A CONTEXT-AWARE AND WORKFLOW-BASED FRAMEWORK FOR PERVASIVE ENVIRONMENTS

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

BY

BİLGİN AVENOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

FEBRUARY 2014

A CONTEXT-AWARE AND WORKFLOW-BASED FRAMEWORK FOR PERVASIVE ENVIRONMENTS

Submitted by **BİLGİN AVENOĞLU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife BAYKAL
Director, **Informatics Institute** _____

Prof. Dr. Yasemin YARDIMCI ÇETİN
Head of Department, **Information Systems** _____

Assist. Prof. Dr. P. Erhan EREN
Supervisor, **Information Systems, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Altan KOÇYİĞİT
Information Systems, METU _____

Assist. Prof. Dr. P. Erhan EREN
Information Systems, METU _____

Assoc. Prof. Dr. Aysu BETİN CAN
Information Systems, METU _____

Assoc. Prof. Dr. İbrahim KÖRPEOĞLU
Computer Engineering, Bilkent University _____

Assoc. Prof. Dr. Alptekin TEMİZEL
Work Based Learning, METU _____

Date: 27.02.2014

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 and Surname : Bilgin AVENOĞLU

Signature : _____

ABSTRACT

A CONTEXT-AWARE AND WORKFLOW-BASED FRAMEWORK FOR PERVASIVE ENVIRONMENTS

Avenoğlu, Bilgin
Ph.D., Department of Information Systems
Supervisor: Assist. Prof. Dr. P. Erhan Eren

February 2014, 73 pages

In this thesis study, a framework offering people help and guidance for organizing their daily activities is proposed. This framework allows users to model their daily activities in the form of workflows. Due to the dynamic nature of pervasive environments, workflows are enabled to be adaptable at runtime according to context information collected in pervasive environments. A workflow engine is used for modeling and management of workflows and a separate rule engine with Complex Event Processing (CEP) capability is incorporated into the framework for both enhancing workflow adaptation and execution and for rule-based context reasoning. The adaptation model in the framework allows modeling activities in a hierarchical manner, from high level abstract activities to more detailed ones. Event-driven Architecture (EDA) is utilized for loosely coupled interaction between the workflow engine and the rule engine, allowing these engines, other context sources and user-level applications to exchange data among each other. Since workflow engines are sufficiently complex, EDA allows incorporation of context information into the workflow models without modifying the workflow language. A higher automation level than the level supported by workflows is proposed by processing events in pervasive environments using CEP. A prototype implementation is developed and the framework is evaluated with some real life examples and experimental tests demonstrating its applicability.

Keywords: Pervasive Computing, Workflow Adaptation, Context-Aware Systems, Complex Event Processing, Event-Driven Architecture

ÖZ

YAYGIN BİLİŞİM ORTAMLARI İÇİN İŞ AKIŞ MODELLERİ TEMELLİ VE BAĞLAM BİLİNÇLİ BİR ÇERÇEVE

Avenoğlu, Bilgin
Doktora, Bilişim Sistemleri
Tez Yöneticisi: Yrd. Doç. Dr. P. Erhan Eren

Şubat 2014, 73 sayfa

Bu çalışmada, insanlara günlük hayattaki etkinliklerinde yardım edebilecek ve onları yönlendirebilecek bir yazılım çerçevesi önerilmiştir. Çerçeve, kullanıcıların günlük etkinliklerini iş akış modelleri ile modellemesine olanak sağlamıştır. Yaygım Bilişim (YB) ortamlarının devingenliği nedeniyle iş akış modellerinin çalışma anında bağlama göre uyarlanması sağlanmıştır. İş akış modellerinin tasarlanması ve yönetilmesi için bir iş akış modeli makinesi kullanılmış, iş akış modellerinin uyarlanması ve çalıştırılmasına yardım edecek ve bağlamdan akıl yürütme yoluyla sonuç çıkarabilecek Karmaşık Olay İşleme (KOİ) tabanlı ayrı bir kural motoru çerçeveye dahil edilmiştir. Çalışmada kullanılan iş akış modeli uyarlama yaklaşımı kullanıcılara günlük aktivitelerini üst düzey soyut etkinliklerden daha alt düzey somut etkinliklere doğru hiyerarşik biçimde modelleme olanağı sunmaktadır. İş akış modeli makinesi ile kural motoru arasında esnek etkileşime olanak veren bir Olay Temelli Yapı (OTY) kullanılmış ve bu yapı aynı zamanda diğer bağlam kaynakları ve kullanıcı uygulamaları ile veri alışverişini de sağlamıştır. OTY, iş akış modeli makinelerinin yeterince karmaşık olmasından dolayı, bağlam bilgisinin iş akış modeli dilinde değişiklik yapmadan bütünleştirilmesini sağlamıştır. İş akış modellerinin sağladığı özdevinim YB ortamlarındaki olayların KOİ motoruyla işlenmesi sonucu daha üst özdevinim düzeylerine çıkarılmıştır. Önerilen çerçeve için bir ön ürün geliştirilmiş ve uygulanabilirliği gerçek hayattan örneklerin modellenmesiyle ve deneysel ölçümlerle gösterilmiştir.

Anahtar Kelimeler: Yaygın Bilişim, İş Akış Modelleri Uyarlaması, Bağlam Bilinçli Sistemler, Karmaşık Olay İşleme, Olay Temelli Yapı

ACKNOWLEDGEMENTS

I express sincere appreciation to Assist. Prof. Dr. P. Erhan Eren, for his guidance and insight throughout the research. Thanks go to other thesis supervising committee members Assoc. Prof. Dr. Altan Koçyiğit and Assoc. Prof. Dr. İbrahim Körpeoğlu for their suggestions and comments. I offer sincere thanks to other examining committee members, Assist. Prof. Dr. Aysu Betin Can and Assoc. Prof. Dr. Alptekin Temizel.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1. INTRODUCTION

1.1. Purpose of The Study

Recent developments in the microlectronics, wireless communication, and mobile technology show that software and hardware systems are further integrated into everyday lives of people. Small sensors with wireless capabilities sense the environment and process the sensed data or send them to another unit for processing, and the inferences as a result of the processing may be sent to people for guidance through mobile devices, or actuators. Moreover, besides sensors, other context information related to environments of users, properties of devices around users and preferences of users may be used in the same way. Sensing/collecting, sending, processing, inferencing and guiding sequence shows that a system is needed between the lowest level or the sensing/collecting level and highest level or guiding level. The main purpose of this research is to produce a framework which bridges the gap between the lowest-level and highest-level, and to construct a baseline for higher-level applications in order to offer help and guidance to people when they are carrying out their daily activities.

1.2. Justification of The Study

Users engage in many activities in their daily lives. They may need help and guidance when they are participating in these activities. Since the aim of technology is to make the life of people easier and help them stay away from making errors, a software architecture may produce a baseline for fullfilling the needs of people.

There are many different software and hardware solutions in the literature providing help and guidance for people. Especially, the number of mobile applications are increasing currently for this purpose. However, most of these software are customized solutions for specific needs, and most of them are unaware of each other which means they do not communicate among themselves. For this reason, people have to install, manage and use many different types of software for integrating the technology in their daily lives. The proposed framework in this research aims to produce a generalized framework for helping and guiding people. Other developed applications can also produce data as input to this proposed framework and can use the information output by this proposed framework.

1.3. Thesis Study

Producing such a framework for pervasive environments is not a trivial process. Five essential properties of such a framework are determined and solutions are provided. These properties are listed as follows:

- Daily activities of people should be modeled first.
- The models produced for user daily activities should be adaptable to the changing conditions.
- Inferences should be made according to context information.
- The context information should be integrated to the produced models in a loosely-coupled manner.
- There should be different levels of help and guidance for people according to user preferences, context of users.

First of all, daily activities of users should be modeled. People carry out numerous activities as part of their daily lives and these activities as well as their orders change according to conditions in the environment. However, despite such changes, daily activities can be seen as "semi-structured" activities. Generally, while the activities exhibit day-to-day similarities, some changes may occur. For example, a student goes to school every day by bus, but if he has limited time on a particular day, he may take a taxi. People generally need help and guidance in order to organize their daily activities. A software system should know the skeleton of these semi-structured activities, track the users' activities, update the skeleton structure according to the changes in the environment and offer help and guidance to the users by making inferences. Developing such a software system calls for a modeling structure in order to model users' daily activities. In other words, we need technology-independent and transferrable models of human activities (Kawsar, Kortuem, & Altakrouri, 2010).

Such systems have been implemented for the business process management and scientific workflows domains. In business process management, business processes are modeled as workflows and these workflows are executed in a workflow management system. According to (Chappell, 2009), using workflows allows applications to be designed in a unified logic, makes them more understandable and also scales well. Moreover, workflows are especially appropriate for long-running processes which require the application to record the state of a process such as an employee hiring process which might take weeks from interviews to hiring of the employee (Chappell, 2009). Similar to the business processes, user activities in daily life may last a day, a week or a month. Users' daily activities resemble the business processes and workflow systems are adapted for use in the pervasive computing domain. In this domain, workflows can be used for modeling and automating user activities and several attempts have been made in the literature (Abbasi, Ahsan, Shaikh, & Nasir, 2010; Ranganathan & McFaddin, 2004; Tiedeken, Kreher, & Reichert, 2010; Wieland, Kopp, Nicklas, & Leymann, 2007).

Second, the models should be adaptable according to the changing conditions. Since pervasive environments are highly dynamic in nature, it is difficult to know all variations at modeling time. For this reason, workflow processes need to adapt according to the changing conditions (Unger, Eberle, Marconi, & Sirbu, 2010). Adaptation of business processes are already addressed by some studies (Marinovic, Twidle, & Dulay, 2010; Pesic, Schonenberg, & Van Der Aalst, 2007; W M P Van Der Aalst, Adams, Hofstede, & Pesic, 2009), (Smanchat, Ling, & Indrawan, 2008). Nevertheless, most popular workflow languages in the literature do not support all of the adaptation approaches as presented in (Schonenberg, Mans, Russell, Mulyar, & Van Der Aalst, 2008). For pervasive environments, an adaptation approach should allow both design-time and run-time adaptation. Because of the dynamically changing conditions of pervasive environments, runtime adaptation is gaining

more importance for workflow systems. Users should be able to define the abstract structure of their activities, and at runtime they can replace abstract tasks by defining new workflows or selecting from existing workflows. Even these newly created or selected workflows should include abstract tasks, and accordingly users can define their tasks hierarchically.

Third, for pervasive environments, raw context data should be processed and used. The inferences acquired by processing context data can offer direct help and guidance to users or they may be used for driving the workflows. As discussed with the hierarchical adaptation above, replacing an abstract task in a workflow with a new or existing workflow requires implementing a selection mechanism. This selection is generally done by using simple rules (Adams, Ter Hofstede, Edmond, & Van Der Aalst, 2006). Rules are also used in other inline structures of workflow systems such as "Conditional Routing" (*YAWL - User Manual Version 2.2*, 2011). Because activities depend on context, rules may be much more complex than expected. For this reason, rules should use context information collected from pervasive environments and complex rule definitions should be allowed. Rules defined simply in workflow systems can be handled by rule engines which are specialized systems designed for complex rule operations. Currently, rule engines support defining rules by examining patterns, correlation and abstraction, hierarchies and relationships between context data or events, which is known as "Complex Event Processing (CEP)" (*Drools Fusion User Guide Version 5.5.0.Final*, 2012). Using a specialized rule engine with CEP capability can provide new capabilities for workflow systems. However, rule engines should not be intertwined with the workflow systems, as this only increases the complexity of these systems and violates the theory of loose coupling systems (Orton & Weick, 1990).

Fourth, the rule engine and workflow system should be separated to produce a loosely coupled architecture and communication mechanism between them. The solution for designing loosely coupled systems calls for implementing an event-driven architecture (EDA) (Architect & Railways, 2006). EDA also reveals another opportunity for workflow systems. Similar to integrating a rule engine with the workflow engine, context information can also be integrated without changing the structure of the workflow engine. Integrating context data with workflows is known as "context-aware workflows" and it has been addressed by some studies (Abbasi et al., 2010; Abbasi & Shaikh, 2009; Ardissono, Furnari, Goy, Petrone, & Segnan, 2007; Cho et al., 2010; Joohyun Han, Cho, Kim, & Choi, 2006; Hsu, Wu, & Wang, 2010). Most of these studies extend a workflow language with context information which increases the complexity and hinders standardization on workflow languages. Integration approaches are gaining more importance because of the complexities of context management systems and workflow systems. Due to this complexity, they must be handled by separate systems specialized in their areas. Workflow systems should not be intertwined with context management systems.

Lastly, different levels of help and guidance should be offered. In the business process management domain, workflows are used for automating business processes (Wil M P Van Der Aalst, Hofstede, & Weske, 2003). This automation concept may refer to preventing errors by putting conditional constraints between activities, organizing activities in some order by using control structures, and distributing tasks to appropriate workers by utilizing resourcing structures. However existing automation level of workflows is not enough for pervasive environments and software systems should provide highest level of automation in order to be unobtrusive. For this reason, events originating from different sources can be examined by the rule engine with CEP capability and user activities and workflows can be

synchronized automatically according to the rules. Also, rules can be defined in order to guide people and prevent incorrect operations in pervasive environments.

1.4. Contributions of The Study

The framework proposed in this study incorporates a user activity modeling system, an event processing system, two implemented modules and an event channel enabling loosely coupled, reliable and asynchronous messaging between these modules. As such, the framework can help guide people and automate some activities in their daily lives. The proposed framework is compatible with Satyanarayanan's vision which implies that the necessary software such as location tracking, speech recognition and online calendars and hardware structures such as computers, wireless sensors, cameras and wireless communications exist, but the solution is to offer an architecture to integrate these components seamlessly (Satyanarayanan, 2001). Towards this vision, the proposed framework offers a system-level engineering solution in the form of a framework incorporating custom developed software and integrating existing software available in the literature in a loosely-coupled manner.

Specifically, this framework makes contributions regarding:

- how smart-workflows are used for modeling daily user activities,
- how hierarchical adaptation is applied for handling dynamically changing conditions related to these activities,
- how a workflow engine is supplemented by an external rule engine with CEP capability,
- how event-driven architecture is used for integrating these engines in a loosely coupled manner and for incorporating context information into workflows,
- how events and CEP are used for automation of workflows.

1.5. Organization of the Study

In the rest of this thesis, related studies are discussed in Chapter 2. The conceptual architecture of the proposed framework and its properties are discussed in Chapter 3. Chapter 4 includes the prototype implementation and its details. The scenario-based evaluation and evaluation by experimental testing and the results are provided in Chapter 5. Chapter 6 gives the conclusion.

CHAPTER

2. LITERATURE REVIEW

2.1. Dynamic Workflows

Workflow is defined as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (Allen, 2001). Workflow technology influenced the enterprises because it provides methods and technologies for modeling, execution and management of business processes (Wieland, Kaczmarczyk, & Nicklas, 2008). Workflows include many logical steps that are known as "activities". Users or other resources such as machines can interact with activities to automate the activities (Allen, 2001). Completing the activities sequentially, in parallel or in any other pattern allows completing the work in an efficient manner. However, many workflows may exist in an organization and they must be managed carefully. Workflow Management Systems (WMS) are developed for managing workflows. Allen (Allen, 2001) defines WMS as "a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications".

Recently workflows have been used in pervasive computing environments because they are appropriate technologies for simplifying and customizing user interaction with his environment (Ranganathan & McFaddin, 2004). However, workflow technology is useful for predictable and repetitive processes (Abbasi & Shaikh, 2009) which are not suitable for dynamic pervasive environments. For this reason, context-awareness and adaptability must be accommodated into the workflows. Adaptation of workflows are examined in several studies (Smanchat et al., 2008) (Schonenberg et al., 2008) (W M P Van Der Aalst, Adams, et al., 2009) (Gu, Pung, & Zhang, 2004) (Adams, Hofstede, Russell, & Aalst, 2009). Since several different classifications exist, the classification for control-flow perspective of the business processes (Schonenberg et al., 2008) is appropriate for pervasive environments. In this classification, adaptation methods (or flexibility types as named by the authors in (Schonenberg et al., 2008) can be classified as *flexibility by design*, *flexibility by deviation*, *flexibility by underspecification* and *flexibility by change*. *Flexibility by design* is the ability for a workflows system to propose alternative execution paths at design time. The best path is selected at runtime for each process instance. *Flexibility by deviation* is the ability for a workflow system to deviate at runtime from the defined path. The process model is not altered. The changes occur in the running process instances. *Flexibility by underspecification* is the ability for a workflow system to execute an incomplete model. The model is completed at runtime. *Flexibility by change* is the ability for a workflow system to allow changing process models at runtime. Process instances are turned into a new process model. Currently, workflow systems do not support all of these adaptation methods. According to (Mans, van der Aalst, Russell, & Bakker, 2009), the ADEPT1 workflow system supports *flexibility by*

design and *flexibility by change*, YAWL supports *flexibility by design*, *flexibility by underspecification* and *flexibility by change*, FLOWer supports *flexibility by design* and *flexibility by deviation* and Declare supports *flexibility by design*, *flexibility by deviation* and *flexibility by change*. Authors in (Mans et al., 2009) examine a healthcare process and they propose that all types of flexibility are useful for different parts of the process and different workflow systems with different types of flexibility can be used together for supporting all types of flexibility.

There are limited studies related with implementing dynamic workflow technology in pervasive environments. McFaddin and Ranganathan (Ranganathan & McFaddin, 2004) propose a prototype for modeling pervasive environments as workflows based on Business Process Execution Language (BPEL). In this study, a task planner service and BPEL runtime engine are two main elements of the proposed prototype. Task planner generates a workflow according to user task requirements. Task planner gets user requirements and chooses a workflow template from a template service offered by local organization such as a department store. Task planner then customizes this template according to user preferences and context requirements. Moreover, task planner can use some predefined rules by comparing/contrasting the current context parameters for customization. Customized BPEL workflows are then deployed to the workflow engine. If required, user interacts with the workflow for making choices or defining new goal details by using produced web pages. This study does not allow working on multiple workflows. In addition, once the workflows are deployed no change can be made even some context change.

One of the most important dynamic workflow projects is ALLOW Project ("ALLOW - Adaptable Pervasive Flows Project," 2008). ALLOW project started in February 2008 and it was completed by January 2011. ALLOW project aims to integrate people seamlessly to pervasive business and working processes. For this reason, they propose the concept of Adaptive Pervasive Flow (APF). It is argued that real life processes resemble the "flows" either explicitly or implicitly. These flows are named APFs and they can be attached to physical objects or people physically or logically. Because, these objects or people are mobile flows can change according to context. Because formal models can model workflows, current and future behavior of workflows can be detected. ALLOW project proposes a workflow engine that is named as Flow Control Engine (FCE). This engine can run the workflows in distributed manner. Beside FCE, they propose a BPEL based flow-model and language (Unger & Hanna, 2008). However, they want to extend BPEL according to requirements in the project.

Dynamic or adaptable workflow concept is not limited to the pervasive domain. There are many studies in business process modeling area. One of the most important work in this area is the "Declarative Workflows" proposed by Van der Aalst et al (W M P Van Der Aalst, Pesic, & Schonenberg, 2009). In this study they propose the "DECLARE" framework and according to them "instead of explicitly defining the ordering of activities in models, Declare models rely on constraints to implicitly determine the possible ordering of activities (any order that does not violate constraints is allowed)". This means that if the constraints in the process model can be defined there is no need to explicitly determine the order of activities at beginning. The activity, which is needed to be run, can be determined at runtime and this gives a flexibility to choose different activities according to context parameters.

Marinovic et al.(Marinovic et al., 2010) proposes an approach similar to the DECLARE framework. In their approach, they used teleo-reactive programming generally

used in programming robots. They say that teleo-reactive programming allows adaptation mechanisms to context changes and it allows recovering from unexpected situations without restructuring the workflow. According to them, "a TR program is written as an ordered list of condition-action rules called a TR procedure"(Marinovic et al., 2010). However the main problem is the difficulty of modeling workflows in TR procedures and there can be more than one way to represent a workflow.

Similarly Sirin et al. (Sirin, Parsia, & Hendler, 2004) proposes a method for template-based semantic web service composition. They say that instead of building fixed workflows, pre-defined workflow templates can be created and concrete services can be bound to workflows at runtime. Their approach includes using ontologies for describing abstract functionalities and encoding functional and nonfunctional preference parameters in workflows. They also extend the OWL-S language to implement this approach.

