VERS L'ÉVALUATION DES SYSTÈMES INFORMATIQUES UBIQUITAIRES

par

Yasir Malik

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 13 Juin 2014

TOWARDS EVALUATING PERVASIVE COMPUTING SYSTEMS

by

Yasir Malik

Thesis submitted to the Department of Informatics
to obtain the degree of Doctor of Philosophy (Ph.D.)

FACULTY OF SCIENCE

UNIVERSITY OF SHERBROOKE

Sherbrooke, Québec, Canada, 13 June 2014

Le 13 June 2014

Le jury a accepté la thèse de Yasir Malik dans sa version finale

Membres du jury

Professeur Bessam Abdulrazak
Directeur
Département d'informatique
Faculté des Sciences
Université de Sherbrooke, Canada

Professeur Stefan D. Bruda
Membre externe
Département of Computer Science
Bishop's University, Canada

Professeur Hamid Mcheick
Membre externe
Département d'informatique et de mathématiques
Université du Québec à Chicoutimi, Canada

Professeur Hugo Larochelle
Président-rapporteur
Département d'informatique
Faculté des Sciences
Université de Sherbrooke, Canada

Résumé

L'informatique diffuse est le passage du paradigme informatique vers l'informatique partout. L'émergence couvre principalement l'informatique mobile et distribuée, les réseaux de capteurs, l'interaction homme-machine et l'intelligence artificielle sous l'égide de l'informatique diffuse. Des efforts considérables ont été mis sur les recherches dans ce domaine, mais il n'existe pas de normes ou des méthodologies communément acceptées pour évaluer ces systèmes et de définir des nouvelles orientations de recherche dans le futur. Cette thèse s'attaque au problème d'évaluation des systèmes informatiques ubiquitaires. La question de recherche notamment le quoi et comment évaluer n'a pas encore été résolue. Dans l'objectif de trouver une réponse à cette question et d'élaborer un cadre général d'évaluation, nous avons procédé comme suit.

Pour répondre à la première partie de la question, "Quoi évaluer", nous avons tout d'abord classé les systèmes en se basant sur certains critères, et nous avons défini ensuite les principaux paramètres pour évaluer ces systèmes. Pour ce faire, nous avons étudié différents aspects de l'informatique diffuse et nous les avons classés en onze différents aspects/domaines d'évaluation. Pour chaque aspect, nous avons identifié les principaux paramètres qui peuvent être caractérisés et mesurés. Cette taxonomie n'est pas assez exhaustive, mais elle reflète le schéma de classification le mieux adapté pour des évaluations effectives. Cependant, pour que l'évaluation soit la plus complète possible, nous avons jugé nécessaire d'incorporer l'utilisateur dans le processus d'évaluation. À cet effet, nous avons proposé un modelé d'évaluation qui prend en compte les besoins de l'utilisateur, le contexte dans lequel la technologie sera utilisée, et l'environnement d'exploitation dans lequel le système va être déployé. Le modelé proposé constitue une première étape vers le développement des directives

et standards d'évaluation qui peuvent être utilisés peuvent être utilisées pendant les évaluations formatives et sommatives.
Une autre question complémentaire à l'évaluation des performances est la validation fonctionnelle d'un système en cours d'exécution, qui confirme que le système est conforme aux exigences fonctionnelles et ne contient pas de failles. Pour répondre à la deuxième partie de la question à savoir "comment évaluer", nous avons adopté les techniques formelles de vérification et de validation. Comme le champ d'application du projet est très large, nous sommes concentrés sur l'évaluation au premier stade de la conception afin de vérifier et de valider l'exactitude fonctionnelle de la conception de systèmes.
Pour la preuve de concept, nous avons appliqué deux méthodes, dans la première méthode, nous avons étudié les approches de vérification automatique et nous avons choisi la technique la plus connue qu'est le "model checking" pour vérifier les exigences fonctionnelles d'un système de gestion des médicaments basé sur le contexte pour des personnes âgées dans une maison Intelligente. Cette approche est complémentaire aux tests et à l'évaluation et permet aux concepteurs de vérifier le comportement de leurs systèmes par rapport aux exigences fonctionnelles avant le développement du prototype de système. Certaines propriétés de base, telles que la disponibilité ou la vivacité, l'interblocage, la comparaison des spécifications et implémentations et l'analyse d'accessibilité, sont également vérifiées à ce stade.
Dans la deuxième méthode, nous avons étudié les approches de vérification d'exécution et nous avons adopté la technique de conception par le contrat pour modéliser et vérifier la sémantique et exigences de l'interopérabilité des services dans les environnements intelligents. L'avantage de cette approche réside dans la vérification automatique en temps réel de l'interopérabilité des services dans les environnements intelligents.

Summary

Pervasive computing is shifting the computing paradigm toward everywhere computing. This emergence covers distributed and mobile computing, sensor networks, human computer interaction and artificial intelligence. Tremendous efforts have been put in the related research, however no standards or commonly accepted methodology exists to evaluate these systems and identify directions for future research. The unsolved research question of ***"what and how to evaluate"*** pervasive computing system has not yet been answered. This thesis addresses the problem of evaluating pervasive computing systems. In an attempt to answer the above questions and designing an evaluation framework, we followed a chronological order of the research question.

To address the first part of the question,"*What to evaluate*", we assembled a user-centered framework that account for the system, user needs, context in which the technology will be used, and the operating environment in which the system will operate or deployed. Within each factor we consider important evaluation areas and identify the key aspects in the evaluation process. Contextual and environmental factors present key issues and aspects in the evaluation process. User associated evaluation areas are studied from the ergonomic point of view, that would help evaluators to evaluate the user acceptance aspects of the system. System evaluation aspects are identified by classifying distinctive features and studying the key performance parameters of interest for pervasive computing systems. Consequently, we studied different aspects of pervasive computing and classified them into eleven different aspects of evaluation. Within each aspect, we identify key parameters that can be characterized and measured. This taxonomy is by no means complete, but merely reflects the classification scheme that is best suited for the purpose of effective performance evaluations. The proposed model is a step towards forming standard evaluation guidelines that can be used during formative and summative evaluations.

A complementary issue to performance evaluation is functional correctness of a running system, which confirms that the system fulfills its functional requirements and does not contain any flaws. To address the second part of the question that is "*how to evaluate*", we have adopted the well-known formal verification and validation techniques. As the scope of the project is very big, the focus of this thesis is on early design stage evaluation to verify and validate the functional correctness of the systems design. For the proof-of-concept, we applied two methods:

In the first method, we studied automatic verification approaches and used a well-known model checking approach to model and verify the functional requirements of a context aware medication management system for the elderly in a Smart House. This approach is complementary to testing and evaluation, it allows designers to verify their system behavior against its functional requirements before developing the system prototype. Some basic properties like the availability or liveliness, deadlock checking, matching of specification and implementation, and reachability analysis are verified.

In the second method, we studied the runtime verification approaches and used design by contract technique to model and verify the semantic and pragmatic service interoperability requirements in smart environments. The analysis of this technique and results are presented. The benefit of the approach is automatic verification of services interoperability in smart environments on the fly.

Keywords: Evaluation, Performance, Taxonomy, System Factors, User Factors, Formal Methods, User-Centered, Design Analysis, Model Checking, Design By Contract.

Acknowledgment

"In the name of God, most Gracious, most Merciful"

I wish to express my sincere appreciation to those who have directly indirectly contributed to this thesis and supported me during this amazing journey.

First and foremost, I want to thank my advisor Professor Bessam Abdulrazak for believing in me and giving me opportunity to work with him. His support, guidance, funding, all the useful discussions, brainstorming sessions, article and thesis revisions and many more are the reason that today I am able to finish this milestone of my life. He not only mentor and support my research but he was always there during the times when I was really down and depressed due to personal problems. He was more like an elder bother to me away from my home.

My sincere gratitude is reserved for Professor Stefan D. Bruda for his invaluable insights and suggestions on the thesis topic. I really appreciate his willingness to meet me, help me with questions and in going through my project proposal and several drafts of my thesis.

In regards to the formal method, I thank Professor Marc Frappier for his invaluable insights and suggestions and valuable tips on the subject. I really appreciate his willingness to meet me, help me and letting me attend his course on the subject.

Besides my advisor, I would like to thank the rest of my thesis committee Professor Hugo Larochelle, Professor Hamid Mcheick, Professor Stefan Bruda, for their insightful comments, suggestions, encouragement. Their support and valuable comment helped to polish my thesis and research.

Very special thanks to the department of computer science for giving me the opportunity to carry out my doctoral research and for their financial support. It would have been impossible for me to even start my study had they not given me a scholarship and facilities for my studies. I would also like to thank the FORCE foundation for giving me scholarship in all years of my studies, which was a great help.

I would also like to thank all my friends from DOMUS lab for all their useful suggestions and for being there to listen when I needed an ear.

Words cannot express the feelings I have for my parents and my in-laws for their constant unconditional support.
Finally, I would like to acknowledge the most important people in my life my wife Jowaria, my daughters Iman and Aroush. They have been a constant source of strength and inspiration. There were times during the past four years when everything seemed hopeless and I didn't have any hope. I can honestly say that it was only my wife and daughters determination and constant encouragement that ultimately made it possible for me to see this milestone through to the end.

Lastly a big thankyou to Sherbrooke city, this small city has become a big and special part of my life.

Abbreviations

ADL Activities of Daily Living

AHP Analytic Hierarchy Process

API Application Programming Interface

CSP Communicating Sequential Processes

CSP# Communicating Sequential Programs

CTL Computation Tree Logic

DbC Design By Contract

ICT Information and Communication Technology

LTL Linear Temporal Logic

MC Model Checking

NIST National Institute of Standards and Technology

NFP Non Functional Properties

NFC Near Field Communication

PAT Process Analysis Toolkit

PDA Personal Digital Assistant

PRIDE PRediction In Dynamic Environment

PerCom Pervasive Computing

QoS Quality of Service

RFID Radio-frequency identification

SAAM Software Architecture Analysis Method

Abbreviations

SCORE	System, Component, and Operationally-Relevant Evaluation
SEQUEL	Solver for circuit EQuations with User-defined ELements
SOAP	Simple Object Access Protocol
SOA	Service Oriented Architecture
TAM	Technology Acceptance Model
UbiCom	Ubiquitous Computing
UI	User Interface
WCF	Windows Communication Foundation
XSD	XML Schema Definition

Contents

Résumé **ii**

Summary **iv**

Acknowledgment **vi**

Abbreviations **viii**

Contents **x**

List of Figures **xii**

List of Tables **xiii**

Introduction **1**

1 Background **11**

1.1 Vision and Paradigm 12

1.2 Concepts in Pervasive Computing 15

1.3 Design and Evaluation Challenges in Pervasive Computing 18

1.4 Summary 25

2 Related Work **26**

2.1 Exiting Solutions and Approaches 28

2.1.1 Evaluation Frameworks 28

2.1.2 Simulation, Testing and Verification Approaches 33

2.1.3 User Centered Approaches 38
2.1.4 Usability Studies 42
2.1.5 Contextual Evaluation Approaches 44
2.1.6 Technology Acceptance Models 45
2.1.7 Environmental and Ecological Evaluation Approaches 47
2.1.8 Analysis 48
2.2 Summary 51

3 What To Evaluate 52
3.1 User Centered Evaluation Framework 53
3.1.1 Environmental Factors 57
3.1.2 Contextual Factors 58
3.1.3 User Factors 59
3.1.4 System Factors 64
3.2 Summary 76

4 How to Evaluate 78
4.1 Early Design Evaluation 79
4.2 Early Design Analysis of Contextaware System: Model Checking Approach 83
4.2.1 System Scenario 84
4.2.2 Designing and Modeling the System 86
4.3 Verification of Dependability Properties 94
4.4 Runtime Analysis of Pervasive System: Design By Contract Approach 97
4.4.1 Requirements Analysis 102
4.5 Summary 107

Conclusion 109

List of Figures

1.1 Pyramid of Pervasive Computing 13

2.1 Design and Evaluation Lifecycle 27

3.1 Design and Evaluation Model for Pervasive Computing System 56

4.1 System Architecture of Contextaware Medication System 87
4.2 System Architecture of PAT 95
4.3 Service Contract Model 103

List of Tables

2.1 Evaluation Challenges And Approaches At Each Design Stage 50

3.1 Summary of Quantitative and Qualitative Parameters of Systems and Users 56
3.2 Summary of User Aspects and Key Elements 64
3.3 Infrastructure Evaluation Parameters 66
3.4 Design Evaluation Parameters 68
3.5 Autonomicity Evaluation Parameters 69
3.6 Integration Evaluation Parameters 70
3.7 Service Availability Evaluation Parameters 71
3.8 Interaction Evaluation Parameters 72
3.9 Interaction Evaluation Parameters 74
3.10 Application Purpose Evaluation Parameters 76

Introduction

Evaluation is an essential process for successful deployment, assessment and acceptance of new technologies. An evaluation is an activity to assess a program or a product for its quality check and benchmarking. The outcome of this process is a report which helps to take decisions on pros and cons and to provide directions to improve the quality of the product or a program. Evaluation processes have been understood differently in different disciplines and thus have resulted in various methodologies and approaches depending on existing knowledge and set standards in subject domain.

Evaluation of computing systems is not a new idea. It started with the rapid growth in development of software and hardware, numerous evaluation and simulation methods have been proposed and widely adopted in hardware/software design and development. The process is applied at every design and development stage, which helps to improve the quality of a product at every stage and helps to decide how to continue the development.

With the recent development in information communication technologies (distributed and mobile computing), computing systems have become more complex and thus evaluation of these systems has become a complex task [130]. The most important aspects in any computing system evaluation (hardware or software) are functionality, reliability, usability, efficiency, maintainability, portability and cost [62]. Although there could be many reasons for a computing system quality to be questioned, the two most significant reasons are:

1. *Functional Failure*: refers to the failure occurred when the input produces a wrong output. A simple example could be addition of two numbers that results in a wrong answer or an air defense tracking system that could not distinguish between a friendly and enemy missile.
2. *Performance Failure*: refers to the failure occurred when system is functionally correct, however computing the output for the input takes longer than expected. This failure can cause serious problems in time critical systems where an output becomes an input for the second process. A simple example can be an output of an addition of two numbers takes longer time than expected, although the result produced is correct. Thus system is said to have failed due to its computing performance.

From the two reasons stated above, we could conclude that in the process of computing system evaluation, it is necessary to guarantee functional and performance.

Pervasive computing *a.k.a* ubiquitous computing (UbiCom) is an emerging computing paradigm that is shifting desktop computing into computing integrated in the user's living environment. The core concept of *PerCom* is anytime and anywhere computing and communication. The vision of *PerCom* is to bring computing and communication capabilities within users living environment and assist them in doing their everyday tasks. This topic has a strong relationship with many computer science disciplines such as mobile computing, distributed systems, human computer interaction, operating systems, artificial intelligence, wireless networks, embedded hardware designs, sensors and sensor networks. Furthermore, the emergence of technology relies on the convergence of recent advancements in smart computing and communication devices like internet and wireless technology. These technologies are embedded in user's living environments, body or cloths and provide them with useful services. The services offered in these environments are intelligent (relaying on the environments context) and autonomous as they can run and perform tasks on their own without having direct user input and attention.

Problem Statement

Its being two decades since Weiser's vision of ubiquitous computing was coined, very few practical or even promising systems have been deployed and generated significant impact, and received acceptance. There exists an uncertainty in measuring the real achievements and advancements in this field, the reasons behind this phenomenon are:

- Firstly no standard measurement and evaluation techniques exist in this field.
- Secondly most of the systems exists in research labs or in dedicated environments, therefore the system evaluation and user acceptance is hard to measure and assess.

A review of the related literature shows that tremendous efforts have been put in the related research over the past two decades. However, unlike other disciplines, no standards or commonly accepted methodology has been established for a comprehensive evaluation of these systems to identify directions for future research. Consequently the unsolved research question of "***what and how to evaluate***" has been difficult to answer.

Goal

The goal of this research is to study and analyze system evaluation techniques to propose a comprehensive evaluation framework for *PerCom* systems. To achieve our goal, the research is centered on the question "*What and how to evaluate*". Analyzing and studying the *PerCom* systems reveals that design and evaluation of pervasive computing systems is a very complex task and requires complete knowledge of the following:

- The system and user behavior.
- The functional and non-functional requirements.
- The environmental and contextual issues.

Hence a one box solution is impractical although desirable.

Approach and Methodology

To address the question of "*What and how to evaluate*", we tried to answer it in a chronological order. To address the first part of the question i.e."*What to evaluate*", we have assembled a user-centered evaluation framework (see figure 3.1) that incorporates important factors from the system, user, context and operating environment. Within each factor we consider important evaluation areas and identify the key aspects in the evaluation process. Contextual and environmental factors present key issues and aspects in the evaluation process. User associated evaluation areas are studied from the ergonomic point of view, that would help evaluators to evaluate the user acceptance aspects of the system. System evaluation aspects are identified by classifying distinctive features and studying the key performance parameters of interest for *PerCom* systems. The diversity of *PerCom* prevents the use of a well-formed hierarchical classification scheme, thus the research has taken a different perspective and identified criteria that define major divisions in operational paradigm. Eleven criteria were chosen that would exhibit vastly different characteristics and helped us to generate the most compelling categories and key aspects associated with each category. Each aspect has key parameters that need to be identified and measured. The specified parameters that are identified are of particular interest to each category, but do not provide an exclusive list of every parameter to be evaluated. There are common parameters that are of interest for all systems such as throughput, response time, user acceptance, etc. This taxonomy can serve as a reference when deciding which parameters are most relevant to a particular *PerCom* system. The proposed conceptual framework model is a step towards forming standard evaluation guidelines that are defined for each factor, and can be used for qualitative and quantitative evaluation of *PerCom* systems during formative and summative evaluation processes.

The second part of the question i.e. "*How to evaluate*" is very difficult to answer. Considering the greater operating space, dynamicity, availability and user requirements of *PerCom* systems, it is not easy and practical to have a one box solution. Different methodologies and techniques have been presented in the literature and they are employed at different design stages of the system development. Our study of literature concludes that, for a comprehensive evaluation of *PerCom* systems, multiple techniques and approaches should be employed at each design stage.

In our effort, we looked at the early design stage evaluation, and studied and proposed evaluation approaches that can be employed at the early design stage. Our motivation for evaluating the *PerCom* system in the early phase of its design is based on the fact that: if the functional requirements of the system are correct then it could be said that the system meets the minimum safety requirements and is dependable. In addition, evaluating and verifying requirements at the early stage helps to identify errors in system requirements that might appear in the later design phases and can reduce the cost of simulation and testing. Early design stage evaluation in *PerCom* is very challenging. Existing approaches to evaluate the *PerCom* system at early design stage mostly centered towards the usability analysis, which employs various techniques like heuristic approaches, wizard of OZ, technology probes, system walkthroughs (c.f. Chapter 2 section 2.1.4 for details). These approaches are employed to get the user experience, factors of adaptability and gathering requirements for system design and development. The development of a *PerCom* systems application is a very difficult job, thus it requires sophisticated verification and validation techniques. To address these issues and requirements, we have studied existing formal verification and validation approaches and have applied them to quantify the functional correctness of *PerCom* systems. The promising benefits of formal methods drove the motivation to use and extend the existing validation and verification techniques for the purpose of early stage evaluation of *PerCom* systems. In this thesis, we studied two methods:

1. Firstly, we studied and applied an automatic verification technique such as

model checking to model and verify the functional requirements of the context aware medication management system. The benefit of this approach helped to validate the system in advance of its development, contrary to simulation and testing. The flaws identified with this approach helped to improve the design of system and increase the confidence level in its correctness and efficiency.

2. For the second method, we, studied runtime verification approach such as "Design by Contract" to verify the service interoperability requirements in smart environments. The benefit of the approach allows on-demand verification (a.k.a runtime verification) that combines formal specifications with an implementation to support runtime verification of software systems during its execution. This approach is contrary to off-line verification methods, such as theorem proving and model checking, that are hard to adopt in the common development methodologies and does not completely support runtime verification.

In the future, we would like to study more applications of runtime verification for *PerCom* system evaluation at different design stage of *PerCom* system evaluation.

Contribution of Thesis

The aim of this project is to evaluate pervasive computing systems. Our research started with the question of "*what and how to evaluate*". We tried to answer it in chronological order, to address the first part of the question what to evaluate:

✓ We assembled a user-centered framework that account for the system, user needs, context in which the technology will be used, and the operating environment in which the system will operate or deployed. A survey of the literature helped us to find "*what to evaluate*" by classifying *PerCom* systems and identify key parameters that can be measured. After the classification, we took a layered approach for evaluating system from the beginning of its design to find approaches for "how to evaluate".

A complementary issue to performance evaluation is functional correctness of a running system, which confirms that a system is conforming to its functional requirements and does not contain any flaws. The benefit of this approach allows designers to verify their system behavior against its functional requirements. This approach is complementary to testing and simulation. It enables to find flaws in the design before developing the system prototype. To address the second part of question that is "*how to evaluate*" (methodology of evaluation), we adopted a well-known formal verification and validation techniques at early design stage and applied two approaches.

- ✓ In the first method, well-known model checking approach has been used to model and verify the functional requirements of the context aware system at early design stage.
- ✓ In the second method, a *Design by Contract* technique has been used to model and verify the semantic and pragmatic service interoperability requirements in smart environments.

Organization of Thesis

The thesis is organized in the following four chapters followed by a conclusion. Each chapter follows the discussion presented in pervious chapters to help the readers understand the objective and research conducted during the thesis.

Chapter 1: Introduces the topic of the thesis, presents a discussion and general overview of the research background. Defines a fundamental vision of *PerCom* systems, fundamental properties and characteristics, and the key concepts related to *PerCom* systems. Later major challenges related to *PerCom* systems design and evaluation are discussed. In addition, the design and evaluation lifecycle for *PerCom* systems is presented. The goal is to better understand and position the evaluation approaches for *PerCom* systems presented in the literature.

Chapter 2: Presents the review of approaches and solutions for evaluating *PerCom* systems. The approaches presented in this chapter are specially used and proposed for *PerCom* systems evaluation. To analyze existing approaches, we have divided the approaches in seven categories. These approaches are as follows:

1. Evaluation frameworks for architectural designs and infrastructure evaluation.
2. Simulation, testing and verification techniques that cover metrics and their measurement techniques for various components of the systems
3. User centered evaluation approaches for field and lab assessments, qualitative and quantitative evaluation, formative and summative

evaluation, In-Suit evaluation of underlying systems.

4. Usability studies.
5. Contextual evaluation approaches for technology and context quality.
6. Environmental and ecological evaluation approaches for environmental implication of technology.
7. Technology Acceptance Models to evaluate the impact of technology on users and their acceptance factors.

Chapter 3: Presents the proposed user-centered evaluation framework for *PerCom* systems evaluation. A discussion on a methodology to answer ”*What and how to evaluate*” is divided into two sections. This chapter describes and discusses the “*What to evaluate*” question and presents the discussion on a proposed evaluation framework. Each factor is discussed in detail and key aspects and associated parameters are also presented.

Chapter 4: Presents the approaches studied to answer “*How to evaluate*”. It introduces early design evaluation approaches in *PerCom* systems and our motivation in evaluating pervasive systems at its early stage. Later, formal methods, their importance and role in early design evaluation are discussed, followed by a discussion on techniques such as model checking and DbC. In addition, the use of model checking for modeling and analyzing context aware applications and analysis of DbC for verifying interoperability requirements for smart environments are presented.

Conclusion: Finally, a conclusion is presented along with a discussion on the lessons learned and future work.

Chapter 1

Background

In this chapter, we will analyze the existing research, concepts and technologies used in ubiquitous or pervasive computing projects. We will, also, review the design and evaluation challenges faced by pervasive system designer. Pervasive computing has shifted the computing paradigm from traditional desktop computing to computing in the physical environment. Recent advancements in computing and communication technologies such as embedded and sensor technologies have been a major driving force in the research and development of pervasive computing. Pervasive computing defines the phenomenon of integrating technology and humans together. It offers a world where technology becomes available to humans virtually and invisible through hundreds of computing and communication devices around them, those fabricated with user clothing, devices embedded in human bodies to user's physical environment, smart utility appliances, smart homes to cars. This will allow system to carrying out information processing tasks and providing users with a set of services adapted to their needs and requirements across the network. The realization of pervasive computing is, when users will take computing and communication for granted and will only notice when it's absent.

1.1 Vision and Paradigm

The pervasive computing paradigm describes the computing environment, where computing and communication is imperceptibly present anytime and anywhere around the user physical environment. Mark Weiser,[1] a chief scientist at Xerox PARC, is considered to be the father of ubiquitous computing. Weiser and his colleagues introduced the vision of ubiquitous computing, also known as pervasive computing, as

> *"The most profound technologies are those that disappear, they weave themselves into the fabric of everyday life until they are indistinguishable from it"* [184]

The spirit of the technology is the development of environments saturated with computing and communication capabilities yet gracefully integrated with users.

> *"The idea is to make a computer so embedded, fitting, natural, that we use it without even thinking about it"*(Mark Weiser)

Pervasive computing has a strong relationship with closely-related fields, and the research is spans over many computer science disciplines like mobile computing, distributed systems, human computer interaction, operating systems, artificial intelligence, wireless networks, embedded hardware design and sensor networks, figure 1.1 shows the pyramid of pervasive computing system. Recent research and

1. http://en.wikipedia.org/wiki/Mark-Weiser

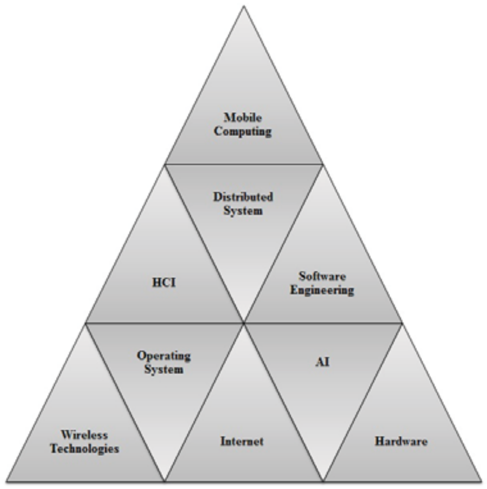

Figure 1.1: Pyramid of Pervasive Computing

developments in these technologies tend to support rapid development of pervasive computing. Although there is no formal definition for pervasive computing, based on its characteristics it can be defined as:

"The access to information and services anytime and anywhere".

Pervasive computing as visioned by Weiser in [184], some fundamental principles of pervasive or ubiquitous computing are as follows:

– The purpose of a computer is to help you do something else, which implies computer should do the job instead of you.

– The best computer is a quiet, invisible servant. This implies that all operations should be transparent and require little user involvement.

– The more you can do by intuition the smarter you are. The computer should extend your subconscious, which implies that the computer system should respond to user's subconscious according to the user and the environment context. This approach gave birth to the idea of context aware computing.

– Technology should create calm. This implies that technology will inform or serve the user and should not demand his/her focus or attention.

Based on the principle's outlined above and on ongoing research, some basic characteristics of pervasive computing can be derived. These characteristics are listed below:

- **Embedded**: In pervasive computing environments, devices should be implanted within the environments transparently. Transparency means users are just using the services and are not involved directly in the systems underlying operations (e.g. adapting services according to user preferences like changing temperature, lights, etc.).
- **Context awareness or Context sensitivity**: Operations and services are aware of the physical and logical context in which they occur and adapt to the environment accordingly.
- **Human-centric**: This is the heart and real implication of the pervasive computing vision. The system should bring computation and communication within the easy reach of humans through natural perceptual interfaces of speech and vision, so it blends into people's lives and enables them to collaborate, access knowledge, automate routine tasks and their environments.
- **Intelligent**: Pervasive environments are embedded with computing devices that have the ability to learn from user behaviors, needs and preferences that help to adapt user's physical environments according to their preferences. Ambient intelligence [12] techniques allow these devices to help people when performing their daily living activities reactively or proactively.
- **Autonomous**: The user doesn't necessarily need to ask for work to be done and

the system delivers services without user's direct interaction and intervention. Based on these concepts, many pervasive computing research projects are undertaken in interdisciplinary, multi-faceted application domains like schools [186], hospitals [14], universities, homes [74], cities [108], war fields [121], healthcare [131], etc. Some notable projects are e.g. Gaia [150], Oxygen [157], Cityware [98], CoolTown [25] and Gator Tech Smart House [74], these are just few to name here. All these systems are designed and developed based on the principles and basic characteristics of ubiquitous/pervasive computing.

1.2 Concepts in Pervasive Computing

Following are some concepts driven form the pervasive computing vision:

- **Calm Technology**

  Calm technology is another term coined by Weiser in the context of ubiquitous computing [181]. The idea of calm technology is to allow users to focus on information that is needed in the center of their attention and ignore information that is peripheral. Ubiquitous technology allows information to be present anytime and everywhere, making available huge amounts of data and information to the user. Calm technology makes the user more focused on the information needed while hiding the unnecessary information. This way the user is not be distracted or slowed down.

- **Internet of Things**

  Internet of things or internet of objects is another concept invented with the vision of pervasive computing [64]. The concept is attributed to the former Auto-ID Center, founded in 1999, based at the Massachusetts Institute of Technology (MIT). The idea is to inter-connect every object present in the environment so that each object can interconnect with other objects in the network to form a self-configuring network e.g. house hold appliances communicating and servicing in home network.

- **Context Awareness**

  Context awareness is another term coined after pervasive computing vision. Within information rich environments, the information in the environment is huge and sometimes irrelevant. Managing and processing large quantity of information to provide the users with the appropriate services is addressed by the research on context aware computing. Although the research in context aware computing is growing, there is an impoverished understanding of what context is and how it can be used. This results in various understandings and definitions of the concepts of context presented in the literature [4, 30, 50, 106, 148, 187]. For the purpose of general understanding we are quoting the definition by Dey and colleague [4].

  > *"Any information that can be used to characterize the situation of entities (i.e. whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves".*

  Based on various understanding and definitions of context, numerous applications have been developed and published in the literature. It ranges from very simple application of lights turning ON when user enters the room to more sophisticated applications where the application changes its behavior based on the sensory input from the living environment of the user.

- **Ambient Intelligence**

  Ambient intelligence is another concept in pervasive computing. The concept is linked with the advancement in electronic device and communication technologies. As progress has been made in embedding computing and communication around human living environments, it will be important these devices communicate and coordinated with an intelligent system embedded into them to form an ambient intelligence environment. A brief definition of ambient intelligence would be:

  "*A digital environment that proactively, but sensibly, supports people in their daily lives*" [12]

  From the definition above, ambient intelligence refers to a system that is embedded with digital intelligence to proactively deliver services to its user. The system has to be intelligent enough so that it can assist users when needed and doesn't interfere in other operations in the environment. Thus, design of ambient systems should be user friendly while providing absolute privacy and security of user data. Another concept that emerges with the ambient intelligence is known as smart environment. Smart environments comprise of sensor, actuator and sophisticated communication technologies. Smart environments are found in healthcare (where various sensors monitors the user's vital signs, location and offer services based on the context), offices (where smart meeting rooms are contextually aware of ongoing events and provide users with services according to their needs), etc.

1.3 Design and Evaluation Challenges in Pervasive Computing

From the concepts stated above, we have seen that, unlike desktop computing paradigm, in pervasive computing *PerCom* users are surrounded by many computing and communicating devices. Thus, in this complex situation, the design and evaluation of pervasive systems becomes a big challenge. As the technology is improving, there is a great need for a standard model to position *PerCom* and define directions for future research. A review of related literature shows that tremendous efforts have been put in the related research over the past two decades, however unlike other disciplines no standards or commonly accepted methodology has been established to evaluate these systems and identify directions for future research. There is a clear consensus among researchers that traditional performance approaches are no longer applicable for pervasive computing environments. Gabriele et al. suggested that due to high QoS requirements, proactive performance tuning activities and interdependencies between user behavior and system, conventional evaluation approaches are not directly applicable for evaluating *PerCom* [99]. In another work, Scott and Jennifer identified some major evaluation challenges in pervasive computing evaluation, such as applicability of metrics, scale, ambiguity and unobtrusiveness [170]. Similarly, Satyanarayanan drew attention towards the measuring and benchmarking the *PerCom* system and presented key challenges in quantifying pervasive systems. The challenges are as summarized as follow [161]:

– *Combining realism with reproducibility, which states that it is not easy to define a benchmark in a way that preserves realism yet gives the same results in* *identical experimental settings.*

- *Sheer scale complexity of pervasive computing systems, which states that it is difficult to interpret measurements and determine the contribution of different system components to those measurements. This makes it difficult to identify bottlenecks on which attention should be focused to improve the whole system.*
- *Multidisciplinary nature of pervasive computing, which state that the end-to-end quantitative analysis of a pervasive computing system would need to integrate the analysis of the user behavior, software behavior, hardware behavior, wireless network behavior etc.*

In the following, we present the major and common design and evaluation challenges for *PerCom* systems. The challenges presented are both technical and non-technical and cover issues that are central to research in pervasive computing and are addressed in this thesis.

- **Interaction**

  Pervasive computing offers completely different challenges in terms of user's interaction with the system. Unlike traditional computing systems, interaction in *PerCom* can be interacting with integrated computing and communication components in the environment with respect to its context. The objective of interaction in pervasive system is to enhance the user experience, so that the user can experience and utilize the available services in an effective and pleasurable way.

- **Integration**

  Integration is one of the major challenges in achieving the vision of *PerCom*. It refers to seamlessly integration of computing and communication components in everyday lives and living environments of the users. The system becomes more like a system of many autonomous systems within it. Thus this heterogeneous nature of various components poses many questions on the seamless integration of components in the environment and the smooth execution of services. This leads to new research challenges and issues in development of *PerCom*.

- **Scalability**

  Scalability is the ability of a computing system or network to meet the growing work demand. Scalability is another major challenge in the development of *PerCom*. Since in the pervasive computing paradigm, communicating and communication is embedded in the environment, thus system designers have to account for scalability issues in their design, e.g. having multiple users using the same resources, the addition of new components that can work with the existing system.

- **Heterogeneity**

  Heterogeneity refers to multiple different autonomous computing units working together in the environment. In *PerCom* environments, it is anticipated that users will carry multiple devices such as a laptop or a SmartPhones, to more sophisticated devices like body wear devices. Consequently connecting all these devices to interact with the environment for the smooth execution of the system poses great challenges such as: application and service adaptation to multiple devices, protocol design for heterogeneous networks, inter-device and network communication between multiple components (device & networks).

- **Interoperability**

  Interoperability refers to the ability of different systems to work together to achieve goals. Due to the presence of multiple devices in the *PerCom* environments, it will be difficult to meet the interoperability requirements of applications implemented on multiple platforms. With the wide range of development platforms, heterogeneity of devices, *PerCom* environment have to meet the requirements of different types of application and services. Requirements may range from interface knowledge, application behavior, syntax, semantics knowledge, pragmatic knowledge etc. Some requirements related to service interoperability in smart environments are presented in chapter 2 section 2.1.2.

- **Invisibility**

  Invisibility is among the basic characteristics of *PerCom*. The goal of fabricating computing and communication in environment present many challenges to system and infrastructure design. To achieve invisibility, systems must keep the focus of the user to the corresponding task, while keeping computing invisible. The application designers have to design application and services that require less user intervention while fulfilling user requirements. In addition, systems and components have to react to any change in user requirements dynamically, without any user intervention or halting system operations.

- **Modulability**

  The modulability of a *PerCom* system describes its capability to adjust and accommodate changes in the environment and user's preferences. The required flexibility of the applications as well as dynamicity of the environment, where the system is deployed greatly affects the most suitable modulability for each system. It is obvious that as more flexibility is incorporated in a system, the higher the programming efforts are required, and these different approaches have distinct characteristics such as in the case of programming and maintainability.

- **Reliability**

  Generally, system reliability is the ability of the system to perform the tasks without the user's intervention and continue to provide services in normal and exceptional conditions. *PerCom* systems tend to provide computing environments where users will perform their Activity of daily living (ADL), thus it is important for these environments to be reliable and fault tolerant. System reliability can be achieved when the system implements dynamic and transparent fault tolerance methods and requires minimal user intervention in handling system operations. Due to openness and diversity in technology, it is hard to build reliable systems. Unlike in traditional computing system, system reliability of *PerCom* environments is hard to achieve due to the following facts:

  – Many heterogeneous and autonomous components (hardware & software) with different implementations have to interact together, which pose a great challenge for designers to maintain the reliable execution among them and design fault tolerance approaches.

  – Pervasive computing systems are complex in nature. It is hard to cover the complete system in test cases and verifying or validating every state of the system. Therefore before hand, a diagnosis of system components requires comprehensive testing and debugging tools, exception handling, dynamic fault tolerant algorithms, transparency of the system and minimal user intervention.

  – Infrastructure and architecture design of *PerCom* systems incorporate many computing and communication devices. Reliability of this environment can be affected by many factors like device failure, communication link failure, and user mobility. Simulating and validating these factors requires a comprehensive knowledge of the entire system, which is not possible due to the presence of multiple vendor components.

The facts stated above and more pose great challenges in the design and evaluation of reliable and fault tolerant system.

- **Privacy and Security**

  Privacy and security issues are of major concern in any computing system, but at the same time, it has great importance in *PerCom* environments. The huge amount of data and wide set of services, with limitless mobility access that moves from one computing environment to another, require a comprehensive and efficient security and privacy mechanism to safeguard user data and maintain his/her privacy. As anticipated, multiple users will be using *PerCom* environments therefore it will be of great importance to define new access control mechanisms to ensure that services and user data are utilized by the legitimate user. For instance, medical data is very personal to the user, thus this data should only be accessible by the patient and his doctor or a person authorized by patient. These immense security and privacy requirements pose a big challenge for system designers to develop highly sophisticated models to ensure security and privacy of the user in *PerCom* environments.

- **Adaptation**

  In pervasive computing systems, adaption is the process to adapt the system's internal and external environment according to the profile and dynamics (i.e. environmental context) of its users. The internal environment refers to the internal adaptation of the application (i.e. context aware application) in response to the changing context of the environment, while the external environment refers to the transparent or prescribed environment adaptation around the application [194]. Adaptation in *PerCom* is necessary to overcome and address the dynamic nature of *PerCom* environments. Unlike traditional computing systems, user mobility and interaction with the environment, changing requirements for available resources, heterogeneous system components (hardware & software), input and output modalities, changes in the physical and virtual environments and more factors cause a great challenge for system adaptation. In addition, interoperability among applications and on-demand user requirements require advanced and complex adaptation methods to support the smooth operation of a *PerCom* environment [72]. System designers have to consider many factors to make their system adaptable to the environment requirements such as:

– Dynamic system adaptation according to the environmental resources (computing and communication) i.e. resource aware adaptation.

– Adapting application components with respect to environment change and user requirements.

– Adapting while interoperating or migrating to other components (hardware & software).

– Ensuring system's adaptation with respect to the functional and behavioral changes required by the user [194].

1.4 Summary

In this chapter, the background of *PerCom* and other related concepts were introduced. It describes the vision and fundamental principles behind *PerCom* along with a discussion of some key concepts in *PerCom* research. Also, this chapter presents a discussion on major design and evaluation challenges for *PerCom* system. The challenges presented are both technical and non-technical and covers issues that are central to research in *PerCom*. In the next chapter, we will review and discuss the approaches and solutions available for *PerCom* system evaluation.

Chapter 2

Related Work

In this chapter, we analyze existing research concepts, approaches, techniques proposed and used in pervasive computing evaluation. We also present the design and evaluation lifecycle which is established to review the related work presented in literature in area of ubiquitous and pervasive computing evaluation. In an effort of pervasive computing systems evaluation, various methods and solutions are proposed in literature. However, most of these solutions and approaches are tailored towards specific projects and technologies at different design stages. To better understand and position evaluation approaches for pervasive computing systems, as a first step we assemble a design and evaluation lifecycle for pervasive computing systems (shown in figure 2.1).

This lifecycle is an abstract representation, showing the steps involved in the system development and deployment. There might be multiple iterations on each step since the system must be implemented and tested before the next step begins. Second, we reviewed the evaluation approaches in practice and divided them in seven different

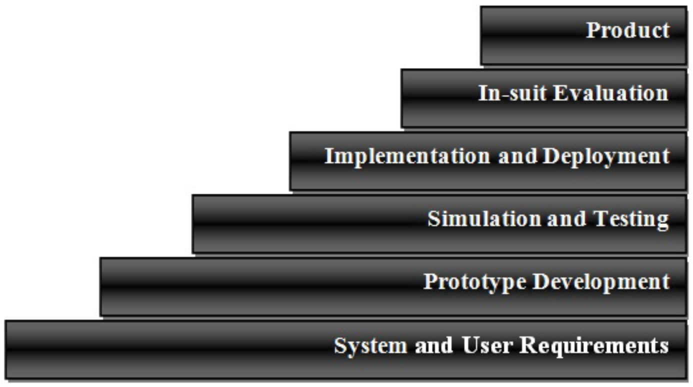

Figure 2.1: Design and Evaluation Lifecycle

categories:

1. Evaluation frameworks for architectural designs and infrastructure evaluation.
2. Simulation, testing and verification techniques that cover metrics and their measurement techniques for various components of the systems.
3. User centered evaluation approaches for field and lab assessments, qualitative and quantitative evaluation, formative and summative evaluation, In-Suit evaluation of underlying systems.
4. Usability studies for the products.
5. Contextual evaluation approaches for technology and context quality.
6. Environmental and ecological evaluation approaches for environmental implication of technology.
7. Technology Acceptance Models (TAM) to evaluate the impact of technology on users and their acceptance factors.

2.1 Exiting Solutions and Approaches

In this section, we discuss and review the approaches best suited under each category listed above.

2.1.1 Evaluation Frameworks

This section presents the review of representative work that addresses different evaluation frameworks, challenges and solutions for *PerCom* systems. Several existing methods and techniques for evaluating distributed and mobile systems can be extended to cover *PerCom* system [191]. Simone and Kazman proposed Software Architecture Analysis Method (SAAM), an analytical method for evaluating software architectures [56]. Kazman et al. developed Architecture Tradeoff Analysis Method (ATAM) a structured technique for understanding the inherent tradeoff in the architecture of software-intensive system [96]. Clements described active review for intermediate designs, methods for reviewing preliminary software designs for suitability in its intended use, context and environment [44]. To evaluate the autonomous and reconfiguration aspects of *PerCom* system, Braunes et al. presented a Reconfiguration-Enabled Compiler and Simulation Toolset (RECAST) an evaluation framework for coarse-grained reconfigurable architectures. The main components of the framework are:

– Profiler based on Stanford University Intermediate Format (SUIF) compiler.

– Synthesis and reconfigurable module.

– Code selector.

The framework combines hardwired and reconfigurable functional units in one template based on architecture description language, which turns a profiler driven retargetable simulator [24].

Scholtz and Consolvo presented a user evaluation areas framework of ubiquitous applications. The goal of this framework was to develop consensus among the research community on evaluating and positioning *PerCom* systems [162]. Researchers from National Institute of Standards and Technology (NIST) have presented an evaluation framework knows as System, Component, and Operationally-Relevant Evaluation (SCORE). The SCORE framework presents a unified set of criteria and software tools to formulate the performance evaluation approaches for complex intelligent systems. The SCORE framework is designed to evaluate technical and user oriented aspects of system components in control and realistic environment. The goal is to quantify the system performance in real user domain [189]. The SCORE has been extended to design the Multi-Relationship Evaluation Design (MRED) framework to cover discounted-factor in evaluating intelligent systems. The MRED framework is capable of producing detailed evaluation blueprints while receiving uncertain input information [190].

In another work, Thompson presented an evaluation framework for evaluating the performance of intelligent systems across large test spaces. The framework is based on spatial variance bounding methodology for designing tests experiments based on a meta-model method for characterizing a discontinuous, stochastic system [173]. The aforementioned evaluation methodologies can be used to study and evaluate the architectural aspects of *PerCom* system.

Web-based services can help users realize the *PerCom* vision. Enrique presented a framework for evaluating QoS of Web services. The framework is designed on the hypothesis of “ensuring QoS of the service is the capability to respond to the constraints, and meet the need according to the preferences of users”. The models tend to evaluate all QoS aspects of Web services as well as an aggregation of non-functional properties (NFP values). NFP depend on context and help evaluate service context [73]. In another work, Liu et al. proposed a framework for evaluating the reliability quality of Web services in service-oriented architectures (SOA). The framework is based on decomposition of reliable quality and focuses on composite Web services. Specific reliability evaluation characteristic of Web services are defined within the framework to find out what kind of reliability should be adapted for a given Web service. The two main aspects that were considered in the framework are availability and accessibility, while other aspects can be included in the framework [107]. Kalasapur et al. focused on SOA in *PerCom*. They presented a mathematical analysis model to derive a set of generic evaluation metrics for SOA, as well as metrics for individual services [97]. Lera et al. employed ontology with performance-related information to study the performance assessment of ambient intelligence systems that allow acquiring knowledge and reasoning about possible performance. This helped them to evaluate how architectural characterization affects the overall performance of distributed intelligent systems [109].

Similarly, Zhang et al. proposed a suit of user-oriented models and methods to evaluate the dynamic QoS requirements (context-awareness, mobility, jitter, reliability, interoperability and robustness, etc.) and adapted service selection. The hypothesis of the proposed model is based on QoS evaluation models that focus on quality criteria associated to user context. The main features of the model are: user-oriented QoS model with hierarchical structure to achieve scalability and flexibility, a context-model with a time dimension to include in QoS evaluation, user preference models based on linguistic variables to calculate the weights of quality criteria and QoS evaluation, and service selection models for first order logic inference and hierarchical fuzzy location evaluation [196]. Similarly, Laurent et al. proposed a generic framework for mobility modeling and evaluation of dynamic behavior of ubiquitous applications [105].

Metrics and benchmarks can help to sustain long-term research and stimulate competition by defining a framework for comparing the effectiveness of alternative approaches. Attempts have been made to identify the key evaluation and assessment parameters for *PerCom* system. Authors in [28] [29] [115] suggested different techniques to gather the data and selecting the metrics from scenarios that are driven by real problems rather than by technology. Romer and Mattern surveyed and studied the design space of wireless sensor networks and classified application designs according to mobility, heterogeneity, network modality, and topology. The results of this study helped designers understand the parameter that should be considered during the design process of such systems and can later be used for performance evaluation [151].

Oh et al. proposed an evaluation framework for QoS of their RFID middleware. They studied and analyzed quality characteristics (like functionality, reliability, usability, efficiency and portability) with reference to software in international standard ISO/IEC 9126, as well as quality elements of standard RFID middleware of EPC Global and extracted the potential aspects for quality of RFID middleware evaluation. They used the Analytic Hierarchy Process (AHP) as a selection method to evaluate the subjective characteristics of stakeholders in an objective way for evaluating quality of RFID middleware. The proposed method can be employed to select RFID middleware best suitable for ubiquitous environments [132].

Schlenff et al. presented a PRediction In Dynamic Environments (PRIDE) evaluation framework to perform moving object prediction for unmanned ground vehicles. The framework is designed on the concept of multi-resolutional hierarchical approach by unifying multiple prediction algorithms into a single algorithm. In addition a traffic control algorithm is presented to evaluate the performance of autonomous vehicle in realistic environment. The framework helps to predict location of moving objects as well as to evaluate vehicle performance in on-road driving scenario without evaluating in real environment with real user [159, 160]. Ranganathan et al. presented a performance evaluation benchmark based on their study of the GAIA system ; they divided the system framework roughly into three layers (system support, application programming support and end-user interface) and identified all of their performance metrics [6,146]. Similarly Kwon and Kim presented a layered methodology to assess the level of ubiquitous services; they proposed a three-layered model (i.e. capabilities of ubiquitous technology, level of ubiquity in technical perspective and level of service quality in behavioral prospective) and conducted the system assessment based on these layers with an expert in the domain. They suggested a seven Likert-type scale for certain technology services to quantify as UbiCom services [95]. This study was related to ubiquitous services and has a significant impact on the evaluation of ubiquitous systems. Grim et al. set some micro-benchmarks to quantify the scalability of migration in *PerCom* applications as well as data storage systems' performance for *PerCom* systems [63, 106]. Shirazi et al. proposed a set of performance metrics to consolidate the QoS (quality of service) model in *PerCom* [171].

2.1.2 Simulation, Testing and Verification Approaches

There are numerous approaches and tools available for simulating *PerCom* systems. We have reviewed some simulation and testing techniques with tools to support them. As mentioned earlier, *PerCom* covers distributed and mobile computing, sensor networks, human computer interaction and artificial intelligence under its umbrella. Various simulation techniques and tools are designed and developed for different disciplines of *PerCom* systems. For instance, to evaluate a wireless sensor network, there are simulation tools that can verify routing protocols, data dissemination, QoS parameters, network throughput and several other issues.

In an effort to simulate and reduce the cost of testing *PerCom* systems, Nerendra et al. suggested using well-known multi-agent-based simulation techniques. The idea is to represent every software and hardware entity as a software agent that can help designers develop and simulate the *PerCom* system. Each agent carries the workflow module responsible for interaction with the interface module of the simulation engine that triggers the workflow event [127]. Reynolds et al. argued that existing simulators are very limited and are not suitable for modeling the desired complexity of UbiCom system and presented some of their initial work on design of generic simulation tools for various ubiquitous system scenarios. In their work, a layered modular approach was used to support the simulation of UbiCom environments without constraining the simulator for covering one of several aspects of system scenarios.

In addition, an emulation framework for middleware and software under development was also provided that can interface with the simulation tool [149]. Min Do et al. drew attention towards the importance of evaluating a distributed sensors environment and the presence of ubiquitous robots in it. They argue that to verify and evaluate the performance of complete systems, the effect of environment sensor should be considered. They presented an integrated simulation framework, according to their approach each robot and sensor in the environment is implemented as components of Robotics Technology Middleware (RT Middleware). The benefit of this approach is to have a common interface module between simulation space and real environment [53].

To improve confidence in the ubiquitous software system, Merdes et al presented a built-in test paradigm by combining it with resource-awareness parameters, a technique that executes tests at runtime for the software under evaluation. This technique is well suited for ubiquitous middleware because of the environment's dynamic nature and inherent resource constraints [123]. Jinseok et al. presented a virtual reality system tool for developing ubiquitous environments. The system supports the prototyping and planning of sensors, display, processing objects and data collection objects. These objects are re-usable and adaptable, and their behaviors can be altered at different abstraction level. This virtual reality system can act as a software-testing platform and can be used by system architects to walk through the system under development and validate various scenarios for interaction usability [165].

Vijayraghavan and Barton from Mobile ad Media Systems HP Laboratories presented a Wireless Infrastructure Simulation Environment (WISE) simulator [180]. WISE is developed to explore the wireless infrastructure, devices, services for these devices and their interaction. The simulator presents an abstracted view of the environment to the end-user, which can be extended for building new scenarios. Similar to WISE, Vijayraghavan and Barton extended QuakeSim simulator [18] to design Ubisim [33]. Ubisim was designed to support the development of systems that integrate the physical and virtual worlds by simulating the physical world of people, the things they interact with, and the places they work, play and live in. The simulator works well with the WISE [180].

Later, Ubisim and WISE were combined to design the UbiWise simulator. The new simulator offers three main features that can help to simulate the ubiquitous system. The first feature is for users to interact with the simulated environment while running experiments or scenarios. The second is for researchers in order to pre-adjust the simulation environment to suit the scenario under consideration. The last feature is for developers to extend the simulator for application under review. The simulator emphasizes computation and communication devices, either integrated with 3D model of physical environments or carried by people [90]. Similarly, TATUS simulator was developed to support software that controls ubiquitous environments. The simulator explores the states of the software under test to develop its own representation of the world. This virtual representation shows the view of the software under test in the environment, and allows making decisions to change the world according to user behavior and mobility. TATUS can be used to verify whether that the environment behaves intelligently according to its set of goals or not [134]. In the following section, we present some discussions on generic simulation models for *PerCom* systems.

Simulators are not particularly designed for *PerCom* systems but can be used to design and simulate various aspects of UbiCom systems. The first simulator to be discussed is the discrete event system specification (DEVS) [10]. DEVS is used to model and analyze distributed discrete systems as modular and hierarchical system. Each module takes an input to transformed output, without knowing the underlying implementation, which makes it a black-box testing system. Ptolemy is another simulator with a graphical user interface [22]. it studies modeling, simulation, and design of concurrent, real-time, embedded systems and allows describing the model of computation that can further describe the interaction between concurrent system components. These components, called actors, are specified in Java technology and can be analyzed for many properties such as dataflow or real-time characteristics. Solver for circuit Equations with User defined Elements (SEQUEL) is another generic simulator widely used for a variety of *PerCom* applications to simulate electronic circuits, its pre-processor includes the library templates required to evaluate user-defined circuits and prepares the associated element subroutines [139]. Moreover there are numerous other formalizing and modeling approaches that come with a simulator that could be used to model and simulate different aspects of *PerCom* systems.

To the best of our knowledge, few attempts have been made to use formal methods as a tool or technique for evaluation, particularly in *PerCom* domain. In an effort to verify the interoperability requirements for *PerCom* systems, Pokraev et al. addressed and identified assessment requirements for semantic and pragmatic interoperability. Based on the requirements, they proposed a method to verify composite systems [117, 145]. The limitation of their approach is the lack of mapping mechanism for service models. Baldoni et al. recommended formalizing interaction protocols using finite state automata to define and verify properties for a service protocol [13]. Their approach is a static analysis that can improve confidence in the system; however, it requires prior knowledge of protocol descriptions under test. A similar approach is presented by Wan et al, they presented a verification algorithm and proposed formalizing the properties of services adapted to Pantagruel, a language that describes and manages services. This approach allows programs to be verified prior to their execution, however it is limited when used to verify composite systems on the fly [183]. Some authors like [2, 36, 37, 40, 46, 47, 49, 84, 114, 147] have drawn attention towards using formal methods for designing and validating *PerCom* systems. The importance of the topic has also been addressed in two recently initiated projects (UbiVal [152] and VERIWARE [102]) funded by European Research Council Advanced Investigators Grant. These projects, also, intend to employ formal method to validate functional and non functional requirements of pervasive system.

Networking is another important aspect of *PerCom* systems. There exists some well-known network simulators that are often used to verify and test the performance of network technologies/protocols for ubiquitous systems. For instance, Network simulator (NS-2) is a discrete event simulator for TCP, routing and multicast protocols over wired and wireless (local and satellite) networks. It can be used as an emulator to implement protocols on real and simulated networks [120]. Global Mobile Information System Simulation Library (GlomoSim) is another parallel discrete event simulator for network technologies best suited for ubiquitous systems [195]. It is built to support network systems using a layered approach (similar to OSI seven layers architecture), which allows the rapid integration of model and solutions developed at layers by different people. There are other existing simulators (e.g. OMNeT++ [178], NetSim [87], OPNET [42]) which are extensively used to simulate the network protocols and implementations for *PerCom* systems.

2.1.3 User Centered Approaches

Numerous user-centered evaluation approaches are devised to design and evaluate *PerCom* system. Petrelli suggested the potential use of user-centered approaches in design and evaluation of interactive information retrieval system. The results of this approach have the potential to first, help assessing the effectiveness of system components by testing the system prototype (micro-level), and secondly, help build a macro view of a system that would impact users and their tasks [140]. Scholtz and Consolvo presented a ubiquitous computing evaluation areas framework [162]. Iqbal et al. tailored existing user-centered design and evaluation approaches for ubiquitous system. In their proposed method, they addressed different dimensions (individual, social and organizational) that are relevant in the design and evaluation of ubiquitous services [82].

Hook suggested using user-centered design and evaluation approaches to achieve the objective of effective ubiquitous application. He presented a two-tiered user-centered evaluation method for evaluating the effectiveness of interfaces by discounting simplistic measurements while focusing on interpretative understandings of what's happening between user and system [76]. Neill et al. presented a 3D virtual reality simulation platform for user-centered design and evaluation of adaptive context-aware systems. The simulator presents a virtual environment that resembles a real environment and captures user context according to changing physical and social dynamics. The platform tends to help in rapid prototyping of adaptive services and in usability evaluation [138]. Similarly, 3D virtual platforms are presented to evaluate services for multi-users in *PerCom* environment [133].

Gross presented a user-centered design and evaluation process by adapting ISO 13407 according to the needs of UbiComp system. The process helps understanding and specifying the context of use, requirements and evaluating design against the requirements in a loop of an iterative cycle [66]. Leichtenstern et al. presented a user-centered prototyping tool called Pervasive Interface Development Toolkit for Mobile Phones (MoPeDT). The tool is an extension and combination of various features available in user-centered prototyping tools. The proposed tool tends to analyze user behavior and evaluate efficiency, effectiveness and satisfaction of interfaces [103].

Privacy is an important factor in *PerCom* system. Samsuri et al. proposed a user-centered evaluation of privacy model for protecting personal medical information. The evaluation is conducted by interviewing users. The interviews covered questions from areas like legislation, ethic, technology and culture. The user's feedback was used as an aid in design and evaluation of privacy models [169]. Similarly, Ali et al. presented a privacy model based on user control over private information; the model helps to evaluate the expressiveness of privacy policies and unobtrusiveness of privacy mechanisms [54].

User-experience is another important area of evaluation in *PerCom*. Arhippainen et al. presented analysis of user-experience evaluation for adaptive mobile application. They conducted interviews with small groups of people to get the information on user-experience while using a PDA application. In addition to the interviewing method, an observation method was also conducted to gather user experience that might have not been expressed during the verbal interview sessions [11]. Isomursu conducted user-experience evaluations with four different methods: (1) methods used before the pilot, (2) methods used during the pilot, (3) methods used immediately after the pilot and (4) follow-up studies to evaluate the user experience data and analyze the results [83]. Ikonen et al. used scenario evaluations to evaluate the user experience while using technology. This method tends to help designers develop functional models of real life and evaluate them before designing the complete system [81]. Vangelis et al. presented a set of attributes with respective metrics that are critical for user-adoption, and then recommended possible solutions for measuring those metrics [122].

Human behavior and contextual factors are also important in the design of *PerCom* environments. A concept of living laboratory was presented to evaluate *PerCom* environments [1] [80]. Results of living laboratories are helpful to identify pitfalls and improvement opportunities in the design of display (interactional aspect). However, the results are not satisfactory when deployed outside a laboratory environment, as participants of the living laboratory study (students or volunteers) are not often representative of the real users of applications. Hence, the interaction and intentional level of these individuals cannot be generalized. To address living laboratory evaluation limitations, domain experts turned to In-situ evaluation and recommended bringing systems to interact with real users in their normal life. In order to capture true user experiences, researchers applied evaluation techniques such as wizard of Oz, experience sampling, analyzing data from sensors, cultural probe, etc. [163]. In-suit evaluation techniques [43] seem promising for *PerCom* system, however these techniques have some limitations like:

– Studies are done on prototypes, therefore results may differ from those with the complete system.

– The time period, which is often 2-3 weeks, is not sufficient.

– The question of how to effectively analyze and utilize the data in evaluation is still not answered.

A summary of selected literature on the topic of In situ evaluation for *PerCom* system is presented by Neely el al. [130].

Qualitative and quantitative evaluation approaches have also been applied for the evaluation of *PerCom*. Researchers in the domain have suggested numerous evaluation approaches. Mark and Chris presented a comparison of quantitative and qualitative evaluation strategies and suggested a hybrid evaluation framework [29]. Jennifer and Scott stated difficulties in evaluation, particularly at the early design stage. They explored the questions that arise in ubiquitous system adoption and evaluation (like: what is useful, usable and what do users actually need). They recommended using unobtrusive methods to gather data by combining both quantitative and qualitative methods, and indicated the importance of diary studies or media-driven diaries to gather data for evaluation [170]. In another study, Consolvo et al. used LAQ sequential analysis (LSA) to assess and evaluate the enhanced biology laboratory [34]. This technique requires hours of manual video coding into different categories.

Beckwith conducted a qualitative evaluation study on user perceptions of house-based sensing technology. The participants of this study lived in a nursing home. The results of the study were useful to obtain the user's perception of these technologies and their adoption mood, however, system design and other technical aspects were ignored in this study [19]. Hutchinson el al. presented a new kind of evaluation technique called *Technology Probe* [75]. This technique has three goals that cover different disciplines of system design and adoption. The first is the social goal that leads to the understanding of needs and desires of users in real world settings. Second is the engineering or technological goal, which explores field-testing of the technology. Third is design goal for motivating users and practitioners to explore new methods and technology for the advancement in the domain. Efforts have also been made to explore formative and summative evaluation approaches [32] [35] [124] [86]. The reason that these approaches have limited impact is that their focus was entirely on gathering user requirements before designing a working system or vice-versa.

2.1.4 Usability Studies

Usability is defined as the ease of use and learning ability of a human-made object, product, device, etc. The ISO 9241-11 standard [61] defines usability as the effective, efficient and satisfactory use of a product, device, etc. in the context of its use. Unlike traditional computing systems, usability evaluation of *PerCom* is very complex due to multiple dimensions of system and user requirements. Bowman et al. conducted a study that highlighted usability issues related to interfaces in virtual environments. They considered three main characteristics: involvement of representative users, context of evaluation, and types of results produced. This study helped position usability evaluation methods in virtual environments [20]. A systematized usability evaluation framework for ubiquitous devices was presented by Kim et al. The framework offered a new usability evaluation method reflecting user, task and device by modifying the context decomposition, allowing evaluating each part of ubiquitous device. This framework has a Web-based implementation where devices can be evaluated and compared by calculating their scores [92]. Zhang et al. presented a generic framework for conducting usability tests for mobile application. The framework tends to serve as guidelines for conducting usability studies of mobile application [193].

In another work, Heo et al. presented an analytical usability evaluation framework based on a multi-level hierarchical model for mobile phone. The framework supports task and interface-based usability evaluation, and helps quantify several usability aspects using the proposed checklist [70]. A heuristic evaluation of HCI was carried out by Bastien and Scapin. They presented ergonomic criteria that can help to evaluate and improve user interfaces completeness and explicitness [31]. In another work, Nielsen presented ten usability heuristics as general principles of user interface design and evaluation of *PerCom* systems [128]. Similarly, Ji et al. presented a task-based usability checklist for evaluating mobile phone user interface based on heuristic during the development process [89].

Numerous usability evaluations of the *PerCom* system are conducted in laboratory and field environments. Rowley conducted a study to evaluate the impact of bringing the system in field environment for studying user behavior and reported the experience in [154]. Nielsen et al. studied the impact of usability studies for mobile systems in laboratory and field environments. They reported that usability studies in field environments are quite difficult compared to laboratory environments. Some of the major problems identified were the user interaction style and cognitive load, which are discounted in the laboratory environments [129]. Lirn duh et al. compared usability studies conducted for mobile phone used in laboratory and field environments. They studied the quantity and quality of severity, performance measures, user satisfaction and subjective feelings and behavioral patterns in both environments, while performing the same tasks [57]. Beck et al. presented six usability evaluation techniques for mobile systems in laboratory settings that involve various aspects of physical motion with either needs for navigation in physical space or division of attention [15].

Kjeldskov et al. evaluated the usability of context-aware mobile systems in the field. They employed our multi-method evaluation approaches (i.e. field-evaluation, laboratory evaluation, heuristic walkthrough and rapid reflection) for evaluating a mobile guide designed to support the use of public transport in Melbourne, Australia. They discussed the limitations and strengths of these approaches in evaluating mobile computing application [94, 101]. Analysis of these studies identified major difficulties in evaluating mobile systems in laboratory and field environments. Other usability studies were carried out and presented in the literature; however, they are not reported here, as most of them are using or improving techniques described previously, or applying them to different applications [5, 7].

2.1.5 Contextual Evaluation Approaches

Pervasive computing systems demand applications that can support highly dynamic self-adaptive environments and require less user intervention. Context information is the key to produce self-adaptive applications, thus it is important to validate context information in *PerCom* environments. To the best of our knowledge, a little work has been done on evaluating the quality of context. Park et al. presented a context-aware simulation system for smart houses. Their idea was to generate a valid context using virtual sensors instead of physical sensors. The simulator tends to help application developers detect rule violations and conflicts in context information [141]. Location is a part of context and an important factor for designing location-based applications.

Morla and Davies evaluated a location-based medical monitoring system by using existing network and context simulators. They presented a hybrid test and simulation environment that focus on component interaction, networking, location changes, and multiple component instances [118]. Neill et al. drew attention toward the importance of context, they suggest that context-aware applications need to operate reliably over a wide variety of situations to behave according to their context of use. They presented a technical architecture to support scalable, cost-effective, runtime experimentation of context-aware applications. The architecture allows informed decision-making in an iterative design cycle [136].

Grace et al. presented a context-based evaluation model and suggested that technologies should be evaluated within the context they will be used [111]. The model does not support *PerCom* evaluation as the contextual factor considered in their model (i.e. environment) is not the only factor to be considered in the case of *PerCom* systems.

2.1.6 Technology Acceptance Models

Pervasive computing systems are available on the market in various forms and users use such technologies in their Activity of Daily Living (ADL) e.g. (PDA, SmartPhones, etc.). Technology Acceptance Model (TAM) is an established theory of information systems that helps to evaluate user acceptance of technologies. In its modeling constructs, TAM accounts for a number of factors that might influence the use of technology. Designing TAM for *PerCom* is a challenging job. To the best of our knowledge, very few attempts have been made in TAM research for *PerCom* system.

Carsten reported the shortcomings of existing TAM models and suggested new factors that should be included in the design of TAM for future technologies [153]. Kay proposed a mathematical predictive model based on user perception of usefulness, ease of use, social influence, trustworthiness and integration to evaluate user acceptance of *PerCom* environments [45]. Dong-Hee presented a Ubiquitous Computing Acceptance Model (UCAM). The construction of models is based on the cognitive and affective attitudes of users. The model helps to identify the factors that influence user attitude and intention while using the ubiquitous system [168]. Shin extended the TAM to examine user acceptance of digital multimedia broadcasting (DMB) system in Korea. The constructs of this model are based on a motivational perspective that helps examine the socioeconomic determinants of DMB technology [167]. Hamner and Qazi extended TAM in their study to examine personal computing technology utilization in government agencies. The constructs of their model are based on external factors such as organization culture, individual and belief factors. This study helped gather diverse versions of adoption and non-adoption problems of employees [79].

Liang et al. in their study extended TAM to examine the use of PDA in healthcare domains. The study concludes that personal innovativeness, relevance and compatibility directly and indirectly affect the usage of PDA for healthcare professionals [112]. June Lu et al. extended TAM to study the use of Internet on handheld devices. The constructs of their model are based on individual differences, technology complexity, facilitating conditions, social influences, and wireless trust environment. The result of the study helps in analyzing factors affecting the use and acceptance of technology [113]. Chismar applied TAM2 [179], an extended version of TAM to evaluate Internet use in healthcare environments [48]. Numerous researchers have applied different constructs to examine *PerCom* and UbiCom systems [58, 156, 175, 182, 192]. However, there is no consensus among researchers over one common model for studying the user acceptance of *PerCom* systems.

2.1.7 Environmental and Ecological Evaluation Approaches

Pervasive computing systems are envisioned to be smartly fabricated into people's lives and environments. Unlike traditional computing systems, where it was considered irrelevant for design and evaluation, environment plays an important role in design and development of *PerCom* systems. In diverse disciplines, techniques and principles have been developed for environmentally friendly execution and deployment of technologies.

Pervasive computing systems are intended to bring comfort in our daily living. However, they have some positive and negative effects on environment. Jain and Wullert studied the environmental aspects of *PerCom* and identified negative impacts (e.g. physical waste and energy consumption, which would come under increasing government and public scrutiny) [91]. It is significantly important for *PerCom* researchers to consider environmental factors during the design of software and devices for *PerCom* environments. The importance of this matter has been articulated in literature [158] [78] [188] [16] [93] [17], however no standard methods or guideline have been developed to evaluate *PerCom* systems

2.1.8 Analysis

In this section, we present the analysis of the literature review. This analysis is presented in table 2.1, that shows the evaluation approaches presented to address the evaluation challenges and at which design stage the approach was applied or can be used. The study of the literature shows that most of the approaches are:

- Applied or developed during the implementation phase or after the prototype has been developed.
- Address specific design and evaluation challenges and ignore or discount other factors in the evaluation.
- Concentration on performance and QoS issues and not the functionality.
- Simulations and tests are performed on limited scenarios and in virtual environments.
- Most of the approaches are application dependent and cannot be applied to other application domains.
- There is no consensus among researchers about the parameters for evaluation.
- No benchmarks have been developed.
- Very limited functional verification has been performed.

The above mentioned shortcomings require a comprehensive approach for evaluating *PerCom* systems. Our analysis of the state of the art suggests that there is uncertainty among researchers about "*what and how to evaluate*". The "*What*" factor is associated to the fact that, which parameters and aspect of system are to be considered. The "*How*" factor is associated with "what" factor, and raises the question of how to evaluate, which approach to follow or adapt for the evaluation. In our analysis, the factor that is of most concern to users is the reliability of the system from the functional prospective. In literature, authors have drawn attention towards using many exiting techniques, but have not provided guidelines that can be applied to validate the functionality of the systems. Secondly there is no information about what parameters to be considered while evaluating reliability.

The main objective of this thesis is to analyze system evaluation techniques to propose comprehensive evaluation framework for *PerCom* systems. Consequently, we have tried to find the best possible solutions for the question "*what and how to evaluate*". In next chapters, we will discuss and present our approach to the research question i.e.*"what and how to evaluate"*.

Table 2.1: Evaluation Challenges And Approaches At Each Design Stage

CHALLENGES	APPROACHES	DESIGN STAGE
Interaction	[1, 11, 20, 24, 27, 31, 33, 70, 73, 80–83, 89, 95, 128, 132, 140, 162, 165, 180, 189, 190]	Prototype Development, In-Suit Evaluation, Implementation And Development
Integration	[24, 33, 53, 63, 106, 127, 132, 149, 180, 189, 190, 195]	Implementation And Development, Simulation And Testing
Scalability	[24, 56, 63, 73, 95, 96, 106, 132, 133, 136, 137, 151, 173, 189, 190, 196]	Implementation And Development, Simulation And Testing
Heterogeneity	[56, 109, 127, 151, 173, 189, 190, 196]	Prototype Development, Simulation And Testing
Interoperability	[13, 73, 142, 143, 183, 189, 190, 196]	Prototype Development, Implementation And Development
Invisibility	[1, 44, 66, 73, 80, 95, 111, 134, 136, 138, 180, 189, 196]	Implementation And Development, Simulation And Testing
Modulability	[24, 56, 97, 109, 149, 173, 189, 190]	Prototype Development, Simulation And Testing
Reliability	[2, 36, 37, 40, 47, 49, 73, 84, 107, 114, 123, 132, 147, 159, 160, 189, 196]	Prototype Development, Implementation And Development, System And User Requirements
Privacy and Security	[54, 104, 130, 158, 169, 189]	Prototype Development, Implementation and Development, Simulation and Testing
Adaptation	[11, 45, 48, 58, 79, 81, 83, 112, 113, 122, 153, 156, 167, 168, 175, 182, 192]	Prototype Development

2.2 Summary

This chapter presented and discussed the evaluation approaches proposed in the literature. These approaches are divided into seven categories according to their methodology and approach. This chapter also presented an analysis of these approaches and their use at different product design level. The analysis is summarized in table 2.1, that can serve as a reference while reviewing state of art evaluation approaches for *PerCom* System.

Chapter 3

What To Evaluate

In this chapter, we present our approach to answer the first part of the question i.e. "**What to Evaluate**". In this context, we present our proposed user-centered evaluation framework model and its building blocks. The evaluation method and approaches presented in literature are quite limited and only focusing on specific areas. Our study shows that pervasive systems are highly diverse in areas such as software architecture, enabling technologies and application domain, thus it is very difficult to establish a generic and comprehensive performance evaluation framework. In our model, we incorporated important system, user, contextual and environmental factors that are necessary for a comprehensive design and evaluation. Within each factor, we identify key parameters that can be characterized and measured for the purpose of evaluation. Our proposed model can serve as a design engineering model and a step towards forming standard evaluation guidelines that can be used during formative and summative evaluation.

3.1 User Centered Evaluation Framework

In this section, we extend our discussion on the proposed model. Based on our observations, we conclude that as a first step towards designing a *PerCom* evaluation framework, it is necessary to examine the common characteristics and differences of *PerCom* systems that separate them apart. We survey the literature to determine the ultimate interest of researchers when they make design decisions and evaluate their prototypes. Consequently, we studied different aspects of *PerCom* and classified them into different areas of evaluation to assemble the taxonomy of *PerCom* system [9].

A taxonomy of *PerCom* systems would allow us to characterize the systems and help to identify the most important performance parameters for evaluation. There is limited research on the classification of *PerCom* due to the heterogeneity of various technologies. Jeon and colleagues presented a taxonomy of ubiquitous applications and suggested three main criteria (i.e. subject, time and place) to classify ubiquitous applications [88]. Similarly Kista and Rajiv presented taxonomy of mobile and pervasive computing applications [55]. Dennis and colleagues presented taxonomy of ubiquitous computing environments [110]. Joanna and colleagues presented a taxonomy of pervasive healthcare systems 138, and Modahl and colleagues presented the taxonomy for a ubiquitous computing software stack called UbiqStack [116]. All these taxonomies and classifications mentioned here are limited to specific domains and do not cover the complete system.

After a careful study of various systems presented in the literature, we analyze the distinctive features of these systems and bring together the most suitable for classification. This taxonomy is by no means complete, but merely reflects the classification scheme that best suits the purpose of effective performance evaluations. Based on the analysis of the distinctive features of pervasive systems, we have chosen eleven criteria that would exhibit vastly different characteristics and can generate the most compelling categories. The diversity of *PerCom* prevents the use of a well-formed hieratical classification scheme. We take a different perspective and identify criteria that define major divisions in operational paradigm. First, we identified differentiating parameters that can be used to categorize pervasive systems. Once the criteria and their differentiating parameters are identified, we, then, define the categories and identify their key aspects and parameters. Each system to be evaluated is based on the eleven different criteria (defined later in subsections), and it will fall into one of the categories each time a different criterion is applied. The candidate parameters of interest for each system could be the union of the common parameters, the differentiating parameters and the key parameters associated with the category. Since there are multiple criteria employed in the taxonomy, any system can belong to multiple categories. Therefore, the set of categories, the system belongs to, can be used to define its character. For instance, a smart house would be considered as a centralized, assistive system that works within a single house. The taxonomy is designed to give researchers a reference when deciding which parameters are most relevant to a particular *PerCom* system. The key aspects and parameters associated with each category are not the only parameters of interests. The specified parameters are of particular interest to each category but do not provide an exclusive list of every performance parameter to be evaluated. There are common parameters that are of interest for all systems such as throughput, response time, and user acceptance. The taxonomy clarifies the scope, commonalities and range of diversity of *PerCom* systems. It also generates a reference and provides guidance when researchers and implementers wish to evaluate and benchmark different systems.

Knowing the complexity and the diversity of pervasive computing systems, different measurements and results collected may not truly reflect the *PerCom* system if comparisons are made among systems with vastly different design approaches and application domains. It is, therefore, crucial to identify not only quantifiable parameters that can be measured and evaluated, but also to use non-quantifiable parameters to characterize important aspects of these systems. In addition, we believe that it is equally important to take into account the contextual and environmental factors for a comprehensive evaluation of pervasive systems. Compared to traditional computing systems, pervasive systems rely on user intention, context and operating environment.

From our studies, we come to the conclusion that the there is a great need for an evaluation framework that considers the user, system, contextual and environmental factors for the comprehensive evaluation of a system. We assemble user-centered evaluation framework model for *PerCom* evaluation (shown in figure 3.1) identified the most important user and system aspects and established a list of quantitative and qualitative parameters for evaluation (summarized in Table 3.1). The system-centric parameters gauge the efficiency and effectiveness of the design decisions and present the assessment of how the system performs and expands under normal and abnormal conditions, as well as the resources employed to achieve the target performance. Similarly, user-centric parameters are used to indicate how well the behaviors of a system correspond to user intentions and expectations, and how much effort is required on the user's behalf to interact with the system. In the following sections,

Table 3.1: Summary of Quantitative and Qualitative Parameters of Systems and Users

	Quantifiable Parameters	**Non-quantifiable Characteristics**
System-Centric Parameters	System Performance Communication Performance and cost Software Footprints Power Profiles Data storage & manipulation Quality of context Programming efficiency Reliability Fault-tolerance Scalability Maintainability Effectivness	Node-level characteristics Service & application Context Characteristics Security & Privacy Knowledge representation Architectural characteristics Standardization Extensibility Backward compatibility Proactivity Adaptability characteristics
System and User Parameters	Adaptability Self-organization Error Explicitness	Economical consideration

we are going to present the key evaluation aspects from the system, user, contextual and environmental perspectives that lay the groundwork to develop a comprehensive evaluation framework and to construct a technology acceptance model [51].

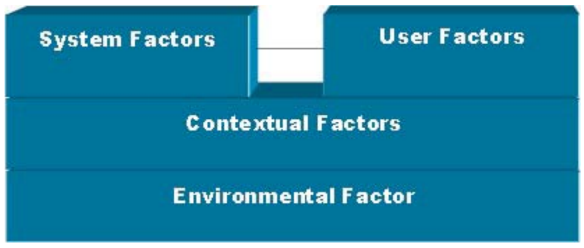

Figure 3.1: Design and Evaluation Model for Pervasive Computing System

3.1.1 Environmental Factors

Pervasive computing systems are envisioned to be smartly fabricated into people's lives and environments. The importance of the matter has been articulated in the literature but, to the best of our knowledge, no standard guidelines or techniques have yet been developed to guide design and evaluation. The environmental factors can be derived from technical and non-technical domains.

From the technical domain, designers and evaluators should consider the implicit and explicit implication of deployed technologies, such as device safety and installation constraints present in the environment, environmental conditions (such as heat, cold and humidity) for the devices[1], energy consumption, physical waste and recycling of devices. In addition, user safety against non-ionizing radiations from mobile communication, environmental dynamics, potential health or environmental effects, stress imposed on the user, restriction of consumers and patients, freedom of choice, threats to ecological sustainability, and dissipation of responsibility in computer-controlled environments [104].

Non-technical factors can be accounted from diverse disciplines such as government regulating authorities, architecture, transportation, engineering, etc. [130], green technology concept, potential impact of applying technologies (Human Health Impacts, local natural environment impacts, social and cultural impacts, global impacts and resource sustainability) identified in Environmental Technology Assessment Manual (EnTA) [77]. We believe that the aforementioned environmental factors can, in one way or another, play a very important role in design and assessment of *PerCom* environments.

1. Pervasive device with built-in computing and communication abilities

3.1.2 Contextual Factors

Pervasive computing demands applications that can support highly dynamic environments and require less user intervention. To meet such challenges, applications should be designed considering user context. There is a rapid development in the design of context aware applications that can react autonomously on behalf of users [38, 71, 125]. Context is one of the most distinctive characteristics of *PerCom* system, thus when there is change in context due to some reason the performance of the system can be affected. The reason behind this affection could be the service provided by the system that could not adapt itself with the current context of the user. We argue that the pervasive computing applications and services should be designed and evaluated taking their contextual factors into account.

For the purpose of clarity, we have studied the most important constructs of context (location, time, identity, activity) [4]. Broadening the scope for contextual factor and considering the interdependency of environment and context, we include environmental attributes as building constructs of context, as change in environmental factor can also bring a change in context. The importance of the contextual factors for evaluation can be illustrated by the following scenario:

At 10:00 AM Mr. Clark leaves his home and visits the doctor at the hospital for a check-up (Considering the visit with his doctor is not scheduled). After his check-up, he unexpectedly sees an old friend, and invites her for a cup of coffee in a cafe to talk about their old days. Later Mr.Clark returns to his home.

In this scenario, the location changes from home to hospital, from hospital to cafe and from cafe to home. There are, also, changes in activities (meeting unexpected friend, inviting and drinking coffee), identities (Considering his friend is not in Mr. Clark contact list), environment (hospital, cafe) and time. The contextual factors in each environment could be different due to the change in location, identities, activities and time, resulting in a change in context. A context aware application that adapts the interface for its user, or provides specific services depending on the context of the user must account for the contextual factors to provide efficient and effective service execution.

Contextual factors can be determined by simulating the scenarios in the environments, to understand human behavior, social situations, the purpose of application and services and environmental factors. More factors can be considered depending on the requirements and the purpose of the application and the definition of the context. As such there are no scales to check the performance of contextual factors other than the quality of the context, a complete characterization of the context in a system, and the dimensions of the context that are of interest. The number of different dimensions of context can be used to correlate the complexity of the system and the utilization of context, for instance, whether the use of the context is proactively or reactively greatly affect the response time and resource usage of the system.

3.1.3 User Factors

Pervasive computing systems are designed to serve users and facilitate their daily activities. They have to satisfy the user's needs, conform to the environmental constraints where the user is located, and must be compatible with their physical and mental characteristics. Therefore, when evaluating pervasive computing systems, it is crucial to consider and evaluate factors not only from the system's perspective, but also from the user's perspective as well. In the following discussion, we will bring forth some user's related factors that are essential for *PerCom* design and evaluation. These factors are derived from our personal experience in [8,67] following the standard guidelines presented in [177]. These factors have qualitative and quantitative importance in evaluation and can be utilized to develop constructs of Technology Acceptance Model (TAM). The key factors and aspects that could be considered for evaluation are summarized in Table 3.2.

- **Personal Factors**

  Personal factors define the user's individual characteristics and surroundings. It considers the user's lifestyle and it focuses on sensory aspects that could affect or stimulate the user's senses such as noise, light, comfortable positions, etc. Pervasive systems have to be adjusted to the user's characteristics, because every person is unique, the "*standard user*" doesn't exist. To design a system that will be useful for our target population (for instance, assistive technology for people with special needs), certain aspects of personal factors need to be taken into account.

  1. Demographics:

     The demographic aspects represent the statistical socio-economic characteristics of a population, such as age, sex, education level, income level, occupation, marital status, country customs and anthropometric data. Populations are diverse and their habits and lifestyle differ from one country to another. It will be important to consider user's demographic factors in the design and evaluation of pervasive systems. Demographic factors can be useful in defining the constructs of a TAM, and regular feedback will help designers to tune the system for the target population.

2. Comfort:

   Comfort aspects are one of the most important aspects in a pervasive system. Indeed, a person will use and adopt the system only if he/she feels comfortable with it. This aspect represents the state of well-being the user feels when he/she uses the system. The term comfortable is not exclusively the user's postural comfort, but also sensory and visual comfort. Indeed, interacting with a pervasive system affects all senses. This is why a system that makes the user uncomfortable to achieve his/her goals will likely be rejected by user. The consideration of comfort aspects in the evaluation will help in identifying the right constructs of a user's acceptance of the technology.

3. Skills Aspects:

   Skills factors are also important in the design of pervasive computing systems. Most pervasive computing applications require the user to understand the basic computing technologies of today (i.e. PC, Cell phones). Skills factors actually help to understand the user's abilities to perform a task. To promote user adoption of a technology, a *"Learning-by-Doing"* approach can be valuable. If we know the potential of the user, we can improve his learning abilities by helping him/her to perform the tasks. It is important to look into the user's experience with the technology, as the target population may contain novice and experienced users of the current technologies (e.g. use of a PDA).

Compared to a novice, the experienced people may have another point of view due to their learning and past experience that can be relevant for the system. Additionally, it is interesting to consider the user's mental workload and capacity. The system will be much more appreciated if it is concise and doesn't involve extra cognitive load. To understand such details, comprehension/comprehensive user's performance and error free user aspects have to be considered in design and evaluation.

- **Organic Factors**

  Organic factors are related to the physiological functions of body and psychological aspects (motivation, perceived use, need for technology), personality traits (e.g. openness, confidence, etc.) which are not usually considered and evaluated in traditional computing system. However, in *PerCom*, it is significantly important to consider and evaluate these factors. Physiological aspects include the user's impairments such as the loss of an organ (impairment after accident), which is likely to be a problem for the user to interact with the system and can limit his activities. Furthermore, the brain is the most important and central cognitive organ and it plays a major role in a user's performance. Cognitive impairments impact a user's capacity, his/her intellectual functions and confidence [3]. It is essential to consider mental workload and capacity in the design of a pervasive system (e.g. Cognitive Orthotic [65]). Additionally, psychological factors enable us to understand the user's personality, frame of mind, and their feelings toward the system. For instance, people who have technophobia will likely not show interest in learning and adapting new technology.

- **Health Factors**

  Health aspects should be carefully considered in the design and evaluation process of pervasive systems. Health aspects relate to the user's health status (e.g. diseases or allergies). Health aspects have to be considered to address the difficulties a user can face while using the system. The user's health condition could restrict him/her in performing some activities. If a user doesn't have an impairment, but later presents difficulties in performing some activities while using the system, the system must notice this change in the user's performance, collect related data (for instance by consulting the user's vital signs and medical history) and propose adopted services.

- **Social and Community Factors**

  Social and community factors directly or indirectly affect or pilot the life we lead in the society. Although demographic aspects are personal factors, these can also be covered in social factors, given that social status and work are social factors. For instance, in the case of pervasive healthcare applications (e.g. assistive technology) for the elderly, demographics and social factors can play an important role. For example, in some countries (especially in oriental ones) people are closer to their family and the social and cultural bond is stronger; their acceptance of the assistive technology will likely be more difficult compared to people living in occidental countries, where social and cultural norms are not as strong. Moreover, religion is an aspect that should be taken into consideration, given that it could have an impact on social life and technology usage [185]. For example, Buddhists might not accept new technologies easily, as they are closer to nature and rely more on their family.

Table 3.2: Summary of User Aspects and Key Elements

User Factors	Aspects	Elements
Personal Factors	Demographic Aspects	Age
		Sex
		Anthropometry
		Social Status and Work
	Comfort Aspect	Postural Comfort and Pressure
		User Effort
		Satisfaction
		Economical Consideration
		Sensory Comfort
		Visual Comfort
	Skill Aspect	Experience
		Use without error
		Mental workload Conciseness
		Symbol interpretation and denomination
		Ease of Learning
		Capacity
		Language abilities
		Willingness
		Openness to experience
Organic Factors	Physiological and Psychological Aspects	Intellectual functions
		Activity limitation
		Impairments
		Confidence
User Factor	Health Aspects	Allergies
		Diseases
		Health state
Social	Social Aspects	Religion and spirituality
		Political life and citizenship
		Community social and civic life
		Social Support/ Social Networking
		Language
		Interpersonal aspects
		Soft skills

Similarly, if the system doesn't fit with the country's policy, it won't be used by the citizens. Furthermore, it is valuable to consider a user's soft skills and interpersonal aspects which can inform on the user's traits of personality and abilities in social interactions and relationships in everyday life.

3.1.4 System Factors

This section describes and discusses the important system evaluation factors. These factors are by no means complete but are best suited for the purpose of effective performance evaluations. We expect that as the *PerCom* research grows and more experiences are shared in its evaluation, more factors and evaluation aspects will be introduced. We found that it will be very effective to first categorize the systems based on some criteria and define key parameters to design a comprehensive evaluation framework. These system evaluation factors are determined from our experience in designing taxonomy of *PerCom* system [9]. This taxonomy is by no means complete, but merely reflects on the classification scheme best suited for the purpose of effective performance evaluations. Based on analysis of distinctive features of pervasive system, we have chosen eleven criteria that exhibit vastly different characteristics and can generate the most compelling categories. In the following section, we present the system evaluation factors and key evaluation parameters associated with each category.

- **Architecture**

  Architecture refers to the conceptual design and functional structure of all hardware and software components in pervasive systems. It provides the blueprint and operational manual during the development and deployment of a pervasive system. We have divided the architectural characteristics of *PerCom* in two major sub categories (infra-structure and design). In the following, we present the details and key evaluation parameters associated with each category.

Table 3.3: Infrastructure Evaluation Parameters

INFRASTRUCTURE				
CATEGORY	**TYPE**	**SUB-TYPE**	**KEY ASPECT**	**KEY PARAMETERS**
Network	Centralized		Resource usage	Software Footprints
				Data Storage Scheme
				Scalability
	Distributed	Stationary	Deployment	Maintainability
			Safety	Security & Privacy
		Grid	Resource usage	Process Management
			Safety	Data Management
				Data Storage Scheme
				Security & Privacy
		Mobile (Infrastructure)	Resource Usage	Software Footprint
			Invisibility	Power Profile
				Data Storage & Manipulation
				Reliability & Fault Tolerance
				Node-level characteristics & Privacy
		Mobile (AdHoc)	Speed & Efficiency	Communication Performance and Cost
			Resource Usage	Data Storage Scheme
			Safety	Software Footprints
			Deployment	Node-level Characteristics
			Compatibility	Security & Privacy
			Invisibility	Power Profile
Geographic Span	Personal-Range		Resource Usage	Software Footprints
			Usability	Power Profile
			Invisibility	Data Storage & Manipulation
				Acceptance
				Node-level Characteristics
	Local-Range		Safety	Reliability and Fault Tolerance
			Deployment	Security & Privacy
				Scalability
	Wide-Range		Resource Usage	Power profile
			Deployment	Data Storage Scheme
			Compatibility	Node-level Characteristics
				Adaptability, Maintainability and Self-organization

- **Infrastructure**

  One of the primary characteristics of *PerCom* is computing and communication anywhere anytime, which allows system to provide service to its users from personal to global scale. We categorize systems according to the distribution of data and control, mobility of users and devices, the infrastructural support of the network and the geographic span. We identify the following differentiating parameters: architectural characteristics at the system level, node-level characteristics, communication performance and cost, and economical considerations that allow us to distinguish one system from another based on their differences in the network infrastructure and geographic span. We present in Table 3.3 the categories, key aspects and parameters associated with this criterion.

- **Design**

  The vision of *PerCom* is to provide users with an access to computational environment anywhere and anytime [184]. Thus, the goal of pervasive computing systems is to design software architectures that support multiple applications and services in a pervasive environment. The diverse nature of *PerCom* has made it difficult for software designers to adapt one common model that can meet all the requirements of *PerCom*. The major challenges that make software design difficult are the ability of software architectures to support interoperability due to various network technologies and implementations, the needs of user and service mobility [52]. After a careful review of different software architectures, we have identified the key differentiating parameters (coordination, coupling, versatility and generation) that can help to classify different software architectures used for *PerCom*. We present in Table 3.4 the categories, key aspects and parameters associated with this criterion.

Table 3.4: Design Evaluation Parameters

DESIGN		
CATEGORY	**KEY ASPECTS**	**KEY PARAMETER**
Application Based Architecture	Modularity Software Dynamics	Coupling and Cohesion
		Dependency Between Application
		Interoperability
Component Oriented Architecture	Modularity Software Dynamics Management Design	Component Compilation
		Orchestration
		Coupling & Cohesion
Service Oriented Architecture	Modularity Compatibility Management Software Dynamics Design	Orchestration
		Runtime Service Generation
		Coupling & Cohesion
		Scalability
		Interoperability
Agent Oriented Architecture	Management Design	Choreography
		Embedded Intelligence
		Autonomy
		Interoperability

- **Autonomicity**

  Pervasive computing systems are distributed, heterogeneous, and dynamic. Unlike computers as traditionally defined, these systems have more diversified software and hardware components making manual management and development much more expensive. Automaticity is an aspect that describes how a *PerCom* system is initialized, how it evolves automatically to accommodate faults and failures, how it adjusts to user requirements, integrates new resources, and how it can identify and fend off potential attacks. The differentiating parameters of this criterion includes the report process of new or changed requirements, the number of people involved in making required changes, and the level of integration between business and programmed logic.

Table 3.5: Autonomicity Evaluation Parameters

AUTONOMICITY			
CATEGORY	**SUB-TYPE**	**KEY ASPECT**	**KEY PARAMETER**
Static		Speed & Efficiency Safety	Computational performance I/O performance Reliability & Fault-tolerance
Dynamic	Self-Learning	Sentience Usability	Quality of context Knowledge representation scheme Error Learning ability Explicitness
	Re-Programmable	Programmability Deployment Compatibility	Ease of programming Maintainability Service & application Extensibility Backward compatibility
	Re-Configurable	Usability Compatibility	Adaptability Ease of programming Self optimization

We present in Table 3.5the categories, key aspects and parameters associated with this criterion.

- **Integration**

  Pervasive computing systems, by its nature, require the integration of many different subsystems with very different characteristics. These subsystems include computational facilities, communication devices, mechanical or chemical sensors and actuators, smart appliances, and existing control systems. Plenty of research efforts have been spent on solving various integration issues, and different implementers have tried different approaches. Based on the approach taken, systems usually exhibit different architectures and therefore present vastly different characteristics. We believe that integration is among the important factors that need to be considered for a comprehensive evaluation. The important aspects like maintainability, standardization, reliability, fault-tolerance, architectural characteristics and scalability would be valuable for

Table 3.6: Integration Evaluation Parameters

Integration		
CATEGORY	Key Aspect	Key Parameter
AdHoc Integration	Method	Designated Black-box
Universal Interface	Method	Analyze data flow
		Analyze pattern
		Analyze content in pipeline
Plug-In	Method	Performance of utilities provided
		Pattern and efficiency
		Integration between application

evaluating the integration methods and to better analyze and check the performance of utilities provided in middleware, patterns and efficiency of integration between application components and middleware. We present in Table 3.6 some differentiating parameters that can be used to categorize the system based on their integration methods and identify the evaluation parameters.

- **Service Availability**

  The goal of pervasive computing system is to provide its user with a rich set of services that are embedded in their physical environment and integrated seamlessly with their everyday tasks. Unlike services that are provided by the Internet, *PerCom* services are invisible, intelligent and invoked automatically depending on the events happening in the environment or the user's context that satisfy their invocation. The quality of pervasive services can be evaluated in many aspects and the key differentiating parameters that can help to categorize services is their ubiquity, interoperability and composition with other services. We categorize the pervasive services based on the definition of *PerCom* (i.e. anywhere anytime). We present in Table 3.7 the category under this criterion, and identify the key aspects and parameters.

Table 3.7: Service Availability Evaluation Parameters

SERVICE AVAILABILITY		
CATEGORY	**KEY ASPECT**	**KEY PARAMETER**
Anywhere & AnyTime	Discovery	Discovery Protocol
	Location	Discovery Protocol
	Adaptation	Service composition
	Availability	Execution
	Mobility	Resource availability

- **Interaction**

  In pervasive computing systems, human-machine interaction and machine to machine interaction are the important components and are becoming highly dynamic and implanted in environment. A system should adapt the interaction and presentation using various interface components available for interfacing based on behavior sensing, service mobility and events happening in the environment. The main objective is to make the system usable and interactive for its user. The easier the system is, the more likely people will use and adopt it. Unlike traditional computing systems, interaction in pervasive computing is done implicitly and explicitly with the user. For this reason, there are so many human aspects that must be considered for the effective design and evaluation. The interaction in pervasive systems is not just interacting with the monitor. Its scope is much bigger when it comes to implicit interaction, where user's activities, gestures and behaviors are observed by implanted sensors in the environment. The systems that do not account for these aspects may lose their credibility and users may not adopt it. The potential aspects that need to be considered when evaluating systems are human to machine and machine to machine interactions, while keeping the contextual factors in mind. We present in Table 3.8 the categories under this criterion and key aspects and parameters.

Table 3.8: Interaction Evaluation Parameters

INTERACTION		
CATEGORY	**KEY ASPECT**	**KEY PARAMETER**
Human to Machine	Human Capabilities Preferences	Perceptual, Cognitive, Motor Interface designs Interaction Mode (Audio, Video, Tangible)
Machine to Machine	Interoperability	Communication Protocols Platforms Computational Capacities

- **Extensibility and Backward Compatibility**

  Extensibility is a major consideration for most computing paradigms, and certainly one of the fundamental factors when evaluating any *PerCom* system. It describes how well the design and the architecture of a system can accommodate new components in the future and the new technologies which come with them. Backward compatibility, on the other hand describes the capability of a system to integrate or collaborate with legacy systems or technologies. Many *PerCom* systems, such as smart houses, structure integrity monitoring or urban computing facilities, are expected to have much longer life-spans compared to traditional computing systems. Many systems are also inter-twined with physical plants that require deployment at the time of construction or risk incurring high costs during retrofitting deployment. With increasing development in the technologies and applications for *PerCom*, the extensibility and compatibility factors will play a vital role in their evaluation. For instance, the geographic span of different pervasive computing systems can vary up to several orders of magnitudes. This will result in a significant impact on the operation behavior and organization of the system. The designer and assessor must take these considerations into account for the successful deployment and operation of their system. The extensibility and backward compatibility can be evaluated by examining characteristics such as:

- The support for dynamic upgrade in firmware and applications.
- Whether there are mechanisms to improve the flexibility in the architecture of the system, such as the use of adapters.
- Whether the nodes support multiple interfaces and standards.
- Whether they are configurable and the extent that they can be adjusted.

- **Invisibility**

  Pervasive computing systems tend to improve the well-being and autonomy of the user by implanting computing devices into their environment and making it invisible. Invisibility can be achieved by using and considering the environmental and contextual factors. Impact of invisibility should be considered to quantify its implicit and explicit effects on its user and system.

- **Intelligence**

  Pervasive computing environments are embedded with computing based devices, able to learn from user behaviors, needs and preferences to adapt the environment accordingly. Ambient intelligence techniques allow these devices to reactively or proactively help people when performing their daily living activities. In our observations, we found that there are two kinds of environments. The first kind interacts and responds to the user's behavior and preferences according to changes in the user's context and behavior. The second environment is personalized according to the user's preferences set in his/her context (profile), and doesn't respond and adapt when the user's context or behaviors are changed. For example, when a person enters into a smart room

Table 3.9: Interaction Evaluation Parameters

INTELLIGENCE		
CATEGORY	**Key Aspect**	**Key Parameter**
ContextAwareness	Context	Quality of context Learning Reasoning

the system recognizes and personalized the environment according to his/her profile but if the person's behavior is changed the environment doesn't adapt to that change. The key aspect in this criterion is the context awareness with the key parameters being quality of context, learning ability and reasoning on context. We, present in Table 3.9 the categories under this criterion and key aspects and parameters.

- **Maintainability**

  The large number of simple and cheap components employed in most pervasive computing systems translates to short mean time to failure (MTTF) and make failure a norm in the operation. To ensure that the system remains operational over a period of time, considerable resources need to be allocated for continuous maintenance. To understand the true cost of real-world deployments, we need to measure MTTF, recovery time, cost of replacing parts or any other maintenance related parameters such as the man-hour required to replace or fix the failed entities, and the size of maintenance crew needed to keep the system operational.

- **Security and Privacy**

  Pervasive computing is supposed to be "calm" and invisible. In order to bring intelligence to an environment that can satisfy the user's desires and needs, the system has to be highly personalized and customizable. This, in turn, depends on the capability to acquire and dissimilate data about the users to various parts of the system. Typical *PerCom* systems hold a lot of fine-grained personal information, such as real- time locations, medical history, biometric data, personal preferences, schedule, and concerns over security and privacy have consequently reached new heights. Any serious discussion on practical deployment would first need to address security and privacy concerns. Examples of characteristics critical to the security and privacy in *PerCom* systems include the expressiveness of the security policy and the strength of security enforcing mechanism. Regarding privacy control, researchers suggest to define appropriate measures and policies on the protection of content, identity and locations.

- **Application Purpose**

  The services and functionalities that pervasive computing systems are designed to provide are extremely diverse. The requirements and emphasis on various performance parameters are heavily dependent on their primary purposes. For example assistive services allow users to enhance and expand their communication, learning, participation, well-being, quality of life and achieve great levels of independence [95]. We have defined some key differentiating parameters such as quality of context, reliability, fault tolerance, security, privacy and effectiveness that can be used to categorize the *PerCom* applications. After analyzing the applications presented in the literature, we identified the key differentiating parameters and categories according to their purposes. We present in Table 3.10 key aspects and parameters that define each category.

Table 3.10: Application Purpose Evaluation Parameters

APPLICATION PURPOSE		
CATEGORY	**KEY ASPECT**	**KEY PARAMETER**
Assurance	Safety	Reliability & Fault Tolerance
	Sentience	Reliability & Fault Tolerance
		Quality of Context
Assistive	Usability	Reliability & Fault Tolerance
	Safety	Node-level characteristics
	Invisibility	Security & privacy
		Modality and Effectiveness
Return on Investment	Speed	System Performance
	Efficiency	Communication Performance & Cost
		Economical Considerations
		Data Storage Scheme
		Learning Ability
		Efficiency
Experience Enhancement	Sentience	Context Characteristics
	Usability	Explicitness
		Learning Ability
		Satisfaction
Exploration	Sentience	Quality of Context
	Deployment	Maintainability

3.2 Summary

In this chapter, efforts have been made to find answer for what to evaluate in *PerCom* system. Pervasive computing system evaluation is different from traditional systems, thus it is not enough to rely only on traditional evaluation area and metrics such as throughput, communication, time, etc. In addition, *PerCom* system operations mostly rely on user intention, context and operating environment. Traditionally these factors were not important and were discounted in evaluation process. There is uncertainty in deciding what to evaluate and what are the areas of evaluation that are central and important to *PerCom* system. We presented, on this chapter a user-centered evaluation framework, which incorporates important system, user, contextual and environmental factors that are necessary for a comprehensive design and evaluation. Within each factor, we identified key parameters that can be characterized and measured for the purpose of evaluation. The proposed model can serve as a design engineering model and a step towards forming standard evaluation guidelines that can be used during formative and summative evaluation. In the next chapter, we will discuss and present our approach to our second question i.e.*"How to evaluate"*.

Chapter 4

How to Evaluate

This chapter presents our approach to address the second part of our research question that is "**How to Evaluate**". The evaluation of a pervasive system is important at all design and development stages. We followed a layered approach to address this question and focused on the early design stage. As mentioned in pervious chapters, evaluating pervasive computing is a very complex job, in particular at the early design stage. We proposed to employ a well-known formal method approach as a solution to early evaluation of *PerCom* systems. At the best of our knowledge, very few attempts have been made to use formal methods as a tool or technique for evaluation particularly in the pervasive computing domain (see chapter 2 section 2.1.2 for details).

For the proof of concept, we applied two methods. In the first method, we analyzed a model checking approach to model and verify the functional requirements of a context aware application deployed in a Smart House. This approach is complementary to testing and evaluation and allows designers to verify their system behavior against its functional requirements before developing the system prototype. Some basic properties like deadlock checking, matching of specification and implementation, and reachability analysis are verified.

In the second method, we looked at runtime verification of *PerCom* systems and tried to verify the service interoperability requirements in smart environments by using a Design by Contract (*DbC*) technique. The benefit of this approach is automatic verification of services interoperability in smart environments.

In the following sections, we present our discussion on two approaches and related concepts. We have, also, positioned our work with the existing evaluation approaches proposed for early evaluation of *PerCom* systems to compare our approach with the existing solutions.

4.1 Early Design Evaluation

Evaluating and verifying systems at the early stage help to identify errors in system requirements that might appear in the later design phases and reduce the cost of simulation and testing. It has been articulated in the literature the difficulties in early design evaluation of *PerCom* systems. There is very little work on early design evaluation of *PerCom* systems. The approaches presented in the literature are mostly centered on the usability analysis, which employs various techniques like heuristic approaches, wizard of OZ, technology probes, system walkthroughs, (c.f. see chapter 2 section 2.1.4 for details). These approaches are employed to get the user experience, factors of adaptability and requirements mining for system design and development.

A complementary issue to performance evaluation is functional correctness of a running system, which confirms that the system conforms to its functional requirements and does not contain any flaws. The system conforming to its functional requirements is said to be reliable, knowing it meets the minimum requirements of safety, which is the base for its certification [174]. System requirements are the first steps towards conceptualizing the system and thus play an important role in the system development. It has been known that system requirements (user needs) are the prerequisite of successful system development. Thus if the system requirements are incorrect or incomplete it will result in faulty system which is not useful and not acceptable by its user.

Our motivation of evaluating a *PerCom* system is in the early phase of its design is based on the fact that if the system design is correct i.e. the functional requirements of the system are satisfied, then it could be said that the system meets the minimum safety requirements and it is dependable. From the study of the literature and in our experience in building *PerCom* systems, we believe that it would be beneficial if the system meets its functional requirements and contain no functional flaws. Hence having met the functional requirements for the underlying system, we can improve the confidence in the system. It has been articulated in literature that there are difficulties in doing the requirements engineering for *PerCom* system. Authors have proposed different methods, such as video-based, contextual-knowledge-based, literature survey and living laboratory, for gathering user requirements [85, 100]. However, these approaches have focused solely on requirements engineering and have provide no information on the verification of the gathered requirements.

Formal methods are often employed for the rigorous analysis of a computing system. These approaches are based on mathematical foundations that provide frameworks to specify and verify the specifications and implementations of the system. Consequently reliability and efficiency of the system can be analyzed and compared. System specifications are formally specified and are checked for any inconsistencies and incompleteness. Next, these specifications along with some desired properties of the system are verified to quantify whether properties are the consequence of the specifications or not. This process is also known as formal verification. Using formal methods, the designer can formalize and model the system specifications, which then help to conceive the system components and their behavior at an abstract level. In addition, the designer can predict the system behavior without executing or building it, which can help to increase the confidence in the specification's correctness. This method is more comprehensive and reliable, contrary to testing and simulation, while allowing the exploration of entire state space and cover every possible system state. There are many formal and semi formal techniques available for system analysis. The most well-known techniques in practice are:

- Theorem proving using deductive methods (manual and automatic approaches).
- Model checking to exhaustively and automatically verify possible system behavior and verification properties.
- Runtime verification or dynamic verification of system components by checking specified properties while the system is running.

Formal method techniques can be applied at different design stages for the formal specification or verification of various aspects of a system (algorithm, protocols, etc). However, we advocate that it is most effective to use formal methods at the early design stage, i.e. during the requirements analysis, specifications, and high-level design of a *PerCom* system. At this stage, the designer can use analysis tools that can help to reason about the design description, verify desired properties and discover problems at early stage of development. Consequently, early design verification with support of analysis tools will play an important role in reducing the cost of testing and improving the reliability of the system while discovering undesired system behaviors. In addition, it will also allow to analyze the system specifications early in the development and to verify that the system satisfies key requirements such as safety and security before developing a complete system or prototype. For the purpose of this thesis and for a proof-of-concept, we studied and applied two methods.

In first method, we use a model checking approach to model and verify the functional requirements of a context aware medication management system for an elderly in a Smart House. The benefits of this approach will help to validate the system early in its development, contrary to simulation and testing which are generally employed after the development of a prototype. This approach will ultimately increase one's confidence in its correctness. In the second method, we verified service interoperability requirements in smart environments by using "Design by Contract" technique. The benefit of using this approach is that it allows an automatic verification of services interoperability requirements for smart environments. In following sections, we present a discussion and our analysis results of each technique.

4.2 Early Design Analysis of Contextaware System: Model Checking Approach

Model checking is a formal verification technique that helps to automatically verify a model of a system against its correctness properties. Contrary to simulation and testing, the model-checking approach verifies the system model based on the exploration of the complete state space. The system models are finite to guarantee reachability and decidability. System models are specified manually or automatically for various levels of abstraction as a state machine that helps to describe the complete system in a concise specification using some high level language such as Petri nets or process algebras. In addition, desired correctness properties are described in some temporal logic such as Computation Tree Logic (CTL) or Linear Temporal Logic (LTL) [26] or in terms of automata. The comprehensive search of the state space is performed to verify that the system model conforms to its specification and, also to verify that its behavior conforms to its specification. Consequently, if the properties do not hold, it is because the abstraction of the model is far from its specification and a counter example exists. The main benefit of a model checking approach is in finding flaws in a model and generating error traces when a property proves false. In the next section, we will study the potential use of a model checking approach in the early stage of analysis for a context aware system. First, we will present a system scenario that will help the reader to understand the system and user requirements, and later, we will explain system modules and their modeling definition.

4.2.1 System Scenario

Mrs. Smith is an old woman of 70 years who lives with her husband. In the last two years, she was assessed at the Alzheimer clinic where she was diagnosed with Alzheimer Disease (AD) in the early stage. Since she has memory problems, she forgets to take her medication. Currently, her husband is taking care of her medication at home, and helps with her prescription at the pharmacy (medication refill) and in preparing her medication when the couple goes out. All these difficulties have a serious impact on the quality of life of the couple. The dependency of Mrs. Smith for her medication on her husband is affecting their life and non-compliance to medication can lead to medication adherence problem. Resulting in hospitalization or other serious health problems. To help Mr. and Mrs. Smith, we propose an integrated technological support that will assist them in medication management and alert for medication on time.

Mrs. Smith lives in an Ambient assisted home where multiple sensors and actuators (screen and speakers) are installed to help her live independently. Mrs. Smith receives a message on her smart phone that she will need to get her monthly medication refills for her prescribed medication. The pharmacy of Mrs. Smith registers the medication by applying RFID tags to the pill bottles, which contains information about the medication, its dosing regimen, number of pills, and weight of each pill. This schedule is also transferred to Mrs. Smith agenda so that caregivers and family can see the updates in the prescription of Mrs. Smith. The pharmacist gives the refills to Mrs. Smith and updates her medication file (changes in the prescription's type, dose, time, etc) on her agenda (view provided to Pharmacist).

He, also, pushes medication information to his mobile device using NFC device. As Mrs. Smith returns home, the mobile agent on her device transfers the medication information to the medication reminder system on the home gateway. The medication reminder system is context aware and accounts for user activities in the home using various sensors. It reminds Mrs. Smith about the upcoming medication, sends back information to the pharmacy to prepare refills and reminds the user for refill pickups. The system reminds Mrs. Smith to take her medication with her if she is going out for lunch or attending her appointments. The medication reminder agent moves to her mobile device and reminds her on time to take medication when she is outside.

The systems share the characteristics of a safety critical system, thus present many challenges for its analysis. Any functional flaw in a system can result in life threatening problems for Mrs. Smith (for instance a wrong medication reminder or no alert on time). In the following, we will describe the medication adherence problem and will present context aware medication management system along with its design and analysis.

Medication adherence refers to whether patients follow their medication regime as prescribed and continues their medication as prescribed [68]. Non-adherence to a medication regime may result in serious health concerns and hospitalization. Although adherence to prescribed medication regimens is difficult for all patients, it is particularly challenging for the elderly. Elderly people who live independently find it hard to follow their medication regime, which causes a serious threat to their lives [126]. There can be a number of factors that can cause non-adherence to a medication regime for the elderly such as, cognitive decline, functional abilities, memory loss that results in forgetting their medication schedules, ordering and picking-up medication from pharmacy, etc. Clinical methods to assess medication adherence includes direct patient questioning, prescription refill records, medication blister pack, etc. These methods tend to address adherence problem but require lots of resources that prevent their adaptability, and are not fault-proof. Recent advancement in ICT has helped to design solutions like medication reminders [176], digital pill boxes [69] that address the issues and limitations discussed above. However, these solutions are limited to research studies, have limited functionality, lack in usability and user acceptance.

To address the medication adherence problem in elderly, we designed a context aware medication management and reminder system. The system is designed for a Smart House where various sensing and actuating components are installed to capture environment data, process it, assist and alert its habitants. The context aware medication system has four modules:

1. Sensing module.
2. Inferring module.
3. Reminding module.
4. Actuating module.

The system architecture is shown in figure 4.1. In the current phase of the project, we have considered medication reminders in the house. The system reminds the users about their medication on the predefined schedule provided by the pharmacy.

4.2.2 Designing and Modeling the System

The design and modeling of *PerCom* systems is very complex task and requires lots of attention in its requirements and specifications. Context aware medication

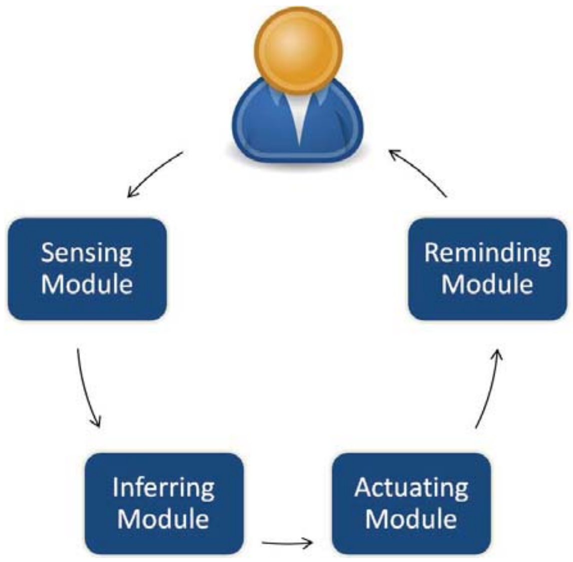

Figure 4.1: System Architecture of Contextaware Medication System

management systems share the characteristics of safety critical systems, where every detail of a system is important to model the system, so that the system performs its task as required and it is safe and dependable. A failure or any functional fault in these systems can result in serious danger and risk the life of the users.

The architecture of a *PerCom* system has a number of devices and applications collaborating, interacting and moving to support the system's operations. A *PerCom* system requires formal methods that support the concurrency among process running in the system and help to specify the mobility of application and the users. Although, there are many formal tools and frameworks available to specify systems, for the purpose of this project, we studied process algebra to specify a *PerCom* system. Process algebra or process calculus is mostly used to model concurrent systems [60]. It offers tools to describe high-level of interactions, communications, and synchronization between a collection of independent agents or processes. Some famous flavors of process calculi include CSP, CCS, ACP, LOTOS, π calculus, ambient calculus, PEPA and the fusion. Among these flavors, we focused on using Communicating Sequential Processes (CSP) [164].

Communicating sequential processes is a high level very expressive language, it uses mathematical objects as the abstraction to represent systems and its processes. CSP has a rich set of operators for modeling the behaviors of concurrent communications systems. System behaviors are described as processes that are well supported by algebraic laws. In addition, patterns of interaction among various processes, for instance devices and applications, can also be modeled as a set of communicating processes that interact with other devices and application which can be described as process expressions. CSP has a set of algebraic operators that allows describing complex relationships of components and behaviors of a system. In this project, we used communicating sequential programs (CSP#) as a modeling language to model the system components and its environment. CSP# integrates high level modeling operators with low-level procedural codes. CSP# implementation are supported by a process analysis toolkit (PAT) model checker (interested readers are refer to [172]). In the next section, we present the modeling of the environment and system modules.

1. **Entity Modeling:**

   As a first step towards modeling the system, we define the user as a process to model all the possible activities associated with the medication system. There are three possible states of the user in our system i.e. he/she could be inside the home, outside the home or taking medication.

$$\textit{User status :Inhome, OutsideHome, Medication}$$

User processes or user activities are associated or followed by many other processes or activities in the house. For instance, entering the home is followed by opening the home door or otherwise. Similarly, taking medication is associated with lifting the medication bottle. Since every activity is important in completing a task, therefore all possible activities are modeled as a process. In the following, we show the definition of a user process while inside or outside the home, followed by the definition of a medication process where the system detects each medication (we simulated two medication bottles) whether medication bottle is lifted or placed on the table.

Status = status of user i.e. inside home or outside home

$User(status) \mathrel{\widehat=} (enterHome(status); user(insideHome))$

$\square\ (leavesHome(status);\ user(outsideHome))$

$\square\ (medication(status);\ user(medication))$

$enterHome(status) \mathrel{\widehat=} [status == outsideHome]enterHome \rightarrow$ Skip

$leavesHome(status) \mathrel{\widehat=} [status == insideHome)leavesHome \rightarrow$ Skip

Medication Bottle Behavior (lifted, placed) i= identification of Medication bottle $status$ = status of medication bottle (lifted or placed).

$medication\ (i,status) \mathrel{\widehat=} (start\_med(i,status);\ medication(i.lifted))$

$\square(finish\_med(i,status);\ medication(i.placed))$

2. **Sensing Module:**

The smart house is composed of many sensors that monitor user activities and changes in their environment. In our project, the sensing module monitors the activities of the users when they enter and leaves home. This sensing module also monitors the information when the medication bottle is lifted and placed. This information will help to infer the user activities and assist them in doing their daily task. In our system, we have considered Radio Frequency Identification (RFID) for user identification and tracking. RFID reader is placed on the home door to detect when the user enters and leaves home. Similarly, a RFID reader is placed beside the medication cabinet, that reads the RFID tags attached to the medication containers and infers which medication is placed or lifted (we assume these tags are placed by the pharmacy or person responsible for medication management for the elderly). This information helps to remind the user of the correct regime of medication incase if they pick the wrong medication, and also help in reporting consumption of medication. We define *HomedoorRFIDReader* as a process to detect the presence of the user in the home or outside. This process takes its input from the home RFID reader that helps to infer the status of the user. Similarly, we define a process of *scaleRFIDReader(i)* which reads the RFID tags attached to the medication bottle. These two processes (i.e. *HomedoorRFIDReader* and *scaleRFIDReader(i)*) are combine into one process termed *Sensitization* using the interleave operator for keeping the process in a concurrent state.

$HomedoorRFIDReader \ \widehat{=}\ (enterHome \rightarrow rfid\_Homedoor\ !(outsideHome)) \rightarrow$ Skip

$\Box\ (leavesHome \rightarrow rfid\_Homedoor\ !(insideHome)) \rightarrow$ Skip

$scaleRFIDReader(i)\ \widehat{=}\ (scaleRFIDReader!\mathrm{i} \rightarrow \mathrm{Skip})$

$Sensitization\ \widehat{=}\ HomedoorRFIDReader\ |||\ scaleRFIDReader(\mathrm{i})$

3. **Inferring Module:**

The inferring module is an important component of the system. Data coming from the sensors are raw information and need to be translated to infer the user activities and assist them in doing their tasks. The inferring module translates the raw sensor data to verify the information with user activity or environmental change. For instance, sensor data coming from the door RFID reader will help to infer the user's presence in the home or otherwise. The inferring module runs an inference engine that has pre-defined rules and these rules are triggered when some conditions are satisfied or some actions are committed. All data coming from sensors are treated as sensor events. The inference engine then matches the rules with the sensor events, triggers the corresponding action based on the satisfied condition. In our system, two activities are monitored: first the presence of the user in the home or outside, second, the status of medication bottle when the user lifts and place back medication bottle. We define *Precensemonitoring* that infers from sensor information on the presence of the user inside or outside the home. Similarly, the *Medicationmonitoring* process is defined to infer from the information of *scaleRFIDReader(i)* which medication model is lifted or placed back. Process definitions are as follows:

(a) **First, the presence of user in home**

$Precensemonitoring \mathrel{\widehat{=}} (rfid\_Homedoor?(status) \rightarrow$

$[status == outsideHome]\, detectenterHome \rightarrow \text{Skip}$

$\Box\ [status == insideHome]\, detectleavesHome \rightarrow \text{Skip});$

$Precensemonitoring$

(b) **Second when user lifts and place medication bottle on the scale/box.**

$Medicationmonitoring \mathrel{\widehat{=}} (scaleRFIDReader?(i, status) \rightarrow$

$[status == lifted]detectstart\_med.i \rightarrow \text{Skip}$

$\Box\ [status == placed]detectfinish\_med.i \rightarrow \text{Skip});$

Medicationmonitoring

4. **Reminding Module:**

   The reminding module is a reminder system that alerts the user about their medication. The reminding module takes input from the inferring module to activate/deactivate the correct reminder. Reminders are predefined and they are prompted based on the input from the inferring module. For instance, when the inference engine evaluates that the user is leaving the house and has to take medication on a scheduled time, it triggers a reminding event. Based on this information, the reminding module translates this information and generates an alert to prompt the user to take medication along with him/her. This will help the user to not skip his/her medication while he/she is away from the house.

   (a) **Take Medication Reminder:**

   This reminder alerts users to take their medication on time. The reminder is very precise and triggers on the scheduled time of medication. This reminder has two parameters: (i = medication that has to be taken and ct = current time). In addition it also takes the sensor input to detect whether the user is in the home or not. In the following, we define the reminder that alerts the user to take the medication.

   $Take\_med\_reminder(i, ct) \mathrel{\widehat{=}} ([detectinhome]alert!(Take\_Medication.i)$ $\rightarrow \text{Skip}$

   WHERE $(med\_timetable.i = ct) \rightarrow \text{Skip});\ Take\_med\_reminder(i, ct)$

(b) **Take the Correct Medication:**

This reminder is very important to address the medication adherence problem. It is triggered after the system detects whether the user lifted the correct medication or not. The correct regime is pre-stored in the system. If the user follows the correct regime, it will prevent the serious consequences of taking the wrong medication regime. In the following, we, define the reminder that alerts the user to take the correct medication:

$worng\_med\_reminder() \mathrel{\widehat{=}} ([detectlifted] \rightarrow alert!(wrong\_medication.i)$ $\rightarrow \text{Skip});$

$worng\_med\_reminder()$

(c) **Bring Medication While Leaving Home:**

This reminder is triggered when the system detects that the user is leaving out from the home and his/her scheduled medication is due in the coming hour. This kind of reminder is helpful for the user to follow their medication schedule even if they are out of their home. In the following, we, define the reminder that alerts user to bring his/her medication while going out:

$bring\_med\_reminder() \mathrel{\widehat{=}} (dectectleavesHome?true \rightarrow alert!(Bring\_Medication))$ $\rightarrow \text{Skip};\ bring\_med\_reminder()$

5. **Actuating Module:**

The actuating module consists of output modalities in the smart house, e.g (audio, video, display). It takes input (reminder to be displayed or played) from the reminding module and notifies the user of various reminders. Here is the implementation of alerting service, which takes its input from the status and then triggers the corresponding reminder:

$Reminding \mathrel{\widehat{=}} alert?status \rightarrow Reminding;$

4.3 Verification of Dependability Properties

With the system model presented above, this section presents the verification of the system's desired properties. In our review of literature, we found that there is a diversity of opinions concerning the verification properties for pervasive computing systems and methods to define and verify them. Generally, researchers have considered and defined different properties for analyzing various aspects of a system such as dependability analysis for fault tolerant system, safety properties for hazard analysis, security properties for security and privacy oriented system or protocols, and customized properties that are tailored to a system's behavior i.e. mostly LTL properties. Since it is hard to generalize properties for pervasive computing, we will define and verify the most important properties for our system. These properties are synthesis of fault tolerance properties, safety properties, security properties and liveness properties. However, these properties are not the only properties for system analysis; many other properties of such kind can be included based on the need and analysis goal of project.

It is highly desirable to have automatic tool support for the modeling and verification of system. Model checking allows to automatically verifying various properties for critical systems using model checkers. There are many model checking tools that support modeling of concurrent systems and their behavior. A survey of such kind is presented by Frappier et al. in [59]. In our project, we used the process analysis toolkit (PAT) to model and verify the properties of the above described system. PAT is a self-contained framework for composing, simulating and reasoning concurrent systems [172], figure 4.2 (borrowed from [172]) shows the system architecture of PAT. In addition, due to its extensible and modularized framework. PAT allows users to build customized model checkers and extend the existing one to support new modeling languages and assertions [172]. We worked with the CSP module of PAT which implements CSP# [172].

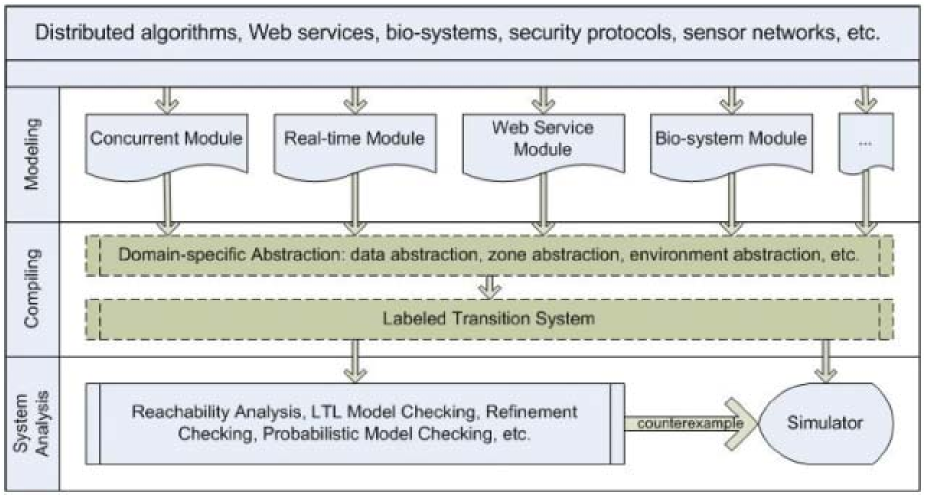
Distributed algorithms, Web services, bio-systems, security protocols, sensor networks, etc.
Modeling
Concurrent Module
Real-time Module
Web Service Module
Bio-system Module
...
Compiling
Domain-specific Abstraction: data abstraction, zone abstraction, environment abstraction, etc.
Labeled Transition System
System Analysis
Reachability Analysis, LTL Model Checking, Refinement Checking, Probabilistic Model Checking, etc.
counterexample
Simulator

Figure 4.2: System Architecture of PAT

- **Deadlock Freeness:** Deadlock is the most common and important property of critical systems. Deadlock can be when two or more processes competing for the same resource or wait for each other to finish. We verify this property as believe it is mandatory for pervasive systems to not have a deadlock in system's operations. For instance, it is not desirable that two reminders triggered at the same time or waiting for each other to happen, or some kind of sensing error that may corrupt inference module operations. These kinds of consequences may lead to a serious threat for the patient life. In the following, we define and verify the deadlock properties with the PAT model checker. Deadlock property is directly supported in PAT using a keyword "*deadlockfree*" e.g.

#assert Process() deadlockfree;

In our system, the result of this analysis was "VALID" which shows the system has no deadlock.

Property 1: ***System is deadlock free***.

```
system()=user()|||sensorization()|||Takemedication_Reminder();
Definition : #assert system() deadlockfree;
********Verification Result********
The Assertion (system() deadlockfree) is VALID.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Visited States:106
Total Transitions:1133
Time Used:0.0183896s
Estimated Memory Used:8973.712KB
```

- **False Reminder:** The medication reminder is the heart of our system, thus it is important to verify if the reminding service is working correctly. We check in our system if the reminder service alerts users according to the environment context, i.e. location, time, etc. The analysis is done by defining a scenario *inconsistentinfo*. In this scenario, the status of the user is outside while the medication bottle is lifted. This is an inconsistent state of the system about the environment and results in a functional bug. We define the scenario in PAT and run the assertion using "*reaches*" keyword. This kind of analysis is also known as reachabiity analysis of the system, where the system knowledge is inconsistent with the environment knowledge, which results in a faulty system. The result of the scenario in “NOT VALID” that means system reached the state defined in the scenario.

```
Property 2: Reachability Analysis (Inconsistent Knowledge of system)
Scenario While user is outside, medication bottle is lifted
#define inconsistentinfo (isOuthome= EMPTY && takemed =Lifted);
Definition #assert system reaches inconsistentinfo;
********Verification Result********
The Assertion (system() reaches inconsistentinfo) is NOT VALID.
********Verification Setting********
Admissible Behavior: All
Search Engine: First Witness Trace using Depth First Search
System Abstraction: False
********Verification Statistics********
Visited States:106
Total Transitions:1133
Time Used:0.0164181s
Estimated Memory Used:8940.744KB
```

4.4 Runtime Analysis of Pervasive System: Design By Contract Approach

We have seen earlier that model checking offers static analysis (off-line verification) while exploring functional flaws in system design. However, it can only be useful while analyzing finite systems, where the system operates in a closed environment and states are not changing with respect to their operating context. Though there are benefits of verifying systems with these techniques, however these techniques are difficult to fit in common system development approaches due to their formality and lack of runtime verification methods.

Pervasive computing systems (specifically context aware systems) are dynamic in nature, thus requiring runtime verification approaches. In such a case a simple static analysis of the system is not sufficient. Combining it with runtime analysis will aid the analysis process. Recently, the research in formal methods have moved from off-line verification to on-demand verification (*a.k.a* runtime verification) that combines formal specification with an implementation to support runtime verification of software systems during its execution. This approach is contrary to off-line verification methods (such as theorem proving, model checking, etc.) that are hard to adopt in the common development methodologies and do not completely support runtime verification of system.

Runtime formal verification allows verification of the system behavior for unexpected dynamic situations and helps to monitor the system performance dynamically. Furthermore, with a runtime verification approach, the system can be analyzed during the development phase and can be monitored while running. We believe that along with static analysis, runtime verification will be favorable for pervasive computing system analysis. In this thesis, we propose the use of Design by Contract as a possible "How" to verify pervasive computing systems on fly. In particular, we highlight the potential use of *DbC* in verifying service interoperability and adaptability requirements in pervasive computing environments, as it is very important that these environments support interoperability of services and components deployed in the environments to achieve common goals [144].

Design by contract is the object oriented design techniques introduces by Bertand Meyer [119]. The technique was introduced to increase the reliability and robustness in software system and make them bug free. It is based on the principle of client supplier relationship from the business world, where a contract is specified between the two on how to interact and deal in a transaction. On the same principle, *DbC* was introduced for software development, where two software components (client and supplier) have a contract specified on how the two will interact. This way it helps to ensure the required conditions and behavior correctness of the software components at design stage. The engineer writes assertions (contracts) for every class and components that can be verified dynamically at runtime, when the component or instance is invoked. Here the assertions/contracts are the boolean expressions that represent the semantic properties of that instance or component. Like business contract, design contracts also have some condition that must be fulfilled in order for smooth execution of system operations. The conditions are known as "*Pre*" and "*Post*" conditions, for instance former has to be met in order to call certain methods or components of the system while later have to fulfill certain condition. The pre and post conditions are inherited by Hoare's correctness triplets of:

$$\{P\}\ \mathrm{A}\ \{Q\}$$

This expression can be interpreted as an execution of process A is a state where P holds and, termination of process A is a state where Q holds. Here P and Q are the logical pre and post condition assertions for the process A respectively. These pre and post conditions are formally known as contracts that software designers can use to formally define precise and verifiable specification for software components. These contracts can be viewed as rights and obligations of the class and its client. The fundamental idea of *DbC* is to attach assertions or "*contracts*" with each component of the application, which allow the runtime verification of the properties that hold.

There are three different kinds of assertions that could be defined for any service, routine or application component:

1. Pre-condition: Pre-conditions are used to specify the assertion that must hold before the execution of the program, or component. In this way the programs or services can be verified before their executions. For instance, it is possible to verify the arguments or inputs required by the service.
2. Post-conditions: Post-conditions are assertions that must hold after the execution of the program. In this way services are verified after their execution against respective contracts.
3. Invariants: Invariants are the assertions in the contract that must hold anytime during the execution of a program.

Design by Contract technique is not exception handling or defensive programming, that allows a system to take and support all possible inputs and outputs. *DbC* allows to formally specifying the behavior of software using specified contracts that help to analyze and guarantee the functional requirements of real-time systems on the fly. We look now at the interoperability requirements in *PerCom* systems and potential use of *DbC* in verifying these requirements.

To support continuity of services in smart environments (SE), these systems must provide seamless interoperability among devices and services. The formal definition of interoperability given in IEEE glossaries is the ability of two or more networks, systems, devices, applications or components to exchange information between them and use the exchanged information. Thus interoperability is the capability of heterogeneous and autonomous systems to interact with each other for efficient service execution [143], It can, also, be defined as transparent &

compatible execution between equipments from different vendors. Researchers have presented many approaches to support interoperability in *PerCom* environments such as [21, 23, 117, 135, 145, 155]. However there are very few works done on verifying service interoperability requirements for these environments. There are three types of interoperability problems that can occur in *PerCom* environments:

1. **Syntactic Interoperability**: *Syntactic interoperability involves a common data format and common protocol to structure any data so that the processing of information will be interpretable from the structure [166].*
2. **Semantic Interoperability**: *Semantic interoperability requires that any two systems will derive the same inferences from the same information [166].*
3. **Pragmatic Interoperability**: *Pragmatic interoperability is reached when the interoperating systems are aware of the methods and procedures that each system is employing. In other words, the use of the data or the context of its application is understood by the participating systems [166].*

Service interoperability verification is a key to ensure service continuity and thus requires a thorough analysis of interoperability requirements. Pokraev and colleagues [142, 143] addressed the issue of interoperability verification and identified assessment requirements for semantic and pragmatic interoperability. These requirements are summarized below (taken as from [143]):

R1: A necessary condition for the semantic interoperability of two systems is the existence of a translation function that maps the entity types, properties and values of the subject domain model of the first system to the respective entity types, properties and values of the subject domain model of the second system.

R2: A necessary condition for pragmatic interoperability of a single interaction is that at least one result that satisfies the constraint of all contributing systems can be established.

R3: A necessary condition for pragmatic interoperability of a service is that *R2* is met for all of its interactions and they can occur in a causal order, allowed by all participating systems.

Based on the above requirements, they proposed a method to verify composite systems. The limitation of their approach is the lack of mapping mechanism for their service model. Baldoni and colleagues in [13] suggested formalizing the interaction protocols using finite state automata to define and verify properties of a service for the given protocol. Their approach is a static analysis of the system which can improve the confidence in the system. However it requires prior knowledge of protocols description under test. A similar approach is presented by Wan and colleagues, authors presented a verification algorithm and propose to formalize the properties of service adapted to Pantagruel (i.e. language that describes and manages services), which will allow programs to be verified prior to their execution. However this approach is limited when used to verify composite systems on the fly [183]. In next section, we present our preliminary analysis of using *DbC* to analyze the requirements presented above.

4.4.1 Requirements Analysis

Based on the requirements presented above, we studied how the requirements can be met if we choose to specify services using a *DbC* design. A service contract model is presented (shown in figure 4.4.1) which is a more like a design pattern to specify service. The components of this model are self-explanatory and can serve as a reference model while specifying services for smart environments. For instance, consider the simple Service Oriented Architecture (SOA) where services are published and advertised in the environments. Using *DbC* a service publisher can specify a service with its service contract and thus help service composition on the fly. It also helps to monitor if a contract of communications and executions is met (which verifies its correct functioning).

In this section, we analyze *DbC* technique to verify the semantic and pragmatic interoperability requirements. Contracts can be used to specify the application requirements that must hold to inter-operate with other systems. A service contract can specify semantics, behavioral, functional and nonfunctional requirements associated with the respective service. In this way, a service provider makes no assumptions about a service consumer, other than those specified in the service contract. Service contract can act as an interface through which service consumers communicate with the other services for service composition. These service contracts can be specified independently of the underlying technologies to support interoperability. The use of *DbC* technique will help to support and verify semantic interoperability requirements in pervasive computing environments. Next, we present the analysis of *DbC* technique in satisfying the semantic and pragmatic interoperability requirements identified in [143].

The first requirement R1 for the semantic interoperability is that the system

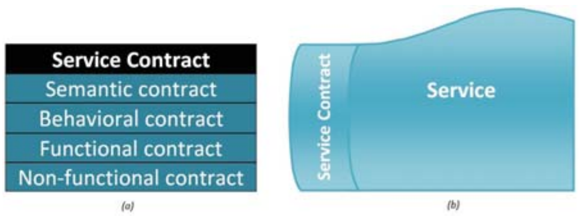

Figure 4.3: Service Contract Model

must have a translation function that maps the entity types, properties and subject domain model of the systems being integrated. This requires the complete knowledge of the system service and its relationships with other components in the environment. This requirement can be satisfied by associating a contract with each component of the system. Thus, when systems are interacting, they can be verified w.r.t to their contracts. This could be the pre or post condition that must hold for any communication. To enrich the contract with the domain knowledge and relations with other components, ontologies can be used. Ontologies are the description of some shared concepts and their relationships for the given domain and they support semantics interoperability based on the domain knowledge. Requirement R2 and R3 address the problem of pragmatic interoperability.

As discussed above, pragmatic interoperability can be reached when the inter-operating systems are aware of each other methods and procedures, also the necessary conditions are met for at least one result that can satisfy constraint. As a result, the context of its application is understood by the participating systems. This requirement can be satisfied by defining the invariants of the contract and can be verified any anytime during the execution, to know the current state of the system. *DbC* can, also, be applied to address the service composition and verification in *PerCom* environment. Authors in [41] have presented an approach to model a service behavior represented by concurrent regular expression and then translate it to label transition system to form a finite state process notation trace to verify the service composition. This approach is not automatic and does not support service interoperability if the composition is between two different systems. *DbC* approach can be applied to support such interoperability and verification of service composition.

The idea is similar to that stated above, every service has its contract which specifies its behavior, functional and semantics requirements.

4.4.1.1 Proof of Concept

For the proof of concept, we simulated a web service to analyze how a design by contract technique can be used to verify the interpretability requirements discussed above. We choose to verify web services as it offers interoperability of languages, platforms and technologies by providing XML-based definitions of services. Furthermore, we have seen efforts in enabling smart environments using web service technologies for example in [184]. It is also one of our motivations to work with web services. As discussed above, *DbC* is a design pattern that can be implemented using various available languages. However, we choose to work with the CodeContract framework. CodeContract supports three types of contracts: pre-conditions, post-conditions and invariants. These contracts offer an Application Programming Interface (API) that consist of methods defined in a class. The ***"requires method"*** helps to define pre-conditions of the method and ***"ensures method"*** helps to define post-conditions of the method. Unlike pre and post conditions, invariants are not directly applied to the methods but are used to verify the state of class instance and see if class is satisfying the contracts defined in it. The class state can be multiple fields, properties or class behavior.

To analyze how *DbC* helps in verifying interoperability requirements of a web service, we implemented a simulate of a web service that computes volume and surface area of a triangle implemented in Windows Communication Foundation (WCF). WCF is a framework for building service-oriented applications. It helps to inter-operate web services while supporting a set of specifications known as the web services specifications. Since basic interoperability can be achieved using WCF, we are interested in how to verify the interoperability requirements discussed above. To do so, a web service is defined using the contracts defined in WCF. WCF offers four types of contracts:

1. First, service contracts (with the attributes of service contract and operational contract) that help to describe which operations the client can perform on the service.
2. Second, data contracts that help to define which data types are passed to and from the service.
3. Third, message contract that helps to define services for interaction with messages. This type of contract is useful in cases where services have to comply with existing message formats.
4. Fourth, fault contract that helps to record the errors occurred during the communication and how the service has behaved during this session.

In WCF there is no contract types available to verify the functional requirements, though we can achieve it using invariants attributes of the *DbC* framework (as discussed above).

In our case, we were interested in analyzing syntactic, semantic and pragmatic interpretability requirements of the web service. By defining services using these contracts, we can verify interoperability requirements of web services. To analyze this, we drive tests using data and message contracts. For the purpose of functional testing a test contract to verify the input values is defined. The service computes the volume and total surface area of a triangle by calling the object of the message and data contract. It exposes two methods: ***MesageContractCalculation*** method that defines the message contract and ***DataContractCalculation*** method that defines the data contract type. The goal of both methods is the same i.e. computing the volume and surface area. However their service call method is different. In the ***MesageContractCalculation*** method, service operation should accept or return only a message contract type and it does not allow any other data types i.e. a service call is sent in a message format and is returned in the same way. The method takes input from the user and verifies its functional contract, if everything is correct it generates a new message contract as a response to the service call and returns the object to the call. If the input value violates the rule, a fault contract is flagged, errors are recorded and the user is prompted about it. In ***DataContractCalculation*** method which implements the data contract type, it takes data contract as input and returns the same. These contracts only describe the data to be transferred between the service requester and the provider, and do not describe any message format that carries the data. Generally for data contracts, interoperability is enabled through the XML Schema Definition (XSD). However, the message contract services can interoperate with any system that communicates through the Simple Object Access Protocol (SOAP). The simulation helps to analyze the pragmatic and syntactic interoperability requirements of web services using data and message contracts.

4.5 Summary

This chapter presented a discussion on "How to Evaluate" i.e. approaches and tools for evaluation. As explained in this chapter, finding one approach or tool for evaluating *PerCom* systems is very difficult. We followed the layered approach and presented possible solutions for early design stage evaluation of *PerCom* systems. Two methods are discussed in this context i.e. model checking and *DbC*. Both methods and their potential use in evaluating *PerCom* systems are discussed and analyzed using example of *PerCom* system. In the next chapter, we conclude our work and present a discussion on the lessons learned and future goals.

Conclusion

In this chapter, we conclude this thesis by summarizing the work, discuss how the contribution of the thesis addresses the research question stated at the beginning and present a discussion on the results. We then discuss and draw directions for future research and topics that go beyond the scope of the current work. Final remarks conclude this thesis.

Summary

The aim of this thesis was to analyze the tools and techniques for evaluating *PerCom* systems. The research started with the open question of "what and how to evaluate". To address the first part of the question, we have presented an evaluation framework for comprehensive evaluation of pervasive computing systems. We advocate that it is important that the design and evaluation of pervasive systems take into consideration the user, environment and context. In our model, we have incorporated and discussed important system, user, contextual and environmental factors that should not be overlooked during the evaluation process of pervasive systems. A survey of literature helped to classify the systems and to identify the key parameters that can be measured, which help to determine "*what to evaluate*".

Conclusion

To address the second part of the question, we, took a layered approach for evaluating a system from the beginning of its design to find approaches for "*how to evaluate*". A complementary issue to performance evaluation is functional correctness of a running system, which confirms that a system conforms to its functional requirements and does not contain any flaws. We studied formal method techniques to verify the functional correctness of the system at the early design stage. This approach is complementary to simulation and testing and can find flaws in the design before developing the system prototype. This approach allows designers to verify their system's behavior against its functional requirements. We studied two kinds of formal analysis methods. At first, we studied the potential use of a model checking approach as an automatic verification method for *PerCom* systems. To demonstrate our approach, we applied a (CSP) modeling framework to a context aware medication management system, and verified dependability properties using a PAT model checker. Secondly, we studied the runtime verification approach of *DbC* for runtime analysis of *PerCom* system. The analysis shows the potential use of using *DbC* in verifying interoperability requirements in smart environments. The interoperability requirements are defined as assertions that facilitate verifying the correctness properties and help to record the associated runs. Since the verification is run for a single piece of code/program or service, thus it can scale well. In addition, these approaches can be easily incorporated with the currently available technologies, however large scale analysis of these approaches are still to be done.

Findings and Discussion

Pervasive computing system evaluation research has recently gained tremendous attention. Researchers around the world are trying to come up with solutions and tools that can be used for comprehensive evaluation. However these approaches and tools are tailored towards specific applications or system modules. In addition, analysis approaches such as simulation, testing, and formal analysis all have limitations when applied to *PerCom* system. For instance existing testing techniques do not support the analysis of applications developed for dynamic environments (context-aware systems), mobility supports, inter-operability and heterogeneity, etc. Similarly, simulation approaches and tools that are widely used for *PerCom* system analysis face similar difficulties when dealing with special characteristics of *PerCom* system. In addition, formal analysis tools such as formal methods are powerful and broad. However they do not completely support the analysis of the unique nature of *PerCom* system.

In this thesis, we have tried to address the research issue and took a systematic approach by studying the available systems to understand and define analysis goals, that helped us to identify and gather relevant performance metrics and later explore appropriate evaluation techniques and tools for evaluation. We focused our attention on evaluating *PerCom* system at early design stage and studied formal methods as an approach for evaluation. Traditionally, formal methods have been used for evaluating complex and safety critical systems, however to the best of our knowledge, it has not been applied in *PerCom* systems. While *PerCom* systems are considered as safety critical systems, yet they are different from traditional critical systems.

Conclusion

Among many differences, the fundamental differences are dynamic environments, context awareness, and service continuation while the user is mobile. Hence these characteristics are not easily modeled with traditional formal frameworks and present many challenges for modeling and verifying *PerCom* systems.

First and foremost was finding the comprehensive modeling framework that would help to model a complete behavior of a system. The design requirements of *PerCom* system are different from a traditional system (such as context awareness, mobility, dynamicity, heterogeneity, etc) which makes existing formal frameworks inadequate for system specification and verification. In this thesis, we modeled and verified a context aware system using model checking approach. Although formal methods such as model checking approach was helpful to identify functional flaws at early stage of system design, however there were few limitation and difficulties faced in using formal methods for the analysis of *PerCom* System. In our experience, we choose to model our application using (CSP) modeling framework that allows describing the patterns of interaction among various processes. Locality and mobility are the important characteristics of pervasive computing system. This mobility could be of software agents to user in environment. (CSP) do not directly support mobility and locality specifications in a straightforward way, therefore modeling system with (CSP) alone is not an optimum solution. Other frameworks such as Ambient calculus [39] (Just to name here) can help to model the mobility requirements. The core concept of the Ambient calculus lies in the ambient that is a bounded place where computation happens. Each ambient can contain a number of applications that are executed in its context. These applications can move in and out, can host other applications and can be used to specify mobility and locality factors of *PerCom* system. However in the best of our knowledge there are no tools available for automatically analyzing specifications, therefore no concrete analysis can be done. In our experience, it was first difficult to find a comprehensive modeling framework that could help to specify all the aspects and properties of *PerCom* systems. For instance, due to CSP limitation presented above, we could not explore mobility scenarios for our system. We believe for comprehensive analysis of *PerCom* system, research community would need to develop specification frameworks and analysis tools that could address the most important properties of *PerCom* system. To date, we have not found any standard method or tool which can be used to model and verify all the evaluation aspects of *PerCom* system discussed in chapter 3. This leads to research issue of extending the utilities of existing modeling frameworks to meet the needs of future computing system modeling and analysis.

Although model checking allows exploring functional flaws in system design, however it can only be useful when analyzing finite systems where system operates in a close environment and states are not changing with respect to their operating environments. While dealing with infinite system, model checking approach faces the well-known problem commonly known as state explosion. This is a major concern in model checking as it is difficult to devise method and data structure to handle large state spaces of complex system. Consequently, this limits its use for open system such as *PerCom* system where multiple systems of systems are running together and system states are changing without the predefined rules. In our case, to deal with the problem, we divided the system in modules. However this approach will not give complete information of system behavior thus system analysis will be incomplete.

While verifying and analyzing the most important properties of *PerCom* system, we observe that contextual information properties of a system cannot be specified and verified easily. Thus requires additional reasoning techniques and tools. Since contextual information and related property verification is central to *PerCom* system analysis, therefore without analyzing such properties, we cannot analyze the adaptation behavior of *PerCom* system. The problem leads towards a solution for using various reasoning approaches in one framework.

We believe that it will not be enough to use model checking for automatic analysis of *PerCom* system. This thesis also advocates the need of runtime verification approaches for runtime analysis of *PerCom* system properties such as anytime, everywhere, context dependent adaption analysis etc. We discussed and presented one way of doing runtime verification using *DbC* for verifying interoperability requirements in smart environments. Although the benefits of runtime verifications are fruitful, however it is hard to boldly say that this approach can be solely used for system analysis. We believe that more research efforts and experiences should be shared for runtime verification to achieve any breakthrough in the analysis of *PerCom* system.

Concluding Remarks and Future Objectives

The above observations lead to the conclusion that *PerCom* system design and analysis is still in its infancy. We have found that research on *PerCom* system evaluation is fragmented in many directions, consequently there is a lack of common evaluation parameters, methodologies, tools, practices and theories that exist and can be used for evaluation. This leads to revisit these approaches and develop new practices for system analysis that can support the verification and analysis of main characteristics of *PerCom* system. We fear unless some standard methodologies and tools are not developed. *PerCom* system will not enjoy the success of traditional computing system and will remain research oriented system in laboratories. Thus developing new analysis tools are necessary for realizing the vision of Mark Weiser. This work has also helped us to conclude that no one box solution is practical for evaluating *PerCom* system. Thus the analysis of *PerCom* system should be conducted in a systematic way that completes its process throughout the development cycle and focuses on main characteristics of *PerCom* system. In addition, evaluation approaches should also account user behavior in analysis process, so that user behavior can be analyzed.

Many related topics are studied in the course of this thesis research; however they did not take-up due to limitation of time and scope of project. In future, we would like to investigate more system analysis approaches, to develop suite of analysis approaches tailor towards *PerCom* system. In particular, we have studied and applied formal methods for design and analysis of context aware system. The results are promising and help to drive research to next level. The next agenda in our research is to explore more system verification and validation approaches, to develop new tools and techniques for *PerCom* system design and analysis. In particular, we would be interested in studying reliable software engineering approaches for pervasive computing system and studying tools and frameworks to verify these requirements on the fly.

Bibliography

[1] G. D. Abowd, C. G. Atkeson, A. F. Bobick, I. A. Essa, B. MacIntyre, E. D. Mynatt, et T. E. Starner, "Living laboratories: the future computing environments group at the Georgia Institute of Technology," dans *CHI 00: CHI 00 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2000, pp. 215–216.

[2] M. Arapinis, M. Calder, L. Denis, M. Fisher, P. Gray, S. Konur, A. Miller, E. Ritter, M. Ryan, S. Schewe, C. Unsworth, et R. Yasmin, "Towards the Verification of Pervasive Systems," *Formal Methods for Interactive Systems*, vol. 22, pp. 1–15, 2009.

[3] B. Abdulrazak, B. Chikhaoui, C. Gouin-Vallerand, et B. Fraikin, "A Standard Ontology for Smart Spaces," *IJWGS*, vol. 6, no. 3, 2010.

[4] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, et P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," dans *HUC 99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK: Springer-Verlag, 1999, pp. 304–307.

[5] W. Abdul, A. Farooque, A. Hina, et S. Ali, "Usability Aspects in Pervasive Computing: Needs and Challenges," *International Journal of Computer Applications*, vol. 32, no. 10, pp. 18–24, 2011.

[6] R. Anand, A.-M. Jalal, B. Jacob, Z. Brian, C. R. H, et B. Brian, "Towards a Pervasive Computing Benchmark," dans *PERCOMW 05: Proceedings of the Third IEEE International Conference on Pervasive*

Computing and Communications Workshops. Washington, DC, USA: IEEE Computer Society, 2005, pp. 194–198.

[7] M. Alshamari et P. Mayhew, "Technical Review: Current Issues of Usability Testing," *IETE Technical Review*, vol. 26, no. 6, pp. 402–406, 2009.

[8] B. Abdulrazak, M. Mokhtari, et B. Grandjean, "Usability of an Assistive Robot Manipulator: Toward a Quantitative User Evaluation," vol. 306, no. 1, pp. 211–220, 2004.

[9] B. Abdulrazak, Y. Malik, et H.-I. Yang, "A Taxonomy Driven Approach towards Evaluating Pervasive Computing System," dans *ICOST 10: Proceedings of the Eight International Conference On Smart homes and health Telematics*. South Korea: Springer Berlin Heidelberg, 2010, pp. 32–42.

[10] T. Antoine Santoni, J. F. Santucci, E. De Gentili, et B. Costa, "Modelling & simulation oriented components of wireless sensor network using DEVS formalism," dans *Proceedings of the 2007 spring simulation multiconference - Volume 2*, série SpringSim 07. San Diego, CA, USA: Society for Computer Simulation International, 2007, pp. 299–306.

[11] L. Arhippainen et M. Tähti, "Empirical evaluation of user experience in two adaptive mobile application prototypes," dans *Proceedings of the 2nd international conference on mobile and ubiquitous multimedia*, 2003, pp. 27–34.

[12] J. C. Augusto, "Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and," *Artificial Intelligence*, pp. 213–234, 2007.

[13] M. Baldoni, C. Baroglio, A. Martelli, et V. Patti, "A priori conformance verification for guaranteeing interoperability in open environments," dans *In Proc. of ICSOC 2006, volume 4294 of LNCS*. Springer, 2006, pp. 339–351.

[14] J. E. Bardram et H. B. Christensen, “Pervasive Computing Support for Hospitals: An overview of the Activity-Based Computing Project,” *IEEE Pervasive Computing*, vol. 6, no. 1, pp. 44–51, janvier 2007.

[15] E. Beck, M. Christiansen, J. Kjeldskov, N. Kolbe, et J. Stage, “Experimental evaluation of techniques for usability testing of mobile systems in a laboratory setting,” dans *proceedings of Ozchi*, 2003, pp. 106–115.

[16] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, et M. Rohs, “Living in a World of Smart Everyday Objects Social, Economic, and Ethical Implications,” *Human and Ecological Risk Assessment: An International Journal*, vol. 10, no. 5, pp. 763–785, 2004.

[17] J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, et M. Rohs, “Social, economic, and ethical implications of ambient intelligence and ubiquitous computing. http://www.vs.inf.ethz.ch/publ/papers/socialambient.pdf, 2004. Institute for Pervasive Computing,” dans *In.* Springer-Verlag, 2004.

[18] M. Bylund et F. Espinoza, “Testing and demonstrating context-aware services with Quake III Arena,” *Commun. ACM*, vol. 45, pp. 46–48, January 2002.

[19] R. Beckwith, “Designing for Ubiquity: The Perception of Privacy,” *IEEE Pervasive Computing*, vol. 2, pp. 40–46, 2003.

[20] D. Bowman, J. Gabbard, et D. Hix, “A survey of usability evaluation in virtual environments: classification and comparison of methods,” *Presence: Teleoperators & Virtual Environments*, vol. 11, no. 4, pp. 404–424, 2002.

[21] A. Bottaro, A. Gerodolle, et P. Lalanda, “Pervasive Service Composition in the Home Network,” dans *Proceedings of the 21st International Conference on Advanced Networking and Applications*, série AINA ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 596–603.

[22] J. Buck, S. Ha, E. A. Lee, et D. G. Messerschmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogenous Systems,” *Int. Journal in Computer Simulation*, vol. 4, no. 2, 1994.

[23] Y.-D. Bromberg et V. Issarny, “INDISS: interoperable discovery system for networked services,” dans *Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, série Middleware ’05. New York, NY, USA: Springer-Verlag New York, Inc., 2005, pp. 164–183.

[24] Braunes, Jens, Kohler, Steffen, et R. Spallek, “RECAST: An Evaluation Framework for Coarse-Grain Reconfigurable Architectures,” dans *Organic and Pervasive Computing ARCS 2004*. Springer Berlin / Heidelberg, 2004, pp. 5–53.

[25] J. Barton et T. Kindberg, “The Cooltown User Experience. Tech Report: HPL-2001-22.” HP Labs., Rapport technique, 2001.

[26] C. Baier, J. Katoen *et al.*, *Principles of model checking*. MIT press, 2008, vol. 26202649.

[27] M. Beaudouin-Lafon et W. Mackay, “Prototyping tools and techniques,” 2003.

[28] J. J. Barton et J. Pierce, “Finding the right nails: Scenarios for evaluating pervasive systems,” dans *Pervasive 07: Common Models and Patterns for Pervasive Computing Workshop*. Canada: Springer-Verlag, 2007.

[29] M. Burnett et C. P. Rainsford, “A Hybrid Evaluation Approach for Ubiquitous Computing. Environments,” dans *In Workshop: Evaluation Methodologies for Ubiquitous Computing*, 2005.

[30] P. J. Brown, “The Stick-e Document: a Framework for Creating Context-aware Applications,” dans *Proceedings of EP96, Palo Alto*. also published in it EPodd, June 1996, pp. 259–272.

[31] J. C. Bastien et D. L. Scapin, “Ergonomic criteria for the evaluation of human-computer interfaces,” INRIA, Rapport technique RT-0156, juin 1993.

[32] V. Bellotti et I. Smith, “Informing the design of an information management system with iterative fieldwork,” dans *DIS 00: Proceedings of the 3rd conference on Designing interactive systems.* New York, NY, USA: ACM, 2000, pp. 227–237.

[33] J. J. Barton et V. Vijayraghavan, “Ubisim Mod for Quake III Arena,” Mobile and Media Systems HP laboratories Palo Alto, Rapport technique, 2001.

[34] S. Consolvo, L. Arnstein, et B. R. Franza, “User Study Techniques in the Design and Evaluation of a Ubicomp Environment,” dans *Proceedings of the 4th international conference on Ubiquitous Computing*, série UbiComp 02. London, UK, UK: Springer-Verlag, 2002, pp. 73–90.

[35] S. Consolvo, L. Arnstein, et B. R. Franza, “User Study Techniques in the Design and Evaluation of a Ubicomp Environment,” dans *UbiComp 02: Proceedings of the 4th international conference on Ubiquitous Computing.* London, UK: Springer-Verlag, 2002, pp. 73–90.

[36] A. Coronato et G. De Pietro, “Formal specification of dependable pervasive applications,” dans *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, 2009, pp. 358 –365.

[37] Z. Chen et S. Fickas, “Do No Harm: Model Checking eHome Applications,” dans *Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, série SEPCASE 07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 8–.

[38] H. Chen, T. Finin, et A. Joshi, “An ontology for context-aware pervasive computing environments,” *Knowl. Eng. Rev.*, vol. 18, no. 3, pp. 197–207, 2003.

[39] L. Cardelli et A. D. Gordon, “Mobile ambients,” dans *Foundations of Software Science and Computation Structures.* Springer, 1998, pp. 140–155.

[40] M. Calder, P. Gray, et C. Unsworth, “Tightly coupled verification of pervasive systems,” *Formal Methods for Interactive Systems*, vol. 22, pp. 1–16, 2009.

[41] J. Chen et L. Huang, “Formal verification of service composition in pervasive computing environments,” dans *Proceedings of the First Asia-Pacific Symposium on Internetware*, série Internetware ’09. New York, NY, USA: ACM, 2009, pp. 19:1–19:5.

[42] X. Chang, “Network Simulations with OPNET,” dans *Proceedings of the 31st Conference on Winter Simulation: Simulation—a Bridge to the Future - Volume 1*, série WSC ’99. New York, NY, USA: ACM, 1999, pp. 307–314.

[43] S. Consolvo, B. Harrison, I. Smith, M. Chen, K. Everitt, J. Froehlich, et J. Landay, “Conducting in situ evaluations for and with ubiquitous computing technologies,” *International Journal of Human-Computer Interaction*, vol. 22, no. 1-2, pp. 103–118, 2007.

[44] P. Clements, R. Kazman, et M. Klein, *Evaluating Software Architectures: Methods and Case Studies.* Addison-Wesley, 2001.

[45] K. Connelly, “On Developing a Technology Acceptance Model for Pervasive Computing,” dans *UBICOMP 07: Proceedings of Ubiquitous System Evaluation (USE)Workshop.* Austria: Springer, Innsbruck, 2007.

[46] A. Coronato et G. D. Pietro, “Formal Specification of a Safety Critical Pervasive Application for a Nuclear Medicine Department,” *Advanced Information Networking and Applications Workshops, International Conference on*, vol. 0, pp. 1043–1048, 2009.

[47] A. Coronato et G. D. Pietro, “Formal specification of wireless and pervasive healthcare applications,” *ACM Trans. Embed. Comput. Syst.*, vol. 10, pp. 12:1–12:18, August 2010.

[48] W. Chismar et S. Wiley-Patton, “Predicting Internet use: Applying the extended technology acceptance model to the healthcare environment,”

E-Health Systems Diffusion and USe: The Innovation, the USer and the USE IT Model. London: Idea Group Publishing, 2006.

[49] A. M. K. Cheng et R. Zheng, "Design and Verification of Pervasive Sensor-Actuator Systems," dans *Proceedings of the 2006 NSF Workshop On Cyber-Physical Systems*, 2006.

[50] A. Dey, G. Abowd, M. Pinkerton, et A. Wood, "CyberDesk: A Framework for Providing Self-Integrating Ubiquitous Software Services," dans *Knowledge-Based Systems*. ACM Press, 1997, pp. 47–54.

[51] F. Davis, "Perceived Usefulness, Perceived Ease Of Use, And User Acceptance Of Information Technology," *MISQ Central MIS Quarterly*, vol. 13, no. 3, 1989.

[52] C. A. da Costa, A. C. Yamin, et C. F. R. Geyer, "Toward a General Software Infrastructure for Ubiquitous Computing," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 64–73, 2008.

[53] H. M. Do, B. K. Kim, Y.-S. Kim, J. H. Lee, K. Ohara, T. Sugawara, T. Tomizawa, X. Liang, T. Tanikawa, et K. Ohba, "Development of simulation framework for ubiquitous robots using RT-middleware," dans *Control, Automation and Systems, 2007. ICCAS 07. International Conference on*, 2007, pp. 2483 –2486.

[54] A. Dehghantanha, R. Mahmod, D. I. Udzir, et Z. A. Zukarnain, "UPEM: User-centered Privacy Evaluation Model in Pervasive Computing Systems," *Ubiquitous Computing and Communication Journal*, vol. 4, no. 4, 2009.

[55] K. M. Dombroviak et R. Ramnath, "A taxonomy of mobile and pervasive applications," dans *Proceedings of the 2007 ACM symposium on Applied computing*, série SAC '07. New York, NY, USA: ACM, 2007, pp. 1609–1615.

[56] M. De Simone et R. Kazman, "Software architectural analysis: an experience report," dans *Proceedings of the 1995 conference of the*

Centre for Advanced Studies on Collaborative research, série CASCON 95. IBM Press, 1995, pp. 18–.

[57] H. Duh, G. Tan, et V. Chen, “Usability evaluation for mobile device: a comparison of laboratory and field tests,” dans *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM, 2006, pp. 181–186.

[58] O. El-Gayar et M. Moran, “College students acceptance of Tablet PCs: an application of the UTAUT model,” *Dakota State University*, vol. 820, 2006.

[59] M. Frappier, B. Fraikin, R. Chossart, R. Chane-Yack-Fa, et M. Ouenzar, “Comparison of model checking tools for information systems,” dans *Proceedings of the 12th international conference on Formal engineering methods and software engineering*, série ICFEM’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 581–596.

[60] W. Fokkink, “Introduction to Process Algebra. Texts in Theoretical Computer Science, An EATCS Series,” *Springer Ver-lag, Jan*, vol. 3, pp. 555–600, 2000.

[61] I. O. for Standardization, *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*, 1998.

[62] G. Gediga, K. christoph Hamborg, U. Osnabruck, F. P. U. Gesundheitswissenschaften, et I. Duntsch, “Evaluation of Software Systems,” *Encyclopedia of Computer Science and Technology*, vol. 45, pp. 166–192, 2001.

[63] R. Grimm, J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, et D. Wetherall, “System support for pervasive applications,” *ACM Trans. Comput. Syst.*, vol. 22, pp. 421–486, November 2004.

[64] N. Gershenfeld, R. Krikorian, et D. Cohen, “The Internet of Things,” *Scientific American*, vol. 291, no. 4, pp. 76–81, October 2004.

[65] S. Giroux, H. Pigot, B. Paccoud, D. Pache, E. Stip, et J. Sablier, “Enhancing a Mobile Cognitive Orthotic: A User-Centered Design Approach,” *International Journal of Assistive Robotics and Mechatronics*, vol. 9, no. 1, pp. 36–47, 2008.

[66] T. Gross, “Designing, Developing, Evaluating the Invisible- Usability Evaluation and Software Development in Ubiquitous Computing.”

[67] S. Helal et B. Abdulrazak, “Toward a Scalable Home-Care Delivery for Frail Elders and People with Special Needs,” dans *ICORR 07: Proceedings of the Tenth International Conference On on Rehabilitation Robotics*. Noordwijk aan Zee: IEEE, 2007, pp. 994 – 998.

[68] P. Ho, C. Bryson, et J. Rumsfeld, “Medication adherence,” *Circulation*, vol. 119, no. 23, pp. 3028–3035, 2009.

[69] T. Hayes, J. Hunt, A. Adami, et J. Kaye, “An Electronic Pillbox for Continuous Monitoring of Medication Adherence,” dans *Engineering in Medicine and Biology Society, 2006. EMBS ’06. 28th Annual International Conference of the IEEE*, 30 2006-sept. 3 2006, pp. 6400 –6403.

[70] J. Heo, D.-H. Ham, S. Park, C. Song, et W. C. Yoon, “A framework for evaluating the usability of mobile phones based on multi-level, hierarchical model of usability factors,” *Interact. Comput.*, vol. 21, no. 4, pp. 263–275, août 2009.

[71] K. Henricksen et J. Indulska, “Developing context-aware pervasive computing applications: Models and approach,” *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 37 – 64, 2006.

[72] K. Henricksen, J. Indulska, et A. Rakotonirainy, “Infrastructure for Pervasive Computing: Challenges,” dans *Workshop on Pervasive Computing INFORMATIK 01, Viena*, 2001, pp. 214–222.

[73] E. Hernandez, M. LAFOREST, et C. RODRIGUEZ, “Evaluation Framework for Quality of Service in Web Services: implementation in a pervasive environment,” Thèse de doctorat, Master thesis in INSA Lyon, 2010.

[74] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, et E. Jansen, “The Gator Tech Smart House: A Programmable Pervasive Space,” *Computer*, vol. 38, no. 3, pp. 50–60, mars 2005.

[75] H. Hutchinson, W. Mackay, B. Westerlund, B. B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, et B. Eiderbäck, “Technology probes: inspiring design for and with families,” dans *Proceedings of the SIGCHI conference on Human factors in computing systems*, série CHI 03. New York, NY, USA: ACM, 2003, pp. 17–24.

[76] K. Höoöok, “User-centred design and evaluation of affective interfaces,” *From brows to trust*, pp. 127–160, 2005.

[77] W. Hoskim, “Environmental technology assessment (EnTA) in cleaner production assessment,” dans *Report prepared for the IX. Balkan Mineral Processing Congress, Istanbul, Turkey*, 2001, pp. 11–13.

[78] L. M. Hilty, C. Som, et A. Kohler, “Assessing the Human, Social, and Environmental Risks of Pervasive Computing,” *Human and Ecological Risk Assessment: An International Journal*, vol. 10, no. 5, pp. 853–874, 2004.

[79] M. Hamner et R. ur Rehman Qazi, “Expanding the Technology Acceptance Model to examine Personal Computing Technology utilization in government agencies in developing countries,” *Government Information Quarterly*, vol. 26, no. 1, pp. 128 – 136, 2009.

[80] S. S. Intille, K. Larson, J. S. Beaudin, J. Nawyn, E. M. Tapia, et P. Kaushik, “A living laboratory for the design and evaluation of ubiquitous computing technologies,” dans *CHI 05: CHI 05 extended abstracts on Human factors in computing systems.* New York, NY, USA: ACM, 2005, pp. 1941–1944.

[81] V. Ikonen et K. Rentto, “Scenario Evaluation for Ubiquitous Computing-Stories Come True,” dans *Position paper for the Ubicomp 2002 conference workshop Evaluation Methods for Ubiquitous Computing. Goteborg, Sweden*, 2002.

[82] R. Iqbal, J. Sturm, O. Kulyk, J. Wang, et J. Terken, “User-centred design and evaluation of ubiquitous services,” dans *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information.* ACM, 2005, pp. 138–145.

[83] M. Isomursu, “Evaluating user experience in technology pilots,” dans *Human-Computer Interaction Symposium.* Springer, 2008, pp. 47–52.

[84] F. Ishikawa, B. Suleiman, K. Yamamoto, et S. Honiden, “Physical interaction in pervasive computing: formal modeling, analysis and verification,” dans *Proceedings of the 2009 international conference on Pervasive services*, série ICPS 09. New York, NY, USA: ACM, 2009, pp. 133–140.

[85] J. Jorgensen et C. Bossen, “Requirements engineering for a pervasive health care system,” dans *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International.* IEEE, 2003, pp. 55–64.

[86] X. Jiang, J. I. Hong, L. A. Takayama, et J. A. Landay, “Ubiquitous computing for firefighters: field studies and prototypes of large displays for incident command,” dans *CHI 04: Proceedings of the SIGCHI conference on Human factors in computing systems.* New York, NY, USA: ACM, 2004, pp. 679–686.

[87] J. R. Jump et S. Lakshmanamurthy, “NETSIM: A general-purpose interconnection network simulator,” dans *Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems.* Society for Computer Simulation International, 1993, pp. 121–125.

[88] N. J. Jeon, C. S. Leem, M. H. Kim, et H. G. Shin, “A taxonomy of ubiquitous computing applications,” *Wirel. Pers. Commun.*, vol. 43, no. 4, pp. 1229–1239, décembre 2007.

[89] Y. Ji, J. Park, C. Lee, et M. Yun, “A usability checklist for the usability evaluation of mobile phone user interface,” *International Journal of Human-Computer Interaction*, vol. 20, no. 3, pp. 207–231, 2006.

[90] J. J.Barton et V. Vijayraghavan, “(UBIWISE) A Simulator for Ubiquitous Computing Systems Design,” Mobile and Media Systems HP laboratories Palo Alto, Rapport technique, 2003.

[91] R. Jain et J. Wullert, II, “Challenges: environmental design for pervasive computing systems,” dans *MobiCom 02: Proceedings of the 8th annual international conference on Mobile computing and networking.* New York, NY, USA: ACM, 2002, pp. 263–270.

[92] H. J. Kim, J. K. Choi, et Y. Ji, “Usability Evaluation Framework for Ubiquitous Computing Device,” dans *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 01.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 164–170.

[93] A. Kohler et L. Erdmann, “Expected Environmental Impacts of Pervasive Computing,” *Human and Ecological Risk Assessment: An International Journal*, vol. 10, no. 5, pp. 831–852, 2004.

[94] J. Kjeldskov, C. Graham, S. Pedell, F. Vetere, S. Howard, R. Balbo, et J. Davies, “Evaluating the usability of a mobile guide: The influence of location, participants and resources,” *Behaviour and Information Technology*, vol. 24, pp. 51–65, 2005.

[95] O. Kwon et J. Kim, “A Multi-layered Assessment Model for Evaluating the Level of Ubiquitous Computing Services,” dans *Ubiquitous Intelligence and Computing*, série Lecture Notes in Computer Science, J. Ma, H. Jin, L. Yang, et J. Tsai, éditeurs. Springer Berlin / Heidelberg, 2006, vol. 4159, pp. 1059–1068.

[96] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, et J. Carriere, “The Architecture Tradeoff Analysis Method,” *Engineering of Complex Computer Systems, IEEE International Conference on*, vol. 0, p. 0068, 1998.

[97] S. Kalasapur, M. Kumar, et B. Shirazi, “Evaluating service oriented architectures (SOA) in pervasive computing,” dans *Pervasive*

Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on, 2006, pp. 10 pp. –285.

[98] V. Kostakos et E. O'Neill, "Cityware: Urban computing to bridge online and real-world social networks," *Handbook of research on urban informatics: The practice and promise of the real-time city*, pp. 195–204, 2008.

[99] G. Kotsis, "Performance management in ubiquitous computing environments," dans *ICCC 02: Proceedings of the 15th international conference on Computer communication.* Washington, DC, USA: International Council for Computer Communication, 2002, pp. 988–997.

[100] L. Kolos-Mazuryk, G. J. Poulisse, et P. A. T. van Eck, "Requirements Engineering for Pervasive Services," dans *Second Workshop on Building Software for Pervasive Computing. Position Papers., San Diego, California, USA*, pp. 18–22.

[101] J. Kjeldskov, M. B. Skov, B. S. Als, et R. T. Hoegh, "Is It Worth the Hassle Exploring the Added Value of Evaluating the Usability of Context Aware Mobile Systems in the Field," dans *Mobile Human Computer Interaction MobileHCI 2004*, série Lecture Notes in Computer Science, S. Brewster et M. Dunlop, éditeurs. Springer Berlin Heidelberg, 2004, vol. 3160, pp. 529–535.

[102] M. Kwiatkowska, "From Software Verification to 'Everyware' Verification," *Computer Science - Research and Development*, vol. 28, no. 4, pp. 295–310, 2013.

[103] K. Leichtenstern et E. André, "MoPeDT: features and evaluation of a user-centred prototyping tool," dans *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems.* ACM, 2010, pp. 93–102.

[104] H. Lorenz, S. Claudia, et K. Andreas, "Assessing the Human Social and Environmental Risks of Pervasive Computing," *International Journal of Human and Ecological Risk Assessment*, vol. 10, pp. 853–674, 2004.

[105] J. S. V. C. Laurent Ciarletta, Tom Leclerc et A. Schaff, “Towards standards for Pervasive Computing evaluation: using the multi-model and multi-agent paradigms for mobility,” LORIA - Campus Scientifique - BP 239 - 54506 Vandouvre-les-Nancy Cedex, Rapport technique, 2008.

[106] E. Lemar, “The Design and Evaluation of a Storage System for Pervasive,” University of Washington, Rapport technique, 2001.

[107] Z. Liu, N. Gu, et G. Yang, “A reliability evaluation framework on service oriented architecture,” dans *Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on.* IEEE, 2007, pp. 466–471.

[108] S. H. Lee, J. H. Han, Y. T. Leem, et T. Yigitcanlar, “Towards ubiquitous city : concept, planning, and experiences in the Republic of Korea,” dans *Knowledge-Based Urban Development : Planning and Applications in the Information Era*, T. Yigitcanlar, K. Velibeyoglu, et S. Baum, éditeurs. Hershey, Pa.: IGI Global, Information Science Reference, 2008, pp. 148–169.

[109] I. Lera, C. Juiz, R. Puigjaner, C. Kurz, G. Haring, et J. Zottl, “Performance assessment on ambient intelligent applications through ontologies,” dans *Proceedings of the 5th international workshop on Software and performance*, série WOSP 05. New York, NY, USA: ACM, 2005, pp. 205–216.

[110] D. Lupiana, C. ODriscoll, et F. Mtenzi, “Taxonomy for ubiquitous computing environments,” dans *Networked Digital Technologies, 2009. NDT ’09. First International Conference on*, july 2009, pp. 469 –475.

[111] G. A. Lewis et L. Wrage, “A Process for Context-Based Technology Evaluation: Examples for the Evaluation of Web Services Technology,” dans *ICCBSS 06: Proceedings of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems.* Washington, DC, USA: IEEE Computer Society, 2006, p. 63.

[112] H. Liang, Y. Xue, et T. Byrd, “PDA usage in healthcare professionals: testing an extended technology acceptance model,” *International Journal of Mobile Communications*, vol. 1, no. 4, pp. 372–389, 2003.

[113] J. Lu, C. Yu, C. Liu, et J. Yao, “Technology acceptance model for wireless internet,” *Internet Research*, vol. 13, no. 3, pp. 206–222, 2003.

[114] Y. Malik et B. Abdulrazak, “Towards Verifying Service Interoperability Requirements for Pervasive Computing Environments,” dans *PECCS*, 2012, pp. 220–223.

[115] J. Mankoff, “Crossing Qualitative and Quantitative Evaluation in the Domain of Ubiquitous Computing. Presented at the CHI 2005 Workshop on Usage Analysis: Combining logging and qualitative methods,” 2005.

[116] M. Modahl, B. Agarwalla, S. Saponas, G. Abowd, et U. Ramachandran, “UbiqStack: a taxonomy for a ubiquitous computing software stack,” *Personal Ubiquitous Comput.*, vol. 10, no. 1, pp. 21–27, décembre 2005.

[117] J. M. Maestre et E. F. Camacho, “Smart home interoperability: the DomoEsi project approach,” *International Journal of Smart Home*, vol. 3, pp. 31–44, 2009.

[118] R. Morla et N. Davies, “Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment,” *IEEE Pervasive Computing*, vol. 3, pp. 48–56, July 2004.

[119] B. Meyer, “Applying "Design by Contract",” *Computer*, vol. 25, pp. 40–51, October 1992.

[120] S. McCanne, S. Floyd, K. Fall, K. Varadhan *et al.*, “Network simulator ns-2,” 1997.

[121] Y.-W. Moon, H.-S. Jung, et C.-S. Jeong, “Context-awareness in Battlefield using Ubiquitous Computing: Network Centric Warfare,” dans *2010 IEEE 10th International Conference on Computer and Information Technology (CIT)*, 29 2010-july 1 2010, pp. 2873 –2877.

[122] V. Metsis, Z. Le, Y. Lei, et F. Makedon, “Towards an evaluation framework for assistive environments,” dans *Proceedings of the 1st*

international conference on PErvasive Technologies Related to Assistive Environments, série PETRA 08. New York, NY, USA: ACM, 2008, pp. 12:1–12:8.

[123] M. Merdes, R. Malaka, D. Suliman, B. Paech, D. Brenner, et C. Atkinson, "Ubiquitous RATs: how resource-aware run-time tests can improve ubiquitous software systems," dans *Proceedings of the 6th international workshop on Software engineering and middleware*, série SEM 06. New York, NY, USA: ACM, 2006, pp. 55–62.

[124] L. Mamykina, E. Mynatt, et M. A. Terry, "Time Aura: interfaces for pacing," dans *CHI 01: Proceedings of the SIGCHI conference on Human factors in computing systems.* New York, NY, USA: ACM, 2001, pp. 144–151.

[125] G. K. Mostfaoui, J. Pasquier-Rocha, et P. Brzillon, "Context-Aware Computing: A Guide for the Pervasive Computing Community," *Pervasive Services, IEEE/ACS International Conference on*, vol. 0, pp. 39–48, 2004.

[126] E. MacLaughlin, C. Raehl, A. Treadway, T. Sterling, D. Zoller, et C. Bond, "Assessing medication adherence in the elderly: which tools to use in clinical practice?" *Drugs & aging*, vol. 22, no. 3, pp. 231–255, 2005.

[127] N. C. Narendra, "Large scale testing of pervasive computing systems using multi-agent simulation," dans *Intelligent Solutions in Embedded Systems, 2005. Third International Workshop on*, 2005, pp. 27–38.

[128] J. Nielsen, "Ten usability heuristics," 2005.

[129] C. M. Nielsen, M. Overgaard, M. B. Pedersen, J. Stage, et S. Stenild, "Its worth the hassle!: the added value of evaluating the usability of mobile systems in the field," dans *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, série NordiCHI 06. New York, NY, USA: ACM, 2006, pp. 272–280.

[130] S. Neely, G. Stevenson, C. Kray, I. Mulder, K. Connelly, et K. A. Siek, “Evaluating Pervasive and Ubiquitous Systems,” *IEEE Pervasive Computing*, vol. 7, pp. 85–88, 2008.

[131] C. Orwat, A. Graefe, et T. Faulwasser, “Towards pervasive computing in health care - A literature review,” *BMC Medical Informatics and Decision Making*, vol. 8, no. 1, p. 26, 2008.

[132] G. Oh, D. Kim, S. Kim, et S. Rhew, “A quality evaluation technique of RFID middleware in ubiquitous computing,” dans *ICHIT’06. International Conference on Hybrid Information Technology, 2006.*, vol. 2. IEEE, 2006, pp. 730–735.

[133] E. O’Neill, M. Klepal, D. Lewis, T. O’Donnell, D. O’Sullivan, et D. Pesch, “A testbed for evaluating human interaction with ubiquitous computing environments,” dans *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on.* IEEE, 2005, pp. 60–69.

[134] E. ONeill, M. Klepal, D. Lewis, T. ODonnell, D. OSullivan, et D. Pesch, “A Testbed for Evaluating Human Interaction with Ubiquitous Computing Environments,” dans *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 60–69.

[135] D. O’Sullivan et D. Lewis, “Semantically driven service interoperability for pervasive computing,” dans *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, série MobiDe ’03. New York, NY, USA: ACM, 2003, pp. 17–24.

[136] E. ONeill, D. Lewis, et O. Conlan, “A simulation-based approach to highly iterative prototyping of ubiquitous computing systems,” dans *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, série Simutools 09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 56:1–56:10.

[137] E. O Neill et D. Lewis, “A Platform for User-Centred Evaluation of Context-Aware Adaptive Services.”

[138] E. O Neill, D. Lewis, K. McGlinn, et S. Dobson, “Rapid user-centred evaluation for context-aware systems,” *Interactive Systems. Design, Specification, and Verification*, pp. 220–233, 2007.

[139] M. Patil, “SEQUEL: A Public-domain Simulation Platform for Electronics and Power Electronics,” *IETE Technical Review*, vol. 26, no. 1, pp. 47–54, 2009.

[140] D. Petrelli, “On the role of user-centred evaluation in the advancement of interactive information retrieval,” *Information processing & management*, vol. 44, no. 1, pp. 22–38, 2008.

[141] J. Park, M. Moon, S. Hwang, et K. Yeom, “CASS: A Context-Aware Simulation System for Smart Home,” *Software Engineering Research, Management and Applications, ACIS International Conference on*, vol. 0, pp. 461–467, 2007.

[142] S. Pokraev, D. A. C. Quartel, M. W. A. Steen, et M. Reichert, “A Method for Formal Verification of Service Interoperability,” dans *Proceedings of 2006 IEEE International Conference on Web Services 18-22 September 2006, Chicago, Illinois, USA*.

[143] S. Pokraev, D. A. C. Quartel, M. W. A. Steen, et M. Reichert, “Requirements and Method for Assessment of Service Interoperability,” dans *Proceedings of the 2006 4th International Conference on Service Oriented Computing, ICSOC ’06, December 4-7, 2006, Chicago, USA*, série Lecture Notes in Computer Science, vol. 4294. Springer, 2006, pp. 1–14.

[144] T. Perumal, A. R. Ramli, C. Y. Leong, S. Mansor, et K. Samsudin, “Interoperability among Heterogeneous Systems in Smart Home Environment,” dans *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, série SITIS ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 177–186.

[145] T. Perumal, A. R. Ramli, C. Y. Leong, S. Mansor, et K. Samsudin, "Interoperability for Smart Home Environment Using Web Services," *International Journal of Smart Home*, vol. 2, pp. 1–16, 2008.

[146] A. Ranganathan, J. Al-Muhtadi, J. Biehl, B. Ziebart, R. H. Campbell, et B. Bailey, "Evaluating Gaia using a Pervasive Computing Benchmark," University of Illinos at Urbana-Champaign,IL, Rapport technique, 2005.

[147] A. Ranganathan et R. H. Campbell, "Provably Correct Pervasive Computing Environments," dans *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 160–169.

[148] T. Rodden, K. Chervest, N. Davies, et A. Dix, "Exploiting Context in HCI Design for Mobile Systems," dans *in Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[149] V. Reynolds, V. Cahill, et A. Senart, "Requirements for an ubiquitous computing simulation and emulation environment," dans *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, série InterSense 06. New York, NY, USA: ACM, 2006.

[150] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, et K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing*, pp. 74–83, octobre 2002.

[151] K. Romer et F. Mattern, "The design space of wireless sensor networks," *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 54 – 61, 2004.

[152] D. Rosenblum, C. Mascolo, M. Kwiatkowska, D. Ghica, M. Ryan, N. Dulay, et E. Lupu, "UbiVal: fundamental approaches to validation of ubiquitous computing applications and infrastructures," *University of Birmingham, University College London and Imperial College*

London, UK, Project Proposal, EPSRC Project GR D, vol. 76625, pp. 2006–2010.

[153] C. Röcker, “Why Traditional Technology Acceptance Models Won’t Work for Future Information Technologies?” *World Academy of Science, Engineering and Technology*, vol. 65, pp. 237–243, 2010.

[154] D. E. Rowley, “Usability testing in the field: bringing the laboratory to the user,” dans *Conference companion on Human factors in computing systems*, série CHI 94. New York, NY, USA: ACM, 1994, pp. 217–.

[155] I. Roussaki, I. Papaioannou, D. Tsesmetzis, J. Kantorovitch, J. Kalaoja, et R. Poortinga, “Ontology Based Service Modelling for Composability in Smart Home Environments,” dans *Constructing Ambient Intelligence*, série Communications in Computer and Information Science. Springer Berlin Heidelberg, 2008, vol. 11, pp. 411–420.

[156] S. Rao et I. Troshani, “A Conceptual Framework and Propositions for the Acceptance of Mobile Services,” 2007.

[157] L. Rudolph, “Project Oxygen: Pervasive, Human-Centric Computing - An Initial Experience,” dans *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, série CAiSE 01. London, UK, UK: Springer-Verlag, 2001, pp. 1–12.

[158] S. Ruth, “Reducing ICT-related Carbon Emissions: An Exemplar for Global Energy Policy?” *IETE Technical Review*, vol. 28, no. 3, pp. 207–211, 2011.

[159] C. Schlenff, J. Ajot, et R. Madhaven, “PRIDE: A Framework for Performance Evaluation of Intelligent Vehicles in Dynamic, On-Road Environments,” DTIC Document, Rapport technique, 2004.

[160] C. Schlenoff, J. Ajot, et R. Madhavan, “Performance evaluation of autonomous vehicle navigation in dynamic, on-road environments,” *Integrated Computer Aided Engineering*, vol. 12, no. 3, pp. 263–278, 2005.

[161] M. Satyanarayanan, “Metrics and Benchmarks for Pervasive Computing,” *IEEE Pervasive Computing*, vol. 4, pp. 4–6, 2005.

[162] J. Scholtz et S. Consolvo, "Toward a Framework for Evaluating Ubiquitous Computing Applications," *IEEE Pervasive Computing*, vol. 3, pp. 82–88, 2004.

[163] J. Scholtz et S. Consolvo, "Conducting In Situ Evaluations for and With Ubiquitous Computing Technologies," *International Journal of Human-Computer Interaction*, vol. 22, pp. 103–118, 2007.

[164] S. Schneider, *Concurrent and Real-time systems.* Wiley Chichester, UK, 2000.

[165] J. Seo, G. Goh, et G. J. Kim, "Creating ubiquitous computing simulators using P-VoT," dans *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, série MUM 05. New York, NY, USA: ACM, 2005, pp. 123–126.

[166] A. P. Sheth, "Changing focus on interoperability in information systems: from system, syntax, structure to semantics," pp. 5–29, 1999.

[167] D. Shin, "Understanding user acceptance of DMB in South Korea using the modified technology acceptance model," *Intl. Journal of Human–Computer Interaction*, vol. 25, no. 3, pp. 173–198, 2009.

[168] D. H. Shin, "Ubiquitous Computing Acceptance Model; end user concern about security, privacy and risk," *Int. Journal of Mobile Communication.*, vol. 8, no. 2, pp. 169–186, février 2010.

[169] S. Samsuri, Z. Ismail, et R. Ahmad, "User-Centered Evaluation of Privacy Models for Protecting Personal Medical Information," *Informatics Engineering and Information Science*, pp. 301–309, 2011.

[170] C. Scott et M. Jennifer, "Challenges for Ubicomp Evaluation," Technical report ucb-csd-04-1331, 2004.

[171] B. Shirazi, M. Kumar, et B. Sung, "QoS middleware support for pervasive computing applications," dans *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, 2004, p. 10 pp.

[172] J. Sun, Y. Liu, J. Dong, et C. Chen, "Integrating specification and programs for system modeling and verification," dans *Theoretical*

Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on. IEEE, 2009, pp. 127–135.

[173] M. Thompson, “Evaluating intelligent systems with performance uncertainty in large test spaces,” dans *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop.* ACM, 2010, pp. 133–135.

[174] E. Tran, “Verification/Validation/Certification,” *Topics in Dependable Embedded Systems. Carnegie Mellon University*, 1999.

[175] I. Troshani et S. Rao Hill, “A proposed framework for mobile services adoption: a review of existing theories, extensions, and future research directions,” *Mobile Multimedia Communications: Concepts, Applications and Challenges, Hershey, PA, USA: Idea Group Publishing*, pp. 85–108, 2008.

[176] L. Tang, X. Zhou, Z. Yu, Y. Liang, D. Zhang, et H. Ni, “MHS: A Multimedia System for Improving Medication Adherence in Elderly Care,” *Systems Journal, IEEE*, vol. 5, no. 4, pp. 506–517, 2011.

[177] T. B. Ustun, S. Chatterji, J. Bickenbach, N. Kostanjsek, et M. Schneider, “The International Classification of Functioning, Disability and Health: a new tool for understanding disability and health.” *Disability and rehabilitation*, vol. 25, no. 11-12, pp. 565–571, 2003.

[178] A. Varga, “OMNeT++,” dans *Modeling and Tools for Network Simulation.* Springer, 2010, pp. 35–59.

[179] V. Venkatesh et F. Davis, “A theoretical extension of the technology acceptance model: Four longitudinal field studies,” *Management science*, vol. 46, no. 2, pp. 186–204, 2000.

[180] J. J. B. Vikram Vijayraghavan, “WISE - A Simulator Toolkit for Ubiquitous Computing Scenarios,” dans *Workshop on Application Models and Programming Tools for Ubiquitous Computing at UBICOMP 2001.* ACM, 2001.

[181] M. Weiser et J. S. Brown, *The coming age of calm technolgy.* New York, NY, USA: Copernicus, 1997, pp. 75–85.

[182] J. Wu, Y. Chen, et L. Lin, "Empirical evaluation of the revised end user computing acceptance model," *Computers in Human Behavior*, vol. 23, no. 1, pp. 162–174, 2007.

[183] H. Wan, Z. Drey, Z. You, et L. Liu, "Formal Modeling and Verification of Services Managements for Pervasive Computing Environment," dans *Proceedings of The 7th International Conference on Service Systems and Service Management*, Tokyo Japan, 06 2010.

[184] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 3–11, 1999.

[185] S. P. Wyche et R. E. Griner, "Extraordinary computing: religion as a lens for reconsidering the home," dans *CHI 09: Proceedings of the 27th international conference on Human factors in computing systems.* New York, NY, USA: ACM, 2009, pp. 749–758.

[186] T. Winkler et M. Herczeg, "Pervasive Computing in Schools Embedding Information Technology into the Ambient Complexities of Physical Group Learning Environments," dans *Proceedings of Society for Information Technology and Teacher Education International Conference 2005.* Phoenix, AZ, USA: AACE, 2005, pp. 2889–2894.

[187] A. Ward, A. Jones, et A. Hopper, "A new location technique for the active office," *IEEE Personal Communications*, vol. 4, no. 5, pp. 42–47, 1997.

[188] A. Woodruff et J. Mankoff, "Environmental Sustainability," *Pervasive Computing, IEEE*, vol. 8, no. 1, pp. 18 –21, jan.-march 2009.

[189] B. Weiss et C. Schlenoff, "Evolution of the SCORE framework to enhance field-based performance evaluations of emerging technologies," dans *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems.* ACM, 2008, pp. 1–8.

[190] B. Weiss et L. Schmidt, "The Multi-Relationship Evaluation Design Framework: Creating Evaluation Blueprints to Assess Advanced and

Intelligent Technologies," dans *Proceedings of the 2010 Performance Metrics for Intelligent Systems (PerMIS) Workshop*, 2010.

[191] H.-I. Yang, C. Chen, B. Abdulrazak, et S. Helal, "A framework for evaluating pervasive systems," *International Journal of Pervasive Computing and Communications*, vol. 6, no. 4, pp. 432–481, 2010.

[192] C. Yoon et S. Kim, "Convenience and TAM in a ubiquitous computing environment: The case of wireless LAN," *Electron. Commer. Rec. Appl.*, vol. 6, no. 1, pp. 102–112, janvier 2007.

[193] D. Zhang et B. Adipat, "Challenges, methodologies, and issues in the usability testing of mobile applications," *International Journal of Human-Computer Interaction*, vol. 18, no. 3, pp. 293–308, 2005.

[194] A. Zaslavsky, "Adaptability and interfaces: key to efficient pervasive computing," dans *NSF Workshop series on Context-Aware Mobile Database Management, Brown University, Providence*, 2002, pp. 24–25.

[195] X. Zeng, R. Bagrodia, et M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," dans *Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on*, mai 1998, pp. 154 –161.

[196] Y. Zhang, S. Zhang, et S. Han, "A new methodology of QoS evaluation and service selection for ubiquitous computing," *Wireless Algorithms, Systems, and Applications*, pp. 69–80, 2006.