A Pervasive Computing Business Reference Architecture: The Basic Requirements Model

Osama M. Khaled, Hoda M. Hosny, and Mohamed Shalan

Department of Computer Science and Engineering. The American University in Cairo (Egypt)
Email: {okhaled, hhosny, mshalan}@aucegypt.edu

ABSTRACT

Pervasive computing is considered one of the most complex computing domains. Our research work attempts to solve some of the business challenges associated with pervasive computing. In this paper, we present a novel business reference architecture which addresses the basic business requirements to build a pervasive computing system by exploring eleven basic quality features and defining their requirements model. It has a detailed trade-off analysis for the selected quality features which guides the user while making decisions on real projects. We found that building a basic business requirements model is a very useful step towards building a business reference architecture, which will lead to a more practical technical reference architecture.

Keywords: Business process re-engineering, Context-aware services, Pervasive computing, Software architecture, Ubiquitous computing

1- INTRODUCTION

Automation aims to reduce efforts that people exert to achieve tasks. It reduces efforts and paper work, speeds up activities to connect remote areas, to transfer information, and to recover from human errors. It is designed to make things easier in our lives and to make them more comfortable. In the pervasive computing world, automation should do the same thing. However, a pervasive computing system is different from normal computer systems as human beings and devices tend to make more movements and activities. It is a system of, usually, small devices distributed in different locations.

Weiser's [1] vision is almost there. Computers are now everywhere, they are small, and contain unprecedented processing capabilities, e.g. Edison by Intel [2] and ARTIK microprocessor by Samsung [3]. They can be used to implement Internet of Things (IoT) applications. On the other hand, supportive technology, as will be shown in this paragraph, has advanced as well. The IPV6 range can acquire billions of smart devices which Gartner expects to reach 5.4 billion by 2020 [4]. A 4G speed network with higher speeds is now expanding in the market and researchers took real steps towards

implementing 5G networks with higher capacity by 2020 [5]. Researchers are working on the 5G Also, researchers are working on new versions of the WiFi and Bluetooth protocols with better IoT capabilities [6]. Renewable energy, e.g. solar power harvesting technology, is now smaller in size and can sustain low-energy consumption devices, like sensors, for longer times. Super-capacitors can also store 10 to 100 times of what electrolytic capacitors can store [7].

The main challenge for a complete pervasive system is the complexity of its architecture with many inherited design challenges due to the features that the system must have, like context sensitivity and adaptive behavior [8]. Some of these features may conflict with each other and subsequently generate more design challenges. For example, it could be required to capture knowledge about users via sensors and at the same time protect their privacy. It could be required to allow devices from any manufacturer to join the system with full compatibility and no errors. Such conflicts increase the complexity of the system and introduce business problems that must be resolved early enough before implementing the system's technical architecture.

Although researchers exerted great efforts to generate reference architectures for pervasive systems, only a few of them addressed the business problems, or challenges, that are supposed to guide the technical model. Building a technical model for an undefined business problem makes the reference model not very practical in all contexts [9]. Our research aims to define a generic business reference architecture layer in order to build a practical and solid technical reference architecture. By business architecture we mean the business concepts, definitions, requirements and processes that form the understanding of a specific domain.

The rest of the paper is organized as follows: Section II cites some of the related research work, Section III explains our research approach, section IV gives details about the essential requirements for basic quality features in pervasive systems, and section V presents a trade-off analysis for the quality features. Section VI concludes the paper.

2- RELATED WORK

As mentioned above, there are many technical reference architecture models that are not based on clear business models. Fortunately, we found some of the reference architectures that based their technical models on business requirements or at least demonstrated their work using business scenarios. We reviewed these models critically in order to establish the basic requirements for our business reference model.

Machado et al [10] present a reference architecture for ubiquitous computing which they called RA-Ubi. The authors built their reference architecture (RA) by following a process in which they i) identify information sources ii) elicit requirements iii) design the RA, and iv) evaluate the reference architecture.

Although the authors claimed that their technical model was based on clear

requirements, they did not show how the ubiquitous computing requirements guided their design decisions nor how the technical architecture can fulfill these requirements. The authors also did not give enough guidance on how to use their RA, and they gave only a description of the high-level components.

Addo et al. [11] introduced a reference architecture to improve security and privacy in the IoT applications. The authors tried to clarify their reference architecture by stating some business scenarios where such quality features should be considered, namely: a) a home automation monitoring service b) an Online Social Networking application, and c) a movie recommendation service.

The authors also identified some of the security, privacy and trust requirements that they considered in their RA. These requirements may be summarized as follows:

1. User identification and validation.
2. Tamper resistance of the physical and logical devices.
3. Content Security.
4. Data privacy.
5. Data communication and storage security.
6. Privacy in ubiquitous devices.

The IoT-A project [12] introduced a reference architecture for IoT systems as well. Its authors stated clearly a list of requirements that they used to support and validate their technical model. They gave details about each piece of requirement to understand its scope of implementation. They did not however provide priorities for the requirements since they considered this practice inappropriate for a reference architecture.

Al Ali et al. [13] introduced a reference model based on pervasive computing and cloud computing. They modeled the system architecture as a chain of nodes that include low-power nodes (e.g. sensors), resource-poor nodes (e.g. mobile phones), resource-rich nodes (e.g. severs), and cloud nodes. The low-power node will send data directly to the resource-poor node. The resource-poor node will aggregate data and send it to the resource-rich node. The resource-rich node makes online processing and delegates long-term tasks to the cloud nodes.

Al Ali et al. [13] focused their discussion on the hardware and network perspectives but gave a shallow discussion about the business requirements. They also did not discuss the impact of security rules on the performance of online data processing.

Our requirements model has gone a few steps further from those earlier models. We analyzed the relationships between the requirements as will be explained in the next section. We also carried out a deep statistical analysis of the relationships between the requirements, and gave a broader view that

can work for multiple business domains.

3- OUR RESEARCH APPROACH

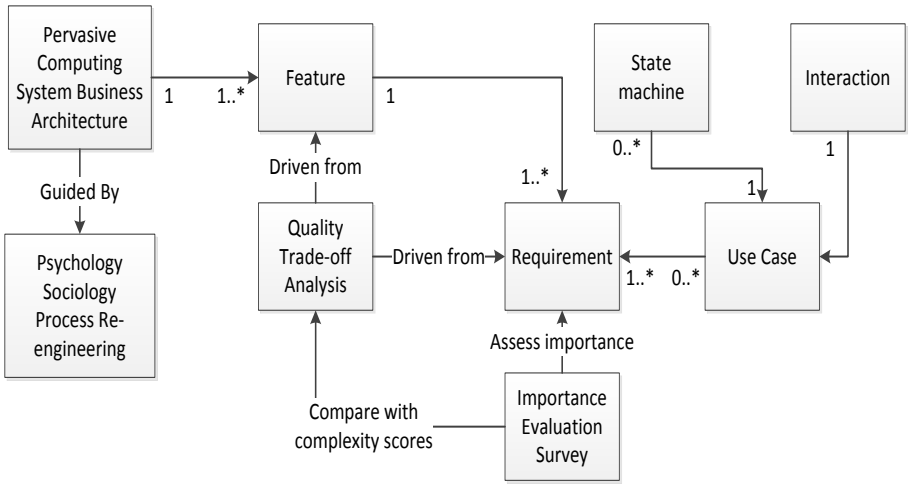

Figure 1 Pervasive Computing Analysis Approach

In order to define a useful business reference architecture for pervasive computing, we analyzed the system from more than one aspect (Fig. 1). We defined the business architecture as a pool of quality features that are driven by some business requirements. We refined our understanding about the requirements by studying possible use cases and state machines. We elicited the list of requirements from the literature and from domain experts by employing business analysis techniques (e.g. workshops and surveys).