One discussion in dynamic workflows concept is to develop a new workflow engine from scratch, like being done in ALLOW project, or select a powerful workflow engine developed previously and extend it. This is not a trivial discussion because many advanced workflow engines are present and even many of them lack capabilities to run all workflow patterns defined by Workflow Patterns initiative (W M P Van Der Aalst, Ter Hofstede, Kiepuszewski, & Barros, 2003). Yet Another Workflow Language (YAWL) is one of these workflow engines that can run all these patterns (W M P Van Der Aalst, Aldred, Dumas, & Ter Hofstede, 2004). YAWL was developed by combining the workflow patterns information with the benefits of Petri Nets. Petri Nets provide a theoretical approach to workflows (W M P Van Der Aalst et al., 2004). YAWL supports both control-flow and data-flow between activities in the flow. However, YAWL workflow engine is not an adaptive engine for context changes. For this reason, they propose the "Worklet" mechanism. In the worklet mechanism, process modeler defines standard activities and deviations. Many deviations, i.e. possible worklets (activities) that can occur, can be defined and one of these worklets can be selected to run at runtime according to context changes. If none of the worklets represent the current context then modeler can define a new worklet. This leads to an evolutionary approach (Adams et al., 2006). Moreover, YAWL workflow engine can be extended and combined with the worklet concept and DECLARE framework to propose more dynamic workflows (W M P Van Der Aalst, Adams, et al., 2009).

As a conclusion, in pervasive environments workflows are used not only for web service composition but also for modeling user activities. Users may define their activities by using abstract components and after that they instantiate these abstract components with more concrete workflows by modeling their activities or by composing software service around them. The time period or the size of the workflow may change according to activities of the user and users have the option of defining their activities from high-level abstract activities to low-level concrete activities. One of the main important challenges is to make workflows adaptable according to changing conditions of the pervasive environments without changing the workflow language and inline working structure of the workflow systems.

2.2. Context-Aware Systems

Context can be defined as any information that can be used to characterize the situation of an entity. This entity can be a person, place, or object (Lee, Ko, Lee, Lee, & Helal, 2007). Context information in mobile environments can be related to user's location,

mobile device's screen characteristics, temperature of the environment, current time and date, user's preferences etc. Mobile and pervasive applications require that these applications must be aware of and adapt to dynamically changing environments. The number of mobile applications and their services has been increasing and the solution for preventing the complexity of services is to detect changes in the environment and adapt to them dynamically (Gu et al., 2004). According to Yang et al. (Yang, Zhang, & Chen, 2008) web services should be context-aware to give users the right information and services in the right place. Context information can be used for service discovery (Lee et al., 2007) and service composition (Boari, Lodolo, Monti, & Pasini, 2008). Moreover, context information can be used with semantic web languages to achieve context reasoning which is an important aspect of smart environments (Chen, Finin, & Joshi, 2004) (Gu et al., 2004).

Modeling context information is one of the most important aspects for context-aware systems since a well-designed context model enables systems to access and process context information (Strang & Linnhoff-Popien, 2004). In a survey (Strang & Linnhoff-Popien, 2004), context modeling approaches are listed as key-value models, markup scheme models, graphical models, object oriented models, logic based models and ontology based models. These context modeling approaches are evaluated according to their properties which are support for distributed composition (dc), partial validation (pv), richness and quality of information (qua), incompleteness and ambiguity (inc), level of formality (for) and applicability to existing environments (app). Table 1 (Strang & Linnhoff-Popien, 2004) shows the results of this evaluation. According to the table, ontology based modeling satisfies most of the requirements. However, some evaluation criteria may be included in this table such as resource consumption when processing the model, and easiness of modeling. Since ontology based modeling cannot satisfy these requirements best, advantages of other approaches may come into prominence.

Table 1 Appropriateness Indication of Context Modeling Approaches

Approach - Requirement	**dc**	**pv**	**qua**	**inc**	**for**	**app**
Key-Value Models	-	-	-	-	-	+
Markup Scheme Models	+	++	-	-	+	++
Graphical Models	-	-	+	-	+	+
Object Oriented Models	++	+	+	+	+	+
Logic Based Models	++	-	-	-	++	-
Ontology Based Models	++	++	+	+	++	+

Truong and Dustdar (Truong & Dustdar, 2009) propose a survey for analyzing techniques and methods for context-aware web service systems. They examine and analyze several systems and propose the components of a web service-based context-aware system. They differentiate the services and applications, which utilize the context information from the components and services, which sense and provide context information. The proposed architecture can be seen in the figure below:

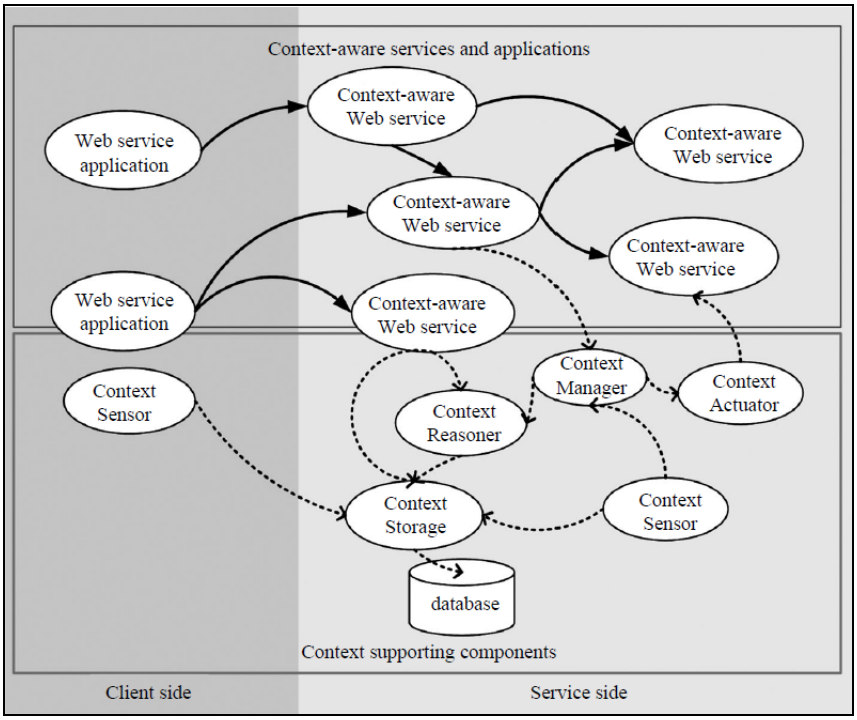

Figure 1 Basic Components In A Web-Service-Based Context-Aware Systems

At the top part of the above figure there exist context-aware services and applications. All the communication between web services and web service applications are based on standard web services protocols such as HTTP and SOAP. At the bottom part of the above figure there exist supporting components. Authors emphasize that the components are generic enough for current web service-based context-aware systems. The figure can also be examined from left to right. At the left, there are client applications. However, client side may also include context sensors. Client applications utilize the web services at the server side. At the right, there are context-aware services, which produce the context for client side. These services can also utilize context components between themselves.

Another context-aware system is Amigo. Amigo Project ("Amigo - Ambient intelligence for the networked home environment," 2008) proposes a middleware for running heterogeneous systems in an interoperable way. This middleware enables to run home appliances, multimedia players and mobile devices to produce a home network even though different companies manufacture them. The main components of Amigo Project's architecture are the programming and deployment framework, context management service, awareness and notification, privacy and security and user modeling and profiling. Amigo's Context Management Service (CMS) provides human-readable results according to sensed data. It has an implementation based on Java technology. CMS can be run on Oscar OSGi framework. CMS components can be installed as bundles to the Oscar. CMS has a CMS Broker component for discovering and using context sources. Meaningful information can be extracted from this context by using Jena toolkit and RDF-based ontology models. CMS allows creating context sources and clients and querying the context information using SPARQL language (Leutnant, Schmalenstroeer, & Poortinga, 2007).

2.3. Event Driven Systems and Complex Event Processing

Michelson (Michelson, 2006) defines an event as "a notable thing that happens inside or outside your business. An event (business or system) may signify a problem or an impending problem, an opportunity, a threshold, or a deviation". In an Event-Driven Architecture, these events are distributed to all interested subscribers and they evaluate these messages and take actions. These actions can be an invocation of services, execution of a business process or information publication (Michelson, 2006). Publish/Subscribe type communication is a major paradigm for event-driven architecture. In Publish/Subscribe communication, messages are sent asynchronously and they are sent to all subscribers, publishers need not know anything about subscribers, recipients are determined by subscriptions instead of being selected explicitly by senders and messages are sent to subscribers who have subscribed before the event is published (Cugola & Jacobsen, 2002). Asynchronous communication is suitable for mobile systems because mobile devices are often turned off or disconnected from the network (Huang & Garcia-Molina, 2004).

In EDA, events are processed according to some rules and the necessary actions are taken (Michelson, 2006). Processing different types of events and analyzing the correlation between events such as casual, temporal and spatial is generally known as CEP (Michelson, 2006). CEP is preferred for processing high-volume and high-speed data generally stored in memory and this produces advantages compared to the traditional database approach since they store data into disk (Yao, Chu, & Li, 2010). The importance and applicability of CEP have been proved in areas such as logistics, finance, manufacturing and energy management (Buchmann & Koldehofe, 2009).

Events may be used to change and drive workflows. This approach is known as Event-Driven Business Process Management (EDBPM). According to Ammon (von Ammon, Emmersberger, Greiner, Springer, & Wolff, 2008) the business process management and complex event processing platforms run in parallel and both work on business processes. Authors emphasize that "The BPM- and the CEP-platform correspond via events which are produced by the BPM-workflow engine and by the IT services which are associated with the business process steps" (von Ammon et al., 2008). Therefore, events or event streams can construct meaningful information to change and update workflows and their task items.

2.4. Similar Studies

The study defined in (Wieland et al., 2007) is the most related study in terms of the concepts used. In that study, authors extend the BPEL workflow language for integrating context and propose the use of Context4BPEL language. Context management software, Nexus, is used for making context queries, handling context events, and routing control flow by context decision. In that study, the architecture is bound to the Nexus platform and an event scheduler is used as an event server. They use a language, Event Registration Language (ERL), for registering events to the Nexus. In contrast, this study is not bound to any specific application for modeling workflows or for context reasoning. In this study, events may come from outside into the messaging system and go outside from the framework using Java Message Service (JMS) standard. Also, a separate rule-engine with CEP capability is used for event processing which allows complex event processing. One of the most important distinctions of this study is the way the workflows are used. In this study, a personal workflow concept is used and different workflow languages can be used for designing workflows. Moreover, even different workflow languages can be used there is no need to modify the workflow languages.

CEVICHE Framework (Hermosillo, Seinturier, & Duchien, 2010) is another mostly related study which aims dynamic business process adaptation by using CEP. In CEVICHE, CEP engine subscribes to the events and adaptation situations are determined by the rules in the CEP engine. When an adaptation situation is determined, CEP engine gives this to the CEVICHE Aspect Manager which uses AO4BPEL, a wrapper on BPEL allowing runtime process adaptation and the adaptation process is handled by this manager. The rules, business processes and adaptation conditions are loaded as a specific XML file named as Standard Business Process Language (SBPL). CEVICHE is not dependent on any CEP engine and only a translation plug-in is required for using different CEP engines. Compared to the framework proposed in this research, CEVICHE runs only with BPEL and AO4BPEL. In the proposed framework different workflow engines can be used. In CEVICHE adaptation is dependent to the methods of AO4BPEL. In the proposed framework, the adaptation methods are not dependent on any framework, since many workflow engines inherently propose advanced adaptation methods. In the proposed framework, these adaptation methods can be supported by CEP engine. CEVICHE uses EDA only for events cloud. However, in the proposed framework, all the modules run in EDA. Since the proposed framework is designed for pervasive environments, different data sources may send data through EDA and also high-level application may get information through EDA.

Another similar study is described in (Ranganathan & McFaddin, 2004). The study shows how workflows help people to interact with the environment by coordinating different web services in a local environment. In the designed architecture, workflow templates are customized according to users' preferences and environment context. This customized BPEL-based workflow is then executed in the BPEL engine. Rules are used for refining BPEL scripts to select the most appropriate web services. That study does not allow any type of workflow adaptation as they specify in their conclusion section. The current study differs from the discussed one in that the concentration is on workflow adaptation and rules are used for making reasoning on context information by using CEP.

In another study (Abbasi & Shaikh, 2009), a framework is proposed for context-aware workflows. Authors offer not to modify the workflow engine so that any workflow engine can be used for this purpose. They use context manager for selecting appropriate activities or sub-workflows from a pool of workflows for replacing the smart activities defined as a placeholder. The study implies that the selection process is done by making reasoning according to domain specific ontologies and sensor data. However, neither the selection process nor the implementation of the framework is given. The study is also extended with the context-aware workflow designer (CAWD) (Abbasi et al., 2010). CAWD can be used for modeling and verifying workflows. It also allows runtime and design time adaptation according to context information. CAWD uses Microsoft Windows Workflow Foundation 3.5 (WF) as the workflow engine, but any other workflow engine can be used with CAWD. In contrast to WF, the workflow engines utilized in current study (YAWL (W M P Van Der Aalst et al., 2004) and jBPM (*jBPM Documentation Version 6.0.1.Final*, 2013)) include adaptation strategies inherently. The adaptation strategies that they want to implement are mostly implemented by YAWL and jBPM already. The main difference of the approach proposed in the current study from CAWD is that context reasoning is made by utilizing complex event processing and the results are used for effectively driving the YAWL and jBPM adaptation strategies. Moreover, the currently proposed approach is an event-driven system which detects the context changes as events in the environment and responds according to these events.

Pryss et. al. (Rüdiger Pryss, Julian Tiedeken, & Manfred Reichert, 2010; Tiedeken et al., 2010) proposes a mobile process engine and a mediation center, known as MARPLE, for enabling mobile assistance. In that work, mobile devices install some software services and communicate with a mediation center in order to execute some fragments of a workflow on the mobile device. Mobile process engine is developed based on the ADEPT (Reichert, Rinderle, & Dadam, 2003) process model and it includes the ADEPT's correctness notions and verification procedures. However, mobile process engine can only control sequence, parallel and conditional routing execution patterns. In addition to execution patterns, the mobile process engine allows adaptation on the mobile devices. This adaptation only includes addition and deletion of single activities and only human activities can be added or deleted by using local context information. Unlike that work, this one is based on a central workflow engine and mobile devices can access the management interface by using web browsers. Also, the currently proposed framework is not constrained with the use of local context information. A workflow can access global context data if they are in the messaging system. The concentration on the adaptation concept in the framework is limited with the subprocesses adaptation concept. One of the workflow engines, YAWL, does not allow adding or deleting task within the workflow. The other workflow engine, jBPM allows adding or deleting tasks while using "Ad-Hoc Subprocesses" concept. However, since the proposed framework is not dependant on any workflow engine different adaptation concepts may be used and these adaptation concepts can be supplemented by other components in the framework. The main difference of this framework is the utilization of rule-based adaptation and automation extended with complex event-processing.

A software solution is proposed by Red Hat Company as JBoss BRMS ("Red Hat JBoss BRMS," n.d.) for enterprise business process automation. In this solution, business process management and rules management are integrated together for businesses to manage their business processes. This software only allows business rules and processes to run in the same knowledge base and depends on the JBoss products. In contrast, the proposed solution in the current study is proposed for pervasive environments and allows different software to be used for same purposes due to the loosely-coupled integration architecture. Also, rule management is used not only for processes but also for context reasoning and automation.

Project Aura (Garlan, Siewiorek, Smailagic, & Steenkiste, 2002) designs, implements, deploys, and evaluates a system for spanning the personal information to the wearable, handheld, desktop and infrastructure devices. By doing this, the project aims to provide distraction-free ubiquitous computing to the users. According to Project Aura, required hardware and software technologies are available, but the integration of these technologies to provide a whole needs more research. Project Aura was designed based on two scenarios. Project Aura developed a compute or data-staging server for helping resource constraint devices to make processing and storing operations. Project Aura also developed a wireless bandwidth advisor for estimating future congestion level of networks and making appropriate connection decisions. Moreover, Project Aura developed location detection methods. Project Aura's two important services are supporting user mobility and protecting user from variations of resources. Project Aura only supports location detection and reasoning in terms of context processing. Other context parameters are not included in context processing. Project Aura was developed based on two scenarios. It does not propose generic solutions for pervasive environments.

Context Aware Middleware for Ubiquitous Systems (CAMUS) (Kiani, Riaz, Zhung, Lee, & Lee, 2005) was developed for providing context synthesis and provision services to

users in distributed environments. CAMUS can be used for accessing data from sensors and devices. This data can be used with ontology repositories to infer meaningful context. CAMUS offers a centralized middleware solution. In this system, context information is stored into a centralized repository for later context reasoning activities. CAMUS separates the environment into domains such as home domain and university domain. Every domain has its own centralized repository. CAMUS is a SOA based system because it uses Jini technology. CAMUS is limited with only context processing and it does not support event based communication. Moreover, CAMUS does not include composition of different services for achieving a complex task.

Pervaho (Eugster, Garbinato, & Holzer, 2006) is a platform for developing and testing mobile ad hoc applications. Pervaho extends J2ME to provide mobile devices to communicate anonymously and asynchronously through a location-based publish/subscribe service. Pervaho also includes a mobile application testing tool which is Phomo, a phone motion simulator. In Pervaho, client devices include a module to detect location changes and send these changes to the central server. After that, the server examines the potential matches according to location change and sends messages to the related subscribers. Pervaho uses UDP between client-server communications. Pervaho also uses J2ME's location API which is known as JSR-179 as location detection mechanism. Pervaho only supports location information as a context parameter.

Cooltown (Kindberg et al., 2002) project was developed in HP labs to bridge the World Wide Web and the physical world. In this project, web technology is pushed to the things like printers, radios and automobiles. Moreover, Cooltown enables discovering non-electronic things such as CDs, books, printed papers. According to Cooltown, physical things in our life have information in the web but there is not any connection between these physical things and the information. For this reason, they proposed "web presence". Web presence extends "Home Page" concept. A home page can be correlated with the physical things. This home page can be put into the web server in the devices if it has the required processor and storage capacities. For the non-electronic things, home page is put into web server and some access mechanisms such as barcodes can be used to relate the thing with the web page. By doing this, a museum visitor can get information about a picture by reading a barcode on the picture with his PDA's barcode reader. PDA then shows the related information web page from the museum's web server. Cooltown offers different infrastructures for people, places and things. For example, it offers "eSquirt" for printing a page without sending the whole page. Printer can get the web page from the server by reading a barcode or a tag from the thing. It uses internet and web page concept instead of using Jini or CORBA because they think that it is the best middleware concept. Cooltown uses only location information in terms of context. Cooltown does not take into account the events and context changes.

CARMEN (Bellavista, Corradi, Montanari, & Stefanelli, 2003) is a mobile agent based context aware middleware. CARMEN offers services to the mobile users according to the characteristics of the context. According to the part of a given scenario, a user can read news from a laptop connected wirelessly to the server at boarding gate while he is flying. After landing, user can continue to read news but this time his palmtop can connect to server at the new station and it can get localized news according to new location such as weather and traffic reports. This scenario shows that requalification of the accessed resources based on location, enforcing the policies in new station and applying the user's policies are needed. For these types of requirements CARMEN provides Metadata Manager for storing user

preferences and policies and Context Manager for determining location, resource parameters etc. CARMEN uses Mobile Agents because they imply that several proposals offer using proxies acting on behalf of limited devices. Proxies can support disconnected operations and caching results. Also, mobile agents offer management decentralization support which leads to scalable solutions and avoid management bottlenecks. CARMEN includes an Event Manager to detect changes such as a user changing his location and the connected access point. But, this is limited with some standard hardware components because these components must support some software for mobile agents to migrate. CARMEN uses Secure and Open Mobile Agent (SOMA) platform as mobile agent platform. CARMEN uses *shadow proxy* concept which is a personal mobile proxy migrating in fixed networks to follow users movements and act as intermediary between wireless devices and user's context. CARMEN clients include a service component, which announces entering the CARMEN domain and exploits shadow proxy to send/receive messages. In CARMEN these clients are implemented for three environments: J2ME for Pocket PC with IEEE 802.11b connection, C for Ericsson environment with Bluetooth connection, and in Palm OS devices. CARMEN is dependent on hardware platforms.

Wu, Liao and Fu (Wu, Liao, & Fu, 2007) developed a service oriented smart home middleware based on OSGi and mobile agents. They imply that OSGi platform for smart home architectures are used as server-centric and this is not suitable for pervasive computing environments. For this reason they use OSGi bundles on all devices in smart home. These OSGi enabled devices can communicate between them based on service-oriented approach. To support service-oriented approach every OSGi based device installs a web service gateway bundle. Supporting service-oriented approach enables this platform to communicate with non-OSGi platforms. In this design mobile agents help these devices to cooperate. A device can download an agent host bundle to work as an agent client. After that it can communicate through web services. Mobile agents can migrate to other agent hosts on different OSGi platforms and carry some services to them. In reality mobile agent migration mechanism is transporting XML-based MASML documents over web services. Using OSGi on devices can consume much processing power, storage and energy. OSGi is a Java based platform and needs a Java Virtual Machine to work. Moreover, a web services bundle is needed to support service-oriented architecture which also increases the resource usage. This smart home architecture does not include composition of web services for complex operations.

MobiPADS (Chan & Chuang, 2003) is a reflective middleware for context aware mobile computing. MobiPADS platform includes a MobiPADS server connected to the wired networks and MobiPADS clients on mobile devices connected wirelessly to the network. Both MobiPADS server and client include two parts: the system components and MobiPADS service space. System components provide essential services such as service discovery, event register and configuration manager to the service space. Service space includes mobilet chains, contextual event objects and metaobjects. Mobilet chains allow mobile applications above MobiPADS client to benefit from the aggregated functionalities of a mobilet which can access the system components. Contextual event objects monitor context changes and report these changes to the subscribers. Metaobjects reflect the configuration of the composite events and service chain. Through metaobjects event compositions and service chain can be adaptively reconfigured. MobiPADS allows context-awareness by using an event notification model. It supports subscription from system components, the mobilets, and mobile applications. Moreover, MobiPADS event model

allows composite events such as monitoring network load, CPU utilization and battery to construct “high load” event. Another important component of MobiPADS is dynamic service composition. MobiPADS responds to the changes in the context by reconfiguring the mobilets service chains. MobiPADS system is implemented in Java platform. MobiPADS uses SOA approach but it does not use web services. It has proprietary solutions for service compositions. An application must use the MobiPADS API to access the MobiPADS platform components. This hinders interoperable communication with other systems.

AlfredO (Rellermeyer, Riva, & Alonso, 2008) is a lightweight middleware, which allows flexible interaction between mobile devices. AlfredO is a service-based middleware. This allows mobile phones to interact with devices by using stateless service interfaces instead of downloading, and installing software packages. AlfredO is based on flexible client-server architecture. This architecture allows a mobile phone to work as a presentation tier, discards the completion of the work or work as a logic tier, and processes the work in high-load networks. AlfredO is based on R-OSGi middleware which allows OSGi services to be distributed across several devices. AlfredO can produce different rendering of same interfaces on different devices. If mobile phones do not support Java, AlfredO can produce browser-based solutions. Two sample applications are developed for testing AlfredO: MouseController and AlfredOShop. AlfredO resource consumption and latency is tested with these applications and found usable. AlfredO is related with the resource consumption and multimodal data on mobile phones. It does not include context and event-driven modules. It is based on services but it uses Java service interfaces.

Gaia (Roman et al., 2002) is a generic environment that integrates physical spaces and their ubiquitous devices to form a programmable computing and communication system. Gaia resembles the operating systems because it tries to do the tasks that all applications need to use. These tasks include events, presence of entities such as devices, users and services, discovery and naming. Gaia uses CORBA as a distributed computing environment. Gaia allows applications to obtain contextual information. Applications query a context provider or listen to an event channel to get context information. Gaia supports ontologies and uses first-order predicates. This means that predicate name indicates the context such as location and its arguments describe the properties of the context. For example, if the predicate is location, the second argument can be “person” and the third argument can be “entering”. Gaia uses DAML+OIL for writing ontologies. Gaia uses Rule-based context synthesizers and machine learning techniques to make context reasoning (Dargie, 2009). The main disadvantage of Gaia is not using SOA. CORBA is not an interoperable standard for communication. Gaia only provides core services to the applications but it does not provide a solution for coordination of these services.

SOCRADES (De Souza et al., 2008) is a middleware for business integration such as integrating enterprise applications with web services enabled devices. According to SOCRADES project, there must be efficient collaboration between devices and backend business systems such as ERPs. This can be done by adding web services capabilities to the things. However, a middleware is needed between devices and backend systems for reliable and secure integration. SOCRADES is a web service based shop floor integration infrastructure. SOCRADES connects Smart Objects of shop floor (manufacturing machines) to high-level back-end systems such as ERP systems. Devices Profile for Web Services (DPWS) is used as a standard for web services. Authors offer the middleware and show a reference implementation for two smart objects. They show how these smart objects can be

used in a web service based infrastructure. SOCRADES middleware has the following features:

- Brokered Access to Devices: An intermediate component exists between web service clients and servers. By this way, asynchronous invocations of occasionally connected devices can be handled by publish/subscribe systems.
- Service Discovery: A central repository called Device Manager and Monitor stores the devices and its services. By the way, all devices can be accessed by ERP systems whether they have connection or not to the shop floor. However, the discovery area is limited with the local network because DPWS uses UDP multicast, which cannot be used in global networks.
- Device Supervision: Device Management and Monitor and DPWS Historian store information about the state of the DPWS enabled physical devices. This data can be used for later examination and diagnosis.
- Service Life Cycle Management: SOCRADES enables web services on devices to be updated at runtime.
- Cross-Layer Service Catalog: Composed Services Runtime enables service composition at the middleware layer. This BPEL based system is put in the middleware because it is the intersection of shop-floor and enterprise level services.
- Security Support: SOCRADES supports role based security. Event filtering are based on roles is possible. Message integrity and confidentiality is inherently supported by DPWS's WS-Security standard.

SOCRADES middleware lacks complex context reasoning support. It is developed for enterprise automation systems for controlling devices in local networks and does not target global management of devices for everyday activities.

CoCA (Ejigu, Scuturici, & Brunie, 2007) is a context-aware platform for pervasive computing environments. CoCA architecture includes four components. The first component is an "Interface Manager". It is an API-based interface managing the user interface of the architecture and the interaction of CoCA between other systems. It also triggers the actions produced by the architecture such as setting a vibrate mode of a phone according to inferences made by the architecture. The second component is "Data Source". It provides the necessary data to the core service of the architecture. Data is modeled as GCoM model (a generic context management model) which includes context capture, context ontology and rule capture elements. The third component is "Core Service". This component is the main context reasoning component. It populates the ontologies with context data and applies reasoning techniques on them. The last component is "Supplementary Service". It includes a knowledge discovery service for adding learning capacity to the architecture and a collaboration service for distributing resource exploiting processes to other computers. Because CoCA uses semantic reasoning, it requires an ontology and context data to be populated with this ontology. Then, the derivation rules can be used for making inferences. CoCA does not include a modeling component. For this reason, it is hard to track user activities unless they are hard-coded to the rules. Moreover, semantic reasoning is a computational-intensive task and it is affected by the size of the data and the number of rules (Wang, Zhang, Gu, & Pung, 2004).

CHAPTER

3. CONCEPTUAL ARCHITECTURE

In this section, conceptual architecture of the proposed framework, *SOMNIUM*, is given. The conceptual design incorporates a workflow engine, a rule engine, and two *SOMNIUM* modules, together with an event-driven architecture as shown in Figure 1. The input to the framework may come from different data sources such as sensors, devices and a variety of context sources. The input may be raw context data or high level information. The output includes the information which helps and guides people. The format or syntax of the input/output information is out of the scope of this research since several styles can be found readily available in the literature such as XML. The following sections explain why these building blocks are needed and how they provide solutions for helping and guiding users in pervasive environments.

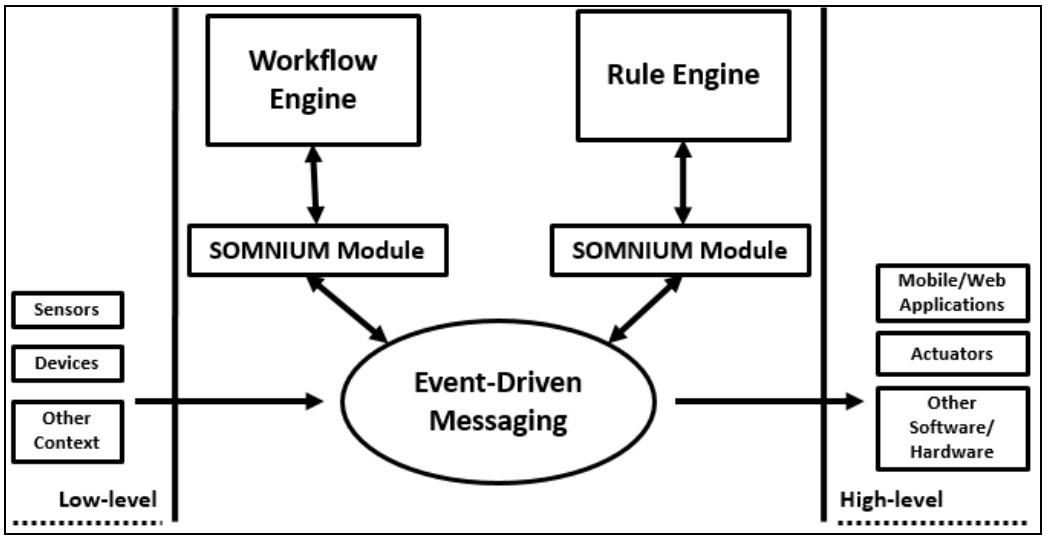

Figure 1 Conceptual Architecture of the *SOMNIUM* Framework

3.1. Workflow Engine

A workflow is defined as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"(Allen, 2001). A workflow is implemented by using a language and is imported to and executed by a Workflow Management System (WfMS) (Eberle, Leymann, & Unger, 2011). "Workflows can be described using a formal model, such as high-level Petri Nets, or through a structured 'programming' language such as BPEL" (Marinovic et al., 2010). The above definition is derived from the business process management domain. In the pervasive computing domain, "smart workflow" concept is offered to bridge the gap between the business processes and the context information (Wieland, Kaczmarczyk, et al., 2008). Workflows which can be adapted automatically or semi-automatically according to context information are defined as "Smart Workflows" (Wieland, Nicklas, & Leymann, 2008). Smart Workflow concept is referred to in different ways by some other studies, such as "Person-centric Flow" (Unger, Eberle, Leymann, & Wagner, 2010), "Adaptable Pervasive Flow" (Wolf, Herrmann, & Rothermel, 2009), and "Situated Flow" (Kortuem, Kawsar, & Altakrouri, 2010). Even though the names used are different, the concepts are similar. Smart Workflows are similar to classical workflows (Kortuem et al., 2010). However, they are used for modeling activities of only one person and tasks can be completed manually by its owner or automatically according to the context. Smart workflows include modeling of past, current and future activities (Herrmann, Rothermel, Kortuem, & Dulay, 2008).

Using workflows allows users to model their activities and make changes on them according to changing conditions. A study shows that workflows are appropriate for modeling human daily activities (Unger, Eberle, & Leymann, 2010). For this reason, workflows are used for pervasive environments in some studies (Herrmann et al., 2008; Kortuem et al., 2010; Ranganathan & McFaddin, 2004). Humans perform activities in daily life according to some order and workflows are the IT-representations of these flow of activities (Unger, Eberle, Leymann, et al., 2010). A person explicitly completes, suspends or cancels tasks in the workflow according to his activities, so that the user activities and the workflow are synchronized.

The synchronization of the user activities with the workflows can be done automatically by using more intelligent techniques. One of these techniques, activity sensing, is used for automatically advancing the workflow (Herrmann et al., 2008). A study examines the relation between Activity Theory and workflows (Adams, Edmond, & Ter Hofstede, 2003). They describe that Activity Theory offers some principles in order to understand human activity. Authors also determine six criteria that a workflow management system should support: flexibility and re-use, adaptation via reflection, dynamic evolution, locality of change, comprehensibility of models, and exceptions as "first-class citizens". They match the principles of Activity Theory with these criteria and show that workflow products support the principles of Activity Theory.

The core component of the *SOMNIUM* framework is the workflow engine, also referred to as the user activity modeling system, where the workflow concept is used for modeling user activities. Studies discussed above show that workflows can be used for modeling daily activities. In the *SOMNIUM* architecture, the workflows represent the ordered activities in the daily life of a person. However, workflows are not dependent on the length or the size of the activity. This means that a workflow can represent activities in a

short period of time such as ten seconds, or longer periods such as days or weeks. Also, a workflow can represent few activities of a person such as when he is at home or many more activities of a person when he is at work.

Generally, for enterprise usage, each task in the workflow is assigned to a different person. Even though the *SOMNIUM* framework supports this, the focus is on the personal workflow concept, where all tasks in a workflow are owned by the same person. Accordingly, processes with tasks to be assigned to different people are not considered. Instead, if a task is to be completed by a person, that task should be a part of his workflow. If there is an interaction or dependency between tasks owned by different people, this can be handled by their own workflows interacting by sending messages into the framework which are then interpreted by other modules in the framework. In *SOMNIUM*, tasks are completed by the owner manually, by the framework automatically, or by the workflow engine (some structures used in the workflow engines may do this such as worklet discussed in section 4.2) automatically. Since a workflow is owned by only one person, there is no concept of "process fragments" (Tiedeken et al., 2010; Unger, Eberle, Marconi, et al., 2010). Also, the framework does not allow the fragments of the workflow to run outside the centralized workflow engine.

In contrast with the systems accessing only local context data (Wolf et al., 2009), workflows in the *SOMNIUM* framework, can access global context information, since all context information is fed into the messaging system.

3.2. Workflow Adaptation

Pervasive environments are highly dynamic; hence the conditions and the characteristics of the environment can change rapidly. Workflow systems should provide support for these changing conditions in order to be useful in pervasive environments. Existing workflow systems support different adaptation methods. The studies presented in (Schonenberg et al., 2008) and (Smanchat et al., 2008) discuss various adaptation approaches. The main discussion about adaptation is presented in (Schonenberg et al., 2008) under the "Process Flexibility" label. In that discussion, the main differentiation is on the structure of the processes. The structure of a process can be defined by using the "Imperative Approach", or the "Declarative Approach". In the imperative approach, a process model is defined as a detailed specification and this specification is executed step by step (Pesic et al., 2007). In the declarative approach, a set of constraints are defined and any order of tasks not violating these constrains is allowed to execute (Pesic et al., 2007). Both approaches have some advantages and disadvantages. Declarative approaches are more appropriate for semi-structured processes. However, if the workflow is large and has many constraints, it is hard to use declarative workflows. Additionally, because the processes are not connected at design time, users cannot understand the overall workflow (Leymann, Unger, & Wagner, 2010). Imperative approaches lead to over-specification in the process definition and they are mostly appropriate for only highly-structured processes. However, the second discrimination in the article which is "Types of Flexibility" can allow imperative languages to be appropriate for semi-structured activities similar to most of the users' daily activities.

One of the "Types of Flexibility" is "Flexibility by Underspecification" (Schonenberg et al., 2008) and this concept is useful if the details of the process cannot be known at design time and the process can change according to runtime conditions. This flexibility type allows defining abstract processes for unknown or uncertain processes. These

abstract processes can be filled by concrete processes at runtime according to the existing conditions, while the concrete processes can be created at runtime or they can be selected from the previously created processes. Even this new concrete process can include abstract processes in itself. By this way, a user can define abstract activities for his daily life and he can hierarchically fill these abstract processes with the concrete ones at that moment. However, if the user wants to select from the already created processes, this selection may not be easy, since it may depend on the runtime context. Context data may be examined according to relations, hierarchies, causalities and patterns between them. For this reason, complex rules should be defined for this selection and currently, workflow engines don't support this type of complex-rule definitions.

In *SOMNIUM*, two imperative workflow languages supporting the "Flexibility by Underspecification" concept are utilized. The *SOMNIUM* framework can support other types of adaptation methods if the used workflow software supports these methods. The aim here is not to be bound to a specific adaptation method. However, in pervasive environments run-time adaptation is more important and describing how the framework supports one of these run-time adaptation methods shows the power of the framework.

Imperative languages supported with flexibility considerations allow easily modeling of users' everyday activities. A set of daily activities generally resembles that of other days, but may be subject to small changes according to changing daily situations. Moreover, people generally plan their activities starting with a high abstraction level, and going into more concrete levels; hence, "Flexibility by Underspecification" concept used in *SOMNIUM* allows people to define workflows hierarchically in order to match to their daily activity plans.

As an example, assume that a person goes to work every day. He can add an abstract task "Go to Work" to his daily workflow. He can later replace this task with another sub-workflow according to his conditions. If he has enough time and prefers to go to work by bus, he can easily use "Go by Bus" workflow as a replacement; or use "Take a taxi" if he is short on time. The hierarchical workflow concept is also used for modeling activities corresponding to different time intervals. For example, a university student can create a workflow which represents the activities of a one semester long course. Assume that this workflow includes homework, projects, exams and lectures organized by date. The student can then replace a lecture with another sub-workflow, which includes the activities such as "go to school, attend lecture, and take a break". Similarly, the student can replace these activities by other sub-workflows such as replacing "Go to School" activity with "Go by Bus" sub-workflow.

3.3. Rule Engine and Complex Event Processing

Generally, workflow systems incorporate rules for their inline operations such as selecting one of the many tasks according to some conditions (conditional routing) and replacing an abstract task by a sub-workflow selected from many existing workflows (subprocess selection) (*YAWL - User Manual Version 2.2*, 2011). However, these rule sets are not enough for use in pervasive environments, where more complex rule sets are needed, such as accumulating user data for one week period and acting on this data. In order to handle such complex rule definitions, the use of an external system specifically designed for this purpose, i.e. a rule engine with CEP capability, is proposed in this study. This rule engine is a software system producing outcomes by applying rules in any form on collected

data ("Drools Expert User Guide," n.d.). A typical rule engine includes an inference engine, a rule base, and a working memory (Hill, 2003). It utilizes some special algorithms, such as Rete (Forgy, 1982), in order to make inferences on the real data in working memory by applying rules stored in the rule base. Real data include static data stored in the memory or dynamic state changes occurring continuously.

State changes are generally known as "Event" (*Drools Fusion User Guide Version 5.5.0.Final*, 2012). CEP refers to processing of these events for the purpose of identifying meaningful events by employing techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes (*Drools Fusion User Guide Version 5.5.0.Final*, 2012). Hence, a rule engine with CEP capability enables us to define complex rule patterns which are typically not available in current workflow engines.

In the *SOMNIUM* framework, a rule engine with CEP capability is utilized. The rule engine examines the context data and produces actions such as "complete task" and "cancel workflow". The rule engine has the capability of examining the historical data and making inferences, with the help of CEP. When using the workflow engine along with the rule engine these two are separated instead of being tightly integrated together. This provides two main advantages for the framework. First, no modifications are made to any of the systems' inline processing structures. So, different workflow engines and rule engines readily available can be used as part of the *SOMNIUM* framework, enabling modularity. Second, running workflows are not affected by changes in the rule base. When the rule base in the rule engine is dynamically changed at runtime, workflows continue to run in the workflow engine without interruption. As an example, assume that a person has a workflow for his daily activities and he has a task in this workflow for outdoor sports. This task depends on the temperature and the time, and if the temperature is higher than 15°C and the time is five o'clock the task is automatically enabled. Today, it is rainy and he recognizes that these rules are not enough for the decision to enable outdoor sports task. He does not have to cancel the workflow or modify it. He only adds another rule to the rule engine, because the rules are not integrated with the workflows.

3.4. Event-Driven Architecture

In everyday life, many events occur such as "the door is closed", "the lecture is started", "the bus left the bus stop". If a software system needs to use these events and take some actions according to them, events should be distributed to all interested parties. The structure which takes events and distributes them to all interested parties immediately for taking necessary actions is referred to as the EDA (Michelson, 2006). EDA uses an asynchronous publish-subscribe type of pattern and as such, publishers do not know anything about subscribers (Architect & Railways, 2006). If the modules are independent and horizontal communication is needed between them, EDA is implemented supporting loose coupling (Architect & Railways, 2006).

Properties of EDA show that the *SOMNIUM* framework fits this architecture. There are three main reasons for using EDA. First, modularity is enabled in the framework, allowing the components to work as independent as possible while communicating with other components as loosely as possible. Second, while the framework becomes context-aware, the context information is not intertwined with the workflow engine. The context-awareness issues are discussed in section 3.5. Third, EDA is used for getting input from

context sources and sending output to high-level hardware/software for helping and guiding people. An event or events may occur by processing the context data coming from different context sources through EDA and when an event occurs, an end-level user or even an actuator may be warned through EDA for taking necessary actions. The rule engine and workflow engine continuously listen to each other and the context sources, and they publish events according to actions that are taken by examining the context data. Subscribers of these events can get the events from the EDA. For example, assume that a student is late for the lecture and he created a rule beforehand which warns him when he is late. The framework makes a decision when the "lecture started" event is constructed in the framework and no event has come from the RFID sensor attached to the classroom's door showing that the student is in the classroom. When the decision is made by the framework, an event is published to the EDA, and because a mobile application on the student's smart phone listens to these events, the student is warned by the mobile application.

The publish-subscribe mechanism widely used in event driven architectures is utilized for loose coupling of the workflow engine and the rule engine. Data sources in the pervasive environments such as sensors, mobile devices, and web services send events or context information into topics created beforehand for every data source. These events are examined by two *SOMNIUM* modules developed for interacting with the workflow engine and the rule engine. These two modules also use topic-based subscription to listen to the data sources and each other. They examine the events and take necessary actions on the workflow engine and the rule engine, such as "complete a task", "cancel a workflow", "listen to a specific event and send it to rule engine". As a result of this, the engines are used without requiring any modifications.

3.5. Context-Aware Workflows

Users interact with their environments while performing daily activities. Characteristics of the environments effect the activities' time, location, order, and the way they are implemented. Because workflows are used in the *SOMNIUM* framework for modeling users' daily activities, these characteristics also affect the structures of the workflows. In pervasive computing, characteristics of the environment, users and the system are referred to as "Context". Because context information affects users' activities, workflow systems should allow context-aware operations. Preprocessing of context, or more generally, context reasoning, is the process of getting more meaningful information. Different types of context data or historical context data can be interpreted together in order to extract high-level information. There are different reasoning techniques in the literature (Bikakis, Patkos, Antoniou, & Plexousakis, 2008). Rule-based reasoning is one of these techniques, where a set of if-then rules is used for inference. Rules can be created easily and it is easy to work with rules (Krumm, 2009). Despite these advantages, rule-based reasoning has some disadvantages. It is hard to detect conflicts between rules, control large number of rules and execute many rules efficiently (Krumm, 2009). However, current research shows that rule-based systems can be extended with CEP techniques. By CEP, patterns, relations or hierarchies between many different context data or context data coming from same source in different time periods are extracted. A CEP engine is a stateful rule engine optimized for long-running and processing multiple event streams coming from different sources while typical rule-based systems are stateless. One of the most important advantages of CEP over rule-based systems is comparing event-histories over-time. For this reason, CEP engines continuously store events and their timestamps. Using these timestamps CEP engines can make a kind of temporal reasoning by examining events over a period of time. While rule-

based systems only process rule-sets, CEP engines use event-based information aggregation through stateful sessions for determining different relationships between events (Vincent, 2007).

There already exist some workflow systems allowing context-aware operations. However, these systems integrate context information into the workflow system and represent context information within the workflow language (Abbasi et al., 2010; Cho, Choi, & Choi, 2007; J Han, Cho, & Choi, 2005; Wieland, Kaczmarczyk, et al., 2008; Wieland et al., 2007), leading to their tight coupling. While the existing workflow languages are already complex due to the nature of complexities of processes, integration of context information with the workflow language increases the complexity further. In the *SOMNIUM* framework, context information is not tightly integrated with the workflows. If context data are needed somewhere in the workflow, these data are examined or processed by the rule engine outside the workflow and only the outputs are sent back to the workflow by mapping them to the variables defined in the workflow. Also adaptation decisions are given by the rule engine which examines the context data published by various data sources in the pervasive environment into the topics defined in the EDA platform. Decoupling of context management system from the workflow system and handling of context reasoning and adaptation-by-context operations in separate modules allow each module to specialize while continuing to work together for producing complete solutions.

3.6. Workflow Automation

In the business process management domain, workflows are used for automating business processes (Wil M P Van Der Aalst et al., 2003). This automation concept may refer to preventing errors by putting conditional constraints between activities, organizing activities in some order by using control structures, and distributing tasks to appropriate workers by utilizing resourcing structures. By this way workflows eliminate human errors and decrease the time needed to complete processes (Allen, 2001). However, automation is more than that. Automation is defined as "full or partial replacement of a function previously carried out by the human operator" (Parasuraman, Sheridan, & Wickens, 2000). Automation has various levels and these levels vary from manual performance to full automation (Parasuraman et al., 2000). The automation level of workflows is not enough for pervasive environments and software systems should provide highest level of automation for being unobtrusive. For this reason, software systems should track user activities, synchronize the system with user activities, inform users about possible operations or act autonomously without asking.

This type of automation can be enabled by using event processing. Events occurring throughout a day, such as "the door is closed", "the course is started", "the bus has just left" are processed by a rule engine with CEP capability and workflows are synchronized according to the results obtained from the complex rules. The event-driven architecture and processing of events by the rules allow the *SOMNIUM* framework to detect activities of the people and automatically complete, suspend, cancel tasks and workflows without manual interaction. Rules are also extended to give the users some help and guidance information. For example, after the lecture ends, system may inform the user about the time left for planned activities and he can select one or more of the activities according to the recommendations.

The *SOMNIUM* framework is designed to represent different levels of automation. Users can use workflows without defining any rules or making subscriptions to context sources. This type of usage is the lowest automation level since the user designs a workflow for his activities and he manually enacts the workflow. He does not use any sensor data or high-level context information. However, he can increase the automation level by making subscriptions to sensors, web services or other data sources that are sending messages to the messaging system. He can use these data to automatically enact workflows by adding rules to the system. For example, when a door RFID sensor sends information upon the user leaving home, a previously defined rule processes this message and automatically completes the work item "Leave Home" in the user's workflow. Moreover, context reasoning can be enabled by rule engine to make more complex inferences from the raw context data. By this way, the system can examine data coming from different sources or examine historical data coming from one or more sources to automatically enact workflows, hence increasing the automation level.

CHAPTER

4. IMPLEMENTATION

Following the conceptual description of the *SOMNIUM* framework, its implementation is described in this section. The conceptual architecture of the *SOMNIUM* framework includes a workflow engine, a rule engine, an event-driven messaging system and two *SOMNIUM* modules. For the prototype implementation, YAWL and JBoss jBPM are selected as workflow engines, JBoss Drools and Esper are selected as rule engines, and Apache ActiveMQ and JBoss HornetQ ("HornetQ - putting the buzz in messaging," n.d.) are selected as messaging systems. A modular approach is preferred in order to avoid dependence on any specific software. As a result, two different types of workflow engines, rule engines and messaging systems are used in the prototype interchangeably to demonstrate this modularity. However, only "Flexibility by Underspecification" adaptation mechanism is demonstrated in the framework. Other workflow adaptation techniques can be used with other workflow engines if the used workflow engines support different adaptation approaches. Two *SOMNIUM* modules referred to as *SomniumBPM* and *SomniumCEP* are implemented as well. *SomniumBPM* manages the YAWL workflow engine and jBPM workflow engine interchangeably and *SomniumCEP* manages Drools rule engine and Esper rule engine interchangeably. Apache ActiveMQ and JBoss HornetQ event-driven messaging systems are also used interchangeably for connecting all components of the *SOMNIUM* framework in a loosely coupled manner. Figure 2 shows the implementation architecture of the *SOMNIUM* framework. Next, the reasons for selecting and implementing these components and how they provide solutions for the given problems are discussed.

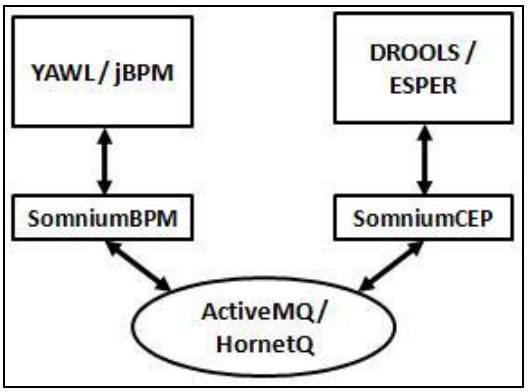

Figure 2 *SOMNIUM* Framework Implementation

4.1 SOMNIUM Modules

Two *SOMNIUM* modules are implemented for managing the workflow engines and the rule engines. *SomniumBPM* for managing YAWL workflow engine and jBPM workflow engine, *SomniumCEP* for managing Drools rule engine and Esper rule engine are implemented.

SomniumBPM has two responsibilities. First, it listens to events occurring in the workflow engines and publishes these events to the topic of the YAWL and jBPM in the messaging system (i.e. Apache ActiveMQ or HornetQ, for details see section 4.4 Event-Driven Messaging). These events may include workflow engine inline operations such as "work item enabled", "work item cancelled", "case completed", "work item status changed", and "case cancelled". Second, *SomniumBPM* subscribes to the rule engine topic, gets the messages from the rule engine, and applies the received commands in the messages to the YAWL or jBPM objects. These commands may include "load or launch workflow specifications", "complete, cancel, suspend cases (running instances of workflow specifications)", "complete, cancel, suspend work item", and "update case or work item data".

SomniumCEP manages the rule engines and it has two responsibilities. First, *SomniumCEP* subscribes and listens to the topics in the messaging system. Different data sources in pervasive environments can publish messages to these topics. When a message arrives, *SomniumCEP* gets the message, parses it, then sends it to the rule engine and fires the rules. Second, *SomniumCEP* waits for the results of firing rules, gets these results from the rule engine and sends them to its topic in the messaging system.

SomniumBPM and *SomniumCEP* modules are implemented as HTTP servlet by using the Java programming language. *SomniumBPM* and the YAWL engine communicate through HTTP messaging. The part of the *SomniumBPM* communicating with the YAWL is implemented as a YAWL Custom Service (Arthur ter Hofstede, 2010) and it also uses InterfaceB for some operations (for details of YAWL Custom Service and InterfaceB, see section 4.2 Workflow Engine). Similarly, *SomniumBPM* communicates with the jBPM through HTTP. jBPM core engine, WS-HumanTask (Amend, Ford, Endpoints, Keller, & Rowley, 2007) based task server, a task listener and a process listener are combined in a Jetty based server application. By this way, *SomniumBPM* and jBPM can send and get HTTP messages between themselves. *SomniumBPM* modules and YAWL and jBPM engines may run on different servers. However, *SomniumCEP* and the Drools rule engine or Esper rule engine should be run on the same Java virtual machine. All *SOMNIUM* modules communicate with ActiveMQ by using TCP and HornetQ by using JNP connections.

4.2 The Workflow Engine

The workflow engines used in the framework allow modeling user daily activities by using workflows. In *SOMNIUM* framework, the YAWL workflow engine and jBPM workflow engine are used.

There are various workflow systems developed for modeling business processes. However, not all of these are suitable for modeling user activities in pervasive environments. A workflow system should be rich in terms of modeling constructs, support many different patterns and allow adaptation in order to serve as a user activity modeling system. Moreover, a workflow system should support a usable software interface which allows regular users to

easily create and manage workflows. YAWL and jBPM support most of these requirements and because of this they are selected as workflow engines for the prototype implementation.

YAWL is a workflow language based on Petri nets (W M P Van Der Aalst et al., 2004) and also uses workflow patterns (W M P Van Der Aalst et al., 2003). One of the main reasons that YAWL is selected as a workflow design and enactment engine is its support for regular users. Regular users with no programming background can design and deploy workflows easily (Russell & Ter Hofstede, 2009). YAWL has an editor for workflow specification and a web-based interface for execution of workflows and management. YAWL allows late binding and modeling, i.e. "Flexibility by Underspecification" adaptation method, through the "Worklet" mechanism (Adams et al., 2006; W M P Van Der Aalst, Adams, et al., 2009). This mechanism allows adaptation at runtime. YAWL has rich modeling structures such as workflow control flow patterns, timer, and composite activity, multiple atomic and composite tasks, cancellation structures, and exception handling. Moreover, external applications can interact with YAWL by using "custom services" approach or using interfaces supported by YAWL.

One of the significant characteristics of YAWL is its solution for adaptation problem known as "Worklet" which is developed based on the "Activity Theory" (Adams et al., 2009). Actually, a worklet is a small and complete workflow developed for replacing an abstract activity at runtime. In YAWL, a task in a workflow can be defined as an abstract activity and several worklets can be assigned to this abstract activity by defining some rules for making selection among these worklets at runtime according to the context data. Moreover, a new worklet and corresponding rules can be added to the workflow engine, when there is a new condition that cannot be handled with the existing rules. Rules in YAWL are defined as Ripple-Down Rules (RDR). RDR rules maintain a rule node hierarchy in a binary-tree structure (*YAWL - User Manual Version 2.2*, 2011). These rules include simple conditional elements for comparisons, and time-based complex events cannot be handled with RDR. Hence, the rules are separated from the workflows in the *SOMNIUM* framework, complex rules are processed by a dedicated rule engine outside the workflow engine, and the outputs are mapped to the corresponding variables in the workflows. So, workflow engine can continue to use the RDR tree based on these variables. This enables loosely-coupled integration of the rule engine and the workflow engine.

As discussed before, users tend to plan their activities from high-level abstract activities to low-level concrete activities. Hence, the worklet mechanism is used in order to provide hierarchical adaptation for modeling users' daily activities hierarchically and executing them in the workflow engine. For using this mechanism, the user defines an activity as "abstract activity". Then, he creates new workflows by using the "YAWL Workflow Editor" (*YAWL - User Manual Version 2.2*, 2011) or use existing workflows for replacing this activity at runtime according to various conditions. These conditions can be defined by the "Worklet Rules Editor" (*YAWL - User Manual Version 2.2*, 2011) if they are simple conditions, and more complex conditions are defined in the rule engine of the framework. This replacement workflow, i.e. worklet, also includes abstract activities and similarly these abstract activities are replaced by other workflows. By this way, activities are modeled hierarchically similar to the real life.

Although, it can be said that YAWL is an appropriate workflow engine for pervasive environments because of the discussed characteristics, there exist criticisms about YAWL and workflow patterns (Börger, 2011) and the rebuttal (W. M. P. Aalst & ter Hofstede, 2012)

about these criticisms. Börger (Börger, 2011) indicates that workflow patterns are not well founded and they do not have empirical validation. Aalst et. al. (W. M. P. Aalst & ter Hofstede, 2012), answer that workflow patterns usage frequencies are examined by some studies and these patterns are formed from experiences over many projects. Also, they say that workflow patterns are appropriate modeling tools for end users and languages such as Abstract State Machines (ASMs), Business Process Modeling and Notation (BPMN), Unified Modeling Language (UML), Event-driven Process Chain (EPCs) are less suitable for end user modeling. In the current proposed *SOMNIUM* framework, workflow patterns also appear as appropriate modeling tools for modeling daily activities in pervasive environments.

One of the most important properties of YAWL is its open architecture for communication with other systems. YAWL supports some software interfaces for managing operations from outside of the workflow engine. InterfaceB is one of these interfaces which is an Application Programming Interface (API) for interacting with YAWL engine (Arthur ter Hofstede, 2010). It provides case and task operations such as launching a case, completing a task etc. InterfaceB interface is used for communication with YAWL and it is utilized in two ways in the framework. First, *SomniumBPM* (for details see section 4.1 *SOMNIUM* Modules section), is implemented as a "YAWL Custom Service" and by this way it can receive notifications from the YAWL engine related to status changes of the tasks and cases (Arthur ter Hofstede, 2010). Second, the API's methods are used for operations on tasks and cases, such as canceling a case or completing a task. The *SomniumBPM* module uses the InterfaceB to apply these commands to the tasks and cases in the workflow engine.

The other workflow engine used in the framework is jBPM and it is a software developed by JBoss Community for flexible business process management. jBPM is developed with Java and beside others, it includes a core workflow engine, a human task service based on Web Services Human Task (WS-HumanTask) specification, an Eclipse-based and a web-based graphical editor for designing workflows and a management console for process and task management. jBPM uses BPMN 2.0 specification language for expressing business processes.

jBoss Community proposes a product, JBoss BRMS, for business process and decision management and this product combines Drools Expert, Drools Fusion, Drools Guvnor and jBPM products. In this product rules are written in separate files in Drools language and integrated with the processes. The aim of this integration is to run processes and rules in the same knowledge session(Salatino & Aliverti, 2012) and get the power of CEP for business process management. This approach uses the rules and events for managing processes similarly with the *SOMNIUM* framework. However, *SOMNIUM* framework uses loosely coupled integration architecture. In *SOMNIUM,* the rule engine and workflow engine run separately. This architecture offers similar capabilities with the BRMS and produces other advantages such as using the rule engine as the context reasoning component.

jBPM offers two different adaptation methodologies. The first one is similar to YAWL's worklet mechanism and users can define abstract processes hierarchically by using a "Reusable Sub-Process" component in a workflow specification. The second one is "Ad-Hoc Sub Process". This approach resembles the Declare's constrained-based declarative approach. In this approach, users can select different process fragments, repeat tasks and add new tasks into the defined ad-hoc process.

BPEL is not selected as a workflow language for the *SOMNIUM* framework. BPEL is a standard for defining the web services behavior from abstraction level to execution level (Ouyang, van der Aalst, Dumas, ter Hofstede, & La Rosa, 2007). Because it is a widely known de-facto standard it has many commercial (Oracle BPEL Process Manager, Microsoft Biztalk Server 2004, IBM WebSphere Application Server Enterprise, etc.) and open source (Apache ODE) implementations. However, BPEL has some disadvantages. In (W M P Van Der Aalst et al., 2005), authors state that BPEL is difficult to use because Extensible Markup Language (XML) representation of BPEL documents and even its graphical implementations can only be understood by experts. They also imply that, even though BPEL supports both abstract and executable processes, BPEL is mostly used as an executable language and it fails for modeling abstract processes. Moreover, according to that study, a language should not support both abstract and executable process levels.

Similar criticisms are listed in (Vigneras, 2008) for BPEL. The author states that BPEL resembles structured languages such as Java rather than BPMN or YAWL and it is not user friendly, hard to read, hard to learn and hard to implement. The example given in the article shows that some workflows cannot be represented equivalently by structured workflow languages.

In pervasive environments, for modeling user daily activities readable workflows are needed which can be easily designed by regular users. Because, BPEL is used for pure cross-organizational processes (W M P Van Der Aalst et al., 2005), it may not be appropriate for user activity modeling. Similarly, in (Rüdiger Pryss et al., 2010; Tiedeken et al., 2010), authors find BPEL too low level for dynamically evolving or adapting mobile processes.

Creating and modifying workflows by end users may seem impractical. However, workflow engines generally provide separate solutions for designing workflows. YAWL has a desktop-based editor for designing workflows and jBPM also has a web-based editor for workflow designing. Moreover, other approaches are also possible such as using templates or constructing workflows by process mining techniques (W.M.P. van der Aalst & Weijters, 2004). Workflow templates can be created by expert users and shared by other novice users. By this way a novice end user can use the workflow template by making some small modifications on it. Process mining algorithms can also be used for extracting workflows from the activities of people. However, because the process mining algorithms run on the recorded activities, activities of the people should be recorded first.

4.3 The Rule Engine

One of the most important design components of the *SOMNIUM* framework is the rule engine which supports CEP. Two different rule engines, JBoss Drools and Esper, with CEP support and highly popular in the literature, are used in the framework for the prototype implementation. The goal here is not to compare these two systems, but instead to show how it is easy to remove a module from the framework and replace it with another similar one.

Two components of Drools are used in the framework: Drools Expert and Drools Fusion. Drools Expert is the rule engine itself and Drools Fusion is the CEP module (Bali, 2009). The Drools platform is selected because its rule language is easy to understand and modify. Drools uses the Rete algorithm, hence the performance of the system does not depend on the number of rules (Bali, 2009).

Esper is also another popular open source rule engine in the literature with CEP support. Esper uses Event Processing Language (EPL) which supports filtering, joins, aggregation, and causality in one language. In contrast to database approach, Esper stores the queries i.e. rules and run the data on these queries which allows Esper to be run in real-time. (*Esper Reference*, 2012).

The rule engine (Drools or Esper) interacts with the module *SomniumCEP* in the framework. *SomniumCEP* gets messages containing context data, from the messaging system and relays it to the rule engine for evaluation. The rule engine fires the rules, infers the results and gives the output to *SomniumCEP* which then publishes the results back to the messaging system.

The rule engine has two important tasks in the framework. First, it processes raw context data in order to obtain high level results. For example, when a door sensor senses that a person enters the room, it sends this context data into the corresponding topic in the messaging system. Because, *SomniumCEP* has subscription to this sensor topic, it gets the message and gives it to the rule engine. After firing the rule, rule engine infers that there is someone in the room. Then, the rule engine activates *SomniumCEP* for sending this inference to the *SomniumCEP's* topic in the messaging system. Second, the rule engine allows automation in the framework. The rules in the engine support managing of the users' activities. For example, when a user completes an activity, this activity is automatically completed in the workflow engine by examining the rules fed by data from the sensors in the pervasive environment. Also, the rules are used for guiding the users. For example, a user can be informed about the time left for an activity, if the remaining time is below a critical threshold. For automation, the rule engine decides to complete, cancel, and suspend workflow tasks and workflows according to the rules. When the rule engine outputs an action related to the workflows or tasks, it sends the command to *SomniumCEP*, which then sends this command to the *SomniumCEP's* topic listened to by *SomniumBPM* for applying commands in the YAWL and jBPM workflow engines.

4.4 Event-Driven Messaging

The *SOMNIUM* framework uses EDA for enabling loose coupling between the workflow engine and the rule engine. Besides this, EDA enables using data coming from different data sources such as sensors, web services or other applications. These data are easily fed into the framework by using publish-subscribe type communication. Moreover, the output generated by the framework can also be utilized by high-level hardware/software. For the prototype implementation, this type of communication is supported by JMS and it has received wide industry support ("Getting Started with Java Message Service (JMS)," 2004). JMS provides a standardized API for sending and receiving messages by using Java programming language. A pervasive framework, such as *SOMNIUM* framework, needs more than sending and receiving messages, such as guaranteed delivery, high availability, high performance and scalability. For this reason, two open source JMS 1.1 compliant messaging server are used: Apache ActiveMQ (Snyder, Bosanac, & Davies, 2011) and JBoss HornetQ.

Pervasive systems have to collect context data from different systems. For example, data coming from sensors, web service calls, mobile devices or manual insertions need to be fed into the pervasive system. In the publish-subscribe type structure, these event-generators feed data to the corresponding topics. Other modules in the system which already have subscriptions to these topics get data and operate on the data. Also, modules themselves

publish data generated by the operations within the modules. This allows two-way communication mechanism between the modules. In the framework, the workflow engines and the rule engines have corresponding topics created beforehand in ActiveMQ and HornetQ. They send their inline operation results to these topics through their *SOMNIUM* modules, *SomniumBPM* and *SomniumCEP*. Also *SomniumBPM* listens to the topic of the *SomniumCEP* and *SomniumCEP* listens to the topic of the *SomniumBPM*. By this way, workflow engine and rule engine exchange data about the operations of each other.

CHAPTER

5. EVALUATION

5.1 Scenario-based Evaluation

In this section, a sample scenario is given for evaluating the framework. In this sample scenario, how workflows can be used for modeling user daily activities and adapted to the changing daily conditions is shown. Moreover, it shows how the components of the framework work together. The scenario contains daily activities of a university student for three days. The student starts with a simple workflow on the first day and he extends it on the second day and the third day. In this example, the student generally leaves his home, goes to school, attends a lecture and goes back to home, every weekday. However, he may make some changes on this flow according to his special needs or changing conditions of the environment. How these changes are handled by the *SOMNIUM* framework is demonstrated. For this demonstration only YAWL workflow engine, YAWL components and adaptation structures are used since jBPM offers the similar structures. However, in the experimental evaluation section, both of the workflow engines are used.

5.1.1 Modeling User Daily Activities by Design Components

A workflow is designed for the first school day and it is used for presenting how some YAWL structures are used for modeling users' daily activities. These structures indicated by rectangles in Figure 3 are examined. On the first day, the student leaves his home at 12:30. Depending on the traffic, he generally arrives at his department between 13:00 and 13:30. On this day he arrives at 13:35 because of heavy traffic. His course is to start at 13:40. So, he does not have time for any other activity and he directly attends the lecture. After the lecture, he leaves the department, at which point he has two optional tasks to carry out. He can select one of these two tasks according to the remaining time, since these two tasks have time constraints. He has the option of bypassing both of these two tasks and selecting the "Go Home" task. Figure 3 shows the workflow for the first day of the student. Also, this workflow is used in section 0 to show how modules execute within the *SOMNIUM* framework.

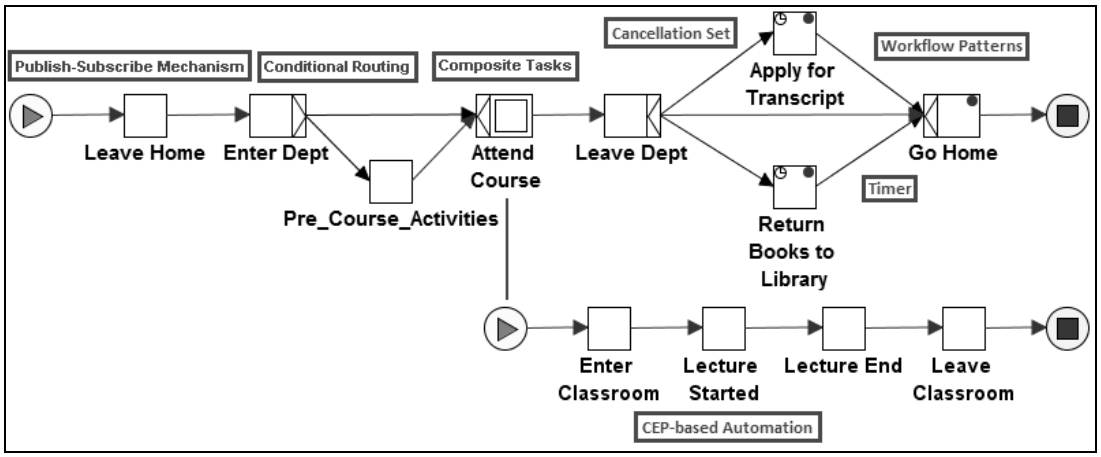

Figure 3 Workflow for the First School Day

Five design components (or structures) are used for modeling the activities on the first day: "Conditional Routing", "Composite Task", "Cancellation Set", "Timer" and "Workflow Patterns".

"Conditional Routing" is used for handling situations when a user needs to select one or more activities according to context information. For the first school day, after the student enters the department, the YAWL engine selects "Attend Course" or "Pre_Course_Activities" tasks according to the remaining time. This situation is modeled by using the "Conditional Routing" structure. A condition which checks the time left value is defined by using the "Conditional Routing" structure of YAWL. According to the value of the time left variable, the "Attend Course" task or the "Pre_Course_Activities" task is selected automatically by YAWL.

In YAWL, a workflow specification can include more than one "sub-net" which includes a set of activities starting from the "Input Condition (a circle including a triangle)" to the "Output Condition (a circle including a square)". One of these sub-nets is selected as the root net (*YAWL - User Manual Version 2.2*, 2011). The other sub-nets can be represented by using "Composite Task" component in this root net. In other words "Composite Task" is a container for sub workflows. In our example, a composite task is used for representing the lecture activities. Using a container decreases the complexity of the main workflow and it is more understandable for end users especially for large workflows. Because the lecture activities may include many tasks, defining them in a different sub-net increase the readability of the root net. Readability of the workflows is important, because the framework is used in pervasive environments by people with different levels of information literacy.

For the first school day, after attending the lecture and leaving the department, the student has three alternatives: he can apply for a transcript and go home, he can go to the library and go home, or he can directly go home because he does not have enough time for applying for a transcript or going to the library. Three design components are used for modeling these activities. First, the "Timer" component is used for controlling the time left for "Apply for Transcript" or "Return Books to Library" tasks. The "Timer" component has a timer which starts when the task associated with the "Timer" component is enabled. In this

case, a timer component is defined for each of the "Apply for Transcript" and "Return Books to Library" tasks, and when they are enabled, the corresponding timers are started. A variable is defined in the workflow specification and the expiration time of the "Timer" components is determined by the value of this variable. When the time determined by this variable goes off, the tasks which have the timer components (in our case "Apply for Transcript" and "Return Books to Library" tasks) are removed from the "Worklist" of the user. Worklist is a software component in YAWL which includes the work items of the user that are ready to be executed (*YAWL - User Manual Version 2.2*, 2011). Second, the "Cancellation Set" is used for constraining the user to select one of the three options discussed above. A cancellation set is defined for a task, and when this task is completed all the other tasks included in its nominated cancellation set are deactivated (*YAWL - User Manual Version 2.2,* 2011). In this example, when "Apply for Transcript" task is completed, the task "Return Books to Library" is removed from the worklist of the user. Similarly, when "Return Books to Library" task is completed, the task "Apply for Transcript" is removed from the worklist of the user. The student also does not have to do one of these two tasks and he can directly go home. When this happens, the tasks "Apply for Transcript" and "Return Books to Library" tasks are removed from the worklist, because they are in the cancellation set of the task "Go Home". Third, "Parallel Split (AND-split)" and "Simple Merge (XOR-join)" workflow patterns are used to be able to model these activities. "Parallel Split (AND-split)" is used because it allows two or more parallel branches to be enabled and executed concurrently. In this case "Apply for Transcript" and "Return Books to Library" tasks are enabled concurrently. "Simple Merge (XOR-join)" is used, because if one of these tasks ("Apply for Transcript" or "Return Books to Library") is completed, the subsequent task which is "Go Home" is enabled.

5.1.2 Operation of the Modules

The operation of the modules in the framework is discussed by using workflows for three school days.

For the first school day (Figure 3), four different parts of the workflow are explained. For the first part, inline operation of the modules is shown when a workflow is started, and the first task which is "Leave Home" is automatically completed. When a workflow starts, the *SOMNIUM* framework needs to make subscriptions and creates rules and task commands such as complete, suspend, for all of the tasks in the workflow specification if the user wants *SOMNIUM* to provide higher-level automation. However, for simplicity, only the first task's requirements are shown on loading and starting of the workflow. Examining the first task can give insight for the overall working mechanism of *SOMNIUM*. Figure 4 represents the sequence diagram for the completion of the "Leave Home" task. Initially, modules make the necessary subscriptions to the topics created manually in ActiveMQ or HornetQ. The *SomniumCEP* module subscribes to the topic "*HomeDoorSensorTopic*", which is fed by the outdoor RFID sensor of the user's home. *SomniumBPM* subscribes to the topic of *SomniumCEP*, which is the "*SomniumCEPTopic*" for getting the commands depending on whether the user leaves his home or not. When the user leaves his home, the RFID sensor sends this information to the "*HomeDoorSensorTopic*" topic in ActiveMQ or HornetQ. After that, *SomniumCEP* gets this message as a subscriber to this topic. Then, *SomniumCEP* gives this message to the Drools or Esper rule engine and executes the rules. The rule written in the Drools rule language or the Esper rule language decides to complete the "Leave Home" task, and Drools or Esper gives this message to *SomniumCEP*. *SomniumCEP* sends this message to the topic the "*SomniumCEPTopic*" in ActiveMQ or HornetQ. The code in Figure 5 shows how this rule is implemented in the Drools rule definition language. Figure 4 and

Figure 5 show the framework implementation with the Drools rule engine, YAWL workflow engine and ActiveMQ messaging system. Figure 6 shows the same rule written in Esper language.

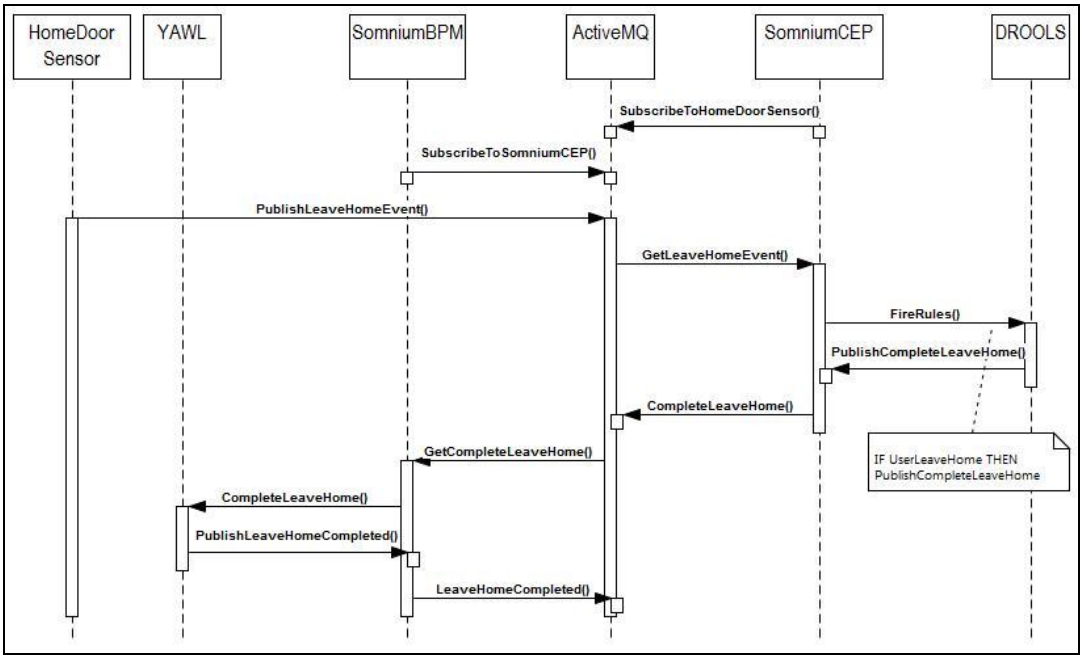

Figure 4 Messaging Structure for Automatic Completion of the "Leave Home" Task

```
rule "User Leaves His Home"
    when
        x : SomniumEVENT($parameter_0 : parameter_0 == "HomeDoorSensorTopic",
                         $parameter_1 : parameter_1 == "Value",
                         $parameter_2 : parameter_2 == 1)
    then
        dp.publishDroolsEvent("194", "194:Leave_Home_3", "COMPLETEWORKITEM", "1");
end
```

Figure 5 The Rule in the Drools Language for Completion of the "Leave Home" Task

```
createEPL("select d.parameter_1 from "
        + "SomniumEVENT(parameter_0='HomeDoorSensorTopic'," 
                     + "parameter_1='Value', "
                     + "parameter_2=1).std:lastevent() as d");
```

Figure 6 The Rule in Esper Language for Completion of the "Leave Home" Task

In Drools, rules are created using the "rule", "when", "then", and "end" keywords. The rule starts with the rule name and then the condition and consequence sections are written (Bali, 2009). In this rule, a rule with the name "*User Leaves His Home*" is created and when a message comes to Drools from the "*HomeDoorSensorTopic*", the parameters' values are evaluated. If the name of the topic represented by "*parameter_0*" is "*HomeDoorSensorTopic*", the type of the parameter represented by "*parameter_1*" is "*Value*" and the value represented by "*parameter_2*" equals to one, the condition evaluates to true and a message is published by the "*publishDroolsEvent*" method, which implies that the task with the name "*Leave_Home_3*" of the workflow with the case number "*194*" is to be completed. The number "*194*" represents the case number of the workflow given automatically to all launched workflow specifications by the YAWL workflow engine. Similarly "*Leave_Home_3*" represents the task identifier of the "*Leave Home*" task in YAWL. The method "*publishDroolsEvent*" sends the "*Complete Task*" command to the topic "*SomniumCEPTopic*" in ActiveMQ. When *SomniumBPM* module gets this completion message from the "*SomniumCEPTopic*" topic which it listens to, it sends the "*Complete*" command to the YAWL engine and the YAWL engine completes the task.

For the second part, the implementation of the "Conditional Routing" structure in the framework is discussed. This example shows how a variable defined in the YAWL workflow specification is updated automatically by the *SOMNIUM* framework. It is assumed that the "Enter Dept" task which is the second task in the sample workflow is completed by the *SOMNIUM* framework, when the user enters the department. This completion process of the "Enter Dept" task is similar to the completion process of the "Leave Home" task discussed above. Instead of repeating the discussion on this process for "Enter Dept" task, it is better to concentrate on the step when "Enter Dept" task is completed. When the "Enter Dept" task is decided to be completed by the rules in Drools, the update of the "TimeLeft" parameter is also decided according to the corresponding rules defined in Drools and sent as a command together with the "Complete Enter Department Task" command. By this way, the "TimeLeft" parameter is updated by *SomniumBPM* because it listens to these commands coming from *SomniumCEP*. After completion of the "Enter Dept" task, YAWL checks whether the "TimeLeft" parameter is greater than five minutes or not. If it is less than or equal to five minutes, YAWL automatically starts the "Attend Course" composite task. Figure 7 shows how "Conditional Routing" is defined in YAWL.

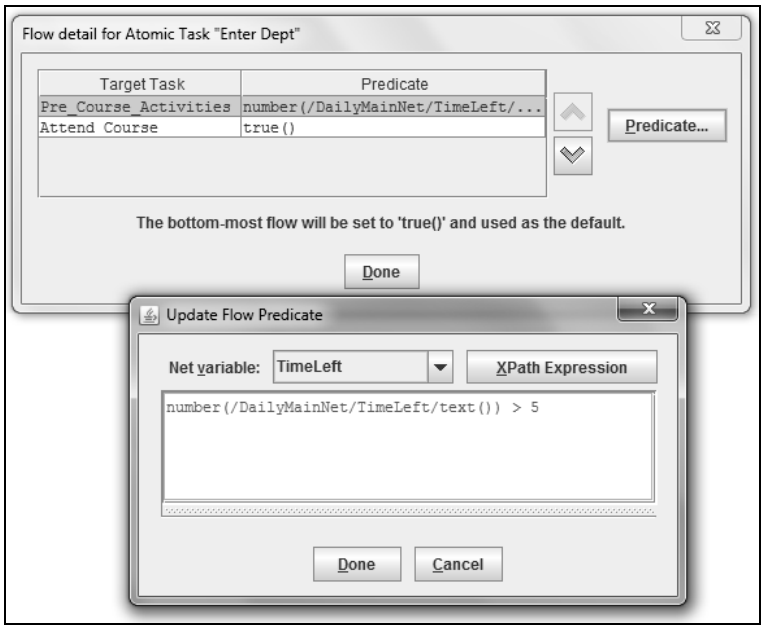

Figure 7 Conditional Routing Definition

For the third part, how context information coming from the users' environments affects the workflows of the users is shown. For this purpose, automatically detecting the "lecture started" event and completing the task "Lecture Started" in the sample workflow is discussed. The "Attend Course" composite task is used for representing the course related activities. The "Lecture Started" task is one of the activities in this composite task related to the lecture. For automatic detection of the "lecture started" event, a simple sensor network is implemented in a classroom for collecting context data. Arduino (Arduino, n.d.) open source electronics prototyping platform is used for this purpose. Sensors in this network sense context data such as motion, light, door status, and noise level. Arduino Uno development boards, Digi XBee ("Digi XBee Wireless RF Module," n.d.) wireless modules and RedBee RFID reader (125 kHz) ("RedBee RFID Reader," n.d.) are used. A rule engine examines the collected data and infers whether the lecture is started, break is given, and the lecture is ended. For this part, only how lecture started event is detected and interpreted by the framework is shown. Several sensor sets are implemented in the classroom for detecting various events in the classroom. However, for detecting "lecture started" event, only an RFID sensor is used for detecting availability of the students and the instructor, and a Hall Effect sensor for detecting whether the door is open or not. A small Java program is implemented for getting the lecture time from the schedule and sending an alert to a topic in ActiveMQ/HornetQ which is listened to by *SomniumCEP*. Using these sensors, the rule in Figure 8 is written for detecting the "Lecture Started" event. Figure 9 shows the same rule written in Esper language. When the event is detected by firing this rule, Drools or Esper sends the complete "Lecture Started" task command to *SomniumCEP*, and then, *SomniumCEP* sends this message to the topic "*SomniumCEPTopic*" in ActiveMQ/HornetQ which is listened to by *SomniumBPM*. The process is similar to the completion of "Leave Home" task. However, for the "Lecture Started" event, the event definition is more complex because it needs to use data coming from different sources. When all the data coming from

different sources is ready, the *SOMNIUM* framework automatically detects the event and completes the task. The rule in Figure 8 shows that, for detecting the "lecture started" event, more than or equal to five students and the instructor must be in the classroom, the door must be closed and the start time of the lecture must have come.

```
rule "Lecture Started Event"
when
    $x1 : SomniumEVENT(parameter_0 == "HALL",parameter_1 == "1",parameter_2 == 1)
    $y1 : SomniumEVENT(parameter_0 == "INSTRUCTOR",parameter_1 == "1",parameter_2 == 1)
    $z1 : SomniumEVENT(parameter_0 == "LECTURE",parameter_1 == "1",parameter_2 == 1)
    $t1 : Number(intValue >= 5) from accumulate(StudentEVENT(parameter_0 == "STUDENT",
                                                parameter_1 == "1",
                                                parameter_2 == 1), count(1))
then
    dp.publishDroolsEvent("1", "1:Leave_Home", "COMPLETEWORKITEM", "1");
end
```

Figure 8 The Rule in Drools Language for Detecting "Lecture Started" Event

```
createEPL("select d.parameter_1 from "
        + "SomniumEVENT(parameter_0 = 'HALL',parameter_2=1).win:length(1000) as a, "
        + "SomniumEVENT(parameter_0 = 'INSTRUCTOR',parameter_2=1).win:length(1000) as b,"
        + "SomniumEVENT(parameter_0 = 'LECTURE',parameter_2=1).win:length(1000) as c,"
        + "SomniumEVENT(parameter_0 = 'STUDENT',parameter_2=1).win:length(1000) as d "
        + "where a.parameter_1 = b.parameter_1 and "
        + "b.parameter_1=c.parameter_1 and "
        + "c.parameter_1 = d.parameter_1 "
        + "group by d.parameter_1 "
        + "having count(d.parameter_1)=5");
```

Figure 9 The Rule in Esper Language for Detecting "Lecture Started" Event

For the last part, some other important structures within YAWL and how they produce solutions for modeling users' daily activities are discussed. After the lecture is ended, the student leaves the department. The time is 16:30 when he leaves the department. After that time he can go to library for returning books or he can apply for transcript. Because public institutions in his location close at 17:00, he can complete only one of these tasks. If he does not want to do any of these two tasks, he can directly go home. When the user leaves the department, "Apply for Transcript", "Return Books to Library" and "Go Home" tasks become enabled. If the user completes the "Apply for Transcript" task, the "Return Books to Library" task automatically is deactivated and vice versa. If the user completes the "Go Home" task, the tasks "Apply for Transcript" and "Return Books to Library" are automatically deactivated. If the user does not complete any of these three tasks, "Apply for Transcript" and "Return Books to Library" tasks are automatically deactivated after thirty minutes from the completion of "Leave Dept" task, due to the "Timer" structure.

On the second day, the student leaves his home at 12:30 and he arrives at the department at 13:20. His course again will start at 13:40. He has enough time for talking to

his instructor. He modifies his workflow by adding new sub-workflows to the "Pre_Course_Activities" task. He adds, "Talk to Instructor", "Go to Bookstore" and "Drink Tea" workflows. If he has time more than five minutes, and the instructor is in the office, the system automatically selects the "Talk to Instructor" workflow. If the instructor is not in his office, the workflow engine selects either one of the other two workflows according to the remaining time. If the remaining time is more than 10 minutes, the workflow engine selects the "Go to Bookstore" workflow; otherwise it selects "Drink Tea" workflow. For this second day, he has ten minutes and his instructor is in the office. The workflow engine can easily check the "InstructorInOffice" parameter updated by the *SOMNIUM* framework, when the RFID sensor in front of the office door detects that the instructor comes to his office. After that, RDR rules defined in the YAWL's rule editor are enough for selecting the correct workflow. However, in pervasive environments rule definitions are typically not as simple as checking only one parameter. There may be other students in the instructor office or he may be talking on the phone. For detecting other students, more RFID sensor events should be tracked, and for detecting whether the instructor is talking on the phone, the noise level should be tracked continuously. These types of complex events cannot be handled by RDR rules. Assume that these sensor sets are implemented and the *SOMNIUM* framework detects that the instructor is available. So the student talks to his instructor, and then attends the lecture. After that, he leaves the campus to go home. Figure 10 shows the workflow for the second day of the student.

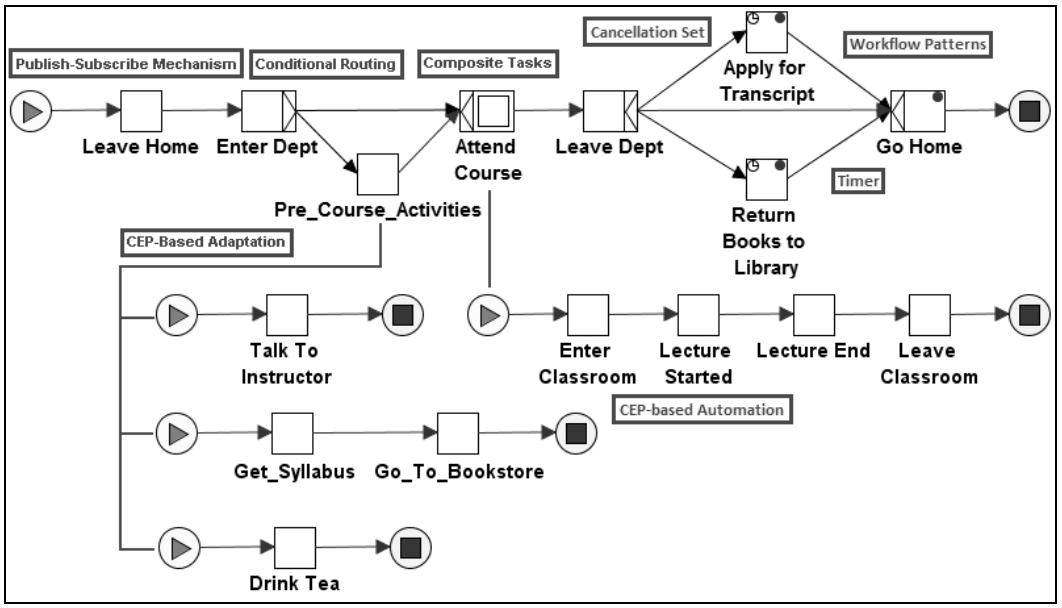

Figure 10 Workflow for the Second School Day

On the third day, the student again leaves his home at 12:30 and arrives at the department at 13:25. His course again will start at 13:40. Today his instructor is not in the department. He has fifteen minutes for doing something before attending the course. He can talk to the teaching assistant, get the course syllabus from him and go to bookstore for buying the course book given in the syllabus. However, he cannot go to bookstore by walking or by using ring shuttle bus, because fifteen minutes is not enough for these options. He modifies his workflow by adding new sub-workflows to the "Go_To_Bookstore" task. He adds, "Go by Walking", "Go by Ring" and "Find another Solution" workflows. If he has more than thirty minutes, the system automatically selects the "Go by Walking" workflow. If he has more than twenty minutes and less than or equal to thirty minutes, the system automatically selects "Go by Ring" workflow. If he has more than ten minutes but less than or equal to twenty minutes then the system selects the "Find another Solution" workflow. The user can replace this workflow by another workflow at runtime. For example, if he has a friend with a car, he can go with him to bookstore and replace the "Find another Solution" workflow with "Go with a Friend" workflow. On this day the student takes a taxi for going to bookstore and replaces "Find another Solution" workflow with "Go by Taxi" workflow. By this way he can hierarchically adapt his workflow according to changing conditions. Figure 11 shows the workflow for this third school day. In this workflow, the "Pre_Course_Activities" task and the "Go_To_Bookstore" task are defined in the form of worklet.

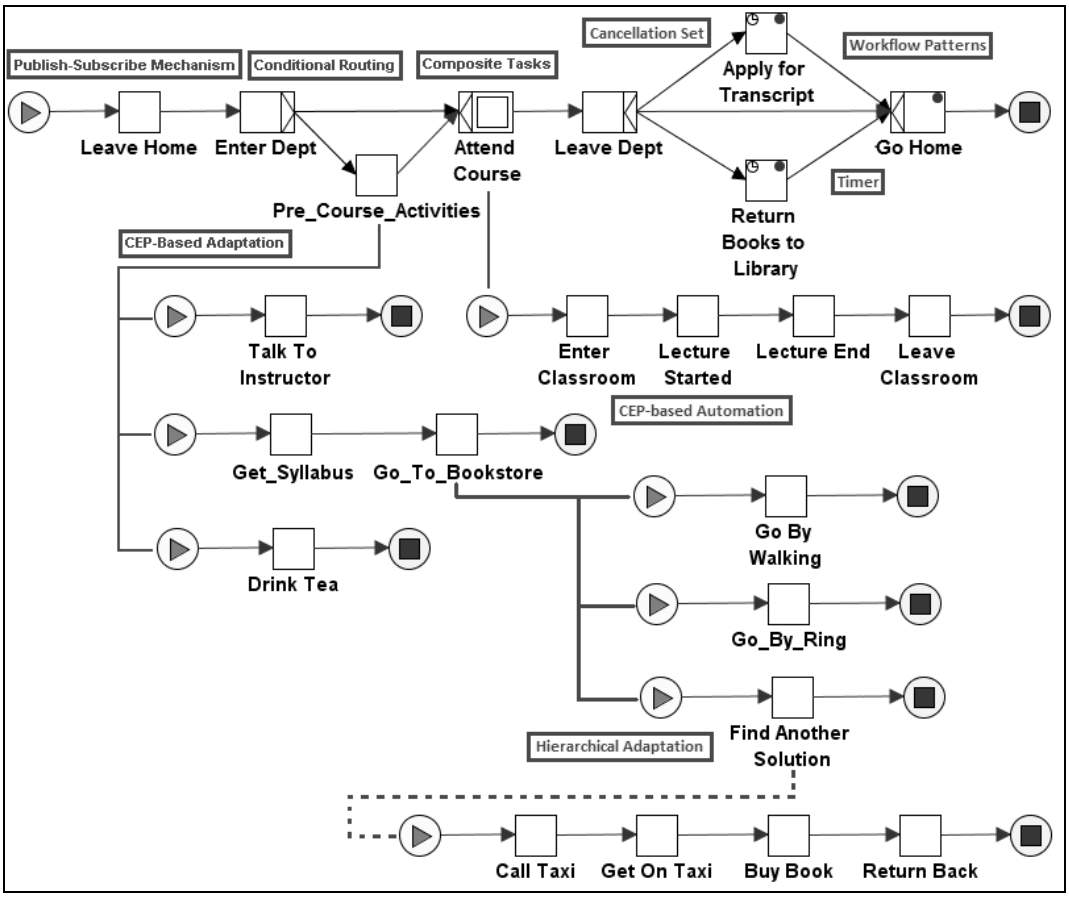

Figure 11 Workflow for the Third School Day

5.2 Experimental Evaluation

The performance and operation of inline messaging architecture in the *SOMNIUM* framework are evaluated. For this evaluation Drools and Esper CEP engines, YAWL and jBPM workflow engines and ActiveMQ and HornetQ messaging systems are used. However, the aim of this experimental evaluation is not to assess or compare the performances of the individual components used in the framework. The aim is to show how the framework is operating, how the replacement of the components by its counterparts affects the working structure of the framework and which parts of the framework consume how much time. Because the aim is not to assess or compare the software, YAWL, jBPM, Drools, Esper, ActiveMQ and HornetQ are used as they are in their default versions. No parameters related to performance, operation, security of these software are modified from their default values.

In the experiments, calculating the timestamps of the messages includes starting and completing of two different tasks. The first task that is used is completing the “Leave Home” task. This task includes a simple rule (Figure 5) and this rule gets only a sensor event and decides to complete the task according to the value of the sensor. The second task that is used is completing the “Lecture Started” task. This task includes a complex rule (Figure 8) and it needs at least eight events for completion: the time of the lecture must be over, the

instructor must be in the classroom, the door must be closed and at least five students must be in the classroom.

Apache JMeter load testing tool ("Apache JMeter," n.d.) is used for the experimental evaluation of the framework. A publisher and a subscriber are created in JMeter. The publisher simulates the sensor events by publishing 2 and 101 events. We omit the initial messages, because they are affected by the initialization delays of the software components in modules. By this way we make the analysis of 1 and 100 messages. Threads are used for publishing messages. For simple rule, 2 messages are sent within 8 seconds when there are 2 rules in CEP modules and 2 workflows in BPM modules and 101 messages are sent within 404 seconds when there are 101 rules in CEP modules and 101 workflows in BPM modules. For complex rule, 2 messages are sent within 16 seconds when there are 2 rules in CEP modules and 2 workflows in BPM modules, and 808 messages are sent within 808 seconds when there are 101 rules in CEP modules and 101 workflows in BPM modules. Experimental tests are repeated 5 times and the average of the scores are used when there are 2 rules and workflows in the framework for eliminating the errors which occur due to network congestion, CPU overloading etc. Messages are processed by the framework and the JMeter subscriber gets the final result messages. Figure 12 shows how these messages are processed in the framework. The timestamps at 7 different points in the framework are measured and recorded to a database for analysis.

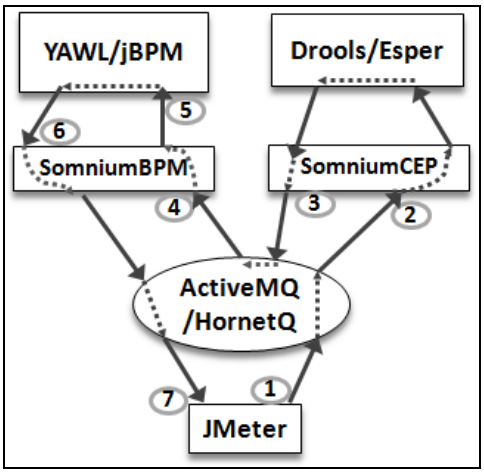

Figure 12 The *SOMNIUM* Framework Load Test Measurement Points.

(1 - JMeter sends messages to the ActiveMQ/HornetQ topic in place of sensors. 2- *SomniumCEP* gets these messages from the ActiveMQ/HornetQ topic, parses the messages and gives them to Drools or Esper. 3 - Drools or Esper fires the rules and gives results (including YAWL/jBPM commands) back to *SomniumCEP*, which sends messages to ActiveMQ/HornetQ topic. 4 - *SomniumBPM* gets the messages from ActiveMQ/HornetQ, parses messages and applies commands to YAWL/jBPM. 5 – Commands are get by YAWL/jBPM and related workflow tasks are started. 6 – Commands are processed by YAWL/jBPM, tasks are completed and results are given to *SomniumBPM*. 7 - *SomniumBPM* sends the result messages to the corresponding ActiveMQ/HornetQ topic and jMeter gets the results.)

When making these measurements, three computers are used. The first one is a 64-bit Windows 7 Home Premium based laptop computer with an Intel Core i7-2630QM 2.00 GHz processor, and 8 GB RAM. This computer runs YAWL's YAWL4Study 2.3final and jBPM 5.4.0 Final workflow engines, *SomniumCEP* and *SomniumBPM* modules, Apache JMeter 2.11 and JBoss Drools 6.0.1 Final and Esper 4.6.0 rule engines. YAWL software and *SomniumCEP* (including Drools or Esper) and *SomniumBPM* modules are deployed to Apache Tomcat 6.0.18 servlet container. jBPM workflow engine is deployed to Jetty server version 9.0.4. The second computer is a 64-bit Windows 7 Home Premium based laptop with an Intel Core i7-2630Q740 1.73 GHz processor, and 4GB RAM, and runs Apache ActiveMQ 5.6.0 and JBoss HornetQ 2.3.0 Final. The third computer is a 32-bit Windows Vista Home Premium based laptop computer with an Intel Core 2 Duo-P8600 2.40 GHz processor, and 3 GB RAM and it runs Oracle 11g Express Edition database for storing calculated timestamps. The first computer and third computer are run in the same LAN and the second computer which includes messaging software is run in another LAN.

When making these measurements, two different architectures (Figure 13 and Figure 14) are used because of the different implementation architectures of the YAWL and jBPM. For both of the architectures, the implementation of the event-driven messaging server and CEP engines and *SomniumCEP* module are same. The messaging servers are standalone Java-based applications and they are run continuously in the second computer. The only difference between messaging servers are the communication protocols. ActiveMQ uses TCP communication and HornetQ uses JNP (socket/RMI based protocol used by JBoss) ("Naming on JBoss," 2004). CEP engines, Drools and Esper, are run with *SomniumCEP* module and they are loaded to the Apache Tomcat servlet container as servlet. *SomniumCEP* module runs the CEP engines, fires the rules and gets the results by using API calls

In YAWL-based implementation architecture (Figure 13), YAWL engine is deployed to Apache Tomcat and the communication with the engine is done through its InterfaceB servlet-based interface using HTTP calls. In this experimental evaluation, only starting and completing a task commands are used. *SomniumBPM* gives start and complete commands to YAWL. At step 5, YAWL starts the related task, completes it and sends a message to the YAWL Custom Service listening the YAWL engine. Calculating the time between the start task event and the complete task event is different. YAWL registers the completion time, after making an HTTP call to the YAWL Custom Service for announcing the completion event. For this reason, completion time also includes the networking delay between YAWL and YAWL Custom Service. In other words, the elapsed time between Step 5 and Step 6 includes the announcement of the completion message to the YAWL custom service.

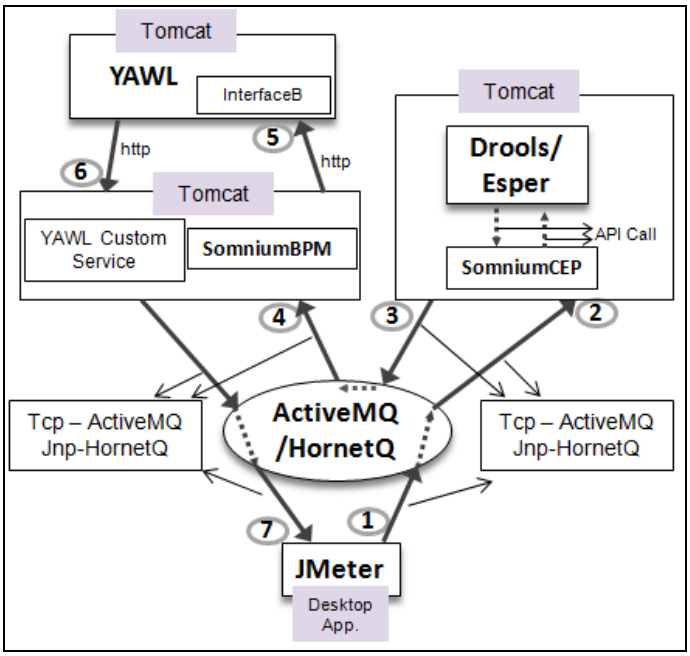

Figure 13 Implementation Architecture for YAWL

In jBPM-based implementation architecture (Figure 14), jBPM Business Process Management Suit is deployed to the Jetty servlet engine and http server. jBPM can also be run as a standalone Java application. However, for calculating consistent results between different components of the framework the implementation architecture of the jBPM is implemented in the same way as YAWL. jBPM engine can be assigned a "Task Event Listener" and this listener listens to the events in the engine. When a new task status change occurs, engine warns the task listener. However, the listener runs with the engine together. Again, for consistency between implementation architectures, a custom listener servlet is implemented and deployed to the Apache Tomcat together with the *SomniumBPM*. By this way an HTTP call is needed between jBPM listener and the custom listener similarly between YAWL and YAWL Custom Service. The elapsed time calculation for starting and completing events is similar for jBPM and YAWL. In jBPM, a WS-HumanTask specification-based human task service is needed for enabling the tasks to interact with the human users. For this reason we implement a human task service together with the jBPM engine. Eventually, jBPM component includes a jBPM engine, a human task service and a task event listener.

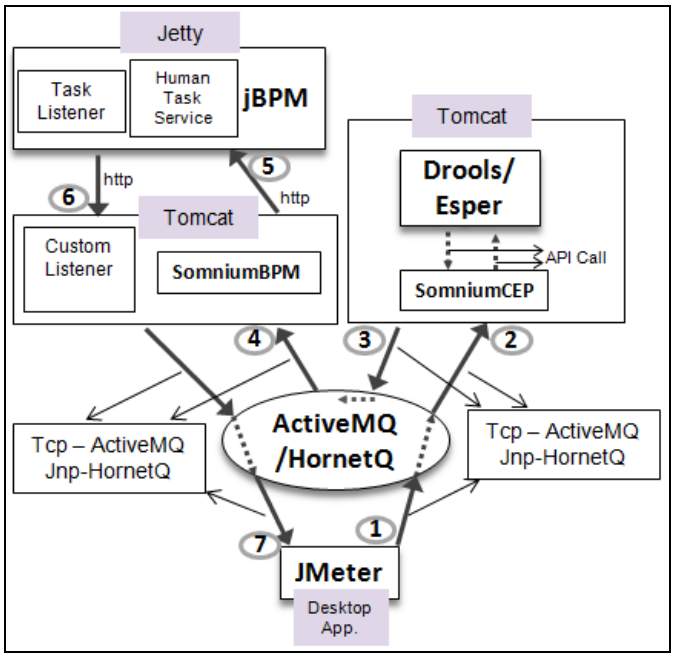

Figure 14 Implementation Architecture for jBPM

Table 2 shows the timestamp results for experimental tests in seconds. In Table 2, "COMB" column shows the different combinations of the software that are used in the tests. In this column, "D" shows the measurements when the Drools rule engine is used, "E" shows the measurements when the Esper rule engine is used, "Y" shows the measurements when the YAWL workflow engine is used, "J" shows the measurements when the jBPM workflow engine is used, "A" shows the measurements when the ActiveMQ messaging system is used and "H" shows the measurements when the HornetQ messaging system is used. The numbers 1 and 2 at the end of the combinations represent the simple rule and complex rule respectively. "TOTAL" column shows the number of rules and workflows in the rule engine and workflow engine respectively. The "GENDIF" column shows the general time difference between steps "1" and "7". The column "S1_2" shows the difference in seconds between steps "1" and "2". Other columns (S2_3, S3_4, S4_5, S5_6 and S6_7) show the results for the corresponding steps. The columns including "_P" string at the end shows the percentage of the difference in the given steps. The columns including "_ST" string at the end shows the standard deviations of the time differences in the given steps.

Figure 15 shows the graphical representation for all experimental test results. According to the Figure 15, it can be inferred that the time consumed by the steps between 4 and 5 takes more percentage than others when YAWL workflow engine is used. This occurs due to the InterfaceB communication interface of YAWL. Since no parameters of YAWL are changed for increasing the performance, this does not mean that YAWL runs slower than the others. The figure also shows that workflow engines consume more time than the complex event processing engines whatever the software combinations are used. Moreover, it can be inferred that, complex event processing engines consume the minimum time compared to other measurement steps.

Table 2 Timestamp Measurements of Experimental Tests

(In “COMB” column, “D” represents Drools, “E” represents “Esper”, “Y” represents “YAWL”, “J” represents “jBPM”, “A” represents “ActiveMQ”, “H” represents “HornetQ”, “1” represents “Simple Rule”, and “2” represents “Complex Rule”. “TOTAL” column shows the number of rules and workflows in the framework. “S1_2” and other “SX_Y” columns show the time difference between steps. Columns end with “_P” string show the percentages and columns end with “_ST” string show the standard deviations of the respective steps.)

COMB	TOTAL	GENDIF	S1_2	S2_3	S3_4	S4_5	S5_6	S6_7	S6_8	S1_2_P	S2_3_P	S3_4_P	S4_5_P	S5_6_P	S6_7_P	S6_8_P	S1_2_ST	S2_3_ST	S3_4_ST	S4_5_ST	S5_6_ST	S6_7_ST	S6_8_ST	GENDIF_ST
DJA1	1	0.098	0.020	0.009	0.020	0.009	0.022	0.023	0.018	20.268	8.805	20.500	9.645	22.536	23.172	18.246	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DJA2	1	0.098	0.021	0.007	0.021	0.010	0.022	0.022	0.018	20.953	7.140	21.174	10.609	22.192	22.217	17.932	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DJH1	1	0.127	0.041	0.007	0.025	0.010	0.021	0.026	0.023	32.157	5.842	19.877	7.579	16.567	20.658	17.977	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DJH2	1	0.125	0.039	0.006	0.025	0.010	0.022	0.026	0.023	31.264	4.484	20.391	8.024	17.577	20.832	18.260	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DYA1	1	0.643	0.020	0.007	0.020	0.398	0.174	0.209	0.024	3.070	1.089	3.157	61.861	27.093	32.667	3.730	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DYA2	1	0.694	0.020	0.008	0.020	0.448	0.179	0.196	0.018	2.887	1.211	2.943	64.552	25.843	28.197	2.564	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DYH1	1	0.719	0.040	0.008	0.026	0.447	0.175	0.190	0.023	5.540	1.087	3.674	62.143	24.325	26.373	3.231	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DYH2	1	0.728	0.039	0.007	0.026	0.452	0.181	0.194	0.023	5.363	0.960	3.631	62.070	24.813	26.619	3.163	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EJA1	1	0.093	0.019	0.006	0.019	0.010	0.021	0.023	0.018	20.879	6.190	20.236	10.539	22.599	24.453	19.556	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EJA2	1	0.092	0.020	0.004	0.019	0.010	0.022	0.021	0.018	21.440	4.114	20.574	10.614	24.000	22.956	19.258	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EJH1	1	0.125	0.039	0.007	0.026	0.010	0.021	0.028	0.023	31.410	5.288	20.511	7.691	16.988	22.285	18.111	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EJH2	1	0.135	0.042	0.007	0.030	0.010	0.021	0.028	0.024	31.163	5.386	21.939	7.814	15.657	20.934	18.040	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EYA1	1	0.689	0.021	0.007	0.020	0.445	0.177	0.190	0.020	2.993	1.015	2.907	64.568	25.616	27.607	2.902	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EYA2	1	0.671	0.020	0.005	0.020	0.437	0.169	0.184	0.019	2.953	0.743	3.044	65.229	25.139	27.374	2.892	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EYH1	1	0.672	0.061	0.007	0.026	0.388	0.166	0.180	0.023	8.779	1.108	3.882	57.946	24.883	26.876	3.402	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EYH2	1	0.643	0.039	0.006	0.025	0.384	0.165	0.182	0.023	6.125	0.871	3.918	59.739	25.711	28.265	3.636	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DJA1	100	0.108	0.019	0.009	0.019	0.014	0.028	0.023	0.018	17.549	8.497	18.003	12.930	26.277	21.085	16.744	0.001	0.009	0.001	0.003	0.004	0.003	0.001	0.011
DJA2	100	0.118	0.019	0.012	0.023	0.015	0.030	0.023	0.018	16.324	9.983	19.847	12.903	25.569	19.414	15.374	0.001	0.013	0.033	0.002	0.005	0.004	0.001	0.037
DJH1	100	0.135	0.039	0.010	0.023	0.013	0.028	0.028	0.021	29.148	7.583	17.193	9.973	20.855	20.648	15.247	0.002	0.011	0.004	0.002	0.004	0.012	0.003	0.013
DJH2	100	0.138	0.040	0.011	0.023	0.015	0.029	0.028	0.020	28.559	7.952	16.750	10.914	21.026	20.339	14.800	0.003	0.011	0.002	0.004	0.004	0.011	0.002	0.014
DYA1	100	0.623	0.020	0.009	0.021	0.386	0.165	0.174	0.021	3.280	1.488	3.420	61.961	26.471	27.906	3.380	0.005	0.009	0.004	0.020	0.014	0.014	0.021	0.032
DYA2	100	0.669	0.019	0.009	0.020	0.420	0.181	0.192	0.018	2.894	1.403	3.060	62.778	27.124	28.648	2.741	0.001	0.007	0.001	0.020	0.013	0.015	0.001	0.023
DYH1	100	0.638	0.039	0.012	0.023	0.380	0.164	0.173	0.020	6.064	1.887	3.635	59.516	25.715	27.167	3.183	0.002	0.014	0.001	0.018	0.013	0.014	0.001	0.028
DYH2	100	0.636	0.039	0.009	0.024	0.379	0.165	0.173	0.020	6.124	1.364	3.728	59.658	25.918	27.153	3.208	0.002	0.003	0.001	0.017	0.014	0.014	0.001	0.020
EJA1	100	0.106	0.019	0.007	0.019	0.014	0.029	0.023	0.018	17.997	6.197	18.035	13.371	27.522	21.429	16.878	0.001	0.006	0.001	0.004	0.004	0.004	0.001	0.010
EJA2	100	0.109	0.019	0.009	0.020	0.015	0.029	0.026	0.018	17.278	8.104	17.854	13.519	26.562	23.617	16.683	0.001	0.008	0.001	0.003	0.005	0.008	0.001	0.011
EJH1	100	0.137	0.040	0.008	0.024	0.014	0.028	0.029	0.022	29.347	6.071	17.512	10.009	20.726	21.018	16.335	0.005	0.005	0.009	0.002	0.004	0.015	0.011	0.023
EJH2	100	0.140	0.039	0.012	0.023	0.015	0.030	0.031	0.021	28.098	8.619	16.580	10.802	21.147	22.077	14.754	0.002	0.042	0.001	0.002	0.004	0.016	0.001	0.042
EYA1	100	0.616	0.019	0.007	0.020	0.389	0.163	0.176	0.018	3.136	1.079	3.179	63.143	26.477	28.488	2.986	0.003	0.006	0.002	0.019	0.010	0.013	0.004	0.022
EYA2	100	0.638	0.021	0.010	0.022	0.390	0.170	0.181	0.025	3.292	1.598	3.399	61.081	26.654	28.355	3.976	0.012	0.013	0.010	0.017	0.014	0.029	0.032	0.051
EYH1	100	0.631	0.038	0.011	0.023	0.377	0.162	0.174	0.021	6.059	1.755	3.675	59.639	25.619	27.507	3.254	0.001	0.011	0.001	0.017	0.012	0.016	0.002	0.023
EYH2	100	0.642	0.039	0.007	0.023	0.383	0.168	0.173	0.022	6.100	1.087	3.635	59.592	26.191	26.923	3.396	0.002	0.004	0.001	0.018	0.025	0.013	0.010	0.030

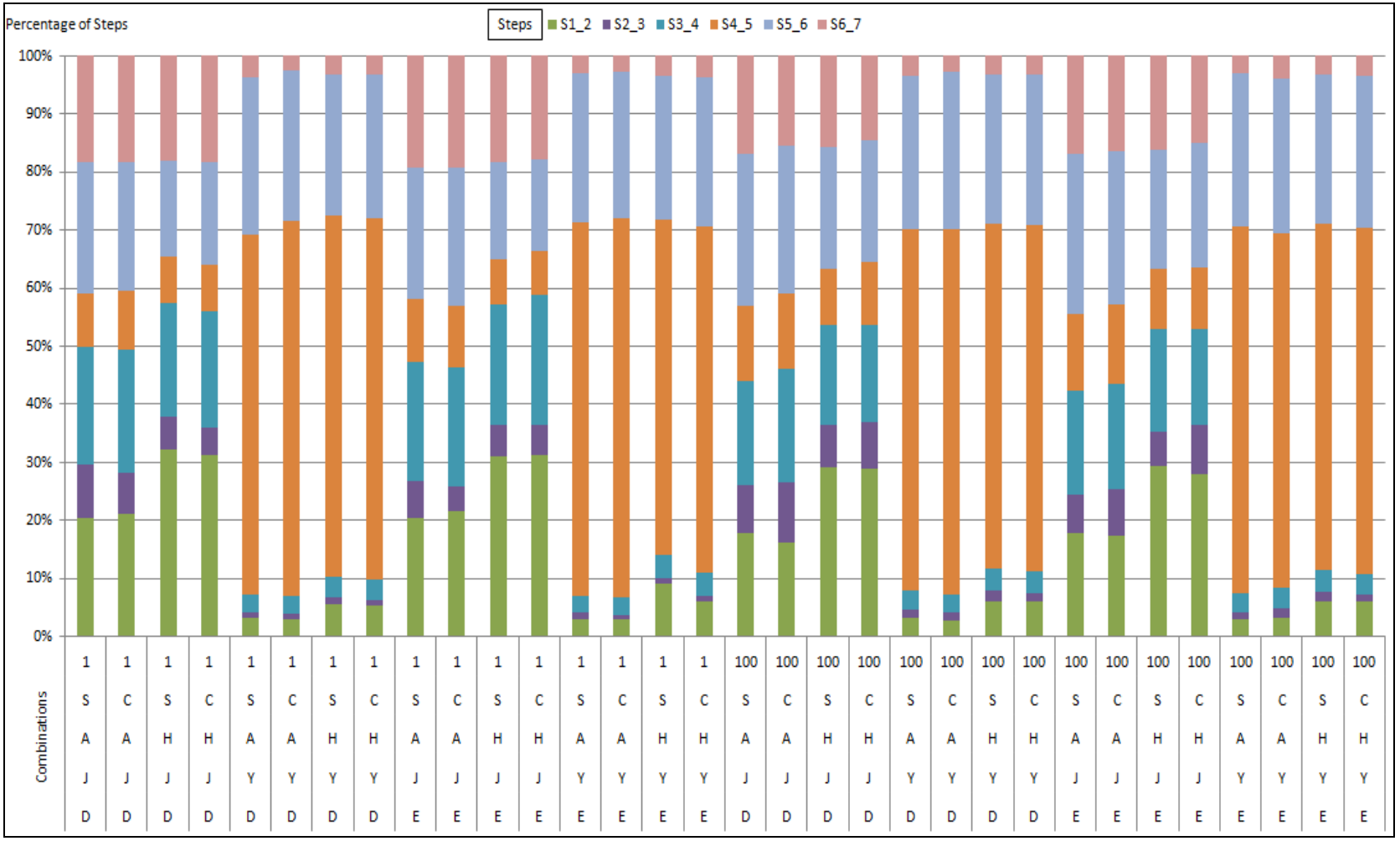

Figure 15 *SOMNIUM* Framework Experimental Test Results

Figure 16 shows the proportions of consumed time by the rule engine and the workflow engine. According to the Figure 16 the workflow engines consume more time than the CEP engines. When calculating these scores it is inferred that, the processing of the first message takes much longer than the other messages at both the rule engines and workflow engines. This probably occurs due to initialization of the software components in the engines. For this reason, scores are calculated by excluding the first messages (since 2 and 101 rules and workflows are created in the framework the figure shows the scores excluding the first messages). Certainly, these results depend on the memory and processing capabilities of the used computers and the configuration parameters of the software. Default configuration parameters of the given software versions are used in the experimental testing.

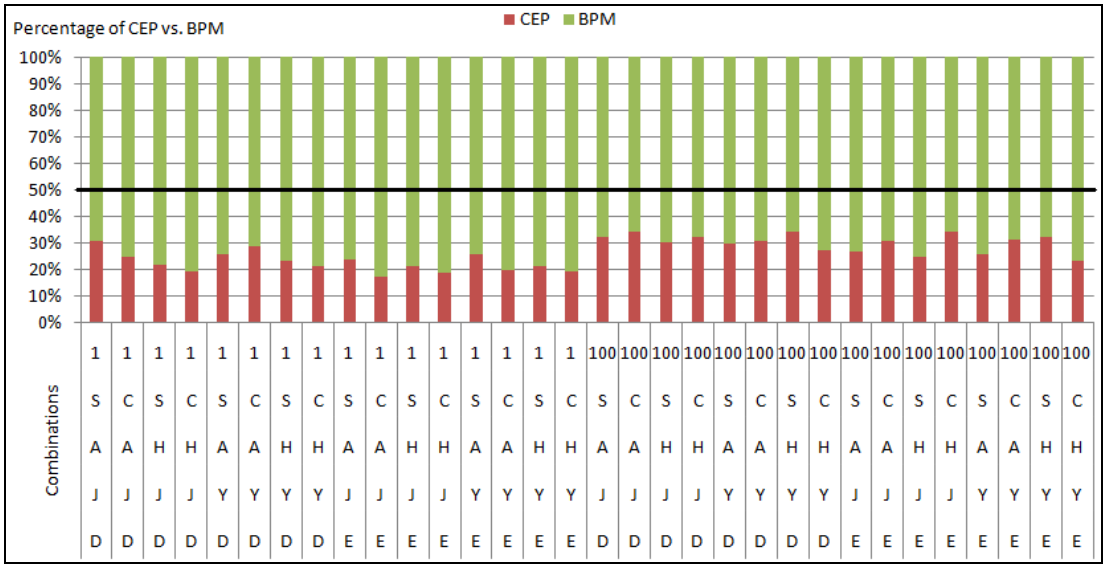

Figure 16 Comparison of Rule Engines and Workflow Engines

Figure 17 compares the different rule engine and workflow engine combinations. This graphic takes the average of the rule engine and workflow engine scores. In other words, the graphic shows the average proportions of steps S2_3 and S5_6 without considering the messaging systems, rule complexity and number of rules/workflows. According to this graphic Esper performs slightly better than Drools. Even though the average scores of the Drools (0.0089 and 0.0086) and Esper (0.0075 for both) do not change when they are run with jBPM and YAWL, the proportions are changing. This occurs because of the InterfaceB communication interface of YAWL which consumes more time than the jBPM communication interface implemented by the researcher.

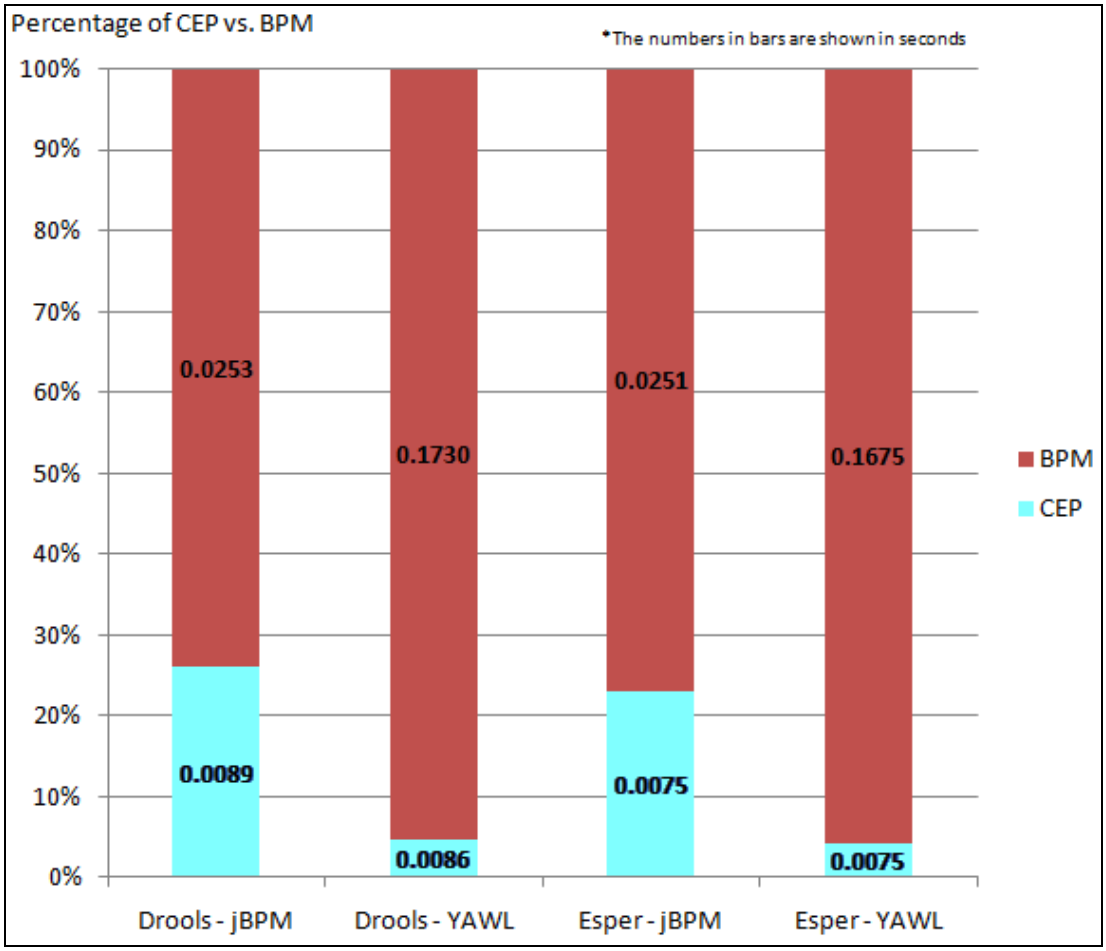

Figure 17 Paired Comparison of Rule Engines and Workflow Engines

Figure 18 compares the consumed time by Drools and Esper rule engines according to rule complexity (simple rule and complex rule) and the number of rules (1 rule and 100 rules). The affects of the messaging systems and workflow engines are eliminated by averaging the values. This graphic shows that Esper performs slightly better than Drools. Moreover, there is a significant difference between 1 and 100 rules regardless of rule engine and rule complexity. However, the rule complexity (simple or complex) does not make significant difference whether there is 1 rule or 100 rules in the rule engines. When the rule engines are run with 1 rule, processing of simple rule takes much time than processing of the complex rule unexpectedly. This occurs due to the approach for sending messages to the framework. In this experimental testing, messages are sent one by one and these messages are put in the rule engine as soon as they come. When the first message arrives at the rule engine, the related rule is activated in the memory and the other messages (other 7 messages for complex rule) do not repeat this memory activation process. After the last message (7^{th} message) arrives, the rule is executed and this execution process is similar with the simple rules. If the messages are stored in temporary storage without inserting them to the rule engine and if they are inserted to the rule engine all together when the last message arrives

then the processing of the complex rules should take more time than the processing of the simple rules.

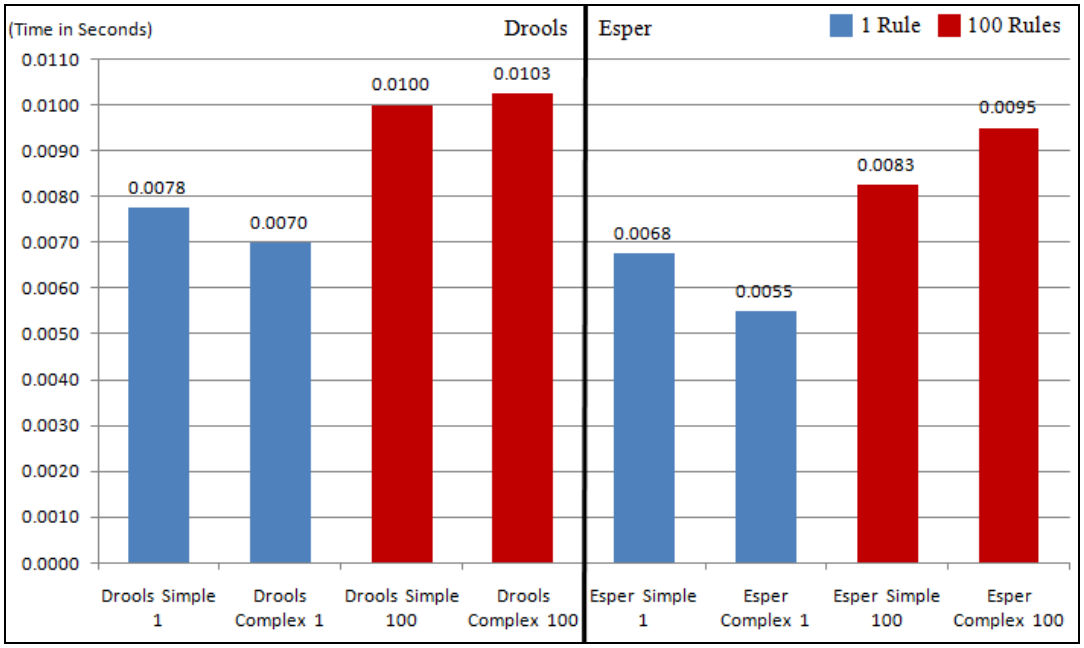

Figure 18 Rule Engines Comparison

Figure 19 compares the times consumed in different steps when jBPM and YAWL workflow engines are used. The effects of the messaging systems, rule engines and rule complexity are excluded by taking the average scores. This graphic shows that jBPM significantly performs better than YAWL. This occurs due to InterfaceB communication between step 4 and 5. The time between these two steps also affects the general difference when the YAWL engine is used. The time between steps 5 and 6 shows the elapsed time between starting a task and completing a task. This difference is also significantly higher for YAWL than jBPM. According to this figure, it cannot be said that there is a significant difference between the consumed times when the number of workflows loaded in the workflow engines are different (1 workflow or 100 workflows).

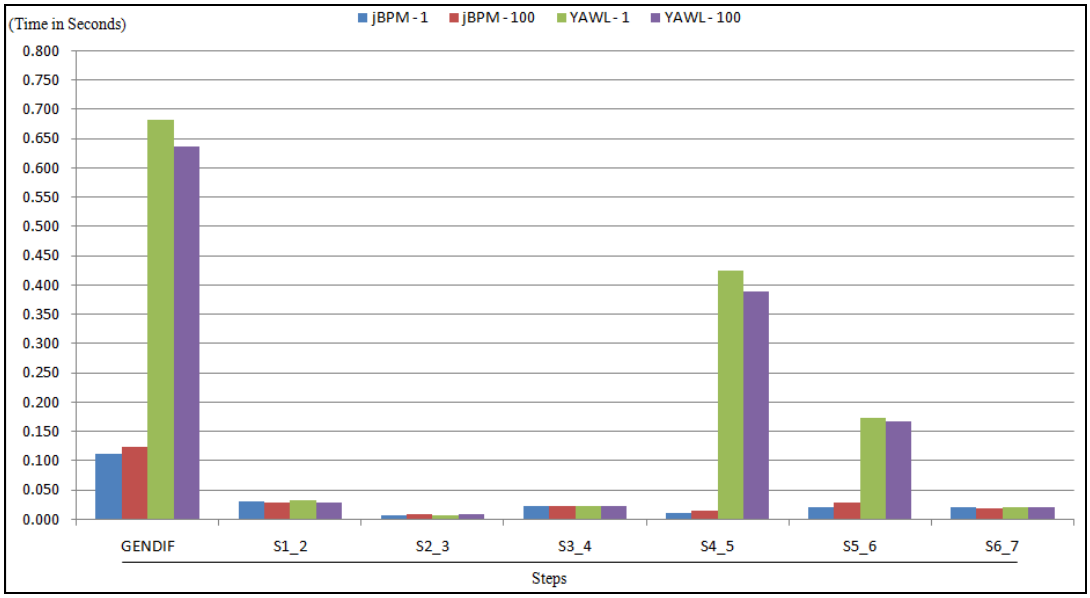

Figure 19 Workflow Engines and The Number of Workflows Comparison

Figure 20 compares ActiveMQ and HornetQ messaging systems. The effects of the rule engines, workflow engines and rule complexity are excluded by taking the average scores. According to this graphic, ActiveMQ significantly performs better than HornetQ. The number of rules in the rule engines (1 or 100) and the number of workflows in the workflow engines (1 or 100) do not significantly affect the messaging system when ActiveMQ is used. However, there is a slight difference for HornetQ when 1 rule/workflow is loaded to engines and 100 rules/workflows are loaded to the engines.

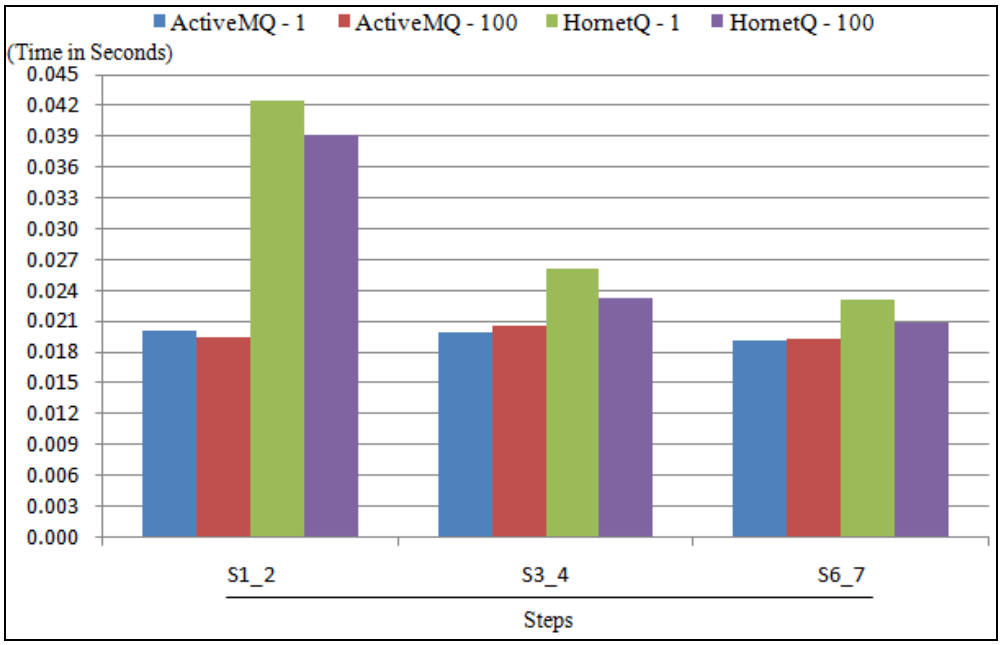

Figure 20 Messaging Systems Comparison

Figure 21 compares the time differences between different steps according to the rule complexity. The effects of the messaging systems, rule engines, workflow engines and the number of rules/workflows are eliminated by averaging the scores. There is no significant difference between any steps in terms of the rule complexity.

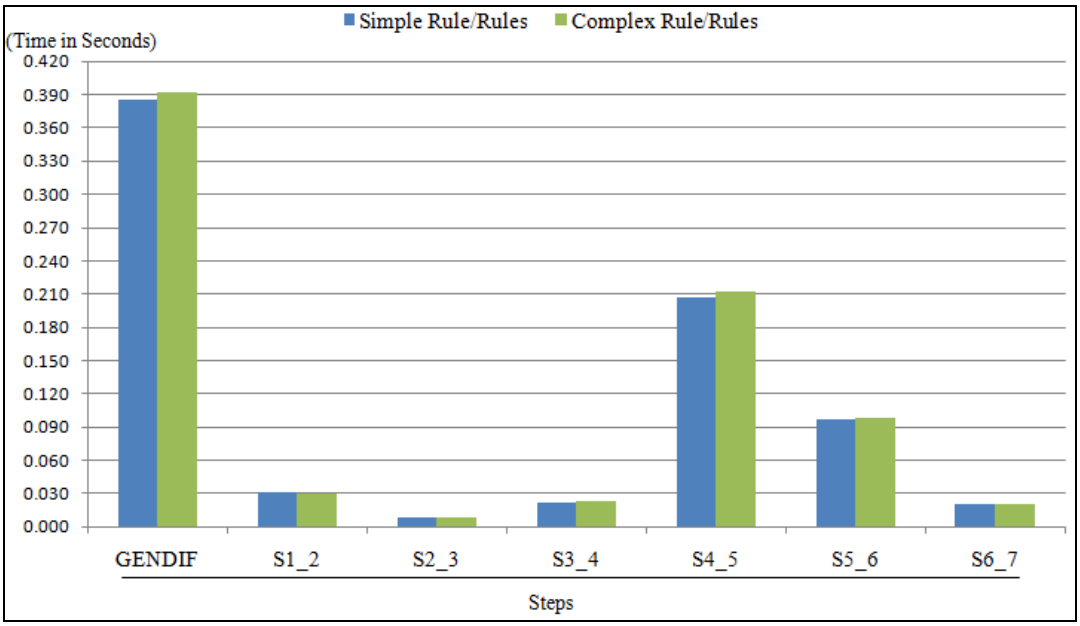

Figure 21 Simple Rule - Complex Rule Comparison

Figure 22 shows the consumed times in different steps when 1 rule/workflow and 100 rules/workflows are run in rule engines and workflow engines. There is no significant difference between number of rules/workflows except for the steps 4 and 5. Between step 4 and 5 running 1 rule/workflow takes slightly much time than 100 rules/workflows. This difference also affects the general difference.

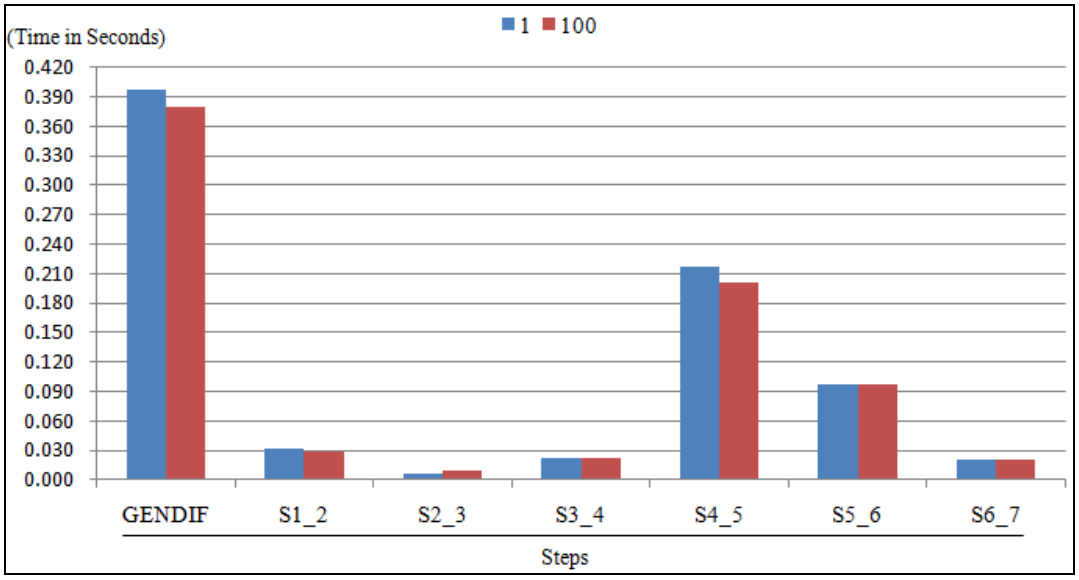

Figure 22 1 Rule/Workflow and 100 Rules/Workflows Comparison

Figure 23 shows the comparison of the consumed times at different steps for 8 different combinations of software modules that are used for experimental evaluation. The rule complexity and the number of rules/workflows are not taken into account and the scores are averaged. According to this figure, the best combination is “Esper-jBPM-ActiveMQ”. Moreover, the combinations including the jBPM always perform better between steps 4-5 and 5-6 and in the general difference.

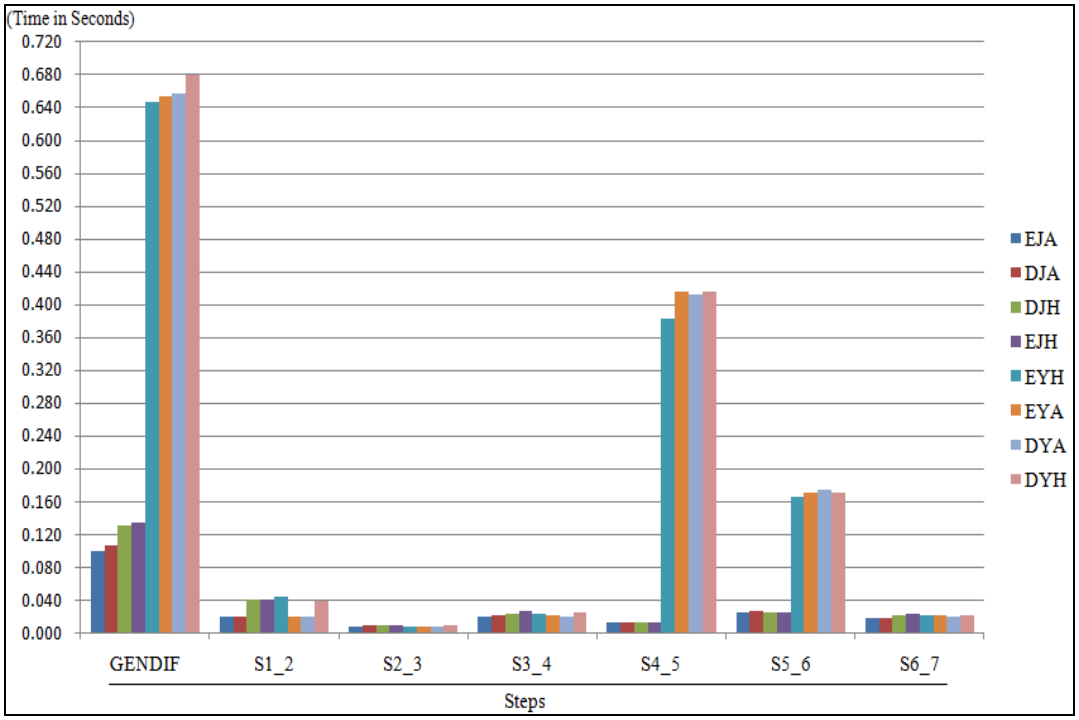

Figure 23 Comparison of Different Software Combinations

CHAPTER

6. CONCLUSION

This study proposes a framework for helping and guiding users when they are carrying out their daily activities. Hence, users' daily activities need to be modeled first, and for this purpose, workflows are used. Most of the workflow systems in the literature offer a user interface for designing workflows. However, while some of them are user-friendly, most of them are produced for designers who have technical knowledge. This is the case for workflow systems that are used in this study, YAWL and jBPM. They have user-friendly editors for designing workflows but they are designed for people who have technical knowledge. End users should be able to design their activities without any technical knowledge. Using workflow templates may ease the designing process and decrease the time to prepare a complete workflow model. Moreover, since this research proposes context-aware workflows, context information should be included into the designers. Users should easily relate the context information, raw data or high-level information, with the activities within the workflows and the workflows themselves. Additionally, user friendly mobile applications such as (Tüysüz, 2013), may allow end users to use this framework in daily life. Since this framework is used as a baseline, such applications are needed for integrating this framework into daily life.

The current study proposes a method for adaptation of workflows according to changes in users' daily life by using hierarchical adaptation concept with CEP capability. This adaptation approach is useful and it represents the real adaptation requirements of users in pervasive environments. However, other adaptation approaches used in the literature may also be used for pervasive environments, such as adding a new task to a workflow and deleting a task from the workflow at runtime. The applicability of different adaptation concepts can be proposed and implemented for the framework.

Software for pervasive environments should include a context-aware system, since users interact with their environments continuously throughout a day. The solutions proposed in the literature integrate context with the workflow language which increases the complexity of the workflows. This study proposes an EDA-based approach for loosely coupling of context information with the workflows. This loosely coupling approach allows workflow systems and context-management systems to evolve on their own. Currently, many improvements have been done in the literature for both of the concepts. Workflow patterns are offered for workflow systems which allow enhancing the design process. Similarly, context modeling and reasoning methods are continuously developing. For this reason, producing a new language by integrating the workflow systems by context management systems may constrain the evolution of these systems

.

Using EDA enables the main components of the framework to evolve on their own without restrictions coming from the tight integration. EDA also enables this framework to become a modular system because the components of the system can easily be replaced by their similar counterparts. For the implementation part, a JMS-based publish-subscribe type messaging system is used for implementing the event-driven architecture. JMS is generally used for local systems communicating through TCP. However, the framework can be used and evaluated using an internet-based publish-subscribe approach. Such an approach may allow this framework to scale well for very large number of users, since internet-based publish-subscribe approaches offer decentralized architectures.

This framework also proposes different levels of automation which in fact is a complex concept. In this framework, users can change the automation level by adding or subtracting rules, and subscribing or unsubscribing to the data sources. If a user makes necessary subscriptions and writes the rules, the framework tracks his activities automatically and provides help when it is really needed. Unobtrusiveness is one of the most important properties of the software and hardware systems for pervasive environments. Adjusting automation level according to users' preferences allows software and hardware technologies to be invisible and turns the technology to "calm technology".

The scalability of a framework for pervasive environments is an important concept since large number of users, devices and resources are continuously interacting within these environments. Experimental tests show that the current implementation of the prototype may not be enough in terms of the consumption of computer resources when this framework is open to public. Several solutions can be suggested. First of all, the workflow engine and rule engine should be run in different cluster of computers running in a load balanced fashion. Second, experimental tests show that the HTTP calls to the workflow engines for task and case operations take much time to complete. Instead of HTTP calls, TCP-based or JNP-based communication types may be selected for overcoming this bottleneck. Third, if many concurrent users use the framework the messaging systems may be another bottleneck. As a solution, most of the messaging systems propose clustered broker implementation as is the case with ActiveMQ and HornetQ. Since the framework implemented in this research is a prototype for testing functionality, the scalability aspects of the prototype is left for future study.

Extensibility of a framework for pervasive environments is another important concept. Event-driven architectures used in this research support flexibility and extensibility which allow systems to evolve (Taylor, Yochem, Phillips, & Martinez, 2009). Other systems can be easily plugged into the framework and subscribe to low-level and high-level events thanks to the EDA. It is estimated that two important components may increase the value of the framework through using this extensibility feature of the proposed framework. First, since rule-based context reasoning is provided by the framework and the context data is modeled in key-value pairs, a semantic context reasoning component may be plugged to the framework. Semantic reasoning applications use ontologies and ontologies have high and formal expressiveness (Baldauf, Dustdar, & Rosenberg, 2007). Moreover, since defining an ontology offers a uniform way of describing the structure of a domain, ontologies enable knowledge sharing and reuse (Strang & Linnhoff-Popien, 2004). Second, another important component that may increase the value of the framework is knowledge discovery component. The framework may automatically produce different workflows and offer them to the end users by examining the everyday activities of the users. Data mining and process

mining techniques may be used for discovering the activities and producing rules automatically.

Another extensibility issue for the framework is the modular approach that is used. Thanks to this modular approach, different software can be used for workflow engine, rule engine and messaging system and the effort needed for replacing one of these with another may affect the implementation decisions. For replacing a component, an adaptor-like program should be developed and integrated to the *SOMNIUM* modules. Since modules used are complex, replacing one module with another may be time-consuming if the details of the replacing component is unknown and needs to be done by professionals. However, if the implementers have knowledge about the replacing component, several lines of code can be easily written for integration. Table 3 shows the number of lines of code to be written in Java language for the components used in the framework. Besides this, if the used rule engine component is replaced with another rule engine, rules should also be converted to the language of the new software. The rules required for pervasive environments may be complex and converting these rules may be time-consuming if the details of the new language are unknown. Moreover, if the workflow engine is replaced with another, the workflow definition files should also be converted. Obviously, if both of the workflow engines support a common standard known as BPMN, this conversion is not needed.

Table 3 Lines of Code Needed for Replacing Modules

Module	**Lines of Code**
YAWL	43
jBPM	106
Drools	55
Esper	39
ActiveMQ	26
HornetQ	29

In the *SOMNIUM* framework, it is proposed that workflow patterns are appropriate modeling tools for modeling user daily activities in pervasive environments. Although these patterns are produced mostly by examining the business processes, they also exist in daily activity patterns of the people. A future study may show how other workflow patterns can be used for modeling widely used activity patterns in everyday life.

REFERENCES

Aalst, W. M. P. Van Der, