaces. Audio equipments, haptic interfaces, or augmented reality can be exploited to access services of an Aml system.

Along the lines of “intelligent user interfaces” we may also expect a large range of adaptivity of Aml user interfaces (UI). The domain of adaption can be categorized as follows:

- Device: Aml UIs should be able to integrate and adapt to various input/output devices exploiting voice and gestures seamlessly.
- Function: If additional functions become available in a dynamic system, the UI should be able represent them and to effectively accommodate them as well.
- Context: If the brightness of a light or the volume of a sound change, the interface should adapt itself to the changes.
- User: Depending on the skills and preferences of the user, the interface should be self-adapting as well.

2.5.2 Communication

The sensors deployed in an environment may fail to operate properly due to an unexpected accident or depletion of battery power. The Aml system should adapt to these failures and continue to provide its service without human intervention such as replacement of devices and maintenance operation. In particular, the following adaptive services are necessary to support seamless communication among Aml devices and/or systems.

- Self-organization of ad-hoc network or sensor network
- Self-healing capability
- Self-management
- Reconfiguration of network when the network components are changed

- QoS support: To satisfy this requirement, Aml systems adapt to the change of network states and resources. This feature is closely related to proactivity.

2.5.3 Computing

Depending on the available information and services, an Aml system is requested to upgrade or downgrade its service to a user. In addition, the QoS of communication also may vary adaptively in all layers of protocols.

2.5.4 Spontaneity

An Aml system interacts with its environment *spontaneously* and *seamlessly*. As mentioned above, when an Aml system is faced with a change of circumstances, it should adapt to the new ambience without human intervention. That is, joining the new network, exchanging information with other nodes, and collaborating with the environment to fulfill the mission of the system should happen without explicit commands.

2.6 Open Systems

Aml systems may consist of heterogeneous devices, which are connected with various network protocols. They may include: sensors, actuators, various devices for interaction (i.e., PDAs based on Palm OS, PDAs based on Windows CE .NET, PCs running different OSES), and heterogeneous networks (i.e. Wired/Wireless LAN, Bluetooth, ZigBee).

Therefore, open system technology is indispensable for the seamless integration and cooperation among Aml devices. We define open systems in three aspects:

- 1) Interoperability: An application collaborates with other applications on local and remote systems.
- 2) Portability: A software/hardware should adapt to other software/hardware so that it will function in a different computing environment than the one for which it was originally written.
- 3) Connectivity: An application can exchange information with other applications without modification of communication facilities.

For example, in the application of Assisted Bicycle Team Training, the following devices take part in the operation:

- Heart rate sensor: sensing the heart beat per minute
- Wind sensor: sensing the speed of wind

- Pedal power sensor: sensing the pedal power of a team member
- LCD display: displaying the heart rate
- PDA: displaying the sensor data
- Head phone set: voice communication among the racing members
- Communication device: RS-232, Mica.

These different devices should be interoperable and connected seamlessly for the normal operation of all the systems during the training period.

In the MMI perspective, a user may interact with his/her ambience using various kinds of MMI devices such as cameras, head phones, data gloves, special sensors, smart phones, augmented reality glasses, etc. Therefore, standard interface schemes should be devised for seamless integration of these devices. Hardware and software are required to collaborate without error not only on a local platform but also between remote platforms. For connectivity, the standard protocol should be supported by the operating system.

The openness of an Aml system can be achieved by support in the OS layer along with user layer application functions. Total support of openness may be difficult in embedded implementation because it has limitations in memory size and performance. Standardization for the cooperating devices in an Aml application is necessary, but interoperability with remote applications may be achieved by the base station or gateways.

2.7 Dependability

To avoid ambiguity, we will use the definition of dependability as follows [15]:

“Dependability is the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behavior as it is perceived by its user(s).”

More precisely, a dependable system should satisfy four attributes:

- Availability: to be ready for usage
- Reliability: to continue a service without error
- Safety: to avoid catastrophic consequences
- Security: to prevent unauthorized access and/or handling of information.

Security contains not only confidentiality but also other complex issues such as authentication, integrity, digital signature, and repudiation. Therefore, it will be handled in a separate section.

In the mobile communication area, some critical messages require guaranteed delivery. For fast and reliable delivery of a user request, deterministic

communication should be supported by an operating system. Flooding and multi-path routing in network layer is needed for real-time communication service. Largely, this task is to be achieved on the application level.

From the perspective of microelectronic/Microsystems, Ami systems require dependability. Devices in an Ami system have the characteristics relevant to dependability as follows.

2.7.1 Accurate diagnostics

For the reliable operation of Ami devices such as sensors and actuators, each device should be equipped with the functionality to diagnose its respective state. Remote calibration is also required because manual service for precise calibration is rarely possible.

2.7.2 Fault tolerance

An Ami system has distributed implementation and interacts with a communication network. A node among the devices may fail either due to lack of energy or due to physical destruction. Especially, nodes in sensor networks are prone to fail. Even if a node fails, the operation of the whole system is requested to be maintained. In some applications, a fail-safe or fail-silence property is necessary. Early detection of faults is important in safety-critical systems to prevent faults. For example, in car communication application, the vehicle traffic on highway or in a congested area of the city should be monitored and faults should be anticipated, so that a car will not crash into others. In factory applications of Ami, process faults or system faults should be detected early to prevent catastrophic results.

2.8 Scalability

Scalability is defined as the ability of an Ami application or device (hardware or software) to continue to function well when it (or its context) is changed in size or volume in order to meet a user need.

Due to the holistic approach to various fields of technologies, Ami environments will encompass a tremendous number of users, applications, networked devices, and their interactions, which has never happened before [10]. As environmental smartness grows, so will the number of devices connected to the environment and the intensity of human-machine interactions.

Traditional development requires new development of the application for each new device. However, due to the number of devices and interactions in Ami, reuse of developed logic is required as a solution. Scalability should be

considered on the software level as well as on the hardware, communication, and interface level in an Aml application.

2.9 Security

Embedded systems in Aml interact continuously with humans and with other devices using communication networks. In some cases, their malfunction can cause property damage and personal injury, which is exemplified in the case of ambient traffic light control. Hence, Aml systems should forearm against various types of attacks that are expected as shown in Figure 2.

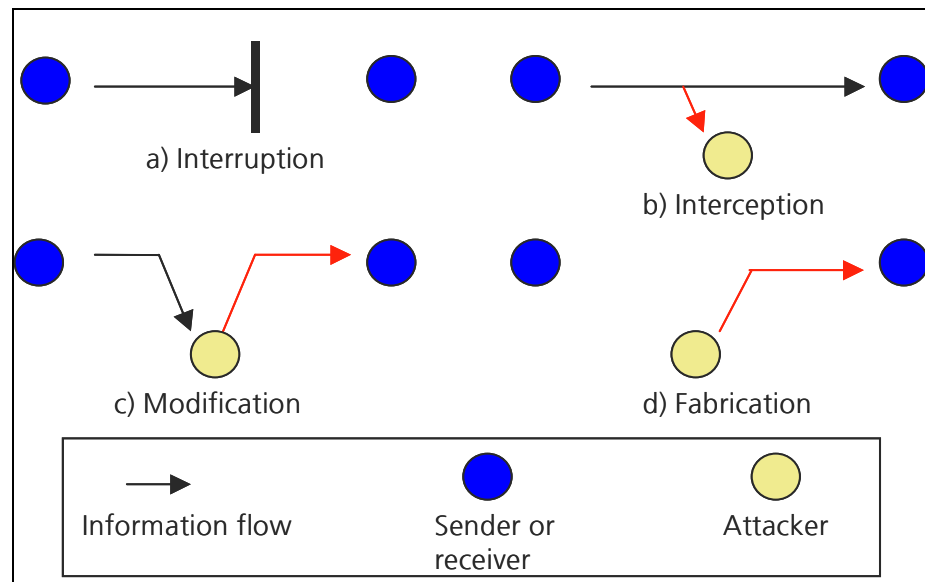


Figure 2 Various types of security attacks

However, because embedded systems in Aml often rely greatly on cost effectiveness and power efficiency, CPU and other resources are restricted in performance. For example, an 8-bit CPU cannot be enough to encrypt/decrypt data with a long cryptographic key [21]. This is one of the important challenging issues in general embedded system applications as well as in Aml applications. Requested security services in Aml applications are as follows:

- Confidentiality: protection against inappropriate disclosure
- Authentication: protection against fabrication
- Availability: protection against denial of service
- Integrity: protection against unauthorized modification.

Security from malicious attacks, in particular, is important for safety-critical application such as “Ambient Traffic Light Control [8],” “Assisted Living [9],” and “Assisted Working [7].”

3 Operating Systems Properties

An operating system is an intermediate part between the user and the computing hardware resources; its purpose is to provide an environment that lets the user execute its programs in an efficient and convenient manner [24]. The OS is responsible for managing the hardware computing resources, and for hiding the complexity of exposing them to the users. Nowadays, it is very difficult to define what is a part of an OS and what is not: Modern operating systems come with text editors, compilers, etc. However, in general, an OS should contain the following [24]:

- A process manager: A process in an OS is a unit of work that can be executed. An OS should provide a management infrastructure in order to support this concept, which contains, in general, a CPU scheduler, a process synchronization unit, deadlock avoidance strategies, etc.
- A storage manager: An essential part of an OS is the manager of storage devices including memory, hard-drives, floppy disks, etc. An OS should provide a management facility for the storage of data.
- An I/O system: Important parts of a computing environment are I/Os; an OS should provide a management facility to support I/Os.

An OS can also provide support for the following facilities:

- Distribution: provide a computing environment consisting of many computing nodes that communicate by exchanging messages over a communication network [25]. The OS supports the concept of transparency for distribution.
- Real-time computing infrastructure: A real-time computing environment is a computing environment in which tasks are fulfilled within a pre-defined time deadline. Real-time computing can be divided into two categories: hard real-time and soft real-time computing environments. Some OSes provide support for such concepts.
- Protection and Security mechanisms: Some OSes support the concept of protection (protection, in general, is the fact of protecting users' objects from other users [26]). Protection is a means for implementing security. Security goals, in general, are: confidentiality, integrity, and availability [26]. On the other hand, security treats are: interception, interruption, modification, and fabrication [26].

In general, the OS can be described using the following figure:

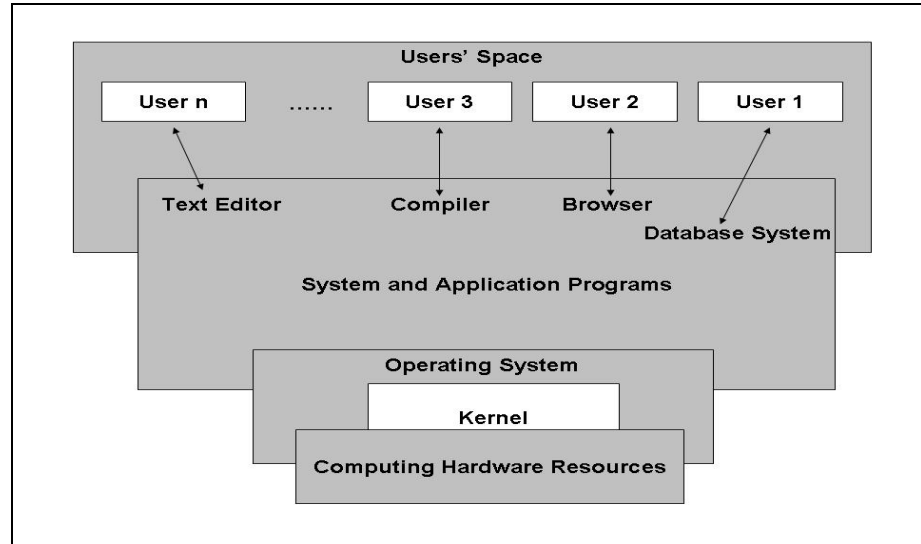


Figure 3 Operating Systems in Computing

Figure 3 shows where an OS is located in a computing environment. The kernel, is the lowest OS part that is running all the time and takes care of the basic task resource allocations [24].

Nowadays, there is no common definition of the architecture of how an OS should look like: Each OS vendor has its own architecture, and more over, it has its own definition of what makes up an OS [24]. However, there are many common modules and features offered in an OS. In the following sub-sections, we will list the most important ones with some definitions.

3.1 Real-time

A real-time computing environment is an environment where (some) task are executed (completed) within a predefined time deadline from start to completion [27]. As stated in chapter two, the real-time property exists in two formats [27]:

- Hard real-time property: hard real-time property is said about deterministic systems where tasks are fulfilling the real-time requirement all the time.
- Soft real-time property: the soft real-time property means that the system is tolerant of missing some of the timing requirements (most of the time the probabilistic approach is used with this category).

3.2 Scalability

Scalability is the ability to scale hardware and software to support larger or smaller volumes of data and more or less users. To be scaled to the needs

of Aml systems, previously developed software needs to run on other systems connected by mobile networks. This requirement may be acquired by a system environment on which the software is able to run in other system platforms. Due to the distributed nature of Aml systems, the processors may be *loosely coupled*. Thus, an OS with a centralized task scheduler will soon become a bottleneck. The following features are of importance to support scalability for Aml systems:

- Virtual engine for processor management
- Mobile communication
- Load balancing technique.

3.3 User interface

The modern OSes should interact with people in a large number of ways. The most notable are: voice interfaces, graphical user interfaces in all varieties, even robotic interfaces or input via gestures are envisioned. In this document, we classify user interfaces into three categories:

- Graphical User Interface (GUI) for the OS: The GUI is a facility that provides a set of graphical objects (i.e., windows, icons, pictures, boxes, etc.) that can be used by the user as an easy approach to exploit the OS resources.
- Voice recognition facilities: Modern OS provide a voice recognition mechanism as user interface to exploit some of the OS facilities that use voice (commands).
- I/Os: OSes should provide rich libraries for supporting the exploitation of the resources they offer via certain I/O devices such as screens, joysticks, mice, keyboards and keypads, etc.

Through all these facilities, the OS can provide its services in a user-friendly manner; the goal is to provide transparent mechanisms for the use of OSes.

3.4 Modularity

Modularity is an important concept in OSes. Because of the fact that OSes come in a general format, modularity gives them the possibility of being well customized for particular domains. As an example, we point out Linux, which can be used in embedded systems without a keyboard or a mouse. The fact that Linux is modular allows the use of modules that are only needed for this kind of applications (no mouse, no keyboard).

3.5 Resources

Platforms on which Oses run are characterized by a big diversity in terms of resources. The resources may differ in the Central Processing Unit (CPU) architecture; they may use different memory technologies, etc. In this report, we will make distinction between four categories of platforms in terms of resources:

- Tiny: this category contains those systems with CPUs using less than 8bits, and memories with ranging from few bytes to few Kbytes.
- Small: this category contains those systems with CPUs using from 8bits to 16 bits, and memory ranging from few Kbytes to few Mbytes.
- Medium: this category contains those systems with CPUs using from 16bits to 32 bits, and memory ranging from few Mbytes to hundreds of Mbytes.
- Large: this category contains those systems with CPUs using more than 32 bits, and memory with more than hundred of Mbytes.

3.6 Protection and security

Security in computing can be defined as the guarantee of offering the following properties [26]:

- Confidentiality: means that only authorized people/users can see/access protected data.
- Integrity: means that only authorized people/users/processes can modify the data in acceptable ways.
- Availability: This is also a complex property; availability expectations are presence of objects/services in usable form, capacity to meet service needs, progress (bound waiting time), and adequate time/timeliness of service. Goals of availability are: timely response, fair allocation, fault tolerant, utility or usability (can be used as intended), and controlled concurrency.

On the other hand, security breaches are: interception, interruption, modification, and fabrication [26]. Interception happens when an unauthorized party will get access to a computing facility; interruption happens when one of the available computing resources is lost, unavailable, or unusable; modification happens when an unauthorized party modifies the content of an object in the computing environment; and fabrication happens when an unauthorized party fabricates an object in the computing environment.

3.7 Reliability

In OS, reliability is the fact that the OS operates consistently with its specification [31]. In theory, a reliable OS is totally free from technical errors; in practice, vendors express this by a percentage, and weight affected to OSes' tasks/components.

3.8 Open systems

Open systems are a concept that uses standards for allowing the operating system to communicate/interact with other systems. Some of the important features of open systems are: connectivity, interoperability, and portability. In the following paragraphs, we will identify the characteristics of each feature stated above.

3.8.1 Connectivity

OSes should provide mechanisms to enable communication between many computing nodes; these mechanisms are achieved by the implementation of standard communication protocols. Especially, wireless networks are of importance for connectivity among nodes. As an instance, we state a variety of Wireless network protocols such as IEEE 802.11, IEEE 802.15.4, and Zig-Bee as well as TCP/IP.

3.8.2 Interoperability

In [29], interoperability is defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. In OSes, this functionality is supported via many techniques like networking, file formats, data coding, etc. In this document, we will use the term interoperability in terms of applications; that is, applications should be interoperable through some OS mechanisms.

3.8.3 Portability

Portability means the ability to port objects (software or hardware) for use in other environments. In this document, we will define portability in terms of applications. That is, portability means the ability to port applications that should run from one platform to another.

3.9 POSIX compliant

Portable Operating System Interface for UNIX (POSIX) is a standard from the IEEE and ISO communities that defines an interface between programs

and OSes [32]. Most OS vendors are trying to certify their OSes as compliant with the POSIX specification.

3.10 Multi-tasking

Multi-tasking is the ability to enable the CPU in a computing environment to execute more than one task at a time. This requires the OS to implement a process scheduling mechanism. Multi-tasking in one single program can best be explained by threads. Nowadays, most OSes implement this feature.

3.11 Distribution

Distribution in operating systems is the concept of interconnecting many computing nodes through communication networks to be presented to the OS users as one computing system [25]. In practice, OSes partially implement distribution: they implement distributed file systems, distributed memory management schemas, distributed processors environments, etc.

In fact, distribution can take many formats; here is a list of distribution forms with a few examples:

3.11.1 Distributed processing

Distributed processing is the ability to distribute processes over many processing nodes in a way that is transparent to users [25]. We state as an example the SUN RPC (Remote Procedure Call). In order to implement system distribution in processing, many features are required; these include distributed mutual exclusion, clock synchronization, distributed deadlock detection, and agreement protocols.

3.11.2 Distributed file systems

Distributed file systems mean the ability to implement a common file system that can be shared by all computing nodes in the distributed system [25]. The Network File System (NFS) is an example. Many properties such as distributed mounting, distributed caching, encryption (for security purposes), availability, and consistency are required to implement distributed file systems.

3.11.3 Distributed shared memory

Distributed shared memory means the ability to use some/all of the memories present in computing nodes in a distributed system, as one logical

memory. Examples include IVY (Integrated Shared Memory at Yale), which was implemented in the Apollo DOMAIN environment, Mirage, which was implemented at the University of California Los Angeles, and Clouds, developed at the Georgia Institute of Technology [25]. This property is supported by many other concepts like granularity, paging algorithms, etc. (for more information, please refer to [25]).

3.11.4 Distributed scheduling

Distributed scheduling deals with process scheduling in a distributed system. It deals with load balancing, load sharing, task migration, etc. For more information, a good list of readings is provided in [25] page 291.

In distribution, many other subjects are of relevance, such as fault tolerance, security, failure recovery, and concurrency control. In theory, in order to classify an OS as a distributed OS, all aspects stated should be fulfilled; in practice, only some of these features are present.

4 Support of Aml Characteristics by Operating Systems: a Mapping Table

Aml systems are intended to provide their users with a surrounding environment to support them in some of their activities. This support is provided by many applications that require, in general, the use of an OS. This OS should provide some characteristics necessary to support the special characteristics that an Aml system may have. However, the relationship between the Aml characteristics and the OS characteristics is not evident. More over, commercial Oses are provided with only a sub-set of the characteristics described in section 3. In order to understand how we can select which OS to use in an Aml system, we need to analyze the relationship between the Aml- and the OS- relevant characteristics. Based on this analysis, we should present a selection table for commercial Oses.

In the following, we will present a table that summarizes the existing relationship between OS and Aml characteristics; the table also provides a set of commercial Oses and their characteristics:

- **X**: means that there is strong correlation between the corresponding attributes in the table or an attribute is supported by the corresponding OS.
- Δ : means that there is weak correlation between an Aml characteristic and the corresponding OS characteristic.
- Ω : means that an OS does not fully support the corresponding OS characteristic.

To illustrate how Table 1 can be used to select an OS based on the Aml characteristics, we provide the following example:

The HomeCare company wants to engineer a node of an Aml system in the assisted living domain. The analysis of the node reveals the following characteristics: it is small and embedded, mobile, has to be aware of the position, and its communication should be adapted depending on energy consumption. At first, the person in charge of selecting an appropriate OS identifies the respective rows in the upper part of the (rotated) table. In each of these lines, he identifies the X or \square s and follows the columns down to the corresponding OS characteristics. X means that the OS characteristic is required, \square means that the OS characteristic may be required. In this example, real-time property may be of interest; modularity and resources have to be considered. Furthermore the OS has to provide an appropriate connectivity. With the identified OS characteristics, he can then find the listed Oses that meet these requirements.

5 Conclusion

Aml is a totally new paradigm shift in information technology. Aml applications need holistic approaches consisting of systems and technologies that are sensitive, embedded, distributed, heterogeneous, highly mobile, adaptive, contextualized, transparent, dependable, and intelligent. In this report, we present the characteristics of Aml systems and features of operating systems to support them. The relations between two characteristics were summarized in a table along with support of current commercial OSES for Aml characteristics. Because Aml applications cover huge areas, we understand that the selection of the appropriate OS to fulfill a variety of Aml characteristics is not a trivial task. Some Aml characteristics might be handled on the application level. However, others need support on the OS level. As a whole, current OSES are inadequate for supporting most Aml systems and applications because those are based on a new technological paradigm. Hence, a new type of operating system architecture is a challenging issue to support Aml systems and applications. The following attributes of an OS are often required for Aml systems:

- Small size and power management: to be used in sensor nodes and resource-constrained systems.
- Modularity and openness: to be ported in various hardware and software architectures.
- Concurrency: to execute high level processing for users such as proactive behavior, and low level input/output functions such as sensing, concurrently.
- Mobility: to support highly mobile ad hoc networks.
- Direct access to hardware: to control power usage directly in applications.

In particular, power is the most important resource in Aml systems. Accordingly, power management should be considered in every aspect of the OS and all components need to be designed to save power when not active. In order to facilitate power management, it is desirable that direct and precise control over the hardware of Aml systems should be provided to applications. Layered architecture, which is common in conventional OSES and platforms for hardware abstraction, may have to be avoided for better efficiencies in power consumption.

On the other hand, direct access to hardware may cause bad effects on software portability. Hence, the trade-off between direct access and abstraction needs to be made depending on the functional and hierarchical level of an Aml system in applications.

Proactivity and context-awareness are significant factors differentiating Aml from current IT technologies. Although these characteristics may be implemented on the application level, some functions for intelligence and inference related to proactivity and tracking of user intent should be provided on the OS level to support them, which has not been considered yet in conventional OSes.

As a whole, considering all the constraints of Aml systems mentioned previously, a modular, flexible and very small sized Operating System, namely, a nano-Operating System, is desired for prevalent deployment of Aml.

References

- [1] <http://www.uk.research.att.com/ab.html>, The Active Badge System
- [2] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/chall_in_Aml.pdf "Challenges in Ambient Intelligence: A Microelectronic/Microsystem Perspective."
- [3] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/hunmob_12.pdf, "Research Proposal for a Joint Project on Mobile Communications."
- [4] <http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/SE-Whitepaper.pdf>, "A Software Engineering Perspective on Ambient Intelligence"
- [5] http://agrausch.informatik.uni-kl.de/budapest/kompetenzen/documents/HMI_white_paper.pdf, white paper in the area of MMI for the research topic "Ambient Intelligence"
- [6] <http://agrausch.informatik.uni-kl.de/budapest/szenarien/documents/bicycle%20scenario.pdf>, Scenario and Demonstrator "Assisted Bicycle Team Training"
- [7] <http://www.eit.uni-kl.de/ami/en/inhalte.html>, "Scenario : Human centered manufacturing"
- [8] <http://agrausch.informatik.uni-kl.de/budapest/szenarien/documents/carcomm%20text.pdf>, "Inter-Car Communications Scenario"
- [9] http://agrausch.informatik.uni-kl.de/budapest/szenarien/documents/ass_living.pdf, "Assisted Living Scenario"
- [10] <http://www.cordis.lu/ist/istag.htm>, "Ambient Intelligence: from vision to reality".
- [11] Ian Broster, "Flexibility in Dependable real-time system," Ph. D thesis, Univ. of York, Aug. 2003
- [12] Hermann Kopetz and Gunther Bauer, "The Time-Triggered Architecture," Proc. of the IEEE, Jun. 2002.

[13] Shin K., and Ramanathan P. "Real-time Computing: A New Discipline of Computer Science and Engineering," Proceedings of the IEEE, vol. 82, no. 1. 1994.

[14] <http://www.faqs.org/faqs/realtime-computing/faq/>, Comp.realtime: Frequently Asked Questions

[15] J. C. Laprie, "Dependability—Basic Concepts and Terminology," vol. 5 of Dependable Computing and Fault-tolerant Systems. Springer-Verlag, IFIP WG 10.4 1992.

[16] Guanling Chen and David Kotz, "A Survey of Context-Aware Mobile Computing Research," Technical Report TR2000-381, Dartmouth College, Dept. of Computer Science, 2000

[17] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, pp. 10-17 Aug. 2001

[18] Paolo Remagnino and Gian Luca Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm," IEEE Tr. On Systems, Man, and Cybernetics, vol. 35, no. 1, pp. 1-6, Jan. 2005

[19] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, "System Architecture Directions for Networked Sensors," In Tenth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 93-104, November 2000.

[20] Nojeong Heo and Pramod K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," IEEE Tr. On Systems, Man, and Cybernetics, vol. 35, no. 1, pp. 78-92, Jan. 2005

[21] Philip Koopman, "Embedded system security," IEEE Computer, vol. 37, no. 7, pp. 95-97, Jul. 2004

[22] <http://www.ercim.org/>.

[23] BelAmi Project, "Project Outline", <http://www.iese.fraunhofer.de/BelAmi>

- [24] Operating System Concept, Fifth Edition, Silberschatz Galvin; Addison Wesley.
- [25] Advanced Concepts in Operating Systems; Mukesh Singhal, Niranjana G. Shivaratri; Mc Graw Hill.
- [26] Security in Computing, Second edition; Charles P. Pfleeger; Prentice Hall PTR.
- [27] Real-Time Systems; Jane W.S. Liu; p-28-32; Prentice Hall.
- [28] <http://www.kde.org>.
- [29] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.
- [30] http://www.sei.cmu.edu/str/descriptions/trusted_body.html.
- [31] http://whatis.techtarget.com/definition/0,,sid9_gci752461_00.html.
- [32] <http://www.webopedia.com/TERM/P/POSIX.html>.

Document Information

Title: Operating System Characteristics for Ambient Intelligent Systems.

BelAmI-Report: No. 001.05/E

IESE-Report: No. 103.05/E

Date: November 16, 2005

Status: Final

Distribution: Public

Copyright 2005, Fraunhofer IESE.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means including, without limitation, photocopying, recording, or otherwise, without the prior written permission of the publisher. Written permission is not needed if this publication is distributed for non-commercial purposes.