

A WORKFLOW-BASED MOBILE GUIDANCE FRAMEWORK FOR MANAGING
PERSONAL ACTIVITIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

GÖKHAN TÜYSÜZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INFORMATION SYSTEMS

SEPTEMBER 2013

Approval of the thesis:

**A WORKFLOW-BASED MOBILE GUIDANCE FRAMEWORK FOR
MANAGING PERSONAL ACTIVITIES**

submitted by **GÖKHAN TÜYSÜZ** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Nazife Baykal
Director, **Informatics Institute**

Prof. Dr. Yasemin Yardımcı Çetin
Head of Department, **Information Systems**

Assist. Prof. Dr. P. Erhan Eren
Supervisor, **Information Systems Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. Altan Koçyiğit
Information Systems Department, METU

Assist. Prof. Dr. P. Erhan Eren
Information Systems Department, METU

Assoc. Prof. Dr. Aysu Betin Can
Information Systems Department, METU

Dr. Nail Çadallı
KAREL A.Ş.

Assist. Prof. Dr. Alptekin Temizel
Information Systems Department, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: GÖKHAN TÜYSÜZ

Signature :

ABSTRACT

A WORKFLOW-BASED MOBILE GUIDANCE FRAMEWORK FOR MANAGING PERSONAL ACTIVITIES

Tüysüz, Gökhan

M.S., Department of Information Systems

Supervisor : Assist. Prof. Dr. P. Erhan Eren

September 2013, 69 pages

In daily life, people have to perform a large number of activities typically in a limited amount of time. Thus, they may need help and guidance provided by support systems in order to accomplish these activities accurately and in the correct order. Accordingly, in this study, we propose a software framework based on workflows and supported by a mobile application to assist users in pervasive environments for managing their personal activities. Pervasive computing enables to ease and automate the execution of the activities by integrating user's context into management of activities. Therefore, the framework augments user activities with context information, and gives a broader and a customized meaning to them, so provides advanced assistance by using sensors and devices in the environment, making web service calls, and utilizing mobile phone features. Moreover, the relationship between activities and context resources enable to automate user's tasks by defining rules within the framework. In parallel with this information, a prototype implementation is developed, and tested with scenarios from various domains. Consequently, applicability of these scenarios indicates workableness and feasibility of this framework in pervasive environments.

Keywords: context-aware systems, ubiquitous computing, mobile application, personal activity, assisted living

ÖZ

İŞ AKIŞ YÖNETİM SİSTEMİ TEMELLİ KİŞİSEL İŞLERİ YÖNETMEYİ SAĞLAYAN MOBİL ASİSTAN ÇERÇEVESİ

Tüysüz, Gökhan

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. P. Erhan Eren

Eylül 2013 , 69 sayfa

İnsanlar, günlük hayatlarında çok sayıda işi, genellikle de kısıtlı bir zaman içinde yapmak zorundadırlar. Bu yüzden, bu işlerini düzgün ve doğru bir sırada yapabilmelerini sağlayacak bir destek sisteminine ihtiyaç duyabilirler. Bu tez çalışması ile, insanları yaygın bilişim altyapısına sahip alanlarda kişisel işlerini yerine getirmelerinde destekleyecek, iş akış yönetimi sistemi temelli ve mobil uygulama ile kullanımı desteklenmiş bir yazılım çerçevesi önerilmektedir. Yaygın bilişim, insanların işlerini çevrelerindeki kaynaklarla ilişkilendirerek, bu işlerin daha kolay ve otomatikleştirilmiş bir şekilde yapılabilmesini sağlamaktadır. Bu sebeple, bu yazılım çerçevesi bu işlere daha geniş ve özelleştirilmiş yeni bir anlam kazandırarak, çevresindeki sensörlerle ve cihazlarla haberleşmesini, internetteki kaynaklara ulaşmasını, ve kendi cep telefonunu özelliklerinden faydalanmasını sağlayacak gelişmiş bir rehberlik hizmeti sunmayı amaçlamaktadır. Buna ek olarak, bu ilişki sayesinde tanımlanabilecek kurallar ile işlerin otomatize edilebilmesi mümkün olmaktadır. Bu bilgiler doğrultusunda, örnek bir uygulama geliştirilmiş ve bu uygulama farklı alanlardaki senaryolarda test edilmiştir. Sonuç olarak, bu senaryoların yapılabilirliği, önerdiğimiz çerçevenin uygunluğunu göstermektedir.

Anahtar Kelimeler: çevre farkında sistemler, yaygın bilişim, mobil uygulama, kişisel etkinlik, destekli yaşam

To my family

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciations to my supervisor Assist. Prof. Dr. Erhan Eren for his encouragement, guidance and insight throughout this thesis research.

I would like to thank Bilgin Avenoglu for his support and for sharing his knowledge while developing the topic of this research. Also, I would like to thank Süleyman Özarslan for all the helps he had done while setting up this thesis' prototype implementation in Wireless Lab.

I would like to thank my colleague Ebru Gökalp for sharing her experience and knowledge while preparing this thesis study.

I am extremely grateful to my parents for their continuous supports in the completion and preparation of this study as in every moment of my life. I would like to thank my brother for his guidance and advices throughout my life, and his little daughter for making my life more enjoyable during this study.

I would like to thank my friends and colleagues, this thesis would not have been possible without their support.

TABLE OF CONTENTS

| | |
|---|------|
| ABSTRACT | iv |
| ÖZ | v |
| ACKNOWLEDGMENTS | vii |
| TABLE OF CONTENTS | viii |
| LIST OF FIGURES | xi |
| LIST OF ABBREVIATIONS | xiii |
| CHAPTERS | |
| 1 INTRODUCTION | 1 |
| 1.1 Problem Definition and Motivation | 1 |
| 1.2 Thesis Outline | 3 |
| 2 BACKGROUND AND RELATED WORK | 5 |
| 2.1 Ubiquitous Computing | 5 |
| 2.2 Workflow Management System | 8 |
| 2.3 Context-Aware Systems | 10 |
| 2.4 Publish/Subscribe Based Messaging | 12 |
| 2.5 Mobile Computing | 14 |
| 2.6 Related Work | 15 |

| | | |
|-------|--|----|
| 3 | PROPOSED FRAMEWORK | 19 |
| 3.1 | Workflow Management System | 19 |
| 3.2 | Messaging System | 21 |
| 3.3 | Mobile Application | 23 |
| 3.4 | Coordination Management System | 24 |
| 4 | PROTOTYPE IMPLEMENTATION | 27 |
| 4.1 | Workflow Management System | 28 |
| 4.2 | Messaging System | 28 |
| 4.3 | Mobile Application | 32 |
| 4.4 | Coordination Management System | 36 |
| 4.4.1 | Workflow Coordinator Module | 36 |
| 4.4.2 | Communication Module | 37 |
| 4.4.3 | Context Module | 39 |
| 4.4.4 | Automation Module | 42 |
| 5 | FRAMEWORK SCENARIOS | 45 |
| 5.1 | Smart Environment Scenario | 45 |
| 5.2 | Travel Scenario | 48 |
| 5.3 | Health Scenario | 50 |
| 6 | CONCLUSION | 55 |
| 6.1 | Summary and Contributions | 55 |
| 6.2 | Future Work | 57 |
| | REFERENCES | 59 |

APPENDICES

| | | |
|---|---|----|
| A | YAWL | 63 |
| | A.0.1 Introduction to YAWL Architecture | 63 |
| | A.0.2 YAWL Elements | 64 |
| | A.0.3 YAWL Data Transfer | 66 |
| | A.0.4 YAWL Editor | 68 |

LIST OF FIGURES

FIGURES

| | | |
|------------|---|----|
| Figure 2.1 | Computing Eras | 7 |
| Figure 2.2 | Mobile Device Usage | 8 |
| Figure 2.3 | Early System Architectures [30] | 9 |
| Figure 2.4 | Publish / Subscribe Architecture | 13 |
| Figure 2.5 | Space Time Synchronization Decoupling [10] | 13 |
| Figure 2.6 | Mobile Users vs Desktop Users | 15 |
| Figure 3.1 | Framework Conceptual Architecture | 20 |
| Figure 3.2 | Framework Task Definition | 21 |
| Figure 3.3 | Sample Messaging Cases | 22 |
| Figure 3.4 | Navigation Screen of an Installed Application | 23 |
| Figure 3.5 | Coordination Management System Modules | 24 |
| Figure 3.6 | Context Task Relation | 26 |
| Figure 4.1 | Implementation Architecture | 27 |
| Figure 4.2 | Workflow Patterns | 29 |
| Figure 4.3 | MQTT Framework Architecture | 30 |
| Figure 4.4 | Messaging XML Schemas | 30 |
| Figure 4.5 | A Sample Request Message | 31 |
| Figure 4.6 | A Sample Response Message | 31 |
| Figure 4.7 | MQTT Topics | 32 |
| Figure 4.8 | Application MQTT Services | 33 |
| Figure 4.9 | MQTT Publisher Android Service Binder Code Sample | 34 |

| | |
|--|----|
| Figure 4.10 MQTT Subscriber Android Service Binder Code Sample | 34 |
| Figure 4.11 Application Base Activity | 34 |
| Figure 4.12 Mobile Application Screens for Workflow Operations | 35 |
| Figure 4.13 Workflow Management Service Class Diagram | 36 |
| Figure 4.14 Communication Module Architecture | 38 |
| Figure 4.15 Context Module ER Diagram | 40 |
| Figure 4.16 Sample Task Records | 40 |
| Figure 4.17 Framework Task Activity Types | 41 |
| Figure 4.18 Sample Task Related Parameter Records | 41 |
| Figure 5.1 Smart Environment Workflow | 46 |
| Figure 5.2 Close Window Task Execution Flow | 47 |
| Figure 5.3 Smart Environment Scenario Mobile Application Screens | 47 |
| Figure 5.4 Data Flow between Components | 48 |
| Figure 5.5 Travel Workflow | 49 |
| Figure 5.6 Travel Scenario Mobile Application Screens | 50 |
| Figure 5.7 Learn Schedule of Ferry Task Execution Flow | 51 |
| Figure 5.8 Patient Treatment Workflow | 52 |
| Figure 5.9 Health Scenario Mobile Application Screens | 52 |
| Figure A.1 Yawl Architecture | 64 |
| Figure A.2 Yawl Element Types | 65 |
| Figure A.3 Yawl Task Lifecycle | 66 |
| Figure A.4 Yawl Parameter Passing Options | 67 |
| Figure A.5 Yawl Editor Screen | 68 |
| Figure A.6 YAWL Process Flow Definition | 69 |

LIST OF ABBREVIATIONS

| | |
|------|-----------------------------------|
| BPM | Business Process Modeling |
| CoMS | Coordination Management System |
| DBMS | Database Management System |
| ER | Entity Relationship |
| HTTP | Hypertext Transfer Protocol |
| IoT | Internet of Things |
| LGPL | Lesser General Public License |
| MCC | Mobile Cloud Computing |
| MQTT | Message Queue Telemetry Transport |
| QR | Quick Response |
| REST | Representational State Transfer |
| RSMB | Really Small Message Broker |
| UIMS | User Interface Management System |
| WfMC | Workflow Management Coalition |
| WFMS | Workflow Management System |
| YAWL | Yet Another Workflow Language |

CHAPTER 1

INTRODUCTION

In this thesis study, a mobile assisted guidance framework is proposed in order to assist users while performing their personal activities. For this purpose, first of all, personal activities are modeled using workflows. In business environments, the routine processes that do not differ so often and has to be done according to some pre-defined procedures have been modeled as workflows for years. In a similar manner, a daily life routine, a process driven by legislation, a travel plan, a post-operative treatment of a patient, a care plan for elder people may also be modeled as workflows. Then, by utilizing the opportunities of pervasive computing environments it is possible to create smart workflows for personal activities. By this means, workflows will be aware of their context and adapt to their environment accordingly. This dynamism enables the context-awareness of the system. Additionally, a mobile application is needed in order to provide a communication link between the user and the framework, so it will also make possible to benefit from mobile computing and wireless network features. Lastly, a publish / subscribe based messaging system is used in the design of the framework since HTTP based communication infrastructure is not proper for such a framework because users are in need of constant bi-directional communication that enables interaction with all the devices in the environments, even the devices that are power and memory resource constrained. In this thesis, it is also intend to demonstrate the feasibility and effectiveness of such a framework in a pervasive environment. For this reason, scenarios in different domains are explained in detail.

1.1 Problem Definition and Motivation

In daily life, people are in need of carrying out many tasks in a limited amount of time. In addition, some of these tasks may be quite complex and put extra burden on people's mental capacity. Mistakes and omissions become inevitable due to such increased pressure for being more productive. For instance, a person may forget to pay his electric bill when he is dealing with fixing a problem on his computer; or, in the case of accomplishing a task for the first time, people need extra time, and they are more prone to making errors. To illustrate; since a university student may not

know the order of tasks needed for completing a registration process, he can lose time or make a mistake while trying to follow the required steps.

On the other hand, advances in computing provide an opportunity in helping people with their tasks. In particular, ubiquitous computing is necessary in making such technology transparently available in our environments, at home, at work, or outside. Ubiquitous computing provides increased communication capabilities, awareness and functionalities [40] by using high-speed and low-powered wireless communications, small sensors and devices such as smart mobile phones.

The aim of this thesis study is to provide a framework which utilizes ubiquitous computing technologies in order to help people organize their daily activities. At its core, the framework incorporates workflows for modeling such activities as well as assisting its users in carrying out their activities more effectively according to the needs. Contrary to workflows in business environments, users of the framework are mobile most of the time, but need to have constant interaction with the framework. For this purpose, a mobile phone application is developed as part of the framework. In addition, smart phones may also be used as an intermediary unit which enables to use and store data about task's context. For instance, a user may arrive at the location of his current task by using a navigation application on his phone, or he may write a note, upload a photo, or record a voice comment in order to help successor users of the same workflow. Furthermore, the framework collects context data related to the user's current situation from electronic devices such as environmental sensors, mobile phone sensors and web services. It matches those context data with the tasks of users in their workflows, and then makes inferences related to the user activities by examining the context data, and presents these inferences to the user for assistance.

Consequently, users should be presented with a more advanced framework rather than a simple application that tells them what to do and how to do. Activities should be correctly structured, and tasks in these activities should be related with context sources such that context information should be obtained, enriched and used for easing and automating user responsibilities. However, developing such a framework is not trivial.

First of all, it requires knowledge about the user activities and it has to store the definition, order and start/end times of the activities. In other words, the framework needs the structured model of the user activities. In Business Process Modeling (BPM), workflows have already been used to model the processes for a long time, and they increase efficiency of the business processes by concentrating on the routine aspects of the work activities [12]. Similarly, the activities in people's daily lives may also be modeled and organized by using workflows. If an activity is modeled as a flow, people can track their assignments step-by-step so that they do not get confused about the order of the assignments, and most importantly do not skip a required step.

Another significant issue is the ability to know and use current context of the user. By this way, daily life activities can be linked with the ubiquitous computing environment [24], and workflow tasks can be augmented with the help of context information. The knowledge of how the weather will be throughout a day can be necessary for a tourist who has outside activity plans. Similarly, a heat sensor in the environment may serve as a data resource for a process in which temperature of the room should be kept in an acceptable range. Context is of capital importance since it helps define and augment the tasks [21]. In addition, this enables that every task will be instantiated in its particular context, and so can be customized. In other words, tasks will become unique according to their contexts.

Also, customized tasks provide an opportunity to enrich tasks with supplementary text, audio, image and video resources uploaded by users and workflow designers. For example, when a user is designing a workflow or is working on an activity, he can upload a video record that will help users of the same workflow to perform that activity more comfortably. As a result, the general task concept in a workflow system is upgraded into a higher level where tasks wrap up the context information and supplementary resources together with their workflow definitions.

The framework also needs to know which context data are needed for which activities, and when a user should be informed or an action should be taken on behalf of the user context information. This is achieved by defining rules that examine the context data and by determining the action that will be implicitly taken. So, workflow can be automated. For example, when a student leaves his home to go to school, the framework reminds him to take his library books with him because it is their due date. In a similar way, since an activity can be related with a context resource, a sensor value may result in proactive completion of an activity again according to pre-defined rules, with the framework taking the initiative. For instance, when the temperature is raised to an adequate level, data coming from heat sensor may trigger the completion of a task in the workflow.

1.2 Thesis Outline

This thesis study consists of six chapters. First (this) chapter introduces the problem and motivation for this study and explains the overall concept. Chapter 2 includes the background information about Workflow Management Systems, Ubiquitous Computing, Context-aware Systems, Mobile Computing and Publish / Subscribe based Messaging Systems technologies and discusses the previous works and related literature information. Chapter 3 explains the proposed framework architecture and gives the conceptual description of the components. In Chapter 4, realization of the framework architecture is explained in detail. Software and hardware components used for building the framework are expressed, and implementation of the software modules

and mobile application are displayed. In Chapter 5, the feasibility of the framework is demonstrated in scenarios from different domains. Finally, Chapter 6 provides conclusion and possible future work of this study.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter provides literature information about previous studies and researches carried out in the subject domain. In Section 2.1, Ubiquitous Computing vision (introduced by Mark Weiser) and its general characteristics are explained. In Section 2.2, Workflow Management Systems (WFMS) are described in detail; the evolution of WFMS, features and fields of usage are analyzed. Section 2.3 defines context-aware systems and asserts what features must be provided in order to be determined as context-aware. In Section 2.4, Publish / Subscribe based messaging systems is described, and the reasons why such a system is needed are explained. In Section 2.5, importance and growing popularity of mobile computing are covered. Finally, in Section 2.6, literature studies aiming to guide users in pervasive environments by using these explained technologies (partially or fully) are discussed.

Originally, this thesis study is inspired from ongoing PhD. study [1] of Bilgin Avenoglu and a conceptual architecture is designed to realize the insights in that study. However, this framework mainly focuses on the higher level parts (especially for the end users) and demonstrates the applicability of the concept and the framework in different domains. During our researches and studies in this field, some major contributions are made discriminately from the PhD. study. Although, these contributions will be explained more clearly in the following chapters, they can be briefly summarized as: enabling context-awareness of the process, augmentation of activities in the process, enrichment of context information about an activity (with supplementary resources), automation of activities without user prevention. All of these contributions are supported by our proposed mobile application.

2.1 Ubiquitous Computing

Mark Weiser envisioned that in future computing will be so common in everywhere, yet no one will notice its existence, and they will ubiquitously reach the benefits of the computing [35]. Later, his vision on ubiquitous computing is studied with other titles but almost the same concepts such as “pervasive computing”, “ambient intelligence”,

“Internet of Things”. Although each term slightly emphasizes on different aspects, their main differences are rather at an academic level. General features common to all fields are [11]:

- Decentralized architecture so are managed with comprehensive network
- Embedded computer hardware and software into the objects that we use daily
- User ability to reach information with services anywhere and anytime by mobile support
- Context-awareness that they are adaptable to the dynamic context changes
- Routine tasks that are automatically recognized and processed without user intervention

Ever since people started to use computing technology for their needs, computers have experienced different evolution phases. Weiser separates computing history into three eras [37]: “Mainframe Computing”, “Personal Computing”, and “Ubiquitous Computing” (its evolution with time is displayed in Fig. 2.1). In Mainframe Computing, many people shared one computer due to the technological and economical limitations. Computers were run by experts and not used for personal needs. Second era began in 1984 when number of people using a personal computer passed the number of people using shared computers. In Personal Computing era, people were responsible from their computers and kept their personal resources in their computers. Through the rapid development in technology, computers became smaller and cheaper which makes it more likely to own and use more than one computer for each person; thus, Ubiquitous Computing era began. In this era, there are lots of computers sharing each of us. We may interact tens of computers even browsing in the internet. Moreover, we have a constant interaction with computers anywhere and anytime such as driving the car (e.g. radio, sensors, automatic transmission, navigation device), staying at the home (e.g. remote controller, alarm clocks, washing machines, ovens), working at the office (e.g. boards, RFID, mobile devices, printers).

In ubiquitous computing paradigm, it is proposed that people will benefit from the ubiquitous computing (power of the many computing devices and systems at the same time) as it is a daily routine activity; yet, people will not be even aware of they are using it. Mark Weiser [35] explains that being unaware situation; "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it".

Another key point is the importance of location and scaling in the computing world. Although the primary goal of ubiquitous computing is providing ubiquitous services and networks to the people, location and scalability are also supporting features for achieving this goal. Computers should be aware of where they are, so they should

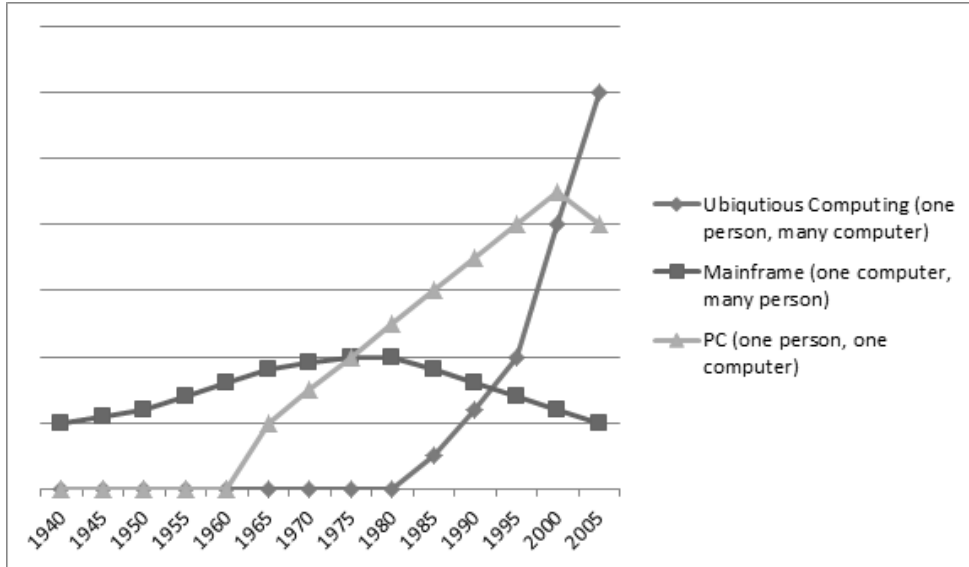


Figure 2.1: Computing Eras

adapt their behavior intelligently according to their environment. By emphasizing the importance of being aware of the location, indeed Mark Weiser points out the importance of the context of a computer. Ubiquitous computing environment should be quite saturated with computing and communication. In daily lives, people are generally on the move to go to work, to go shopping or to go out, so pervasive technology must support mobility in order to provide them the opportunities of the computing [27]. Additionally, users must be aware of their environment and be able to interact with other devices in their environment while they are moving.

Ubiquitous computing proposes that there will be very small and wirelessly interconnected sensors and microprocessors in the environment which are embedded into the objects that we use daily [36]. These devices will have an information processing and communication capabilities that turn ordinary environments into smart environments. Thus, it will be possible to have knowledge about the context of the people, communicate with the other computational devices in their vicinity, and assist people when needed.

Along with the technological advancements, computers are becoming smaller and cheaper, by this means they are integrating into our daily life more permanently. The most explicit practice of this integration can be seen in the usage of mobile devices which have a steady increase as shown in Fig. 2.2 [4]. According to the survey of Cisco Visual Networking Index; Global Mobile Data, mobile devices will outnumber the humans on the earth by the end of 2013. As mobile phone users are increasing every day, mobile phone is becoming more indispensable tool of a human. People are using them for activities such as communicating, payment, socializing, games. Mobile phones are transforming into a door opening to the ubiquitous environment. Hence, mobile

applications are becoming essential requirements to meet users with the ubiquitous computing environments’ opportunities.

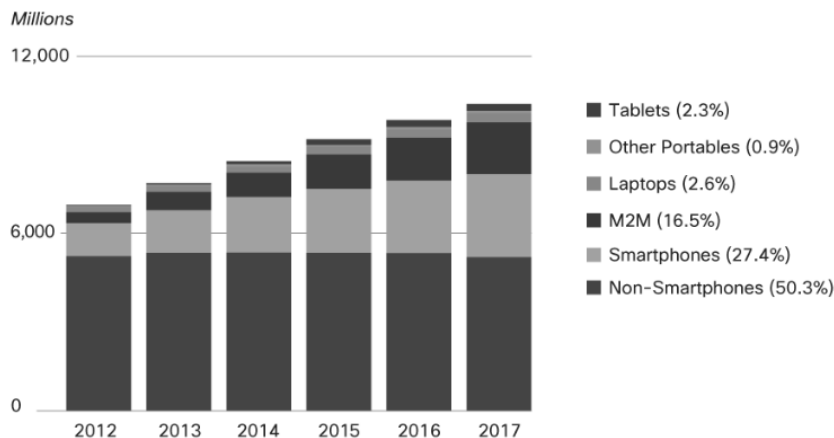


Figure 2.2: Mobile Device Usage

2.2 Workflow Management System

In the sixties’ information systems, applications were running stand alone. For each application, an application specific user interface and database system has to be developed in order to support specific user routines, and database storage and retrieval operations. In the seventies, data layer was removed from the applications and Database Management Systems (DBMS) are begun to be used. By this means, data management was not a burden for applications anymore. In the eighties, user interface layer had faced a similar evolvment, and user interfaces were also extracted from the application by the emergence of the User Interface Management Systems (UIMS). Lastly, W.M.P. van der Aalst proposed that the 90s will be marked as the emergence of the WFMS in order to remove business procedures out of the applications and emphasized that benefits of WFMS is comparable to DBMSs and UIMSs [30]. Although, WFMS is not as breakthrough as DBMS and UIMS, it is used intensely in many domains to handle complex operations. For instance, banking (e.g. credit card, loan approval), advertisement, manufacturing are some of them.

Consequently, process orientation in general and process management in particular are improved in order to support the evolution of enterprise system architectures. This evolution is guided by Dijkstra’s “Separation of Concerns” principle which means each part should focus upon one aspect; in such a manner WFMS are responsible from business procedures in the application. It is also a fundamental principle while dealing with the complexity by computer scientists [38].

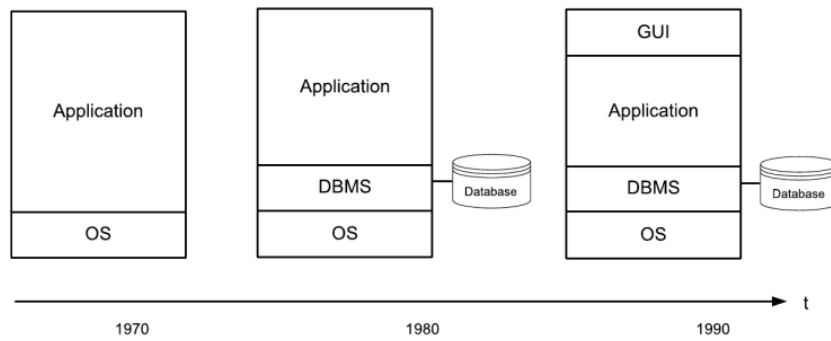


Figure 2.3: Early System Architectures [30]

Workflow Management Coalition defines Workflow: “the automation of a business process, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”. Then, WFMS are developed in order to design the workflows and manage the execution of those workflows. WFMS have workflow engine software module that interprets the process definition, interacts with workflow participants, and invokes information services where necessary. [38]

The workflow concept extends the concepts of process definitions. In industry, processes have already been used to increase efficiency by focusing on the routine aspects of the work activities. In general, they divide work activities into well-defined tasks, procedures, rules and roles that regulate how to perform those activities. Early on, humans performed work activities from beginning to end via manipulating physical objects. However, starting from the information era, computers began to play an important role on workflow activities’ execution. It is aimed to partially or fully automate the processes, computers may perform some tasks or may enforce some pre-defined rules to complete tasks [12].

Recently, several workflow languages (commercial, free, open-source) have been defined in order to manage workflows, yet there are no commonly agreed formal rules among these languages. One of the main reason of this problem is the lack of the accepted formal foundation for workflows [32]. Although, Workflow Management Coalition (WfMC) has efforts on the issue, most of the products use proprietary languages that are not tool independent [33]. Compared to other workflow languages, petri-net based languages are more favorable since they have great advantages while modeling a process. Most importantly, petri-nets are Turing complete, in other words they can provide any functionality in terms of an algorithm [29]. Viriyasitavat et al. refers the advantages of petri-nets among others [34]:

- mathematically-based formal semantics makes it self-documenting and powerful design tool
- large number of techniques for analysis
- support of the concept of a hierarchy net
- distinguish between places (services) and transitions (tasks)
- graph-like representations that have useful links both to graph theory and to algebra

The Workflow Patterns Initiative developed a collection of workflow patterns in order to assess the expressive power of workflow languages. Power of workflow languages is defined according to the number and type of patterns that they support [22]. The aim of this study is to describe the potential capabilities that a WFMS may have during the execution of a business process. There is a range of patterns specified from very simple to the very complex and cover the tasks that can be performed within the most business processes [39]. Thus, they are used for benchmarking the features of workflow languages [32].

As Mark Weiser [35] proposed in his paper, nowadays people are more interactive with their environment due to the growing availability of internet. While people have been carrying out their task with offline workflows, now they are trying to use online collaborative system to achieve higher level interactions [14]. Thus, there is a tendency to use WFMS with real-time interactive applications. Personal workflows may constitute a baseline for assisting users in their daily lives, caring the elderly people, monitoring the patients and orientating people in an unfamiliar environment.

2.3 Context-Aware Systems

Mobile device (notebook, PDA, smart phones) usage is increasing every day, and it makes pervasive (ubiquitous) systems more significant. A key characteristic of ubiquitous systems is that they weave into daily life in such a way that no one will notice its existence after a while. For this reason, being context-aware is an important concept since context-aware systems can offer services according to the current context without explicit user intervention and can automatically adapt to their changing contexts, so it also increases the usability and the effectiveness of the system [3].

In early studies context is defined as the location of the user. After a while, it is understood that location is merely not enough for describing the context information. Environment, identity, and time are added to the definition. However, all these explanations are so particular and is not a de-facto explanation. In their study [8], Dey et al give the most comprehensive definition: “Context is 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. Context is typically the location, identity, and state of people, groups, and computational and physical objects”.

Being a context-aware system means to be able to exploit context information in order to provide relevant services and information to the users. Then, what should be done to use context information effectively. Dey et al [7] proposes a set of context-aware functions that must be implemented in order to reach a context-aware system. This set includes: presenting information and services, executing services, and storage of the context information for further usage.

In the first category, presenting information or service, a context-aware system should be capable either present context information to the user; or it should offer an appropriate service according to the user context. For instance, a navigation application should point out the location of the user in a map and/or should offer some interest places according to the user context.

Secondly, automatically executing a service has practical importance since it may assist users without explicit intervention. A context-aware system should trigger or suppress an action according to the user’s context information, even more it should reconfigure the system when the context of the user changes. For example, a navigation application may warn the user when he arrived at the target location, and in a case where user turned to the wrong direction, it should recalculate the new routing information.

Lastly, services should attach context information to the resources for later retrieval. Keeping related resources in storage and presenting them to the user when necessary has practical uses. It assists the user to remember what he had done previously. For instance, in a field research, a picture kept with its location and time may be very helpful for further analysis. Also, users may reach information resources of other users who had been previously in the same context. A social network application may display users the tags or the comments of previous user in the same vicinity, and make a suggestion by matching the context data of user with the data kept at the system’s storage.

Recently, Hong et al also emphasized the user preferences for context-aware systems [17]. They explain the importance of user preferences that services that a user wishes to reach may differ although they are in the same context. Yet, predicting users’ desire according to only sensor data are not quite possible. System should know preferences and status of the user. For instance, in a shopping mall environment, service offerings on behalf of age and sex information will be clearly more reliable. In order to benefit from the context-aware system efficiently, utilizing context history and user preferences are also important; thereby, it is possible to make more reliable suggestions to the users.

2.4 Publish/Subscribe Based Messaging

Internet of Things (IoT) vision aimed at seamless integration between physical world and the digital world. Thus, objects that we use in our daily lives must be connected to our world. However, in this case two new problems arise. First one is scalability issue. Billions of devices must be interconnected which cause challenges for both internet and those devices. Secondly, there is a need for a standard language that every object can support. In other words, they have to speak a common language to communicate.

In pervasive environments, context of a device may constantly change, so there is a need for a flexible and dynamic communication model. HTTP (Hypertext Transfer Protocol) is not appropriate protocol for IoT vision, since point-to-point and synchronous communication models (as HTTP) lead to static systems. Also, HTTP is a high level protocol, it cannot be supported by all embedded devices in physical world since it does not support persistent communication and not designed for lightweight devices' communication [5].

Scalability problem can be handled by using data-centric communication approach in which information is sent to the devices according to their contents or interest, not to the network address. Publish/Subscribe based messaging protocols are the well-known and widely-used example of data centric approach. "A publish/subscribe system is a middleware communication service that delivers messages from a sender to one or more receivers using the preferences expressed by those receivers, rather than relying on an explicit destination address set by the sender. Specifically, a sender publishes messages, while receivers subscribe for messages that are of interest to them; the system is responsible for delivering published messages to matching subscribers" [2]. In Publish/Subscribe based systems, it is straightforward to add a new receiver or sender to the system, even replacing an existing one. Thus, scalability is managed effectively and dynamic communication topology is supported [18].

The part who registers for an interest is called as Subscriber. On the other hand, the part which produces certain information related with the interest area is called as Publisher. Lastly, the part which ensures the arrival of the data coming from publisher going to the subscriber is called as Broker. Broker is also the entity that coordinates subscriptions. Related communication model is displayed in Fig. 2.4 [18].

Publish/subscribe system gets its strength from decoupling in time, space and synchronization [10]. This paradigm is shown in Fig. 2.5. Firstly, a publisher sends a message as an event to the Event Service (i.e. Broker), and subscriber receives the messages of its interest areas. Space decoupling provides that publisher and subscriber do not have to know each other. Neither part holds a reference to the each other. This decoupling eases the dynamic adaptation of the both parts. Secondly, time decoupling means that both parts do not have to be online at the same time in order to communicating. Publisher may send a message even if subscriber is offline, and conversely subscriber

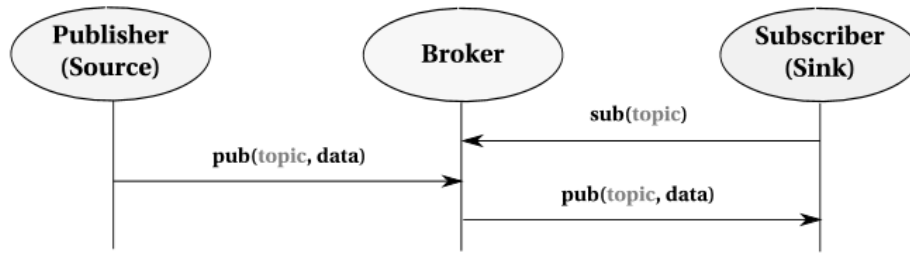


Figure 2.4: Publish / Subscribe Architecture

may receive a message even if publisher is offline. Also, some advanced event services own a specific buffer to cache message, and transmit it to the subscriber when it becomes online. Lastly and most importantly, synchronization decoupling enables producers are not blocked when they are generating a message and subscribers can receive message asynchronously while they are performing another activities.

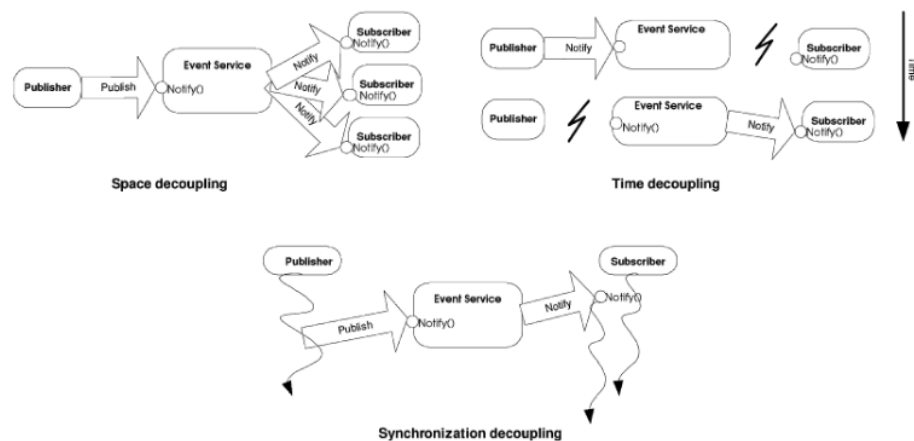


Figure 2.5: Space Time Synchronization Decoupling [10]

Publish/Subscribe systems can be categorized into three groups according to their interests matching strategy: “Type”, “Content” and “Topic”. In type based systems, subscribers explicitly state the type of data they want to receive. For instance, a type based subscriber may wish to receive data about temperature. Content based systems are more advanced version of type based systems, they describe the content of the message, and e.g. they may subscribe to a temperature data which is below/above a specific degree. In topic based systems, topics are needed to be known in advance, so it is possible to subscribe to only specific sets of topics. Topic based systems supports static and primitive messaging and it enables to develop more efficient architectures. Besides, most systems allow topic names with wildcards which may cover a broader range of interest. On the other side, although content-based systems are effective in

describing the characteristics of the data, it also limits the scalability. Its expressiveness requires sophisticated and expensive protocols and advanced message delivery algorithms which causes a higher overhead [10].

In recent years, software architectures are generally using the messaging based communication frameworks in order to increase speed and scalability of their systems, especially for mobile applications. Furthermore, when cloud computing opportunities are used, message-based architectures are having more appreciable performance.

2.5 Mobile Computing

Interactive communication technologies have a steady growth during the recent years, and allow us to communicate with people and reach information in many ways, in various locations. So these technological developments lead to a significant change in the way that we use computing. Computers transform into the information and communication device rather than a data processor and information storage device. They become smaller and come into the daily lives and bring people together. So, computers enable people to reach huge amount of information and also share this information. Thus, a new paradigm, mobile computing, forms. According to its own inner features, mobile computing should have its specific architecture design, and specific applications and interfaces should be developed.

Zheng defines mobile computing [41]; “a broad set of computing operations that allow a user to access information from portable devices such as laptop computers, PDAs, cell phones, handheld computers, music players, portable game devices and so on”. Mobile computing has two operational modes: disconnected and connected mode. In disconnected mode, user can reach only the local storage of the device. In connected computing, user can reach huge amount of data from other devices and servers via a wireless or wired network access. The latter one is the mode that makes mobile computing an indispensable part of daily life, especially with the benefits of wireless connection technologies. Research results of Microsoft Tag about mobile marketing growth and internet usage in Fig. 2.6 indicates growing (and expected) popularity of mobile device usage [28].

However, widespread use of laptop computers, PDAs, and especially smart phones also raises new problems concerning how to build a distributed system with mobile clients [27]. Thus, serious researches have begun about mobile computing. Although many principles are similar to wired computing, mobile computing has some constraints that make it differ. Firstly, although CPU and memory power of mobile devices have been increased thanks to the technological developments, yet it is not possible to be as powerful as desktop computers, briefly they are resource poor. Secondly, they have security concerns about information compromises. They are more prone to be lost, damaged and stolen. Thirdly, wireless network quality is unpredictable which may

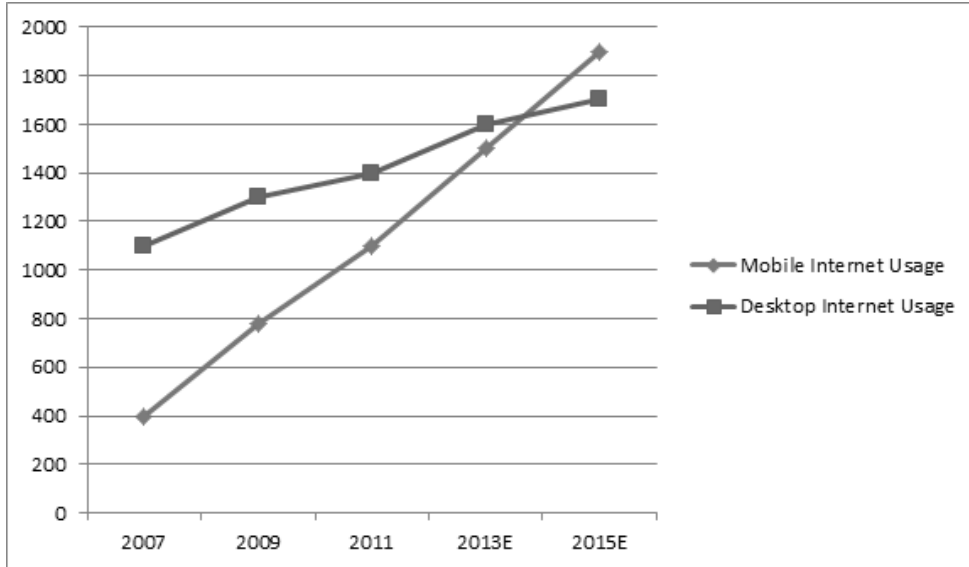


Figure 2.6: Mobile Users vs Desktop Users

cause synchronization and performance problems. Lastly, mobile computing have to concern about power usage as they work with finite energy source batteries.

In order to satisfy the explosive growth of the mobile applications and to respond the mobile computing problems, emerging of cloud computing concept, mobile cloud computing (MCC), has been introduced to be a potential technology for mobile services. "MCC integrates cloud computing into mobile environment and overcomes obstacles related to the performance (e.g., battery life, storage, and bandwidth), environment (e.g., heterogeneity, scalability, and availability), and security (e.g., reliability and privacy) discussed in mobile computing" [9]. MCC aims to move data storage and data process outside of the mobile devices. In other words, computing power and storage resources of cloud technologies are used instead of mobile devices. Thus, mobile applications become lightweight (both in CPU and memory usage), but more bound to the network. In a way, this turns mobile applications into mobile subscribers to the cloud services.

2.6 Related Work

Modeling human activities as a workflow and then guiding them with the opportunities of pervasive environments can be a milestone for managing the daily lives of people in various domains like health, office, home, travel. However, establishing such a framework has some complexities and distinctive features to be considered carefully. In order to achieve an efficient guidance framework, context information should be integrated, since context-awareness enables to offer services according to user and his

environment without explicit intervention, and also enables to automatically adapt to the changing contexts. Moreover, in their daily lives people are generally mobile, and usage of mobile device has a steady incline, thus providing a mobile application to users that allow them using such a framework anytime and anywhere is a facilitator feature for increasing effectiveness. Finally, communication of users with the framework should be handled different than the stationary users. Mobile application needs wireless internet for connectivity and wireless networks are unreliable which may cause performance and accessibility problems. There are some studies in literature to address these problems partially or fully.

One of these studies is MARPLE [25] which is a framework including a Mediation Center and a Mobile Engine. Mediation Center is a server based application which has a visual process modeler for modeling workflows, a repository for storing process and activity templates, a control module for communicating with mobile devices and assigning processes to them and a maintenance module for configuring mobile devices and installing necessary configurations to them. Mobile Engine includes a limited implementation of the ADEPT [6] workflow engine. MARPLE include the process model, fundamental correctness notions, correctness checks and dynamic adaptation concepts from ADEPT. However, the mobile engine only supports sequential operations, conditional routing operations and parallel execution of the process activities.

In MARPLE, activities offer form-based data input for entering information and web service calls. If someone wants to integrate a sensor input to an activity, an activity template designed specifically for this sensor is needed. In contrast to MARPLE, our framework offers many different context interaction methods. A sensor or a web service may be used as a data source, and they can send their data in a common format (XML or JSON) to the topics of the framework. Besides, user's mobile device is an important data source for the framework. Sensors in the mobile device (e.g. location), installed applications may send information about the status of the user. Also, mobile device can send user-generated data to the framework utilizing the keyboard, audio and video capabilities of the device. For instance, a user may send a picture about his activity and upload it to the framework by relating it with the context information. In other words, there is no need to implement additional software to integrate these data sources to the framework.

Another study, Sliver [16] has a small storage and memory footprint, uses Java APIs for supporting different operating systems and supports different communication protocols. It implements a subset of a BPEL-based workflow engine for mobile devices, and so does not support some of BPEL most advanced features (e.g. Serializable Scopes and Event Handlers). It does not fully support the workflow patterns specified by the Workflow Patterns Institution [22]. Sliver currently provides TCP/IP sockets for communication, and if a developer wishes to use another protocol, he has to develop new methods implementing communication methods of Sliver. Hence, there occurs a tight coupling between the mobile application, workflow engine and communication

protocols.

MARPLE and Sliver studies have the same problem due to the implementation of a portion of the workflow engine on the mobile device. A workflow engine on a mobile device consumes a significant amount of processing and memory capacity on the device. Besides, the development efforts of embedding workflow engine to the mobile devices end up with losing major capabilities of workflow engines such as extensive workflow pattern support and adaptation mechanisms. For this purpose, our framework does not include an engine on the mobile devices and separate the mobile devices and the workflow engine and enable publish / subscribe based messaging communication architecture between them. This also allows the framework not to be dependent on a specific workflow engine.

Pajunen et al also implements a BPEL-based mobile workflow engine which executes processes using WSDL interfaces and SOAP messages over HTTP [23]. BPEL has limitations for supporting dynamic nature of ubiquitous computing environments, so while the engine is focusing on process coordination to overcome these limitations, it lacks a mechanism for integrating physical elements [13]. In this study, they assume that this work is useful when network based services are not available or not necessary. However, we are developing a framework for pervasive environments, so we assume that the environment has network connectivity and there is a continuous interaction with the context. Moreover, one of the main reasons running workflows in mobile devices is stated to leverage the capabilities of other application in the mobile device, but it also causes tight coupling between workflow engine and mobile device. Our framework uses and benefits from these applications (e.g. use of a browser, map application, audio and video capabilities) without a workflow engine in a mobile device. Additionally, in our framework users can send resources (text, audio or video record) to the centralized framework, and these resources are kept for assisting another user while performing the same task.

Ranganathan et al also work on modeling daily activities using workflows and build a prototype using BPEL workflow language [26]. Although this study is similar to Pajunen's study, in contrast to Pajunen they enable to decide which service to be used in the workflow according to the rules they define in the task templates. However, tasks are related with web-services statically since BPEL only supports static bindings. For this reason, one has to define the references to the web service it will call before deploying the workflow. Our framework also makes use of services in the mobile device in addition to the services in the environment, and tasks and services can be dynamically related with each other.

In studies of Pajunen and Ranganathan, references to the web services have to be determined before deploying the workflow, which make lose the flexibility of workflows. On the contrary, our framework separates workflow definition, so it is possible to adapt the tasks within the workflows using the context information, even at run time.

Services and resources are offered according to users' context. Thus, the framework becomes more dynamic and responsive to the changes in the context. Another key point is that; our framework is built upon a petri-net based workflow language in contrast to these two studies. By this means, it is possible to support multiple users to engage in the workflow and also to offer alternative paths, cancellation paths or even new runtime path as long as workflow is designed properly.

Our framework extends the concepts presented in Presto, which is a pluggable software architecture for developing mobile workflow support in pervasive environments [13]. In Presto, users are offered different tasks according to their roles and context. For instance, when a student approaches a book in the library, a borrowing book task is offered to him. However, if a librarian approaches, placing book to the shelf task is offered. In addition to offering different tasks in different contexts, our framework also overrides tasks execution by changing the properties of same task according to the context at runtime.

Presto provides two operation modes: task-driven and object-driven. "In the object driven mode, the system senses the physical context first, and then proposes related tasks to the user. In the task-driven mode, the user explicitly indicates which task he or she is performing". Similarly, our framework implements task-driven mode by using the workflow's alternative path and new path capabilities. Also, in our framework it is possible to offer different tasks to the user according to their context similar to object-driven mode in Presto. On the other hand, our framework enables customizing the same task according to user's context at runtime. For instance, a task calling a web service to get information may reach different addresses according to the room that user is present (in other words, according to the user context). Another key difference is that the framework allows writing rules for cancellation, suppression, and completion of a task using the context information. Rules enable automatization of tasks without preventing user, and inform user about occurrence of events, and so ease performing the activities.

CHAPTER 3

PROPOSED FRAMEWORK

This chapter provides an overview and conceptual design of the framework. The framework has a layered architecture; in other words, it is separated into components that are allocated for a specific responsibility. There is not a tight coupling between components, and they communicate with each other through message exchanges. This allows ease of replacement of any component with another one providing the same functionality. By separating the framework into layers, any component of the framework can be modified or changed without reconstructing the entire framework.

Four main components that are utilized by the framework are displayed in Fig. 3.1, and they are explained conceptually in order to define their basic functionalities. Details of the implementation are given in Chapter 4. In section 3.1, WFMS is explained. WFMS is used to define and manage the activities considering the power of workflows in modeling the processes. Section 3.2 gives details of the Coordination Management System (CoMS) which is developed for handling the context information associated with users and taking advantage of the opportunities that are provided by pervasive environments. Section 3.3 introduces the mobile application developed in order to enable users to ubiquitously interact with the framework components. In section 3.4, Messaging System is discussed to set up a more appropriate communication infrastructure between the mobile application and the CoMS.

3.1 Workflow Management System

General usage of WFMS in business environments proceeds as follows; workflows are defined in a static source file by using a script language or a data structure type, and then, these files are loaded to a workflow engine in order to execute these steps in the flow, and so tasks in the processes become organized. By this means, while working on the procedures that have to be performed in a strict order according to the defined regulations, WFMS enables the integrity of the process since it secures that the whole steps in the process will be executed or at least will be evaluated for the execution. Ability to define the process and to organize the execution of the activities

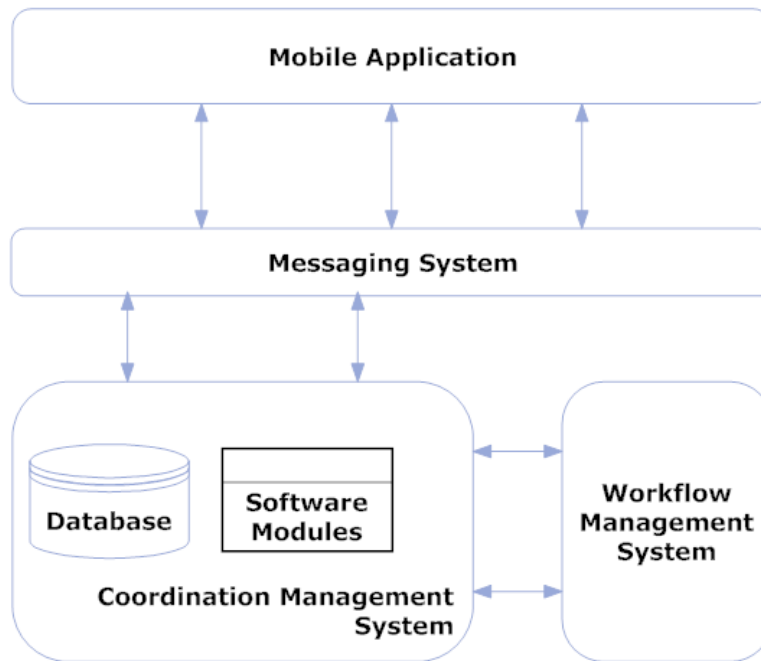


Figure 3.1: Framework Conceptual Architecture

are the primary considerations that we intend to use a WFMS. It is straightforward and more reliable to put business logic (i.e. management of the executions) into a central mechanism (i.e. WFMS) where its efficiency is proven during the usage in the business environments for years.

WFMS may be resembled to a navigation device that guides users. A navigation application assists the user to which road he should use step-by-step. In a similar way, WFMS assists the user about its current task and order of the task, and more significantly, it also handle the data flow between the tasks, distributes tasks to users, and provides some advanced patterns for complex processes. Thus, if a process is modeled and the related workflow is loaded to WFMS, users can perform the tasks even if they are inexperienced about the process, like arriving at the target location by the help of a navigation application, but in a more assistive and advanced way.

Nevertheless, assisting users in their daily lives is not as straightforward as routing a car, so stand-alone usage of WFMS is not enough for managing processes in mobile and pervasive environments; since, providing personalized assistance to a mobile user has challenges in such environments [25]. For this reason, we build some extra components that enable the utilization of WFMS. Tasks should be adapted according to a specific user or environment since static (pre-defined) tasks are not favorable in pervasive environments. Tasks definition should change according to the context that user belongs, even more according to the user itself. For instance, when a user enters a room, he should dynamically interact with the devices in that specific room, not

with devices in that room that is statically defined before. Because, it is not applicable to define a new workflow for each context (location, person). In order to enable integration with the pervasive environments, the ability of reaching and using context information should be increased, so the feasibility of workflows for mobile users in pervasive environments is satisfied. For this purpose, as depicted in Fig. 3.2, workflow task concept is upgraded into a higher level framework task that wraps workflow task definition, task context information, task properties (e.g. the location of a task), and task supplementary resources (i.e. text, picture, audio and video record about a task). This also enables customizing workflows. In other words, in our framework, a task gains a new meaning in addition to its workflow definition. Therefore, same tasks of same workflow are not identical anymore if they are instantiated in different contexts, although their workflow definitions are same.

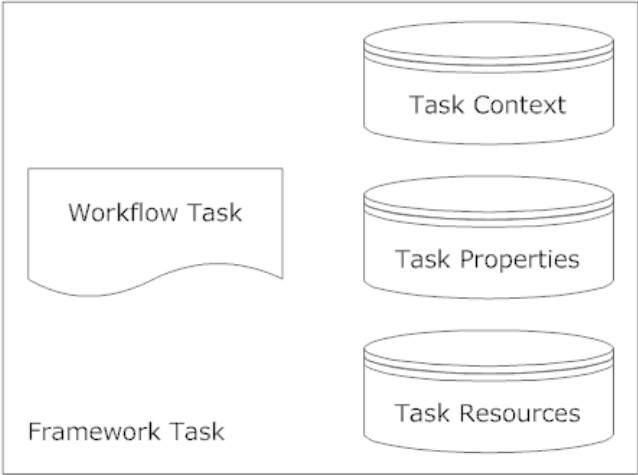


Figure 3.2: Framework Task Definition

3.2 Messaging System

CoMS and Mobile Application are two components of the framework which constantly need to exchange data through a bi-directional communication channel. User interacts with framework using a mobile application, and these interactions are responded by CoMS and related response data are transmitted to a user. Therefore, clients are in need of a communication tunnel that should not be established for each interaction request; since, it wastes the usage of network, CPU and battery sources. Also, the communication tunnel must be suitable for both way data transmission without blocking each other. In other words, CoMS and mobile application have to be able to send data to each other at the same time.

Furthermore, messages should be received by mobile application in a push-based way, it should not constantly poll a medium to get new messages or notifications. Also,

mobile application may receive a message (notification) without making a request. For instance, mobile application may pop up a notification to warn the user without continuously making a request (or even without a request) to the server. Indeed, having push-based infrastructure is a key factor for enabling a real time communication system. User will be aware of a notification and get information as soon as framework sends it. Satisfying this requirement with a polling is an unfavorable design choice since it puts extra (and unnecessary) burden on the network and CPU usage (so decreases the battery life).

For this purposes, Messaging System component is set up. It makes use of a topic-based publish/subscribe protocol that also minimizes network and resource usage by providing bi-directional push-based messaging infrastructure. Fig. 3.3 displays some cases how messaging may occur between mobile application and CoMS. Messaging System also enables loose coupling between framework components since they communicate via messages (in a well-defined data structure) and are not dependent on the inner functionalities of each other. Sending and receiving messages are the only ways that they can interact.

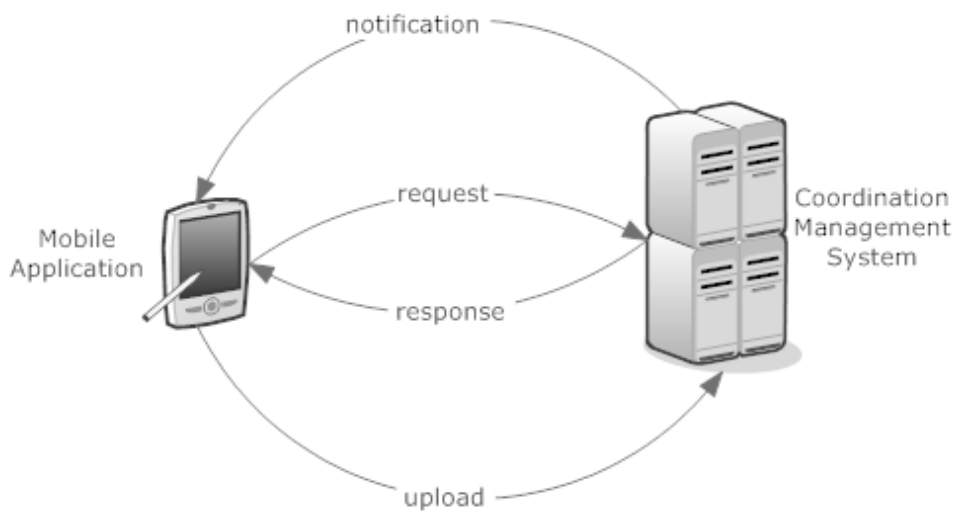


Figure 3.3: Sample Messaging Cases

Moreover, topic-based systems are good at scalability since communicating parties do not have to know exact addresses (IP address) of each other. They interact according to their interests (topic addresses). In other words, more than one sensor may send a data just sending a message to the related topic of the framework. Similarly, just by sending a single message, the framework may transmit a message to more than one clients that are subscribed to the related (and same) topic. This characteristic gains more importance when working in a pervasive environment where there are great numbers of sensors, devices to interact.

3.3 Mobile Application

While performing their personal activities, people are usually mobile, and they are away from their personal computers. However, nowadays majority of people are using handheld devices (e.g. tablet, PDA, smart phone) to achieve counterpart features of a personal computer. This framework is designed for guiding users in pervasive environments, so handheld devices are key enabler for efficiency and effectiveness of the framework. Especially, smart phones are more significant for the framework due to its widespread usage among people, and built-in hardware features and software applications.

Sensors, camera and audio capabilities, installed applications (e.g. Map, QR Reader) are some of the features that enable smart phones to assist people while they are performing their tasks. Knowing context information is important in order to relate this context information with user's tasks. By this way, it becomes possible to propose an advanced framework rather than a framework in which users simply complete the tasks in their lists. To illustrate, a sensor on the mobile phone may inform the framework about the current status of the user, such as a GPS sensor sending the location of the user to the framework. Or, a user may manually send information about a task, for instance he may fill a form that determines the execution of the activity. Correlatively, framework may also inform the user and the user may receive assistance by his mobile phone while he is trying to complete a task. A map application may navigate a user to arrive at the location of his current task (current location of the user is gotten from mobile phone, and target location is get from the framework) as displayed in Fig. 3.4.

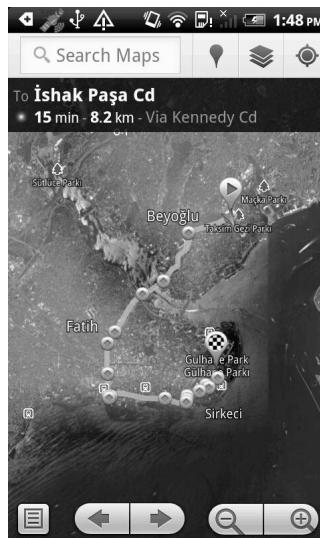


Figure 3.4: Navigation Screen of an Installed Application

Another advantage of a mobile phone is that it can be used as a medium for enriching the tasks with supplementary resources uploaded by users. Users can send data (text,

image, audio or video) to the framework, and these data are related with a task as a supplementary resource. Then, these data may be used both by a user to review his activity and by another user to get experience about that task. For instance, a user may upload a picture about his task to further use it in his research. Or, a user may upload a video record displaying how to set up the connections of a device that will clearly make it easier for other users who will perform the same task.

In addition to utilizing the features of smart phones, ability to communicate with the framework whenever and wherever necessary is the other motivation for the development of a mobile application. By using the application, users enjoy the opportunities of pervasive environments and benefit from the framework’s capabilities as long as there is an internet connection. In other words, the mobile application serves as a bridge which connecting functionalities of the framework to the users.

3.4 Coordination Management System

In order to coordinate the execution of workflows, the communication between the user and the framework, and the context data transfer, there is a need of a central controlling mechanism. For this purpose, CoMS is implemented which manages overall execution in the framework via four modules (displayed in Fig. 3.5) it incorporates: Workflow Coordinator Module, Communication Module, Context Module and Automation Module. The separation into four modules provides less coupling and more cohesion between framework components. Thus, when a change in one technical medium (workflow engine, database system, messaging infrastructure) occurs, it does not affect the whole system and only related module has to be adapted. Also, gathering related functions inside the same module makes it easier to develop new functionalities, and to understand the programmatic structure of the framework.

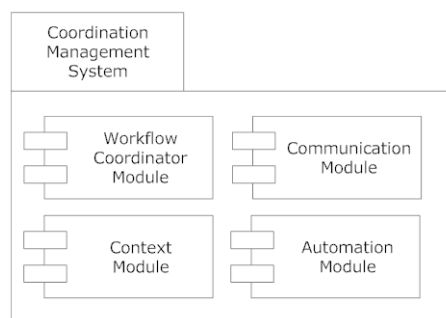


Figure 3.5: Coordination Management System Modules

The Workflow Coordinator Module handles data transfer related about workflow tasks among WFMS and other modules, and executes operations to manage these tasks. This module incorporates essential functions like learning information about a specific

task, getting the list of all tasks of a specific user, or giving input about a task which effects the execution of the workflow. Also, this module presents executive functions to complete a task of the user, launch a new workflow for a user.

As it is explained in messaging system section, communication infrastructure of a framework for pervasive environments especially with a mobile client needs an extra attention. It should be treated as an important part of the framework, and so it should be handled in its own module. Communication Module handles the data transfer between user, CoMS and environment. To state more clearly, it is responsible for processing and delivery of messages to the right addresses. However, setting up a communication infrastructure for mobile clients in pervasive environments is not straightforward; this module should satisfy the specific requirements to conduct the messaging infrastructure of a client having a restricted power and resource capacity, and in an environment which probably has an unreliable network.

In order to increase the effects and usability of the framework, workflow task should be improved by integrating context information and supplementary resources into their definitions. Knowing the context information of tasks enables accessing the functionalities and interacting with various devices in the environment and resources in the internet. Some of these interactions are as displayed in Fig. 3.6. For this purpose, Context Module is implemented to enhance the WFMS task concept using the data stored in an attached database. By this means, tasks are customized with regard to their context information and so they are uniquely identified in their own contexts. Accordingly, supplementary resources (image, audio, video, and text) may be managed with respect to this unique identifier. In other words, users of the framework are also context resources for the framework and they can contribute and enrich the tasks by entering a text comment about a task or uploading a picture explaining how to perform a task.

Further significant goal of the framework is to manage the execution of the personal activities automatically without preventing users. For this reason, whenever possible, it is required to complete a task automatically by evaluating data coming from environment, user, mobile phone. Thus, Automation Module that enables to define rules and then to execute these rules and at the end to take related action is implemented. Automation Module keeps track of and executes the defined rules according to the data collected from various sources in the environment. Rules are used to infer an outcome using the collected data. Accordingly, a task may be completed or suppressed, and so execution of an activity will have been automated without preventing user. To illustrate, by evaluating the collected data, this module may decide that following task is not applicable or not required to be performed in the current context. Consequently, it automatically suppresses that task and the next task in the queue is offered to the user.

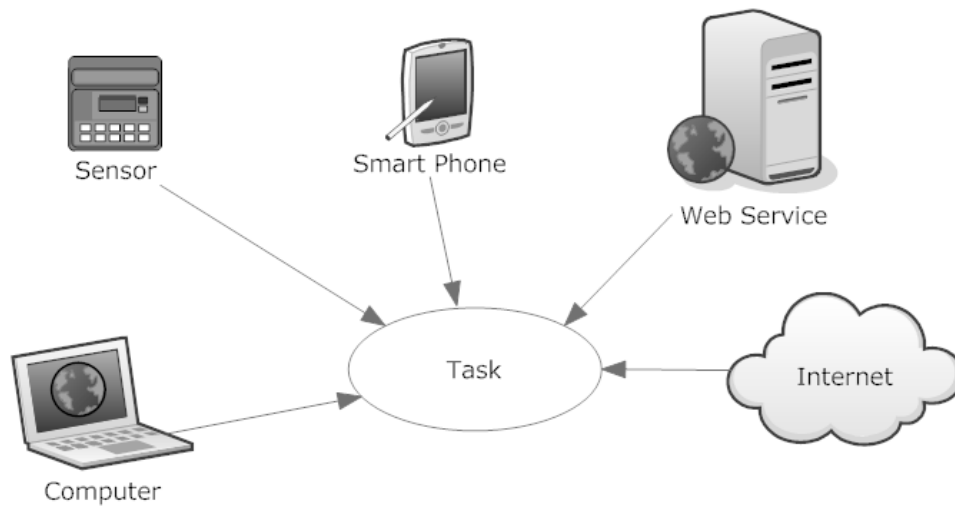


Figure 3.6: Context Task Relation

CHAPTER 4

PROTOTYPE IMPLEMENTATION

In this chapter, the prototype implementation of “A Workflow-based Mobile Guidance Framework for Managing Personal Activities” will be described and explained. Components of the framework and their interactions are displayed in Fig. 4.1. This prototype is built in order to demonstrate the feasibility of the framework, and it is the realization of the general framework architecture that is already presented in Chapter 3 and explains the conceptual description of the proposed model.

First component in the architecture is implemented by using the YAWL (Yet Another Workflow Language) engine, which is an open source workflow engine with fully-featured workflow modeling environment and execution capabilities [33]. Second, we develop software modules by using Java programming language in order to handle workflow, messaging, context and automation operations. Third, in order to provide ubiquitous access to the framework by users, a mobile application is developed on the Android OS. Finally, we set up a communication infrastructure based on the MQTT (Message Queue Telemetry Transport) protocol, which is a topic based publish subscribe system enabling lightweight and asynchronous bi-directional connection [18].

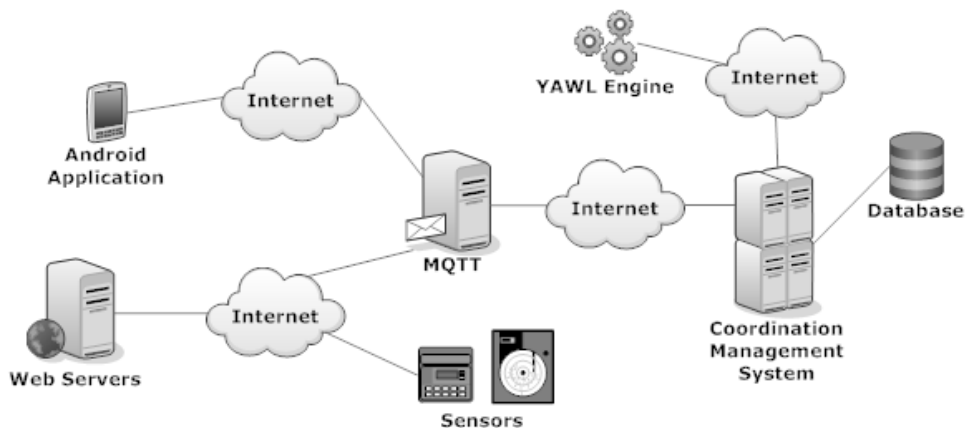


Figure 4.1: Implementation Architecture

4.1 Workflow Management System

In addition to its well-known benefits, WFMS is also an important component for our framework, owing to the fact that workflows are very successful in simplification and customization of user interactions with the pervasive environments [26]. Pervasive environments include a vast number of devices, sensors, computers to communicate and presents various services to be called. The management of communication and execution of these in such an environment is a compelling process without a central organizing mechanism. Thus, at the center of our framework, we utilize a WFMS.

Among various available WFMSs, petri net based systems are more suitable for our purposes since they have the ability to display a graphical representation which makes it easier to understand the process, and they are used in formal analysis, verification and validation of the model [40]. Moreover, petri-nets are more expressive since they support a large number of primitive functions to model a workflow. It supports all routing possibilities (sequential, parallel, conditional and iterative) paths to construct a workflow definition [31].

YAWL, developed in an academia, is a WFMS which is based on petri nets; However, it interests both academic and business environments due to its easy and flexible modeling environment [33]. It is an open source project in contrast to most of the other WFMS so it can be extended according to the particular needs. Additionally, it supports all workflow patterns specified by the Workflow Patterns Institution (displayed in Fig. 4.2) [22] which enables the potentiality to model a large number of real life processes. Also, YAWL supports practical service APIs to manage workflow operations. Therefore, we utilize YAWL for managing the workflows in the framework.

In our framework, we use the latest version (currently) of YAWL implementation which can be downloaded from the website of YAWL located at <http://www.yawlfoundation.org/>. It is an open-source project and is licensed under the terms of Lesser General Public License (LGPL) conditions (details and conditions of the LGPL license are available at http://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License).

Detailed information about usage of YAWL in our framework is given in Appendix A

4.2 Messaging System

Representational State Transfer (REST) is a software architecture style designed for distributed systems. Since its emergence, it dominated the data transfer in web. As creating the loosely coupled services in the web, REST enables the reuse of these services. For this purpose, embedding web servers (employing REST architecture) into the devices is necessary. REST uses HTTP for its application protocol. HTTP provides a Client-Pull interaction model in which client-initiated connections are established to

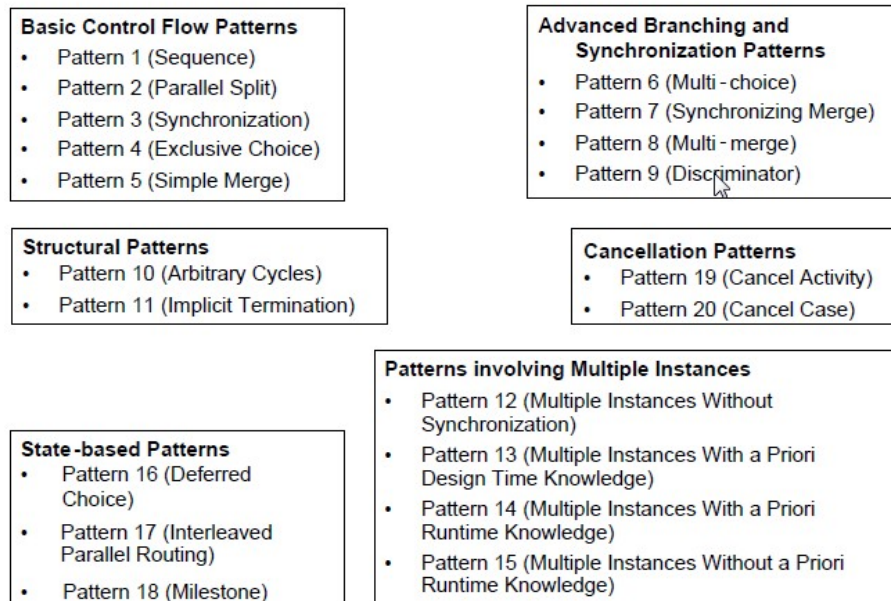


Figure 4.2: Workflow Patterns

the servers. However, this model is not fully compatible with our framework goals since it is not well suited for event-based systems and does not support asynchronous bi-directional connection [15]. In our case, while the client is sending an event from a mobile application and waiting for the response, simultaneously a notification or data must require a response by the client without a polling mechanism.

For this purposes, there are various researches in order to enable push based communication. In push based interactions, requests are initiated by the publisher or server and where data are asynchronously transmitted to the clients as soon as it is produced. The most common and appropriate technologies are developed using Publish/Subscribe based Messaging infrastructure. Thus, it is possible to connect with the objects that we use in our daily lives. One of these researches is MQTT which is a publish / subscribe based messaging protocol invented by IBM workers.

Our framework uses MQTT in order to regulate the data transfer among the components of the framework and the context devices. Unlike the HTTP protocol, MQTT is connection oriented in which the client sets up a connection to the broker before subscribing to and publishing any data, then this connection enables lightweight and asynchronous bi-directional communication. Thus, MQTT enables push-based communication which reduces network traffic and is able to run with limited processor and memory resources [19]. Moreover, implementation of an MQTT client is simple and the complexity resides on the server side, hence it is suitable for developing mobile applications.

In the architecture of the messaging framework, there is a need of central mechanism that is responsible for distributing the message between clients. Multiple clients may subscribe a topic to get data sent by publisher. Also, multiple clients may publish data to the same topic. In MQTT terminology, this central mechanism responsible for distributing the messages, keeping subscriptions is referred as Message Broker. In our framework, we use the IBM RSMB (Really Small Message Broker) component for transmitting data in the form of messages to and from applications and devices over TCP/IP network connections [20]. Communication between the framework components is depicted in Fig. 4.3, and details of the implementation will be given in CoMS section.

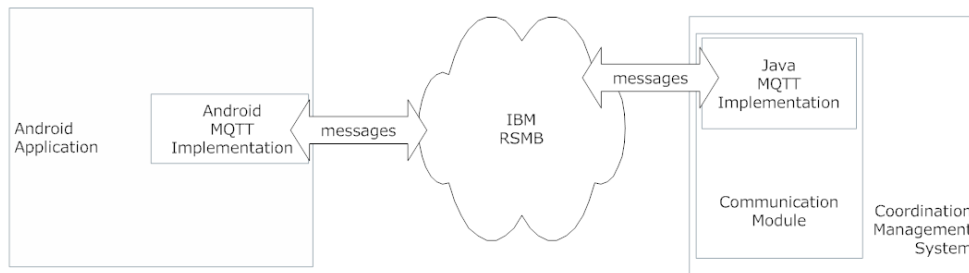


Figure 4.3: MQTT Framework Architecture

Client implementations in mobile application and CoMS are able to communicate with each other just sending messages to the related topics. MQTT supports agnostic data type which enables transmission of any type or content data; in other words, it is possible to send just a single character byte, or megabytes of graphics data [19]. Broker, mobile application and CoMS are decoupled from each other; by this means, any of them may be replaced with correlating component. For instance, it is straightforward to integrate a new mobile application, developed in another platform, into the framework. However, they should communicate with same data type standards in order to understand the messages. For this purpose, messages are built according to the pre-defined XML schemas. Definitions of Request and Response messages are displayed in Fig. 4.4.



Figure 4.4: Messaging XML Schemas

A sample request and response data are given in Fig. 4.5 and Fig. 4.6. Firstly, mobile application sends a request to learn the list of the offered tasks (work items in YAWL terminology) belongs to the currently logged user to the mobile application. Then, CoMS sends a response of the related request including the list of the offered tasks as

data in XML format. Lastly, mobile application turns XML data to Java Object, and it is rendered to the mobile application screen.

```
<request>
  <id>1354312760807</id>
  <name>OfferedWorkItemList</name>
</request>
```

Figure 4.5: A Sample Request Message

```
<response>
  <id>1354312760807</id>
  <name>OfferedWorkItemList</name>
  <value><![CDATA[
    <workitemrecords>
      <workItemRecord>
        <id>13:register</id>
        <specversion>1.5</specversion>
        <specuri>SimpleMakeTripProcess.ywl</specuri>
        <caseid>13</caseid>
        <taskid>register</taskid>
        <taskname>register</taskname>
        <enablementTime>Dec:01, 2012 0:01:43</enablementTime>
        <firingTime/>
        <startTime/>
        <completionTime/>
        <enablementTimeMs>1354312903690</enablementTimeMs>
        <status>Enabled</status>
        <resourceStatus>Offered</resourceStatus>
      </workItemRecord>
    </workitemrecords>
  ]]>
</value>
</response>
```

Figure 4.6: A Sample Response Message

In order to pull data from CoMS, the mobile application sends a Request data and CoMS processes this request and sends a Response data via MQTT. It is possible to replace the XML structure with other data formats such as JSON. However, since the YAWL Interfaces offer XML based responses, we also prefer XML as the communication data structure for compatibility purposes.

MQTT uses character strings to provide support of hierarchical topics. It also supports two types of wildcards. + is wildcard for a single level hierarchy, and # is wildcard for all remaining hierarchy levels [19]. Subscribed and published topics in our framework are depicted in Fig. 4.7. Mobile application is subscribed to /mobile/clientId and publishes to /proxy/clientId where clientId is a unique number that is identifying each

smart phone. On the other side, CMS subscribes to /proxy/+ where + is a wild card that enables to subscribe any topic just one level below the hierarchy of proxy topic. In other words, by subscribing the /proxy/+ topic, CoMS are able to get messages from topics such as /proxy/1234, /proxy/4567. Then, CMS publishes to respective /mobile/clientId address according to the topic address where message comes from.

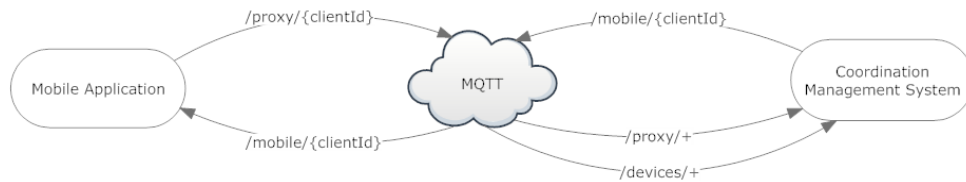


Figure 4.7: MQTT Topics

4.3 Mobile Application

Users need a mobile application through which they ubiquitously reach the framework, see their current tasks as well as instructions and information about these tasks, and can be notified when necessary. For this purpose, an Android OS based application, which communicates with the CoMS component, is developed.

In the development of an Android application, there are two fundamental component types to be considered: Activity and Service. Activities are used for designing screens with a user interface. Each activity represents a single screen. For instance, Login Screen is designed and implemented via an Activity that users interact and send their credentials. Secondly, Services are components that are running in the background and generally used for handling operations that have long-running time or have to run through the application’s lifecycle. For instance, a Music Service may play songs in the background without intervening user or the application’s other screens.

In our mobile application, Service components are used for creating seamless communication to the framework components by using the MQTT protocol. Services are responsible for implementing MQTT messaging operations: publishing data to a topic, subscribing a topic, and listening messages from a topic. For this reasons, applications uses two implementation classes: MqttPublisherService and MqttSubscriberService. Both services are instantiated from the same base class MqttService. Class diagram displaying the inheritances and properties of MqttService is displayed in Fig. 4.8.

MqttService extends Android Service class in order to be developed as a Service running in the background. It also implements MqttSimpleCallback class inside “wmqtt.jar” in order to be registered to be able to receive messages using MQTT Protocol. “wmqtt” is a collection of MQTT Java implementation classes that are provided by IBM. Usage of MqttService class is straightforward. Firstly, “connectToBroker” method is called,

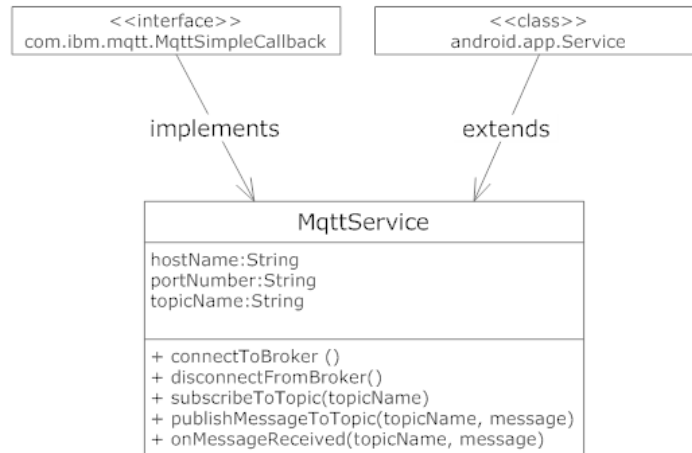


Figure 4.8: Application MQTT Services

so a communication tunnel is built from client to the broker. Then, MqttPublisherService use “publishMessageToTopic” method in order to send data to the related topics. On the other hand, MqttSubscriberService has to call “subscribeToTopic” before starting to listen the topic. After this, when a new message is published to the topic, “onMessageReceived” method is called. Thus, Activity classes using the MqttSubscriberService override this method in order to implement their logic.

Services are created in the global context of the Android application. In our application context, MqttPublisherService and MqttSubscriberService are created. Thus, every Activity class in the application does not create its own Service; indeed it does not. By this means, the service class in the global context should be injected to the local context of the activity’s context. However, it is not straightforward. In order to do this, Service class must be bound to the Activity class using the code snippets in Fig. 4.9.

Also, Activity classes using MqttSubscriberService has to implement a “Receiver” class in order to get messages from the related topic. Basic implementation of a receiver class is displayed in Fig. 4.10.

Instead of implementing to each of these functions and classes in every Activity classes. A base class is developed which extends Android Activity class. It presents simple methods to send and receive messages. Activity classes should override onMessageReceived method and implement their logic. Also, it handles binding and unbinding processes to the Services by using onResume and onPause methods. Then, it also makes code simpler and more robust. Moreover, it also decouples Activity class from the MQTT Implementations. Structure of a base Activity class is displayed in Fig. 4.11.

Since MqttPublisherService and MqttSubscriberService are using different topics, two instantiations of the same class is needed. Topic relations are displayed in Figure X.Y.

```

bindService(new Intent(this, MqttPublisherService.class), new ServiceConnection() {
    public void onServiceDisconnected(ComponentName name) {
    }

    @SuppressWarnings({ "unchecked", "unused" })
    public void onServiceConnected(ComponentName name, IBinder service) {
        MqttPublisherService publisherService = ((tr.metu.androidappl.service.
            MqttPublisherService.LocalBinder<MqttPublisherService>) service)
            .getService();

        // implement logic

        unbindService(this);
    }
}, 0);

```

Figure 4.9: MQTT Publisher Android Service Binder Code Sample

```

public class MQTTMessageReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundleData = intent.getExtras();
        String message = bundleData.getString("MqttMessage");

        // implement logic
    }
}

```

Figure 4.10: MQTT Subscriber Android Service Binder Code Sample

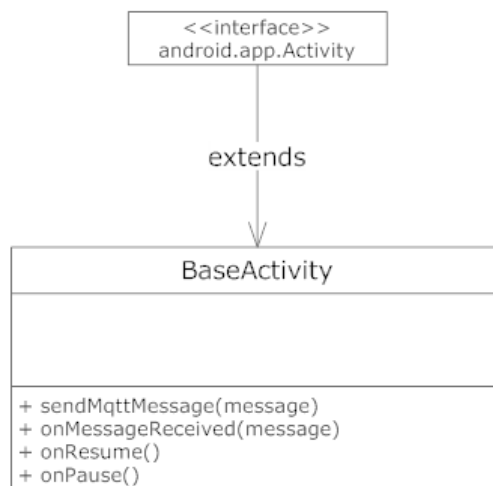


Figure 4.11: Application Base Activity

in Messaging section. `MqttPublisherService` publishes messages to `/proxy/clientId` topic, on the other hand, `MqttSubscriberService` are responsible for receiving messages coming from `/mobile/clientId` topic.

The mobile application is the presentation layer of the framework which provides user interfaces to manage workflow operations, assist a user when performing activities and bi-directional data transfer between client and framework. At first, mobile screens that offer essential operations of WFMS as if using native user interfaces of WFMS are developed. For instance, in Fig. 4.12a, main application screen is displayed in which the user can reach the functionalities about his tasks, or launch a new workflow. In Fig. 4.12b, list of the workflows loaded into the WFMS is displayed, and user can choose one of them by touching and then launch it. Lastly, in the screen in Fig. 4.12c, user can perform start, complete or edit operations, or upload data about the task. List of resources (Comment, Picture, Audio Record, Video Record) uploaded by users or workflow designer is displayed.



Figure 4.12: Mobile Application Screens for Workflow Operations

Additionally, mobile application utilizes smart phone capabilities to assist the user when he is working on his activities. Sensors (e.g. NFC, gyroscope), embedded hardware (e.g. Camera, Audio), and the installed applications (e.g. Map, Browser) are the important features of a smart phone. For instance, a user may use an NFC sensor to complete a task, or a user may upload text, image, audio content to the framework related to a task, or a navigation application may be used to route the user to the location of the task. Utilization of smart phone features will be explained in detail in Chapter 5 where execution of the framework in various domains will be illustrated.

Besides, mobile phone is also a context resource for the framework. They are becoming more and more powerful every day, and becoming to have nearly all senses as humans, but differently than us, they are also able to augment them. This is the reason modern mobile applications differ from web applications, and so this gives an opportunity to offer applications in domains like interactive games, location based applications, healthcare, logistics. However, in general mobile phone is mainly used to find the current location of the user and so his context is determined. Also, other sensors (temperature, orientation, altitude, light, humidity) in the phone give information about users' current situation which enriches the context information and makes possible to have more reliable inferences about them.

4.4 Coordination Management System

CoMS incorporates four modules (Workflow Coordinator Module, Communication Module, Context Module and Automation Module) which manage the overall execution of the framework and interactions among the framework components (WFMS, Messaging System, Mobile Application and CoMS). The combination of these modules, implemented by using Java programming language, constitutes the core component of the framework and provides the coordination of all the operations in the framework.

4.4.1 Workflow Coordinator Module

Workflow Coordinator Module provides a “WorkflowManagementService” class which includes methods to facilitate interactions with YAWL engine. A class diagram including its properties and methods are displayed in Fig. 4.1 in order to give an idea about its usage. Indeed, WorkflowManagementService uses ResourceGatewayClient and WorkQueueGatewayClient classes of “yawl-lib-2.3.jar” for these interactions. This jar is distributed with YAWL installation or can be downloaded from the web site of Yawl Foundation.



Figure 4.13: Workflow Management Service Class Diagram

ResourceGatewayClient presents an interface for retrieving information about users. Roles, groups, capabilities of users are gotten by using this class. It is required because before offering, allocating a task to the user, there is a need of additional information more than user's name or identification number.

WorkQueueGatewayClient presents an interface for performing operations about tasks and workflows. For instance, list of the offered work items may be retrieved by calling "getTaskList" method with an "Offered" task status. Other functionalities are also displayed in Fig. 4.1 (excluding some of the internal methods implemented for helping these methods).

With the help of these interfaces, it is possible to control the YAWL engine by sending XML requests over a HTTP connection. Then, responses are returned again in XML format, and WorkflowManagementService converts them to the Yawl objects for usage in the framework. While other modules are communicating with the workflow engine, they use the methods offered by this service. Thus, in case of replacing workflow engine with another one, it is enough to implement only this service without effecting other modules or components.

4.4.2 Communication Module

Communication Module incorporates a Java MQTT client implementation which subscribes and publishes to the related topics that are determined for communication channels. MQTT implementation is based on WMQTT IA92 Java Utility which is part of a collection of Java implementations of MQTT client APIs which enables connecting to the broker from a local or remote machine by supplying necessary parameters (e.g. IP address, port number). Class diagram in Fig. 4.14 displays the general class hierarchy concept and usage of MQTT in the CoMS.

SimpleMqttClient implements IBM MqttSimpleCallback class in order to be registered to be able to receive messages using MQTT Protocol (as developed in Android application). Then, ProxyClient class is developed by extending SimpleMqttClient class, so have necessary MQTT attributes and methods to use MQTT protocol. ProxyClient also includes methods to interact with other modules in CoMS. For instance, in order to communicate with workflow engine, it encapsulates a WorkflowManagementService class.

MqttProxy is the central MQTT class that manages the communication topics between the mobile application and the CoMS. To explain more precisely, when a mobile application sends a message to proxy topic (e.g. /proxy/clientId) first time, MqttProxy instantiates a ProxyClient for this mobile device, and subsequent messages are forwarded to the related ProxyClient object by MqttProxy. By this means, in addition to keeping mapping of topics, MqttProxy also handles the situations that connections

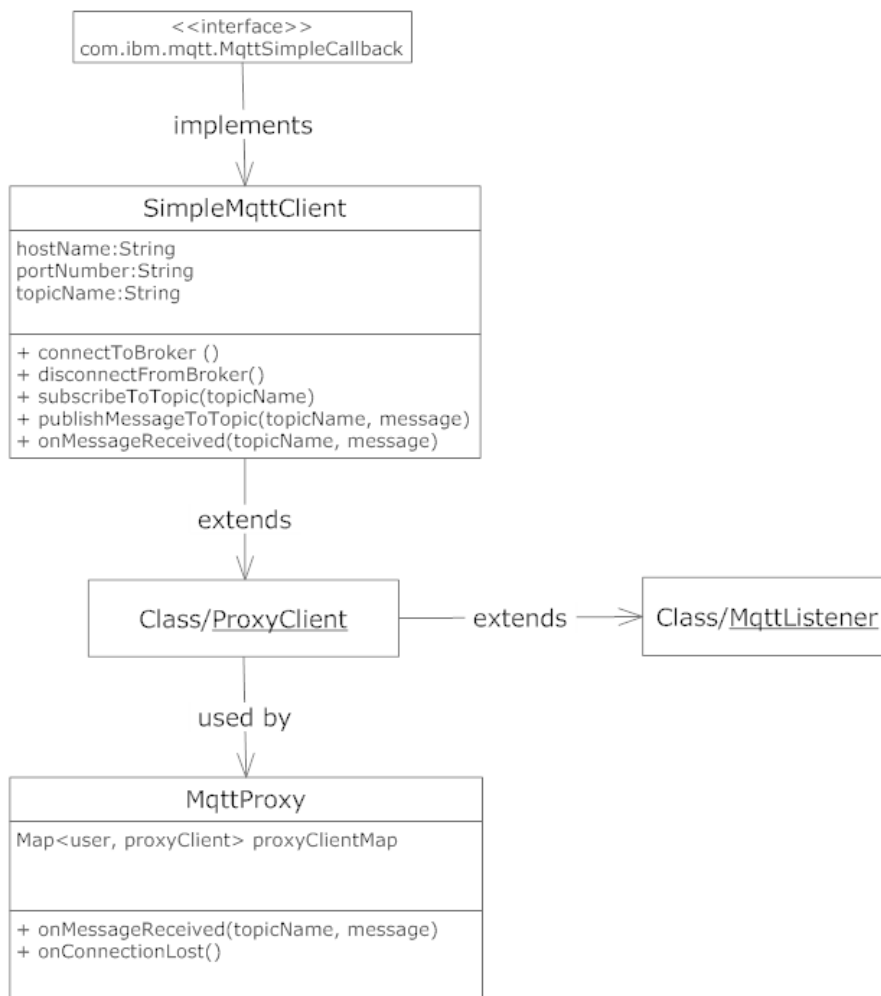


Figure 4.14: Communication Module Architecture

to the MQTT broker are lost. Moreover, a central mechanism that may collect all messages coming to CoMS may be used for further analysis to evaluate the framework and make researches and improvements about the execution of the framework.

MqttListener class is developed in order to listen messages coming from environment (sensors, devices, web services), and then relevant actions are taken. Execution principle of MqttListener will be explained in Automation Module section.

4.4.3 Context Module

In standard WFMS approach, workflows are designed into static files, so tasks in that workflow becomes also static. However, workflows that are designed for pervasive environments should be dynamic and adaptive to the context changes. For instance, a task calls a web service to learn the schedule of departures in a train station. If a web service address is embedded into the workflow definition file, then there is a need of a unique workflow for each station, and even worse users have to choose the correct version of the workflow in a crowded workflow list. Thus, it is not suitable for pervasive environments. Instead, Context Module keeps specific data about tasks according to their context information, and when a user asserts his context (manually or automatically), this specific information are injected to the task definition. In other word, Context Module enables upgrading the task concept in WFMS.

Hence, collection of data should be kept in an organized state, and it should be quick and easy to modify and retrieve the data. Since, DBMS can satisfy these requirements, Apache Derby Relational DBMS installation is integrated into the framework due to its small footprint and compatibility with SQL standards. Also, it is an open-source project. Its installation files and manual can be downloaded from its website (<http://db.apache.org/derby/>).

In order to comprehend the data structure of the framework's database, Entity-Relationship (ER) diagram is depicted in Fig. 4.15. Each entity will be explained in detail by also giving example records.

Firstly, Task is the main and most important entity in the database. It is used to upgrade the task concept in the WFMS. Tasks are defined with their names and contexts. In this prototype implementation, tasks are identified according to the single level of context hierarchy for ease of implementation, but it is possible to increase the number of parameters specifying the context of the task. For instance, in a workflow definition, there exists a task named as "SensorTask" which is designed to listen a sensor data in the room. In order to uniquely identify this task in a different context, our framework keeps its records with its location information (e.g. Sensor Task in Room A, Sensor Task in Room B) as in Fig. 4.16.

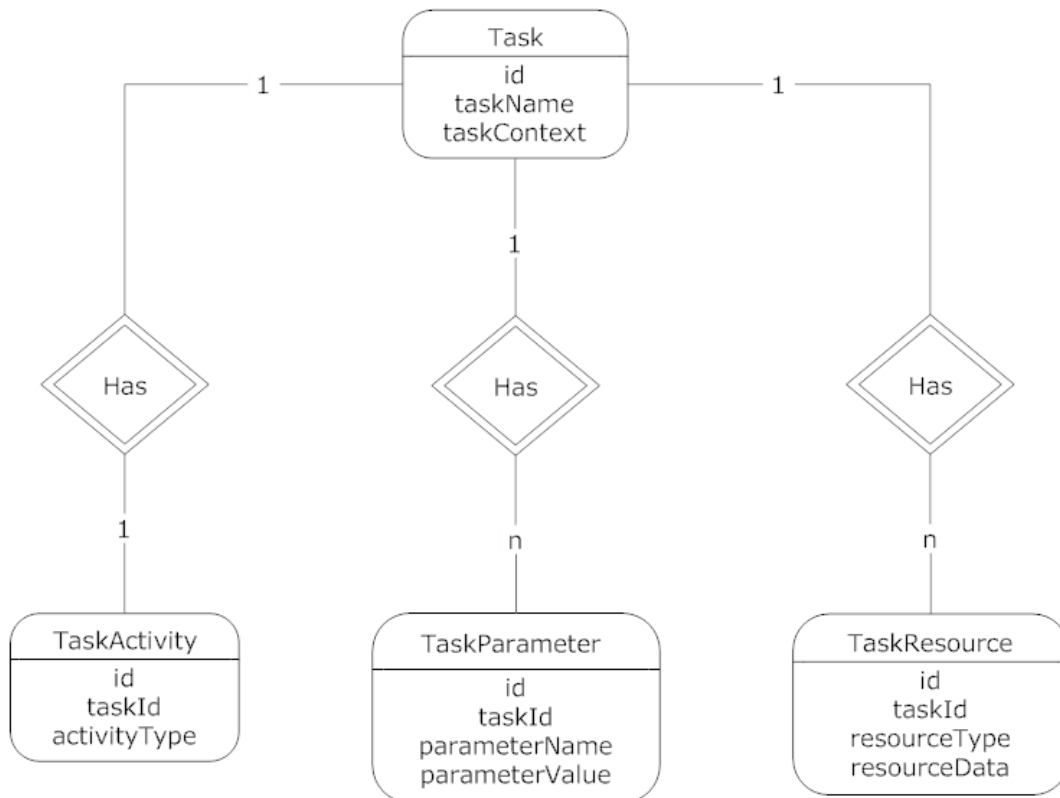


Figure 4.15: Context Module ER Diagram

| Task | | |
|-------------|-----------------|--------------------|
| <u>id</u> | <u>taskName</u> | <u>taskContext</u> |
| 1 | SensorTask | RoomA |
| 2 | SensorTask | RoomB |

Figure 4.16: Sample Task Records

Secondly, in our framework each task is associated with a special type of activity to assist users. This association is kept in TaskActivity table. Some of the specific activity types (implemented so far) are listed in Fig. 4.17. For instance, each of SensorTask is recorded with Sensor activity type. The tasks for which no explicit task type is given treated as a default task type. Task activity types give only extra assistance to users differently than the default types. To illustrate, “Navigation” type assist the user to arrive at the location of the task by displaying a map and routing information on the screen.

| Task Activity Types | |
|----------------------------|---|
| Nfc | reads and writes to NFC tags |
| Scanner | scans QR code or Barcode |
| Sensor | listens sensor data |
| Navigation | navigates user to the task location |
| WebService | calls a webservice and displays response to the user |
| WebView | displays a website in a browser |
| Weather | gets weather forecast reports in the location of the task |

Figure 4.17: Framework Task Activity Types

Accordingly, customized tasks carry unique identifiers, and we are able to insert and update features of tasks at run time. For this purpose, “TaskParameter” entity is created. For instance, the topic address of the SensorTask is assigned in this table, and it can also be modified while workflow is also executing. In a similar way, location of the task may be inserted into the TaskParameter table (with its latitude and longitude information) as in Fig. 4.18, then in addition to routing him, application also becomes able to notify the user that he arrived at the location of the task, and even automatically complete this task (automatic completion will be explained in Automation Module section).

| TaskParameter | | | |
|----------------------|---------------|----------------------|-----------------------|
| id | taskId | parameterName | parameterValue |
| 1 | 3 | lat | 39.892394 |
| 2 | 3 | long | 32.778101 |

Figure 4.18: Sample Task Related Parameter Records

In addition, the customization of tasks also allows keeping additional resource (associated with tasks) recorded by workflow designer or users. Prototype implementation enables 4 types of resources to be uploaded to the framework and to be displayed to the user in the application screen. These are Text, Image, Audio and Video. These resources may have information about how to complete an activity or what to pay attention while performing an activity, so they may be very helpful while performing

tasks. For instance, a user may upload a photo that shows how the cables of a projector device should be connected to the computer while setting up a presentation room.

Keeping user generated contents enables the user to interact and share information with each other. This provides a social collaboration perspective to the framework. Also, it increases the efficiency and usability of the framework with the advanced assistance in contrast to the standard workflows.

4.4.4 Automation Module

An important feature of the framework is the capability of managing the status of tasks without user intervention. This can be achieved by automatically completing a task, or suppressing a task in the flow. Indeed, it is the essence of user assistance that performing much and more qualified work within a shorter period of time by putting in lesser effort.

For this purpose, Automation Module manages the execution of the pre-defined rules. Prototype implementation includes rules only for data coming over MQTT Topic. MQTT class hierarchy is already explained in Communication Module section. MqttListener class (be explained in Communication Module section) is implemented for Automation Module to receive messages using MQTT protocol, so it subscribes to “/devices/+” topic. Then, according to the source of coming message (i.e. /devices/source), Automation Module checks if whether there is any rule defined. If so, it takes the action that is defined in the rule.

To illustrate the execution of a rule in the framework, it is possible to define a rule that checks a range criterion for completion of the task. Let’s assume a task responsible for listening to data from a light sensor to ensure the lights in the room are turned off. When this workflow is started, MqttListener begins to listen to data coming from a light sensor. As light sensor publishes data to the related topic, it checks if whether the lights are turned off or not. When data inside the accepted range for completion come from the sensor, automation module automatically completes the task and notifies the user about completion by publishing a message to the topic of the user.

Furthermore, automation module may alter the execution of the process. For instance, a generic workflow is designed to be able to be used in several contexts, and in this generic workflow user have to perform some collection of sequential tasks. However, one of the tasks is not applicable for the current context (maybe the associated device does not exist in the context, and so it is not required to be performed). Then, by defining a rule, Automation Module may suppress this task before it is offered to the user, and so the task is marked as if it is completed.

For now, rules are defined for only sensor’s data in the environment, and location information coming from a mobile device. At the moment, they are hard coded in the

class implementations. The prototype implementation does not include an advanced rule and context-reasoning component since it is beyond the scope of this thesis study. However, it is an open and leading research field and is asserted as a future work in the Conclusion chapter.

CHAPTER 5

FRAMEWORK SCENARIOS

In previous sections, problem definition and motivation of the thesis are described, and responses (distinctive features from other studies in the literature) addressing to these problems are explained. After discussing the architecture of the framework, general concepts and intended purposes of the components in the framework are briefly explained. Lastly, implementation details are given to demonstrate how the prototype implementation of the framework is developed and integrated, and the reasons of selecting particular components (YAWL, MQTT, and Android) are discussed. In this chapter, execution of the prototype implementation will be illustrated by giving detailed information and examples about designing a workflow, managing it by a mobile application, and getting assistance from mobile phone by using itself and devices in the context. Thus, efficiency and feasibility of the framework will have been asserted in various domains.

Workflows of the scenarios are designed by using YAWL editor, and specific environments are setup to realize the execution of each scenario. Consequently, each scenario is successfully completed from beginning to the end by using framework components and mobile application. Details, screen shots and flows of some tasks can be seen in the following scenario sections.

5.1 Smart Environment Scenario

First scenario is about assisting an employee who is assigned to prepare the meeting room in a university campus for a presentation. Campus and room are assumed to have smart environment features in which it is possible to interact with the devices in the room and campus, and there is availability for using computing devices and network connection. The design of the corresponding workflow is depicted in Fig. 5.1.

In his first task, employee needs to find the meeting room, so he checks his current task “Navigation to Presentation Room”. When he starts the task, a navigation application (installed in the mobile phone) displays routing information from current location to the room’s location by learning location information of the room from the

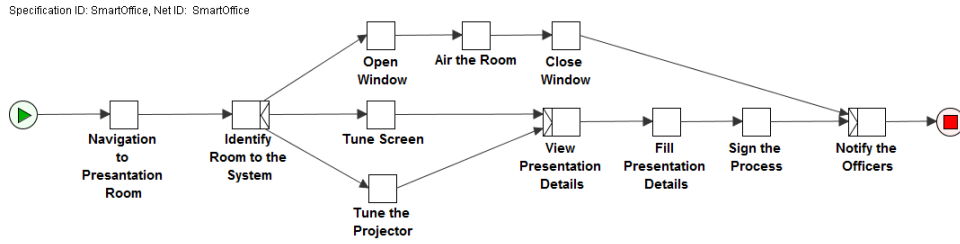


Figure 5.1: Smart Environment Workflow

framework. When employee comes closer to the room, the mobile application notifies employee about the arrival at the correct location and warns that he should complete the navigation task.

As soon as employee enters the room, he uses his mobile phone to scan the Quick Response (QR) code in the room, and QR code information are immediately sent to the framework (by mobile application) for identification of the room by the framework. Completion of this task initiates three paths that make it possible that an employee can work on parallel.

After the completion of the “Identify Room to the System” task, the subsequent tasks are activated according to the context information of the room. For example, in the “Open Window” task, the window sensor inside that particular room is started to be listened by the framework, and when employee opens the window, framework automatically completes the task according to the data coming from the window sensor, and notifies the employee about the completion of the task as displayed in Fig. 5.3a. Similarly, execution lifecycle of some tasks can be managed by the framework. In the “Air the Room” task, the framework decides the duration of allowing fresh air into the room by a rule that checks the outside temperature (5 minutes if it is lower than 23 degrees Celsius, and 10 minutes if higher than 23 degrees Celsius). While an employee is working on other tasks, the framework also continues managing the process according to the context information coming from the environment (room, sensors, web services). After 10 minutes passes since the employee opening the window, framework automatically completes “Air the Room” task and a notification is sent to the employee. Then, in the next task employee closes the window and the corresponding sensor sends a notification to the framework. If the sensor data are in acceptable value range which indicates that the window is closed, “Close Window” task is also automatically completed by the framework without user intervention, and a notification is displayed to the user. Automatic completion of “Close Window” is displayed as a sequence diagram in Fig. 5.2 that depicts the data transfer between the components of the framework.

User is assigned to the tasks for tuning screen and projector device for the presentation. These types of tasks may be cumbersome and complex for the people who have

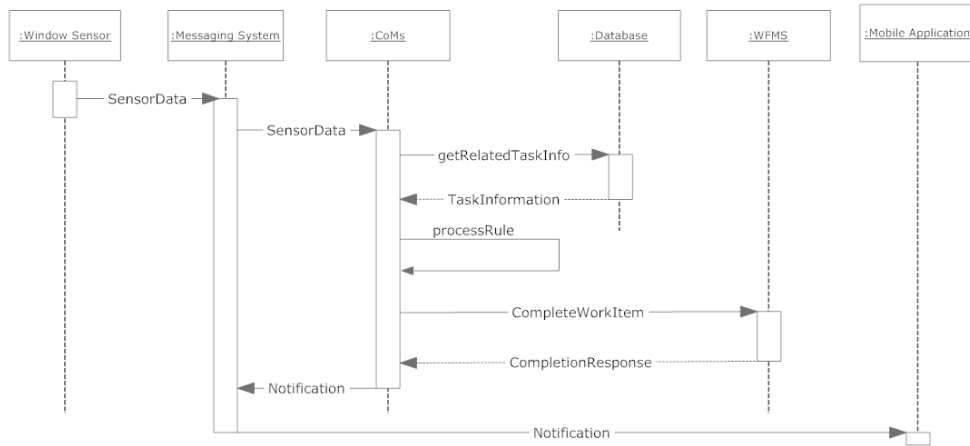


Figure 5.2: Close Window Task Execution Flow

not done it before. Therefore, giving clues or insights with a comment, picture or record to them may be very assistive while they are performing tasks (as if someone in near of him is describing how to do). For this purpose, tasks are enriched with supplementary resources (text, image, audio, and video record) which may be provided by a previous user or a workflow designer. Fig. 5.3b, such resources associated with the “Tune the Projector” task are shown. To illustrate, a photo is highlighting the correct communication ports on the projector device.

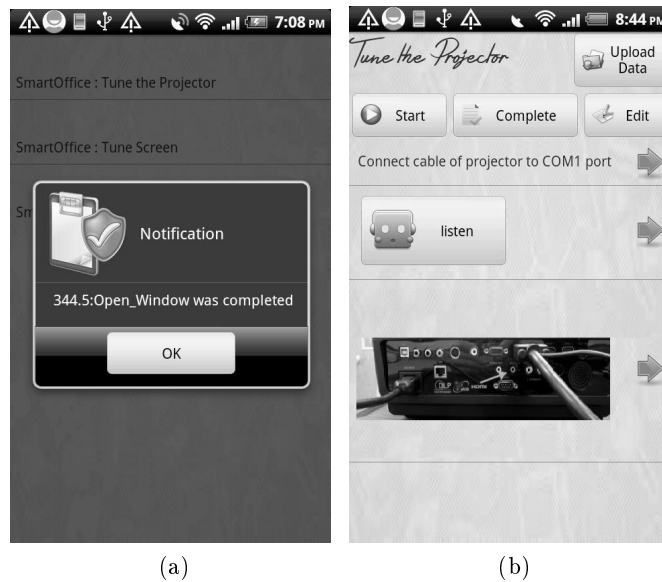


Figure 5.3: Smart Environment Scenario Mobile Application Screens

In “View Presentation Details” task, user is directed to a web site (by mobile application) where he can find details of the presentation, and then check the accuracy of the presentation with his notes. Later on, in “Fill Presentation Details” task, he com-

municates with the framework for updating information (e.g. title, speaker, summary, time, room) about presentation. After the employee completes his tasks for preparation of the room, in “Sign the Process” task, he scans NFC tag of the room (via mobile application), and room identification number (which proves that user resided in the room) is transmitted to the framework. Lastly, in the “Notify the Officers” task, user is presented with an optional form that he may inform officers about a situation (e.g. room is ready for the presentation; however, projector device’s remote controller is not working properly). When he completes the task, the framework makes a web service call, and information about the completion of the presentation room (together with the employee’s notes) are sent to the officers.

Information flow may occur among user, framework and context as depicted in tasks in Fig. 5.4. In the “Open Window” task, when sensors are introduced to the framework, data flow may occur between the environment and the framework. Similarly, in “Fill Presentation Details” task, user may communicate with the framework by filling a form. Lastly, in the “Sign the Process” user may directly interact with the room’s context by scanning the room’s NFC tag. All of these communications are provided using MQTT messaging system.

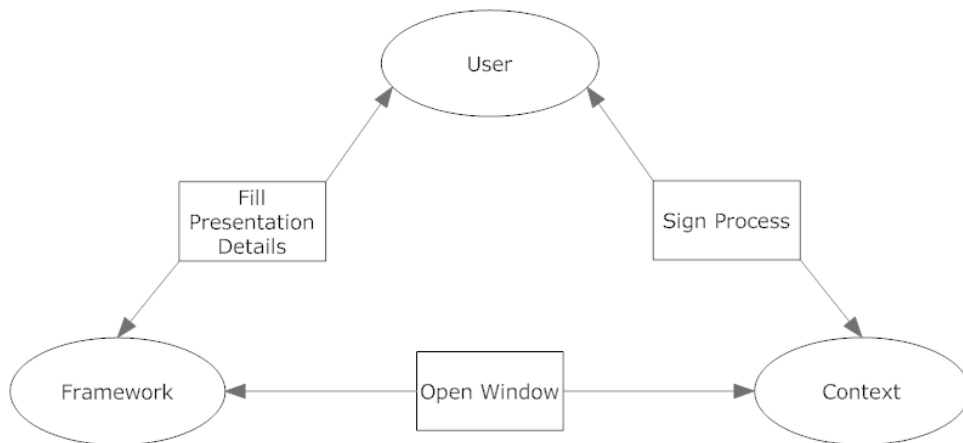


Figure 5.4: Data Flow between Components

5.2 Travel Scenario

In this section, a simplified version of one day travel scenario will be illustrated. Travel scenario is selected to demonstrate the feasibility of our framework in another domain, and the framework can be also very assistive in other activities of people’s lives rather than at work or office. As it is asserted in the previous chapters, workflows are very helpful for guiding people in unfamiliar environments. Having regard to that feature, this workflow is assumed to be designed for a person coming to visit a city that she has never been before, and workflow covers a part of her day travel as depicted in Fig. 5.5.

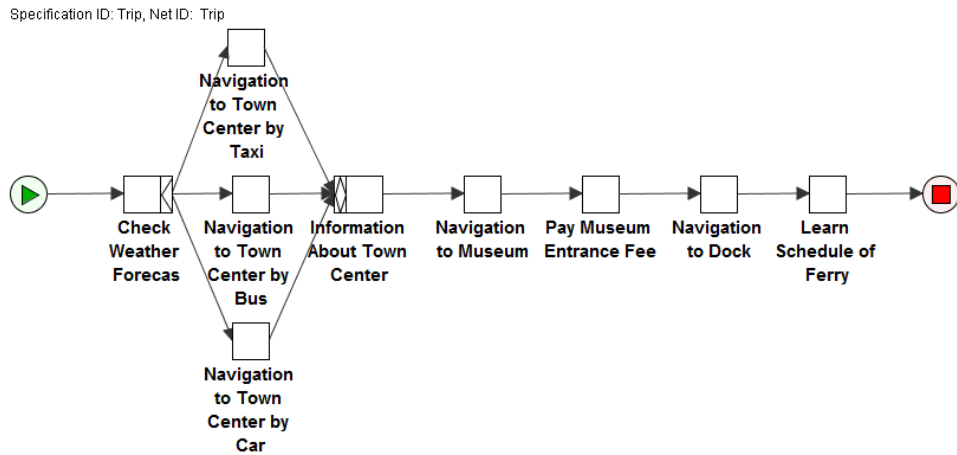


Figure 5.5: Travel Workflow

When user launches her workflow, in her first task framework displays a list of weather forecast stations that she can check the current weather forecast. Moreover, in this task it is also possible to define rules to warn user according to the weather conditions. For instance, framework gets the current weather forecast and in terms of conditions defined in the rule, it may notify user such as “Today, it will rain. Don’t forget to take your umbrella!”.

After user checks weather forecast, she should go to the place where she will begin to visit city. She may be presented with more than one option to arrive at the town center i.e. by taxi, by bus or by car. She may go over each option by looking resources (comment, suggestion). Then by the help of the framework, she may ease the process. She may find out the nearest bus stations, and get the bus schedule of the station. Or, she can find the phone number of taxi stands, and by using the navigation application inside taxi, she can be sure that taxi keeps following the correct route. Moreover, when she comes near to the target location (target location information is dynamically got from the framework), mobile application warns her about the arrival at the target location, and she should complete the task as displayed in Fig. 5.6a.

User arrives at the town center, and in the next task she searches for information about what to do in the town center. User reaches the list of suggested activities by her mobile application which gets related web addresses from the framework. In this list, she is directed to the websites where she can find detailed information.

After visiting this part of the city, user is navigated to the museum that she should visit and again she is assisted by the mobile application. Since the museum is near to the town center, she can go by foot by following the paths in the mobile application. In order to do that, mobile application learns the location of the museum from framework.

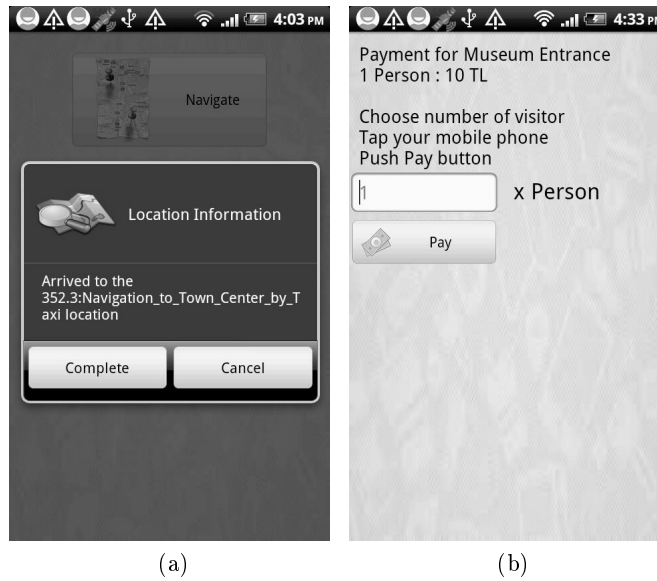


Figure 5.6: Travel Scenario Mobile Application Screens

Not to make an effort about paying the entrance fee of the museum and lose time by waiting in the queue, entrance fee data are integrated into the workflow definition. By this means, when she comes to the museum entrance, she uses her mobile phone NFC capability, and entrance door reads NFC data from mobile phone and easily allows entering the museum. Related application screen is displayed in Fig. 5.6b.

After user visits the museum, he is directed to the docks to use a ferry. She can find detailed information about in which station she should get off in the details of the task.

Before coming to the dock, user can learn the schedule of the ferry by looking at the next task “Learn Schedule of Ferry”. When user starts this task, framework makes a web service call to get the schedule data, and then sends this information to be displayed in mobile application screen. Moreover, in this task it is possible to define rules about catching up to the ferry hours. For instance, framework may send a notification to the user, if she wants to catch the ferry at a specific hour, she should hurry up. Flow of this task (includes displaying schedule operation and warning user operation) is depicted as a sequence diagram in Fig. 5.7. to explain the interactions between components.

5.3 Health Scenario

Health domain is critical since people become very rigorous when their health is subject; therefore, guidance from a professional feels them more comfortable. Hence, such

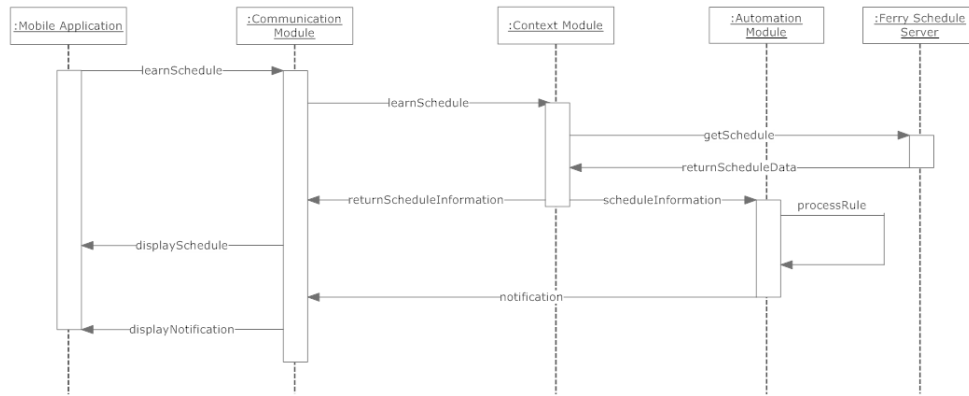


Figure 5.7: Learn Schedule of Ferry Task Execution Flow

a framework may be very helpful in scenarios such as chronic disease treatment, patient care at home, follow up post-operative patient, elderly patient care.

Mobile health scenario may be applied for various situations in people’s lives. In the simplest case, taking medication may be supported especially for the elderly patients. For instance, a patient may be warned to take his medicines on times, and be also supported about the type, dosage, usage of his medicines.

Another key point in the health domain is accurately identifying patients and ubiquitously reaching information about them. Patient may be signed by an attached NFC tag when they are accepted to the hospital, after that all operations (tests, medications, prescriptions) are handled by scanning that NFC tag. By this means, it becomes almost impossible to mistake the identity of the patient even if he is unconscious. Also, by using such a system, doctors may reach a broader range of information about a patient, and instead of paper patient chart, they may use a PDA or mobile phone to monitor the patient situation. It may also be beneficial at nurse shifts, which decreases the possibility of making mistakes.

Organizations are searching for methods to improve treatment opportunities in their hospitals, and to reduce delays and errors in patient care. For this purpose, efficient collaboration between hospital clinicians (doctors, nurses, technicians) is crucial for increasing the quality of treatments. Daily activities of clinicians may be supported by providing them collaboration mechanisms and displaying real time of tasks that they should execute. To illustrate, a scenario that is designed to increase the efficiency of patient admission and guidance will be demonstrated in this section. Related workflow design is depicted in Fig. 5.8. This scenario is designed for a simple and straightforward treatment procedure for the sake of simplicity, since this scenario aims to demonstrate adequateness of the framework assisting users in hospital environments. Advanced patterns (e.g. doctor may request new tests according to the test result of the patient) are omitted.

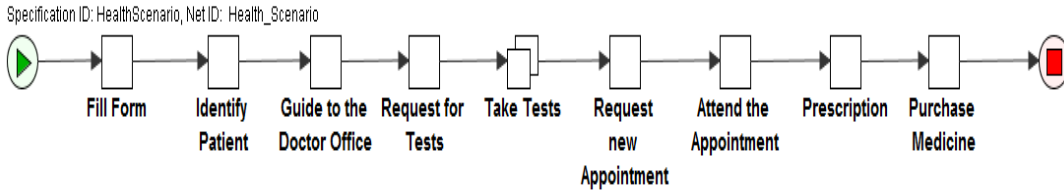


Figure 5.8: Patient Treatment Workflow

When patient arrives at the hospital, he fills a form including his personal details and complaints in his first task “Fill Form”. A sample form definition is displayed in Fig. 5.9a. In the next task “Identify Patient”, he gets his unique patient number by making a web service call. This patient number is also associated with a barcode that will be used throughout his treatment.



Figure 5.9: Health Scenario Mobile Application Screens

Then, according to the patient’s inputs (department and complaints), he is guided to a specific doctor in the hospital. He can find the detailed information about doctor office (floor, room number, and maybe a floor plan) in the “Guide to the Doctor Office” task. After examining the patient, doctor may request for some tests, and patient performs these tests respectively (as displayed in Fig. 5.9b) in “Take Tests” task which is a multiple type task that includes details of each test to guide patient. After tests of the patient is completed, he request for a new appointment from doctor to consult his tests’ results. Since new appointment time may not be very close or at the same day, a notification task is scheduled to warn the patient to attend his appointment.

In this appointment, doctor prescribes necessary medicines with their dosage and usage by using his desktop computer in his office. At the end, patient goes to the pharmacy and gets his medicines by scanning the barcode on his mobile phone.

CHAPTER 6

CONCLUSION

Expression of thesis concludes with this chapter. Firstly, summary of thesis study with its contributions is given at a conceptual level. Then, limits of this study are explained, and possible future works about them are discussed.

6.1 Summary and Contributions

To state it briefly, the aim of this framework is to assist users in pervasive environments while they are performing their personal activities. This does not restrict the framework to be applicable only for particular scenarios in specific environments. In other words, a daily life activity in a house environment is not the only target of this framework. We propose a framework that can be applied to various domains such as smart environment, tourism and health as they are demonstrated in Chapter 5. Yet, it is assumed that environment of the user is saturated with pervasive technologies (Internet, sensors, devices, microprocessors, web services) that the framework can ubiquitously interact. Applicability of the framework into the scenarios from different domains (without any effort or with a slight effort in the implementation of the framework) asserts the feasibility of a common framework that can cover a broad range of user activities, as long as complementary sources are available (e.g. a properly designed workflow, an advanced context-reasoning mechanism and ubiquitous communication with the environment).

Being assisted activity may be a routine work in a daily life or a complex process that will be done for the first time. In each case, the starting point is to assign users certain tasks that they can efficiently perform within a right order. For this purpose, framework is built on top of a WFMS. WFMS has been used in business environments for years, and it is proven to be effective to increase the accuracy and quality of business processes. Same situations are also valid for personal activities, and also in recent studies it has been claimed that workflows are very successful at guiding users in an unfamiliar and complex environment [26].

Thus, we use workflows for simplifying and customizing of personal activities in pervasive environments. However, for managing personal activities, stand-alone usage of workflows is not enough. Guiding users in pervasive environments have some major differences from guiding users in business environments.

Firstly, while users are performing their activities, they will be mobile most of the time. Therefore, a WFMS that can be controlled only from a desktop computer or a browser is not sufficient. Users should be presented with basic functionalities (e.g. get the list of the task, complete a task) via clear user interfaces. Thereby, a mobile application is developed, by which users are able to interact with WFMS. Mobile users need a set of certain and easy-to-access functionalities, so presenting advanced set of operations (as in a desktop version of a WFMS) to them reduces the usability and increases the complexity of the application.

On the other hand, mobility reveals another problem: connectivity. Since users are mobile in their daily life, they do not have a reliable and steady network communication. After a while client makes a request to the server, it may become offline, and when server sends a response, it will be unreachable. For such circumstances, topic based protocols provide a service quality to ensure the delivery of messages and a medium for caching messages. By this means, messages are delivered to the client, when it is back online. Also, mobile devices have finite power and resource limitations. Thus, mobile connectivity should be handled differently than wired devices. Enabling to get information from the server without polling a medium is another key advantage of topic based protocols that also minimize network and memory usage. Decreasing resource usage is already a key principle for mobile application development. For these purposes, a publish / subscribe messaging communication (topic based) infrastructure is set up for the framework. With this infrastructure, it is also possible to transmit kilobytes of image data in addition to simple text messages. By this means, messaging protocols enable to speed up and scale mobile application by getting messages according to interest fields (i.e. topics) and not bothering with IP addresses of each devices in the environment (Technical details how this protocol differs and integrated into the framework are explained in previous chapters).

Lastly, although workflows organize the execution of activities, it is merely not enough for assisting users in pervasive environments. Because, there is a need of more advanced framework rather than an application displaying its users what to do and in which order to do. For this purpose, our framework overloads tasks in workflows with context information. To do that, additional information (related with tasks) is kept in framework database. Definition of tasks is customized with its context data. It can be resembled to “Class – Object Paradigm” in Object Oriented Programming. Each workflow is instantiated with its context information in our framework. This instantiation also makes tasks gain their uniqueness. By this means, each task is handled differently according to the context that it is inside. By this means, our framework enables tasks dynamically adapt to their context.

Additionally, capability to define task uniquely presents another opportunity; augmentation tasks with supplementary information (text, image, audio and video). Users may upload data about tasks to note about their workings, or help other users who are working on the same activity. These resources are also kept at framework database by relating them with tasks (according to their contexts). In general, people want to search and learn other people's experience and knowledge, although they have ideas what to do and how to do. As it is clearly demonstrated in the illustration of scenarios, expression of a work by other people may become practical in many situations.

Most importantly, keeping the relation of tasks with context information enables automate user activities. In the same way that supplementary information about tasks is kept, types and data supply resources of tasks may be kept in the database. For instance, a task asserted as a sensor listener task type may be related with a specific topic address. This enables to define conditions about that task. In other words, it is possible to define a rule that will automatically complete a task with respect to data coming from a sensor.

Regarding the cost of building such a framework, the components that are integrated into the framework and platforms that are used for development purposes are open source software projects. By this means, implementation of a framework may be completed without any cost (except hardware resources and server costs).

6.2 Future Work

Prototype implementation of the framework is developed for proving the feasibility of the framework. Thus, some of the framework component is implemented at the basic level. Improving these components can increase effectiveness and usage of the framework, and so users can be assisted for a broader range of activities.

First study may be about determining context of a task. In our framework, task and context relation is kept in a database. This relation contains single level hierarchy. In other words, each task is associated with only one context data (usually with location). However, as explained in Chapter 2, context may include other properties like identity, state, preferences. Increasing context-awareness of the framework enables more accurate determination of real life events. For instance, in addition to its location information, differentiating a task with regards to user's preferences (e.g. sex, age) will be clearly more meaningful. Hence, context-awareness improvement may be a promising field of study for such frameworks.

While automating user activities, our framework uses pre-defined and programming-driven rules. However, an advanced context-reasoning component can be integrated into the framework since context prediction is a challenging process, and it is not possible to obtain advanced inferences only by using rules. With the help of a context-

reasoning component, event detection can be improved by further analyzing data, and these data may also be used for training the event - condition relations.

Probably, the most significant point that needs an improvement is to secure confidential data about users. Communication infrastructure is built upon a publish / subscribe protocol, so clients may subscribe a topic and get raw bytes of messages as long as they know the address of topics. Although MQTT support authorization and SSL encryption across the network, some additional improvement to increase the notion of trust may be added. To illustrate:

- Restrictions to subscribe and publish data to topics
- Encoding / decoding of messages

In the development of this prototype implementation, we have some assumptions that smooth the implementation, so there exist some limitations about prototype. First of all, we do not make use of history details of user's actions. In other words, while a user is progressing on his process, it is not possible to search and learn what he did in his previous tasks. However, we log data transfers inside the framework that we can use for further analysis, and also we can convert to logging mechanism to another component that framework may use in future studies. Secondly, in the realization of scenarios (so in the mobile application), there is not an advanced error handling, and we assume that users will follow the steps that are displayed them on the mobile application screen. Lastly, interactions with the environment are designed in a simple level. For instance, when a user is assigned to a task in which he is responsible for closing the window, but the window is already closed, so how will framework behave or will sensor transmit data or not is out of scope of this thesis study. Since, this prototype is implemented to demonstrate the feasibility of this study, these advancements are left for a future study to increase the efficiency of the framework.

Our framework enables generating and keeping contents that can be shared between users. In a sense, it lays the foundation of a social collaboration mechanism in our framework. However, since social collaboration part is beyond the scope of this prototype implementation, it is not emphasized in detail. Thus, it may also be researched how to integrate social collaboration into the framework and how such a framework may benefit from it.

REFERENCES

- [1] B. Avenoglu. *A Context-Aware and Workflow-Based Framework for Pervasive Environments*. PhD thesis, Informatics Institute, Middle East Technical University.
- [2] M. Caporuscio, A. Carzaniga, and A. L. Wolf. Design and evaluation of a support service for mobile, wireless publish/subscribe applications. *Software Engineering, IEEE Transactions on*, 29(12):1059–1071, 2003.
- [3] G. Chen, D. Kotz, et al. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.
- [4] Cisco. Cisco visual networking index: Global mobile data, 2013.
- [5] M. Collina, G. E. Corazza, and A. Vanelli-Coralli. Introducing the qest broker: Scaling the iot by bridging mqtt and rest. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pages 36–41. IEEE, 2012.
- [6] P. Dadam and M. Reichert. The adept project: a decade of research and development for robust and flexible process support. *Computer Science-Research and Development*, 23(2):81–97, 2009.
- [7] A. K. Dey and G. D. Abowd. Cybreminder: A context-aware system for supporting reminders. In *Handheld and Ubiquitous Computing*, pages 172–186. Springer, 2000.
- [8] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001.
- [9] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 2011.
- [10] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.
- [11] M. Friedewald and O. Raabe. Ubiquitous computing: An overview of technology impacts. *Telematics and Informatics*, 28(2):55–65, 2011.
- [12] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.

- [13] P. Giner, C. Cetina, J. Fons, and V. Pelechano. Developing mobile business processes for the internet of things. *Pervasive Computing, IEEE*, 9(2):18–26, 2010.
- [14] S. Gogouvtis, K. Konstanteli, S. Waldschmidt, G. Kousiouris, G. Katsaros, A. Menychtas, D. Kyriazis, and T. Varvarigou. Workflow management for soft real-time interactive applications in virtualized environments. *Future generation computer systems*, 28(1):193–209, 2012.
- [15] D. Guinard, V. Trifa, and E. Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.
- [16] G. Hackmann, M. Haitjema, C. Gill, and G.-C. Roman. Sliver: A bpel workflow process execution engine for mobile devices. In *Service-Oriented Computing-ICSOC 2006*, pages 503–508. Springer, 2006.
- [17] J. Hong, E.-H. Suh, J. Kim, and S. Kim. Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications*, 36(4):7448–7457, 2009.
- [18] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtts a publish/subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 791–798. IEEE, 2008.
- [19] IBM. Mq telemetry transport (mqtt) v3.1 protocol specification, 2010.
- [20] IBM. Really small message broker, 2013.
- [21] A. Kocurova, S. Oussena, P. Komisarczuk, and T. Clark. Context-aware content-centric collaborative workflow management for mobile devices. In *COLLA 2012, The Second International Conference on Advanced Collaborative Networks, Systems and Applications*, pages 54–57, 2012.
- [22] N. Lohmann, E. Verbeek, and R. Dijkman. Petri net transformations for business processes—a survey. In *Transactions on Petri Nets and Other Models of Concurrency II*, pages 46–63. Springer, 2009.
- [23] L. Pajunen and S. Chande. Developing workflow engine for mobile devices. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 279–279. IEEE, 2007.
- [24] R. Pryss, J. Tiedeken, U. Kreher, and M. Reichert. Towards flexible process support on mobile devices. In *Information Systems Evolution*, pages 150–165. Springer, 2011.
- [25] R. Pryss, J. Tiedeken, and M. Reichert. Managing processes on mobile devices: The marple approach. 2010.
- [26] A. Ranganathan and S. McFaddin. Using workflows to coordinate web services in pervasive computing environments. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 288–295. IEEE, 2004.
- [27] M. Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

- [28] M. Tag. The growth of mobile marketing and tagging, 2011.
- [29] A. Ter Hofstede. Workflow patterns: On the expressive power of (petri-net-based) workflow languages. In *of DAIMI, University of Aarhus*. Citeseer, 2002.
- [30] W. M. van der Aalst. Three good reasons for using a petri-net-based workflow management system. In *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pages 179–201. Cambridge, Massachusetts, 1996.
- [31] W. M. van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.
- [32] W. M. van der Aalst, L. Aldred, M. Dumas, and A. H. ter Hofstede. Design and implementation of the yawl system. In *Advanced Information Systems Engineering*, pages 142–159. Springer, 2004.
- [33] W. M. Van Der Aalst and A. H. Ter Hofstede. Yawl: yet another workflow language. *Information systems*, 30(4):245–275, 2005.
- [34] W. Viriyasitavat, L. Da Xu, and A. Martin. Swspec: the requirements specification language in service workflow environments. *Industrial Informatics, IEEE Transactions on*, 8(3):631–638, 2012.
- [35] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.
- [36] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.
- [37] M. Weiser and J. S. Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Springer, 1997.
- [38] M. Weske. *Business process management*. Springer, 2012.
- [39] S. A. White. Process modeling notations and workflow patterns. *Workflow Handbook*, 2004:265–294, 2004.
- [40] Z. Xing, Z. Hong, and L. Yulong. A petri-net based context-aware workflow system for smart home. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2336–2342. IEEE, 2012.
- [41] P. Zheng and L. Ni. *Smart phone and next generation mobile computing*. Morgan Kaufmann, 2010.

APPENDIX A

YAWL

A.0.1 Introduction to YAWL Architecture

An overview of the various interfaces of YAWL Environment is shown in Fig. A.1. YAWL has a strong and clear design architecture in which it is possible to reach the all functionalities of the WFMS by using Java APIs (Application Programming Interface). For instance, as it can be seen in the figure, Interface A is responsible for management of the workflows (e.g. upload a new workflow to the system, launch or cancel workflow execution). Another API, Interface B is the responsible for execution of the workflow tasks (e.g. offer a task to the specific user, complete a task, upload data to a task). Functionalities of other APIs and their usages are also explained in the technical manual of YAWL which can be downloaded from YAWL website.

In order to ease the usage of these interfaces in the client side, YAWL encapsulated the methods in InterfaceA and InterfaceB and presented a new APIs named ResourceGatewayClient and WorkQueueGatewayClient respectively. These interfaces are easier to use and to integrate to the application. The execution principle of these interfaces is to make data transfer over HTTP using XML data structure. By using these gateway implementations, it is possible to interact with YAWL engine with YAWL Java Classes. These gateways marshall Java objects into XML type and sends a HTTP request to the YAWL Server. Similarly, the response coming from YAWL Server (in a XML format) is unmarshalled into the Java objects and returned to the client. In other worlds, it presents a seamless communication with YAWL without knowing the internal structure of the engine and dealing with the handling data processes (i.e. transforming XML data type into Java Classes and vice versa).

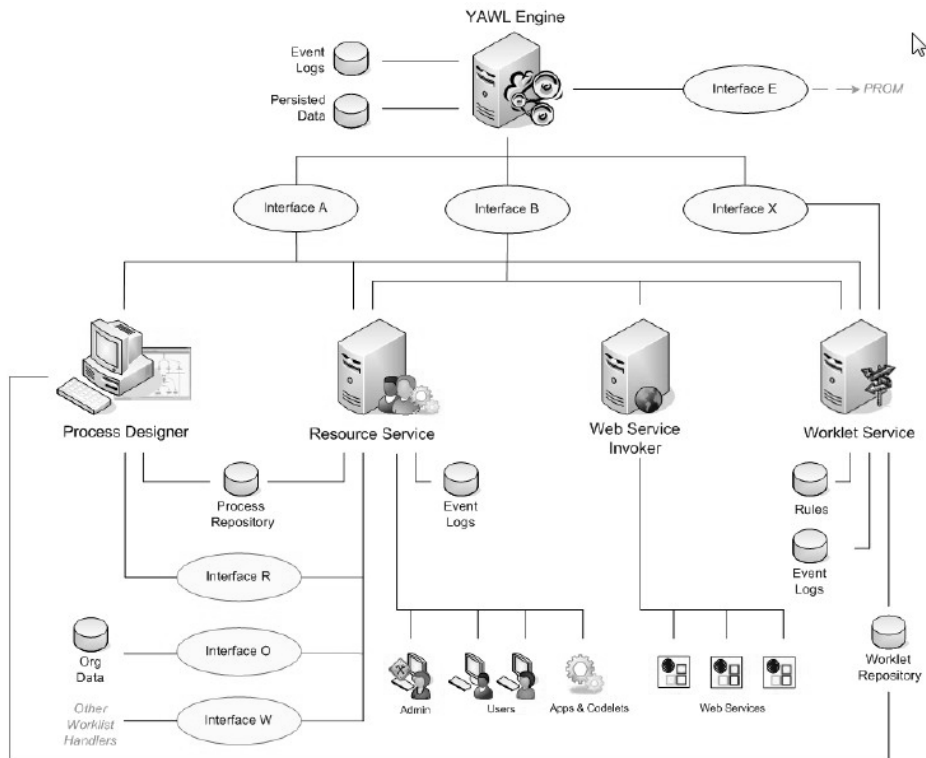


Figure A.1: Yawl Architecture

In order to use these services, at first client must connect to the YAWL engine via supplying necessary information (IP address of the server, username, password). When client send a connect request to the server, a handle information is given as a response (in case of successful connection), and client has to use this handle information in all other requests to the server.

These gateway implementations and their dependencies (required to be used) are bundled in a Java Archive (JAR) file, and is available at the installation folder of YAWL. To use these interfaces it is enough to integrate this file to project.

A.0.2 YAWL Elements

In YAWL, description of business procedure is named as “Workflow Specification”. It includes the details to be able to be loaded into a workflow engine. Workflow Specification defines which task should be performed, the conditions and orders of the tasks, and requirements of the data and resources while performing tasks. “Task” is

a unit of work that has to be performed as a part of the Workflow Specification. In similar to the Object-Oriented Paradigm (i.e. Class is the definition, and Object is the instantiation of the Class), YAWL calls differently to the instantiated version of these elements. For instance, launched Workflow Specification is referred as Case, and started Task is referred as Work Item.

Tasks are connected to each other by incoming and outgoing flow. However, it is not possible to model all type of processes since there may be multiple flows in both incoming and outgoing flows. Additionally, it is necessary to define how to handle these multiple flows. For instance, in a multiple incoming flow; is it enough to completion of a single task to continue to the next task, or should all of the coming tasks be completed? For this reasons, YAWL presents “Joins” and “Splits”, and it is possible to use these with AND, OR, XOR patterns. Type and definitions of these patterns are described in Fig. A.2.







| Name: | Symbol: | Description: |
|---------------------|---|---|
| Split Types: | | |
| XOR-Split |  | The XOR-Split is used to trigger <i>only one</i> outgoing flow. It is best used for automatically choosing between a number of possible exclusive alternatives once a task completes. |
| AND-Split |  | The AND-Split is used to start a number of task instances simultaneously. It can be viewed as a specialisation of the OR-Split, where work will be triggered to start on <i>all</i> outgoing flows. |
| OR-Split |  | The OR-Split is used to trigger some, but <i>not necessarily all</i> outgoing flows to other tasks. It is best used when we won't know until run-time exactly what concurrent resultant work can lead from the completion of a task. |
| Join Types: | | |
| AND-Join |  | A task with an AND-Join will wait to receive completed work from all of its incoming flows before beginning. It is typically used to synchronise pre-requisite activities that must be completed before some new piece of work may begin. |
| XOR-Join |  | Once <i>any</i> work has completed on an incoming flow, a task with an XOR-Join will be capable of beginning work. It is typically used to allow new work to start so long as one of several different pieces of earlier work have been completed. |
| OR-Join |  | The OR-Join ensures that a task waits until all incoming flows have either finished, or will never finish. OR-Joins are “smart”: they will only wait for something if it is necessary to wait. However, understanding models with OR-joins can be tricky and therefore OR-joins should be used sparingly. |

Figure A.2: Yawl Element Types

In YAWL, Tasks can be dynamically assigned into users or roles. Those assigned tasks may be in various states. For instance, a task may be offered to the user, or a task

may be started to be performed by a user. Also, there may be transactions in states of the tasks. For instance, a user, offered to the specific task, may not be available, so the task may be offered to another user. For this purposes, YAWL defines the lifecycle (states and possible transitions) of a task. An overview (but not complete) of this lifecycle is displayed in Fig. A.3.

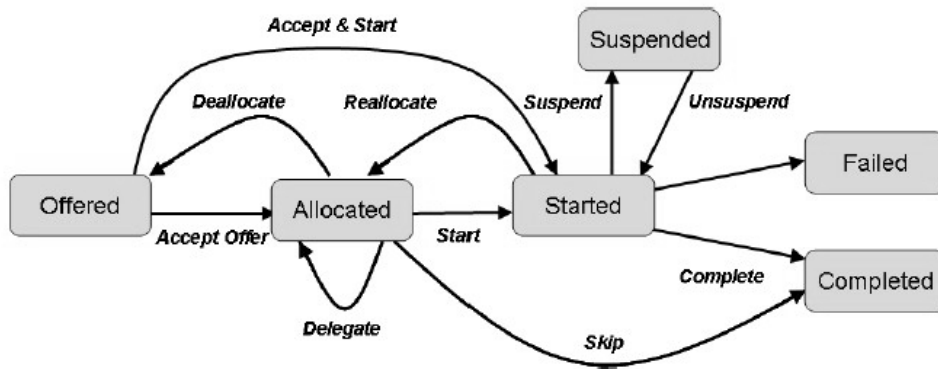


Figure A.3: Yawl Task Lifecycle

Lifecycle of a Task begins with the Creation, and then this task is “Offered” to a user or users in a specific group. When a user accepts the offer, task is “Allocated” to him and he becomes the only one who can perform it. Later, he marks that he is working on the task and it turns into “Started” state. He can suspend or unsuspend a task while he is performing. However, “Failed” task stops the lifecycle of execution. After properly performing, lifecycle comes to end and user marks the status of the task as “Completed”.

A.0.3 YAWL Data Transfer

YAWL defines three types of data, but only two of them (task variable and net variable) are relevant in order to enable data flow among tasks. Task Variables are defined in the context of task definition, they are local to the tasks; in other words, it is only visible for the specific task. Data interaction with the task assignee occurs by these variables. For instance, when a user fills out a form, data are saved into these variables, or the data are already given to these variables may be displayed to the users. To be able to passing out the data, Net Variables must be used. They are defined in the context

of workflow, they are global; in other words, they are accessible by every task in the workflow specification.

Data flow between Task and Net variables occurs by writing simple XQuery expressions. XQuery is a query and functional programming language especially designed for querying data in XML structure. For instance, in Fig. A.4, in the output parameter section, it is defined that “WorkItemValue” task variable will be inserted into the “WorkItemValue” net variable. Similarly, in the input section an XQuery expression is defined to populate data taken from net variable into the task variable. To define the flow, also usage of the variables should be defined as “Input”, “Output” and “Input & Output”.

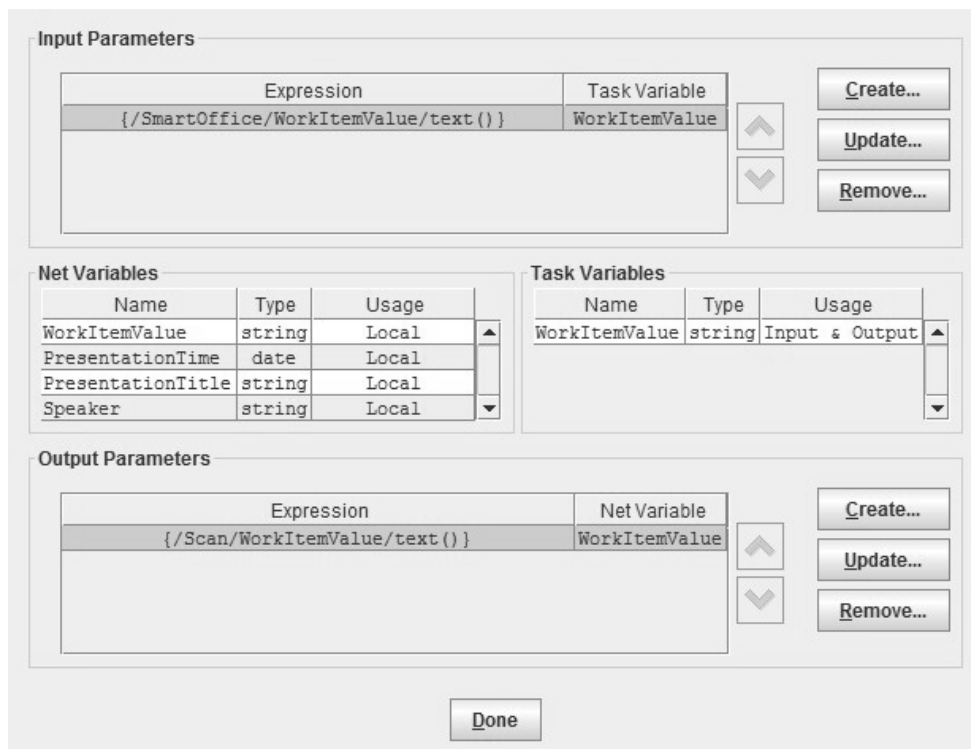


Figure A.4: Yawl Parameter Passing Options

Types of data may be defined by using simple type patterns (e.g. String, Number, Date) like in this definition, but also a workflow designer may create a complex data type by using XML Schema Definition (XSD), and so data are automatically transferred and processed in XML type.

A.0.4 YAWL Editor

For modeling the processes, YAWL offers a visual editor in which it is easy to design and edit workflows by using drag-and-drop feature, and then graphically configuring the properties of the elements, variables, flow. Moreover, it also provides validation and analysis of the workflow specifications. In addition to validation of structural correctness of the workflow, it is possible to get possible warnings and errors before deploying it to the engine by analyzing the workflow via editor. Workspace of the editor is displayed in Fig. A.5 to demonstrate these features. Yawl Editor is installed along with other Yawl Components. It is developed as a Java Desktop Application, so it is portable and runnable in multiple platforms. The detailed information about each feature can be found in YAWL manual.

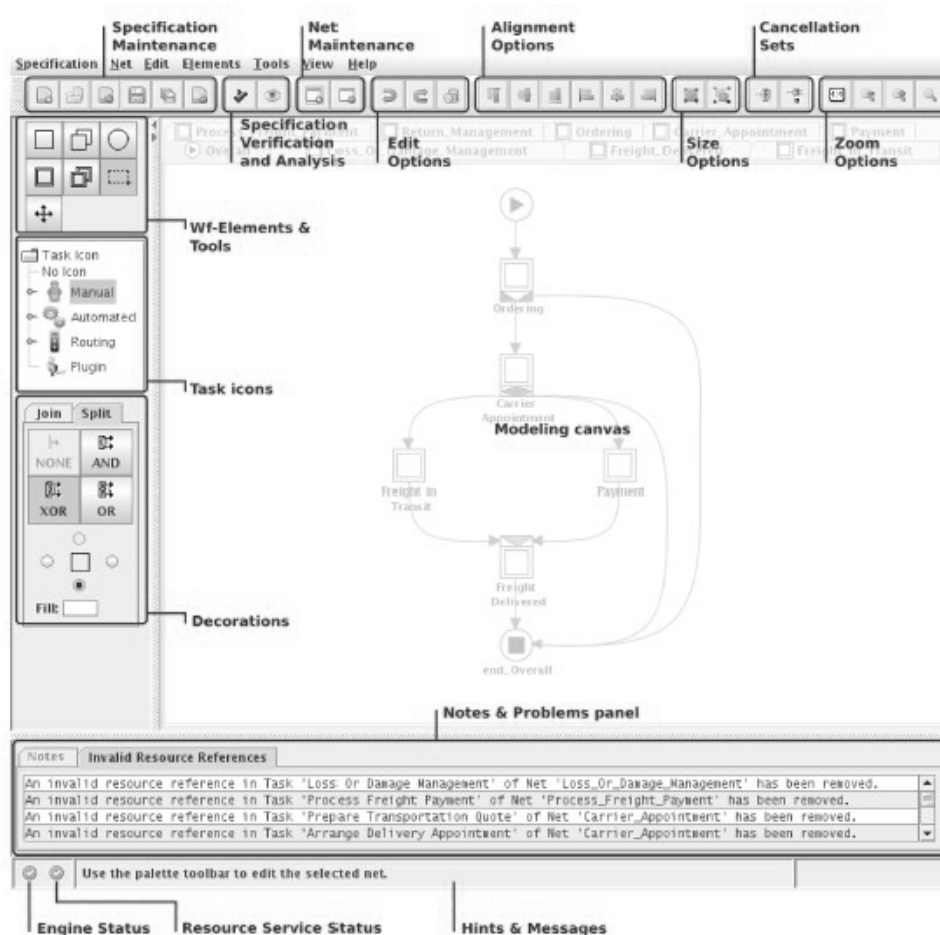


Figure A.5: Yawl Editor Screen

A workflow is designed by constructing a path between start and end node with YAWL components (nodes and edges). A simple definition is demonstrated in Fig. A.6. in which a direct path between source and target node is combined with an edge. In this definition, after completing a task in source node, user is assigned to the task in target node. This is the simplest form of defining a flow; it may be improved by adding splits, joins, and advanced paths (e.g. cancellation path, multiple paths). It is also possible to transfer data between nodes as explained in Yawl Data Transfer section.

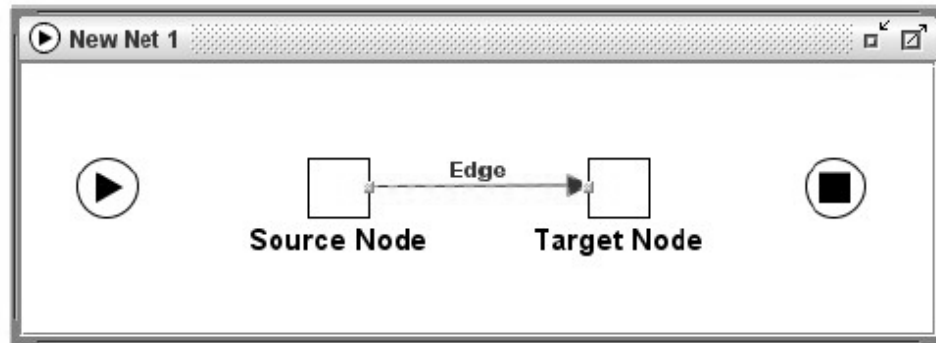


Figure A.6: YAWL Process Flow Definition

The output of the workflow specification is exported into a file with an XML structure. Hence, workflow may be edited without using the editor. Yet, more importantly it is also possible to research on workflow definition by parsing XML document, and so a workflow may be dynamically enhanced with respect to the analyze results.

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

- Fen Bilimleri Enstitüsü
- Sosyal Bilimler Enstitüsü
- Uygulamalı Matematik Enstitüsü
- Enformatik Enstitüsü
- Deniz Bilimleri Enstitüsü

YAZARIN

Soyadı : TÜYSÜZ
Adı : GÖKHAN
Bölümü : BİLİŞİM SİSTEMLERİ

TEZİN ADI (İngilizce) : A Workflow-based Mobile Guidance Framework for Managing Personal Activities

TEZİN TÜRÜ : Yüksek Lisans Doktora

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir.
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir.
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz.

TEZİN KÜTÜPHANEYE TESLİM TARİHİ :