We made a trade-off analysis for the quality features based on the relationships among requirements to prioritize features and to understand their pertaining complexity. The business reference architecture is guided through the study of sociology (activity theory), psychology (Perception), and process re-engineering concepts. The previous theories and concepts are chosen because they are descriptive frameworks for our lives with all its complex interactions. Moreover, the readers will note that Weiser's vision about pervasive systems can be also explained through them. We finally conducted a survey to assess the importance of the requirements. We aggregated the requirements' importance scores by the quality features and compared the results with the complexity score that were generated from the trade-off analysis study.

3-1 ACTIVITY THEORY

The activity theory is one of the descriptive theories that explain human lives. The theory shows that people move around to achieve specific goals (outcomes) within processes. The goal is a desired objective that someone

(subject) wants to achieve. The process is an organized set of activities that should be completed in order to achieve the goal (outcome). People use (tools), physical or mental tools, recognized things or concepts from the world (objects), and manipulate and abide by (rules) to perform the tasks [14] [15]. Moreover, responsibilities are distributed among the people (community) who share the activity according to the (Division of labor) rules (Fig. 2).

An individual who wants to achieve a specific goal for the first time will usually concentrate on the process activities in order to reach the required goal. In other words, his/her mind will be highly alerted not to make any mistake that may spoil the required goal and consequently result in undesired outcomes. For a person who gets used to performing the activities of the process, he/she finds no problem in performing the activities with minimal or no mistakes and he/she usually achieves the goal quite easily [16].

Although the activity theory describes only an activity for an individual, not an activity fulfilled by a group of people [14], its simplicity and clarity are rather appealing to the scope of research since we want only a descriptive model for the environment of the pervasive system. Moreover, it is successful in describing Human-Computer Interaction (HCI) systems [14], which is a major characteristic for a pervasive system where users interact with the system all the time.

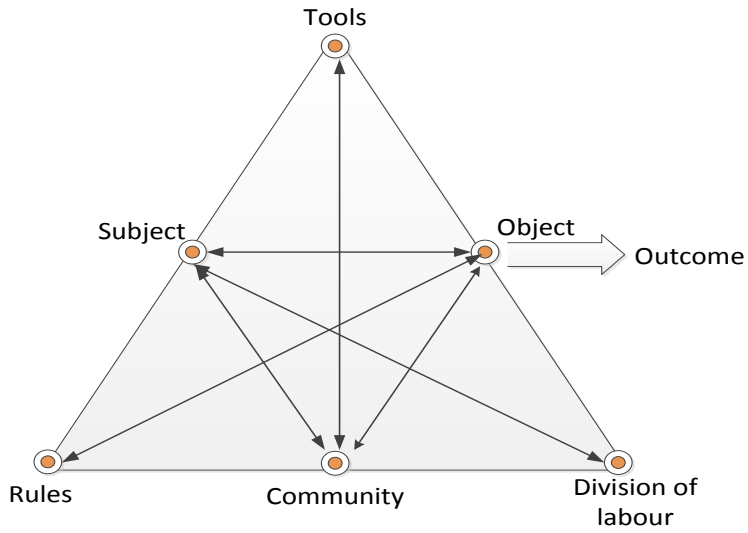

Figure 2 Activity Theory Perspective [12]

3-2 PERCEPTION

Perception represents a natural process that allows human beings to sense the environment and detect its changes through stimuli and interpret them into

useful meanings. We use these meanings to make the proper recognition and devise a suitable response [17] [18]. For example, the environment contains contextual stimuli (e.g. a person whom one knows) that send signals to our sensory system (e.g. eyes or ears) where we use our experience and knowledge to interpret it into a useful meaning (e.g. your friend), recognize it (e.g. your friend Kathrin) and take the proper response (e.g. shake hands).

The process adds to our accumulated knowledge and experience which we use again through our lives within several other perception activities. Our interpretation system mainly depends on detecting specific features about the stimuli [19]. The neural system then reviews these features with the stored knowledge and makes the proper match to recognize the stimuli. There are sets of actions or responses that are also reviewed based on the knowledge about the stimuli, and one or more responses are taken accordingly.

It is very interesting to note that the perception process describes the main activities in pervasive computing in its simple format (context awareness and adaptability). Pervasive computing is similar to the perception process in the sense that it should be invisible and transparent to the users. The perception process is also natural and invisible to the people.

3-3 PROCESS RE-ENGINEERING

Process engineering describes a specific process as a set of activities in order to achieve a specific goal. The process may have different decision conditions, inputs, and outputs. The decision conditions decide on the path that the process will go through which may end up not achieving the main goal of the process. At some point in time, people may find out that the process is no longer efficient and that it needs to be revisited. So, they initiate a reengineering project that aims to revisit the process and recommend solutions.

In process re-engineering, there are 3 major objectives [20]:

1. Maximize the value added tasks that the customer is willing to pay for.
2. Minimize the non-value added tasks which are essential for the process but the customer will not be willing to pay for.
3. Eliminate tasks that are considered a clear waste in the process.

3-4 REQUIREMENTS MODELING APPROACH

A pervasive computing system may automate tasks that people do in their lives. Accordingly, any business requirement that derives functional or architectural requirements should be considered from a process-engineering point of view.

We defined some stereotype notations to understand the *relationships* among the business requirements inspired from the 3 major objectives of process re-engineering:

1. **Minimize**: it is a relationship that shows that one requirement works

on minimizing a non-desired value from another piece of requirement.
2. **Maximize:** it is a relationship that shows that one requirement works on maximizing a desired value from another piece of requirement.
3. **Conflict:** it shows that two requirements could have conflicting values. If this happens, then one of them must supersede the other in order to resolve this conflict. The relationship could be one-directional or bi-directional.

4- QUALITY FEATURES

A reference architecture in pervasive computing should address specific design and architectural challenges that are very common in the domain. We selected the features that pervasive computing applications usually endorse [21]. The following list represents the features that we chose to study in our research and adopt in our model:

1. **Adaptive Behavior (AB):** The system must be capable of dynamically responding to changes in the environment as needed [21].
2. **Context Sensitivity (CS):** The system must have the ability to sense and retrieve data from its environment [21].
3. **Experience Capture (EC):** The system must have the ability to capture and record experiences for later use [21].
4. **Fault Tolerance (FT):** The system must be able to detect errors and take the appropriate recovery actions [21].
5. **Heterogeneity of Devices (HD):** The system must be able to use different device technologies seamlessly [21].
6. **Invisibility (IN):** The system must integrate computing resources to the degree that the user has minimum awareness of them [21].
7. **Privacy and Trust (PT):** The system must ensure that personal operations confidentiality is protected and accessed only by trusted entities [21].
8. **Quality of Service (QoS):** The system must set expectations for its services by setting constraints on the provided services. For example, a system's response may be considered invalid if it is received after a certain period of time [21].
9. **Safety (SY):** The system must ensure highest performance of its hardware and provide immunity for its users and interacting devices from harm and damage.
10. **Security (ST):** is concerned with protecting data from being leaked to unauthorized individuals, protecting data from corruption and alternation, and ensuring accessibility to data whenever requested.
11. **Service Omnipresence (SO):** The system gives its users the feeling

that they carry their computer services wherever they move [21].

The above listed quality features were selected because they represent the basic business and functional requirements for a successful pervasive computing system. They are domain independent, which makes them applicable in any business domain. We chose to cover some classical features like Security, Privacy and Trust, and Fault Tolerance, in order to understand the relevant priority of the pervasive features like adaptable behavior and context sensitivity. None of these features contain architecture requirements. Most of these features were introduced in [21] as the features covered by most of the surveyed systems. We deliberately ignored other features from the business architecture model mentioned by the same authors [21]; namely spontaneous interoperability, service discovery, function composition, openness, scalability, and concurrency since they are more of architectural requirements.

From the perspective of the activity theory, it means that

1. The pervasive system will optimize the usage of tools and signs as will be detailed in the Quality of Service section.
2. The rules will be changed to optimize the process as will be shown in the Adaptive behavior and Context sensitivity sections
3. Responsibilities could be redistributed in the division of labor. The coming sections will show that there are different categories of users with different activities.
4. New members could be introduced to the community to fill in a gap in the process, or removed from the community to eliminate a waste. Heterogeneity of Devices and Service Omnipresence describe some rules that govern the mobility of the users.

The following sections give detailed information about the requirements that derive the quality features. We started the elicitation process by the main categories, which are the quality features, then studied the requirements that make them essential. The requirements are elicited from the literature and derived from our knowledge. They were reviewed in a focus group with experts who added their views as well.

4-1 ADAPTIVE BEHAVIOR

The pervasive system must react dynamically to the changes of its context. In other words, it should adapt itself in a logical way based on specific decision rules. For example, if the pervasive system discussed in the Context-Awareness feature detected that there's been an accident for a specific bus, then it will take a decision that it needs to mobilize a rescue team. The pervasive system in the bus may in this case use its actuators, which are physical or virtual tools that can respond/change the context, to send an SMS to an emergency rescue team, switch on alarming lights, and activate a protection shield for the fuel tank. An adaptable pervasive system may cause further changes to the context and it may subsequently need to adapt to these changes causing further implications.

A generalized concept of the adaptive behavior may be applied on autonomic systems as well where the system adapts itself to system changes in a way that guarantees self-management to its functions and hide intrinsic complexity from users [22].

In summary, a system with an adaptable behavior should fulfill the requirements shown in Table 1.

Table 1 Adaptable behavior Feature Requirements

ID[1]	Name	Note
1	Evaluate/Improve Adaptive actions	The system must continuously review the adaptive actions and improve them to ensure that they satisfy the majority of the users.
2	Has smart decision rules	Such decisions are dependent on the interpretations as sensed from the environment. The decision rules must be taken smartly in favor of a high priority goal maintained by the system.
3	Notify users with changes	The user must be aware of the changes that the system made through its adaptive actions.
4	Possess actuation capabilities	These are the actuators that the system uses to respond to the changes of the environment. These actuators can be virtual or physical.

4.2 CONTEXT SENSITIVITY

Context sensitivity, or context awareness, is a collection of one or more variables used to indicate specific changes in the physical or virtual world. Awareness means that the system has the ability to detect a context and interpret it to a specific decision. For example, a school may have context for buses which contains location, time, and emergency alarm. These three parameters determine the context of the school. Each of these parameters takes specific values:

- Location: Far from school, nearby the school, in garage.
- Time: morning, noon, after noon, night.
- Bus Status: normal, accident, disaster.

There are 36 possible combinations of these variables which produce 36 contexts. One or more contexts may have the same interpretation. So, a context C1= (Garage, night, accident) can be interpreted as a bus has a problem but it is not severe since it is in garage and at night. Another context that may contain C2= (nearby the school, morning, disaster) can be interpreted as an emergency situation that requires immediate reaction to save lives.

The Context Sensitivity requirements are summarized in Table 2.

[1] The ID is used in the Appendix to show relationships among the requirements

Table 2 Context Sensitivity Feature Requirements

ID	Name	Note
5	Equip system with sensors	The sensors are critical for the system in order to collect as much data as possible for analysis [23].
6	Locate interacting objects	At any point in time, the system should locate the objects (smart or dummy). These objects could be interacting with the system, or with a part of it.
7	Provide analytical capability	The system is able to analyze the data collected by the sensors, generate useful information and correct errors if possible and if needed [23].
8	Provide interpretation rules	The system should be able to interpret information using predefined interpretation rules.
9	Record object lifetime	The system must register the lifetime trip of the objects that are part of the system. Statistical records should be available whenever needed.

4-3 EXPERIENCE CAPTURE

According to Spinola and Travassos [21], experience capture is concerned with finding common patterns of the user's behavior or activities and capturing them for later use. For example, in a smart home architecture, a user may have a repeated pattern to enter a room on a specific time, switch on lights, and then switch on the TV. The system can simplify these activities and automate these actions later on. Such a feature needs to be regulated by system policy and clear guidelines.

Moreover, the system should be able to capture knowledge about system users and use it as input for improving pattern capturing [12]. By correlating information and knowledge about users, the system will be able to forecast future user behaviors. If the system is designed for a specific goal that will be used by a specific group of people, then the habits and behaviors of those people could be studied, analyzed and fed into the system, similar to what is practiced in ethnography [24].

The Experience Capture Requirements are summarized in Table 3.

Table 3 Experience Capture Feature Requirements

ID	Name	Note
10	Capture Knowledge about users	Use the personal knowledge smartly to convey to the user that the system is there and recognizes his/her work.
11	Correlate information and knowledge	Correlate information and knowledge to forecast events and anticipate user or object behavior [12].
12	Capture/change behavioral patterns	The system should be able to capture/modify pattern(s) that users or objects repeat when they interact with the system [21].

4-4 FAULT TOLERANCE

Faults are more likely expected in a pervasive computing system due to its complex nature which includes multiple devices with high levels of communication among several software components. There are other reasons in pervasive systems that can cause faults. For example, a smart device may be processing something but it moves unexpectedly which causes its process to fail. The device battery may run out of charge and immediately goes out of service [25].

A fault is a problem that needs to be resolved and the decision of resolution differs with each case. First let's classify the faults as Severe, high, medium, and low based on their consequences:

1. **Severe**: This category includes fatal errors that may result in complete outage of the system, severe financial loss, or total corruption of data and there are no instant resolutions of the problem.
2. **High**: This category of problems does not suffer from complete outage of the system, but may have complete outage in some functions, noticeable financial problems, or impacts a large number of users. There are no instant resolutions for the problem.
3. **Medium**: Such a category has a moderate failure in terms of functions and impacted users and has no financial loss. There could be alternative approaches for the system to complete the required service.
4. **Low**: this category usually includes cosmetic, textual, and partial issues with specific functions. They do not impact the validity of data nor hinder the completion of the user's full scenario. But resolving them can enhance the user's experience.

In the above classification we used fault, error, and failure terms interchangeably. In the literature [26] there is a clear distinction between these terms:

1. **Human error or mistake**: a human behavior that results in system faults.
2. **System Fault**: a characteristic of a software system that can lead to system error.
3. **System error**: an erroneous system state that can lead to unexpected behavior by the users
4. **System Failure**: an event that can occur at a point of time leading the system to deliver unexpected results to the users

Regardless of the classification, there has to be techniques to resolve faulty behaviors. These approaches are classified as follows [26]:

1. **Fault Avoidance**: this approach depends on adopting best practices, tools, programming languages and techniques to minimize error-

prone problems caused by humans.

2. **Fault Detection and Removal**: by using validation and verification techniques to increase the probability of detecting faults before the system is used
3. **Fault Tolerance**: these are techniques that ensure that faults in the system will not cause errors and if there are errors they will not cause failure

The Fault Tolerance Requirements are summarized in Table 4.

Table 4 Fault Tolerance Feature Requirements

ID	Name	Note
13	Detect faults quickly	The system must detect faults very quickly.
14	Minimize Faults	The system must adopt all possible techniques to avoid or minimize faults.
15	Minimize the probability of an object to be offline	The system must ensure the longest number of hours for its object(s) in order to keep providing the automation service for its interacting devices and users.
16	Reduce Error consequences	If an error occurred, then the system must reduce its impact.
17	Show proper error message	The system must show a friendly, descriptive, and directive error message.
18	Take the proper corrective action	The system must take the proper corrective action to rectify the error and reduce its impact.

4-5 HETEROGENEITY OF DEVICES

Heterogeneity of devices implies diverse features that are best functioning within the manufacturer's devices. Only the manufacturer's developers can make the best solution out of their devices. There are of course architectural approaches to resolve this dilemma; however, it is still a dilemma with incomplete and insufficient solutions.

Let's take a single famous example, smart mobile phones. There are different key players in the market like Samsung, Apple, and HTC. Every manufacturer has its own OS. For example, Samsung uses Google Android, Apple uses iOS, HTC uses Android and Windows, and Nokia uses Symbian OS [27]. There are different sizes for the phones, and they come now with bigger sizes that range from handy-small phones to large tablets. Rendering a video on these different devices varies noticeably.

Integration projects are quite costly and they usually exceed their timelines at a very high investment cost, worldwide [28]. The factors that increase the risks of the integration projects include the following:

1. As the number of heterogeneous devices increases, the risks and development time increase as well.

2. The number of integration points: Risks and development time increase as integration points tend to increase.
3. The availability of documentation that describes the device interface: This issue is considered a real problem with legacy systems that depended on developers who did not value the importance of documentation.
4. The availability of good architects: who can understand the whole picture and build a robust integration architectural model.
5. The learning curve: for the developers who should learn the new interfaces and use the knowledge to understand the integration problems.
6. The availability of a development environment: that covers all different integration interfaces. This will minimize the risk of faulty functions during run-time after deploying the developed software.

The Heterogeneity of Devices Requirements are summarized in Table 5.

Table 5 Heterogeneity of Devices Feature Requirements

ID	Name	Note
19	Maximize the number of device technologies	Allow different devices that use different technologies to join/leave the pervasive system with minimal human involvement.
20	Provide a unique identifier for every object	Every object should have a unique identifier that does not conflict with other objects such as IP or MAC address.
21	Render content on maximum number of devices	Allow different devices to render the same content according to their screen dimensions, network bandwidth capacity, and processing capabilities. The content should be visible, readable, and interactive.

4-6 INVISIBILITY

A classical automation system is recognized by the users through integrating its hardware and software assets. The user cannot complete his/her tasks without using the computer explicitly to achieve his/her goal. This classical experience includes some basic activities as follows:

1. Switch on the computer.
2. Log on to the operating system.
3. Go to the software location.
4. Run an executable file of the software.
5. Navigate inside the software and supply it with the required inputs.
6. Apply the changes and wait for the output.

The invisibility feature should ideally eliminate almost all of the above activities and replace them with the implicit input [22] and invisible automation of activities. For example, the system may use user movements, activities,

writings, and gestures as input that guides the system to achieve the goal of the customer. On the other hand, the user may need to interact with the system in some situations, but they should be as minimal as possible.

Invisibility requirements are summarized in Table 6.

Table 6 Invisibility Feature Requirements

ID	Name	Note
22	Minimize unneeded interactions with the system	The system must automate as many activities as possible in order to minimize interaction with the system.
23	Remove unnecessary motions	A pervasive system should reduce the time and effort that people usually take to complete their tasks by making them simple and intuitive.
24	Conceal the part object(s) of the pervasive system	By concealing the system part object(s) in the smart environment fabrications as much as possible.
25	Minimize the use of explicit input	The system should detect inputs implicitly and minimize the use of traditional keyboard and pointing devices [24].

4-7 PRIVACY AND TRUST

We all have private information about ourselves. Humans reveal private information about themselves only to those whom they trust, even those well known to their media. The issue of privacy and trust is crucial for pervasive computing systems. There are always sensors in such systems that collect data about different objects like temperature, images, sounds, locations, etc… We decided to merge privacy and trust as one quality attribute because they are interrelated. As shown by studies and experiments [29], high trust compensates for low privacy and vice versa.

We see the issue of privacy and trust as a three dimensional model. The dimensions are:

1. **Information**: this information could be classified as public, social or private. Information is also captured through direct input from users or detected from their activities, or sensed from the environment.
2. **Trusted entities**: these trusted entities could be classified as highly-trusted, medium-trusted, or low-trusted entities which could be humans or devices.
3. **Situations**: such situations are two-dimensional variables including time and location [30]. For example, people may be willing to reveal private information about themselves with parents or doctors. People may reveal information also whenever they use their personal notebooks or cell phones.

As explained above, information is not always classified as private, social, or public. Moreover, trusted entities are not always on the same level. There

are some entities, human or devices, that are classified as highly-trusted for one person, but those same entities may not be trusted for others. Devices may also be classified as personal, which means they are highly-trusted. For example, headphones are devices that could be used in a private manner [30].

We can summarize the requirements for privacy and trust in a pervasive system as shown in Table 7.

Table 7 Privacy and Trust Feature Requirements

ID	Name	Note
26	Certify trusted entities	Entities that will manipulate information should be certified. For example, a system may require registration with details, and then an admin reviews it in order to grant the right authority level.
27	Classify Information	The system must be able to differentiate among private, social, and public information.
28	Reveal Information controllably	The system must reveal information to authorized entities only based on its own classification, and on the trust level of the authorized entities.
29	Track Information	The system should trace private information to other entities. Traceability may be used later on by the user who owns this information if it is misused.

4-8 QUALITY OF SERVICE

Quality of service in our scope refers to the agreement protocol that the pervasive system signs with users and other systems about its service boundaries. For example, the system may declare that it can serve a user within 0.01 seconds for the requested data and that the time can increase by a maximum of 1 second for a number of concurrent users which does not exceed 1000 at the same time. In other words, it is the ability of the system to meet deadlines [23]. We can classify a deadline as [31]:

1. **Hard deadline**: if the system does not meet its deadline, then the operation is considered failed. This is obviously found in car embedded system, as it is not acceptable that the brake sensor delays its response and causes accidents.
2. **Soft deadline**: where the system may exceed the deadline. The result in this case is controversial, since it could be considered failed, succeeded with a lower percentage or the deadline is just there for reporting and future improvement considerations. For example, if a movie encoder slips its deadline causing a slight pause, it only degrades the QoS and it could be acceptable or rejected according to the situation.

QoS boundaries can be applied across all the system quality features like security, context awareness, and fault tolerance.

The QoS Requirements are summarized in Table 8.

Table 8 Quality of Service Feature Requirements

ID	Name	Note
30	Declare service/quality feature boundaries	The system should specify its acceptable boundaries for each feature or service by which the users can acknowledge the failure of the service if the deadline is breached.
31	Minimize average processing capability	The system should process tasks very quickly and on time.
32	Monitor and improve QoS boundaries	The system must continuously monitor its QoS for the different services and work on improving them whenever possible.
33	Specify hard/soft deadline	The system must flag each response deadline for being a hard or a soft deadline.

4-9 SAFETY

The safety characteristic addresses two aspects of the pervasive system. The first is the system safety, which is concerned with its hardware wellbeing. The second is concerned with the environment's safety where interacting users and machines are kept safe from physical harm or damage [25]. In both cases, safety is very important as it makes no sense to have a system that can damage itself or harm its environment.

When it comes to organizing priorities, then a pervasive system must sustain its hardware healthiness unless this could cause harm to its users. Yang and Helal [32] advise that any solution must cover the four main components of the system which are: device, service, user, and space.

Safety Requirements are summarized in Table 9.

Table 9 Safety Feature Requirements

ID	Name	Note
34	Alert if safety is or about to be compromised	Alerts could be in multiple forms. For example, an alert could appear on a screen and is associated with a high sound.
35	Allow the user to override/cancel system decisions	If the system makes a wrong action that can cause a potential risk for users, then allow the users to override its action or cancel it.
36	Avoid conflicting side effects	The system must take proper actions to avoid side effects on people and devices.
37	Avoid invalid operational directives	The system must provide safety limits for critical operations in order not to cause damage based on wrong user input.
38	Ensure that generated rules do not conflict with system policy	The system may generate new rules driven from its knowledge base. The new rules must not conflict with the system policy that governs the usage of the system.
39	Minimize conflicting usage of shared resources	The system must be able to resolve conflict over shared hardware resources.

40	Override system rules by the regulator	The regulator should have the authority to override system rules in critical situations in order to apply its rules on all concerned members of the society
41	Provide maximum protection for the environment	Interacting users and machines should be protected from injury and damage.
42	Resolve conflicts among objects by an administrator	There should be a way for the administrator to resolve conflicting situations among objects.
43	Respect societal ethics	The system must abide by the societal ethical standards.

4-10 SECURITY

This is a classical and also a critical aspect for any pervasive system. It becomes even more important in a pervasive system whose nature requires high flexibility, openness, mobility, and interaction with new devices which may not be trusted [33]. The eternal goal for this characteristic is to provide data protection and fight system attacks. The term "Data" here refers to any kind of data the system stores or transmits. For example, if a user tries to access the system, it implicitly means that he/she will transmit data (login credentials) to access his/her profile (stored data). The system must ensure the integrity of the users' profiles so that they can access the system later on.

Security risks are handled using three approaches [34]:

1. **Eliminate the threat**: during the design of the system, the risks are identified and the solution is designed in a way that prevents them from the beginning.
2. **Mitigate the risk**: it is not possible to eliminate the risks but the system can take counter-measures to eliminate harm or remove it.
3. **Accepted risk**: this approach can be adapted if risks cannot be eliminated or mitigated. However, users of the system must understand such risks before using the system.

We do not recommend allowing anonymous usage of the system services and resources similar to what was proposed in [12]. Instead, the privacy of the users should be protected and must be revealed only for authorized entities.

There are differences between Security and Privacy and Trust:

1. Security is concerned with the policies that govern the data manipulation and availability while privacy is concerned with the appropriate use of the data.
2. Security rules are embedded in the system, while privacy and trust is about corporate and personal responsibilities.
3. Strong security policies minimize the risk of violating the privacy of information. However, there is no guarantee that responsible people

about the private data will not reveal it to unauthorized entities (e.g. selling data to third-party agencies for digital advertisements).

Security Requirements are summarized in Table 10.

Table 10 Security Feature Requirements

ID	Name	Note
44	Disallow anonymous usage of the system	The system must not allow anonymous access to system resources and services.
45	Enforce Security rules on all objects	The system must ensure that the security policy is applied on all devices that join the system. Devices that fail to fulfill the security requirements must disconnect immediately.
46	Ensure secure data transmission	Data transmission among devices must be secured and protected against intruders [11].
47	Maintain data integrity	The system must ensure corruption-free and alteration-free data.
48	Prevent data leakage	Provide maximum protection for data in order to avoid leakage to unauthorized persons [11].
49	Provide data access rules	Data should be accessed whenever needed by different entities, persons or machines, according to the data security access rules.
50	Take counter-measures to mitigate security threats	The system must take counter-measures to ensure that risks generated from security threats do not cause any harm to system users.
51	Announce malfunctioning smart objects	Some objects may harm the environment, and the community must be aware of such objects in order to put them in the black list.

4-11 SERVICE OMNIPRESENCE

Omnipresence means “present everywhere at the same time.” Service Omnipresence means that the user must get the feeling that he/she is carrying computer services whenever he/she wants and wherever he/she goes. In other words, the user should be able to use his/her computing services whenever he/she wants and in almost any place. Given that it is almost impossible to facilitate computing services everywhere and at any time, it is important that the user gets this feeling.

We will use the term perception instead of feeling in order to provide a better understanding for this quality feature. Perception is the ability to recognize something based on its form. The perception process is dependent on the features of the object and the organization of these features [19]. One can perceive a cat from its main features (head, legs, tail, and sound) but these features have to be allocated correctly in order to call this object a cat. The same happens in pervasive computing systems, the basic features of the pervasive system have to exist, e.g. sensors, context awareness, and actuators. If the pervasive system is spread across a large space, then the sensors should be spread all over the environment professionally and in a

way that serves the user's needs.

The Omnipresence Requirements are summarized in Table 11.

Table 11 Service Omnipresence Feature Requirements

ID	Name	Note
52	Distribute computing power	It is highly recommended to distribute computing capabilities in the environment where a pervasive system operates.
53	Enrich the experience of the highly used scenarios	Such scenarios must get the highest attention and enrichment with the pervasive features (sensors, awareness, actuators, intelligence)
54	Provide Informative messages	Make sure to guide the user and build up experience through hints, tips, and messages.
55	Use a unique user identifier	A unique user identifier that can be used to access different devices which can give the user the feeling that the system knows him/her.
56	Utilize the user mobile phone	Users depend heavily on their mobile phones. Smart phones are now considered a small computer with multiple capabilities. Hence allow the user's mobile phone to be part of the system

5- TRADE-OFF ANALYSIS

In order to analyze relationships among the quality features, we studied the relationships among the requirements of the quality features. For example, we found that the requirement #55 "Use a unique user identifier" has a possible conflict with the requirement #20 "Provide a unique identifier for every object". This is because a user may have more than one device joining the pervasive system, but the user must be identified all time as the same user not a different one. So, we decided to resolve any possible conflict by making the requirement "Use a unique user identifier" supersede because having a unique user identifier will ensure that different rules associated with that user are cascaded properly for devices associated with him/her. This indicates that there is a possible conflict between the Service Omnipresence feature and the Heterogeneity of Devices feature. We believe that our decision is applicable in any business domain based on our rationale.

There are other requirements that maximize or minimize the expected value from other requirements. For example, requirement #52 "Distribute computing power" maximizes the value of requirement #10 "Capture Knowledge about users" since distributed computer power, including sensors, can capture more information about users, their habits, movement patterns, and routine actions over the space of the smart environment. On the other hand, requirement #55 "Use a unique user identifier" minimizes/eliminates the threat in requirement #44 "Disallow anonymous usage of system" as the user will be able to use the system only if he/she is identified. Accordingly, anonymous usage of the system will not be allowed.

We modeled the requirements relationships, shown in the Appendix, and as

explained earlier in our approach, using the conflict, maximize, and minimize stereotypes. We got 44 relationships for 39 requirements. The relationships among the quality features within the research scope are summarized as follows:

1. The Service Omnipresence and Adaptable behavior features have the highest number of outgoing minimize relationships related to 4 and 1 quality features, respectively.
2. The Safety and Fault tolerance features have the highest number of incoming minimize relationships and they related to 2 and 3 quality features, respectively.
3. The Service Omnipresence feature has the highest number of maximize relationships and it is related to 4 quality features.
4. The Safety and Privacy and Trust features have the highest number of maximize relationships and they are related to 4 and 3 quality features, respectively.
5. The Security feature conflicts with the Quality of Service feature in 3 requirements.
6. The Context Sensitivity feature does not conflict with Adaptable Behavior nor Fault Tolerance.
7. The Device Heterogeneity and the Security features have the highest conflict relationships.

Fig. 3 highlights the superiority of quality features, derived from Table 13 (in the appendix), whose requirements may have conflicts and the direction of the decision of the feature that should supersede. The arrow starts from the superseding feature. For example, ST has a higher priority than the QoS since they both conflict on 3 requirements and the resolution is in favor of the requirements in ST. The conflict among the quality features is resolved by resolving the conflicts between the requirements. It is clear that the overall superiority of quality features cannot be detected from the conflict relationship only since the Adaptable behavior and Invisibility quality features are not linked to the other quality features.

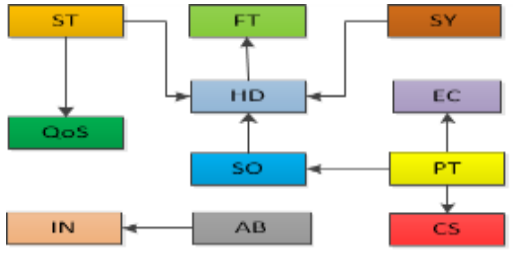

Figure 3 Quality Features Conflict Resolution Priority

In order to determine the overall superiority levels, we analyzed the maximize

and minimize relationships. The following facts could be detected:

1. **There are enabler features:** these are the features that appear as a source with a percentage higher than 50%. Those features are AB, CS, HD, and SO. The fulfillment of the requirements of these features will help other features achieve their requirements. One may think of the enabler feature as a tool similar to what was explained by the activity theory. So, we can define the enabler feature as "**the feature that has the requirements that minimize or maximize the value of other requirements.**"
2. **There are constraining features**: these are the features that appear as a destination with a percentage higher than 50%. The requirements that belong to these quality features are empowered by the enabler features and are enforced on the system as constraints. These features are PT, QoS, SY, ST, FT, and EC. One may think of the constraint feature as a rule similar to what was explained in the activity theory. So, we can define the constraint quality feature as "**the feature that has ruling requirements that must be fulfilled by other quality features.**"
3. **The Invisibility Feature role is unclear**: it is not possible, from the given requirements and relationships, to decide if the Invisibility feature is an enabler or a constraint feature since it appears 50% as source and 50% as destination.

Fig. 4 shows a graphical classification of the enabler and constraint features with their relative proximity from the Enabler and Constraint categories.

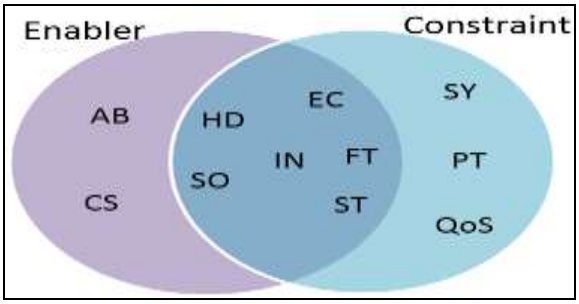

Figure 4 Enabler-Constraint Quality Features Categories

If we follow the chain of superiority depth in Fig. 3, and based on the finding of the Invisibility feature which has no clear classification, and given that the Adaptable behavior feature supersedes Invisibility (Fig. 3), we conclude the following priority layers (pyramid in Fig. 5). The rule is that:

1. Features that have no incoming arrow have higher priority.
2. The next layer includes features that are nested with one incoming arrow, and so on.

3. The Adaptable Behavior and Invisibility are added at the base of the pyramid since both features are not connected with the rest of the quality features.

Figure 5 Quality Features Priority Pyramid

We can further explain the relative weight of every quality feature in terms of complexity and its impact on other features and the number of requirements in every quality feature. We give a score for every quality feature calculated as the number of requirements per feature multiplied by the number of requirements relationships multiplied by the number of features that are covered by these relationships.

We can explain the complexity equation as follows:

1. The requirements in a feature represent its *weight.*
2. The relationships of the requirements in a feature represent its *collaboration weight* with other requirements.
3. The number of covered features represents the *diversity of the collaboration.*

For example the Safety feature has 10 requirements, 11 relationships, and its relationships cover 4 features. By multiplying 10 by 11 by 4, we get the safety complexity score as shown in Fig. 6. Fig. 6, which is based on the total scores of the quality features, shows that 4 quality features (Security, Safety, Service Omnipresence and Fault Tolerance) represent 67.6% of the overall weight for the quality features. In other words, the requirements of these features will need deeper analysis to ensure that the system is implemented on solid basis. It does not mean that the other features are less important. However, in a real project, for example, a decision could be taken to assign more experienced analysts and architects to study these 4 features, or give more time to analyze their requirements.

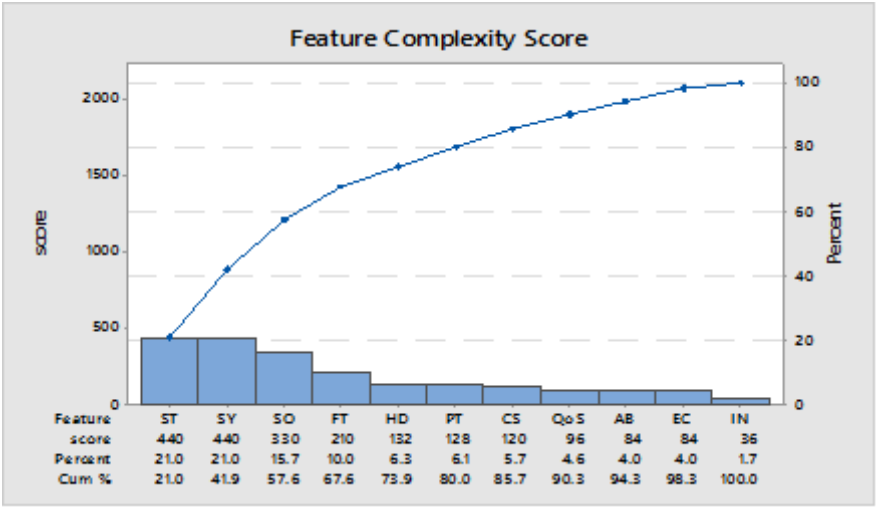

Figure 6 A Pareto diagram for the quality features accumulated complexity weight

In order to evaluate our priority scale of features, we ran a survey with 17 persons asking them to give a score of importance from 1 to 5 for every requirement where 1 means (not important at all) and 5 means (extremely important). The survey was categorized on features with their requirements. All descriptions of the features and the requirements were given to the respondents. The survey was conducted as a blind study where all the knowledge is given in the survey with no examples or detailed explanations. The respondents had different years of experiences in software engineering in general and in different business domains like Telecommunication, mobile applications, web applications, ubiquitous computing, and Human Computer Interaction. Nine of them have over 15 years of experience. Some of the respondents are in management positions and the majority are involved in technical activities.

We averaged the score for every requirement and we then took the average of the requirements that belong to a specific quality feature. We then got a list of 11 quality features ordered according to the given average score.

The results that we got were very interesting. We found that respondents had very close points of views that are similar to our statistical analysis as shown in Fig. 5 and Fig. 6. Although the features were not in exactly the same order as in Fig. 6, the results were segmented almost the same as the priority pyramid in Fig. 5. The standard deviation (SD) of the difference of ranking between the survey order and the complexity order, as shown in Table 12, is 2.3741 which is relatively small. If we divide the number of features by the SD, the result will be 4.8, which indicates that we can segment the ranking of the features into 5 segments.

Table 12 Comparison between the Survey score and the Complexity Score

Feature	Survey Order (SUO)	Complexity order (CXO)	Difference (CXO – SUO)
FT	1	4	3
PT	2	6	4
ST	3	2	-1
SY	4	1	-3
SO	5	3	-2
QoS	6	8	2
CS	7	7	0
AB	8	9	1
HD	9	5	-4
IN	10	11	1
EC	11	10	-1

We derive the following findings from the trade-off analysis study:

1. **Requirements Relationships are indicative of their priority**: a reasonable conclusion about the priority of the system requirements can be reached based on the statistical analysis of the requirements relationships by either using the complexity score method or the priority resolution method.
2. **Priority is not static**: Although the system architect can define a specific priority for every quality feature during the development phase or at runtime, the priority of the feature can be changed according to the context of the problem. The change of ordering could be 2-3 steps up or down as per the SD value. Changes of the priority that exceeds 2 steps must be carefully verified to ensure that the overall goal of the system can still be achieved.
3. **The Constraint Feature is more important**: although the pervasive systems sell for the smart features like context sensitivity and the adaptable behavior, the system will not be usable if features like safety, security, and privacy and trust are not treated equally to the enabler features.

6- CONCLUSION

We introduced a comprehensive reference requirements model for pervasive computing systems that can be easily adapted by software and system architects. Our approach of using the activity theory and process re-engineering concepts to analyze the requirements proved to be an efficient technique. We were able to provide a deeper understanding of the relationships among the requirements and link them with human activities. Business Analysts and Architects can use the requirements model and build relationships with the project's functional requirements, as we did, in order to

build better architectural models. The reference architecture will save the business analyst time as well, letting him/her focus on the project's functional needs.

This reference model can also be useful to study domain areas like the IoT, embedded systems, M2M, and autonomic computing. They all share the same basic requirements represented in our model.

We understand that real projects may have time, budget, and resource constraints that hinder the project team from running a comprehensive study to analyze business and functional requirements of a system. Accordingly, it is a good opportunity to use this reference architecture as a base to study project requirements. Moreover, some of the project timelines could be organized based on priorities as was explained in the trade-off analysis and relationship dependency. To our best knowledge, the subjective evaluation of the requirements is the dominant method used by the software engineers to decide on the priority of the requirements. So, it is an advantage to have a statistical approach that gives accurate results.

The quality features presented above may not be complete. However, they have all the essential requirements. We did not want to add detailed requirements since this would not make the business reference architecture generic enough. Our goal is to introduce a comprehensive business reference architecture that provides a conceptual model for the smart environment. It will also have an extended feature model that will have values and issues driven from the requirements model with helpful breakdown attributes for measuring their performance. The approach breaks down the complexity of the pervasive system and that makes it easier for the architect to define compelling solutions.

We are currently studying some business domains (e.g. retail, emergency, and learning) for which the trade-off analysis results can best be applied. We will research how the weights of requirements in these domains can vary with respect to the weights of the quality features. In other words, we will research the impact of the generic requirements model of the quality features on other requirements models from different business domains.

Our aim is to provide a technical reference architecture that satisfies the requirements of the business model. Hence, we will also study other quality features that we did not address in the scope of this paper like spontaneous interoperability, scalability, openness, service discovery, and function composition [16]. Identifying the referenced business architecture layer for pervasive computing is the first step towards building a practical technical reference architecture. We also have an ambitious plan to carry out both subjective and quantitative evaluations for the generated technical architecture.

APPENDIX

Table 13 shows relationships among the requirements of the quality features.

Table 13 Quality Features requirements relationships

Source Feature	Source Req. ID	Relation	Dest. Feature	Dest. Req. ID	Superseding Req.
AB	2	mi	SY	39	
AB	2	mi	SY	37	
AB	2	mi	SY	36	
AB	1	mx	QoS	32	
AB	3	mx	SY	34	
AB	3	mx	SY	35	
CS	5	mx	PT	27	
CS	5	cf	PT	28	28
CS	5	mx	EC	10	
CS	7	mx	QoS	32	
CS	7	mi	FT	16	
CS	6	mi	FT	16	
EC	10	mx	EC	11	
EC	10	cf	PT	28	28
EC	11	mx	EC	12	
FT	18	mi	IN	22	
FT	13	mi	FT	16	
HD	19	cf	SY	39	39
HD	19	cf	SY	36	36
HD	19	cf	FT	14	19
HD	19	cf	ST	45	45
HD	21	mx	SO	53	
HD	21	mx	SY	35	
HD	20	mx	ST	49	
HD	20	mi	SY	39	
HD	20	mi	ST	44	
IN	24	mx	SY	41	
IN	22	cf	AB	3	3
ST	49	mx	PT	28	
ST	46	cf	QoS	31	46
ST	45	cf	QoS	31	45
ST	45	mx	PT	28	
ST	50	mx	SY	41	
ST	50	cf	QoS	31	50
SO	55	mx	PT	26	
SO	55	cf	HD	20	55
SO	55	mx	ST	49	
SO	55	mi	ST	44	
SO	56	mx	PT	26	
SO	56	mx	HD	21	
SO	52	mi	QoS	31	
SO	52	mx	EC	10	
SO	54	mi	FT	16	
SO	54	cf	PT	28	28

ACKNOWLEDGMENT

We would like to thank the following experts for their help and support in this research work as participants in the focus group workshop study:

- Ahmed Ibrahim and Hassan Ali: IBM Egypt
- Hany Ouda: Etisalat Egypt Telecommunications
- Mohamed H.Abdelrahman: Vodafone Egypt Telecommunications

REFERENCES

[1] M. Weiser, “The Computer for the 21st Century,” Scientific American, September, 1991.

[2] Intel Edison, http://www.intel.com/content/www/us/en/do-it-yourself/edison.html [Last accessed: December 2016].

[3] ARTIK Series, https://www.artik.io/ [Last accessed: December 2016].

[4] B. Connolly, "Business IoT Connections Expected to Hit 5.4 Billion By 2020," http://www.pcadvisor.co.uk/feature/internet/business-iot-connections-expected-to-hit-54-billion-by-2020-3599257/ [Last accessed: December 2016].

[5] B. Bangerter, S. Talwar, R. Arefi and K. Stewart, "Networks and devices for the 5G era," IEEE Communications Magazine, vol. 52, no. 2, pp. 90-96, February 2014. doi: 10.1109/MCOM.2014.6736748.

[6] M. Castelluccio, "A New Bluetooth in 2016," Strategic Finance, pp. 55-56, 2015.

[7] S. Jacobs, "Driving Future Technology With Solar-Powered Energy Harvesting," Product Design & Development, pp. 26-27, 2015.

[8] D. Waltenegus, P. Juha, and V. De Florio, “Existing challenges and new opportunities in context-aware systems,” Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12), ACM, New York, NY, USA, pp. 749-751, 2012.

[9] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, “The concept of reference architectures,” Systems Engineering, vol. 13, pp. 14-27, Jan 2009.

[10] C. Machado, E. Silva, T. Batista, J. Leite, E. Nakagawa, “RA-Ubi: a Reference Architecture for Ubiquitous Computing,” Proceedings of the 8th European Conference on Software Architecture (ECSA), 2014.

[11] I. D. Addo, S. I. Ahamed, S. S. Yau, and A. Buduru, "A Reference Architecture for Improving Security and Privacy in Internet of Things Applications," IEEE International Conference on Mobile Services (MS), pp. 108,115, June 27 -July 2, 2014.

[12] European Lighthouse Integrated Project: Internet of Things Architecture IoT-A Project Deliverable D6.2 – Updated Requirements. http://www.iot-a.eu, January 2011.

[13] R. Al Ali, I. Gerostathopoulos, I. Gonzalez-Herrera, A. Juan-Verdejo, M. Kit, and B. Surajbali, "An Architecture-Based Approach for Compute-Intensive Pervasive Systems in Dynamic Environments", International Workshop on Hot TopiCS in Cloud Cloud service Scalability (HotTopiCS 2014), Dublin, Ireland, Mar 2014.

[14] M. Soegaard and R. Friis Dam, "The Encyclopedia of Human-Computer Interaction", 2nd Ed., the Interaction Design Foundation, 2011.

[15] N. Kim, S. Lee, and T. Ha, "Understanding IoT Through the Human Activity: Analogical Interpretation of IoT by Activity Theory," HCI International, Posters' Extended Abstracts, vol. 528 of the series Communications in Computer and Information Science pp 38-42, Springer International Publishing, 2015.

[16] H. Gleitman, J. Gross, D. Reisberg, "Perception, "Psychology, 8th Ed. W.W.Norton & Company, pp.258-298, 2011.

[17] G. V. Bodenhausen and K. Hugenberg, "Attention, Perception, and Social Cognition," F. Strack & J. Förster (Eds.), Social cognition: The basis of human interaction, Philadelphia: Psychology Press, pp. 1-22, 2009

[18] K. Cherry, "Perception and the Perceptual Process," https://www.verywell.com/perception-and-the-perceptual-process-2795839 [Last accessed: December 2016].

[19] H. Gleitman, J. Gross, D. Reisberg, "Perception, "Psychology, 8th Ed. W.W.Norton & Company, pp.181-217, 2011.

[20] A. Gunasekaran and B. Kobu., "Modelling And Analysis Of Business Process Reengineering," International Journal of Production Research, vol. 40, 2002.

[21] R. Spínola, and G. Travassos, "Towards a framework to characterize ubiquitous software projects," Information and Software Technology, vol. 54, pp. 759-785, 2012.

[22] S. Dobson, R. Sterritt, P. Nixon, M. Hinchey, "Fulfilling the Vision of Autonomic Computing," Computer, Vol.43, no.1, pp.35-41, Jan. 2010.

[23] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, “Mobile And Ubiquitous Computing,” Distributed Systems Concepts and Design, 5th Ed. Addison-Wesley Publishing Company, pp. 817-878, 2012.

[24] G. D. Abowd, E. D. Mynatt, T. Rodden, "The Human Experience [Of Ubiquitous Computing," Pervasive Computing, IEEE , vol.1, no.1, pp.48-57, Jan.-March 2002.

[25] O. M. Khaled, H. M. Hosny, and M. Shalan, “On the Road to a Reference Architecture for Pervasive Computing,” The 5th International Joint Conference on Pervasive and Embedded Computing and Communication Systems, Angers, France, Feb 11-13, 2015.

[26] I. Sommerville, “Dependability and Security,” Software Engineering, 9th Ed, Addison-Wesley Publishing Company, 2011.

[27] M. Nosrati, R. Karimi, and H. A. Hasanvand, “Mobile Computing: Principles, Devices and Operating Systems,” World Applied Programming, vol. 2, issue 7, pp. 399-408, July 2012.

[28] S. Purao, S. Paul, and S. Smith, “Understanding Enterprise Integration Project Risks: A Focus Group Study,” 18th International Conference on Database and Expert Systems Applications, DEXA ’07, pp. 850–854, 3-7 Sept. 2007.

[29] A. N. Joinson, U. Reips, T. Buchanan, and C. B. Paine Schofield, “Privacy, Trust, and Self-Disclosure Online,” Human–Computer Interaction, vol. 25, pp. 1-2, 2010.

[30] V. Kostakos, E. O'Neill, A. Penn, “Designing Urban Pervasive Systems,” Computer, vol. 39, no. 9, pp. 52-59, September 2006.

[31] S. Hua, G. Qu, "A New Quality of Service Metric for Hard/Soft Real-Time Applications," Proceedings of International Conference on Information Technology: Coding and Computing [Computers and Communications], ITCC 2003, pp.347-351, 28-30 April 2003.

[32] H. Yang and A. Helal, “Safety Enhancing Mechanisms for Pervasive Computing Systems in Intelligent Environments,” Sixth Annual IEEE International Conference on Pervasive Computing and Communications, 2008.

[33] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, “Security in Distributed Systems Concepts and Design,” 5th Ed. Addison-Wesley Publishing Company, pp. 463-518, 2012.

[34] A. Ray and R. Cleaveland, “An Analysis Method for Medical Device Security,” Proceedings of the 2014 Symposium and Bootcamp on the Science of Security (HotSoS '14), ACM, New York, NY, USA, , Article 16 , 2 pages, 2014.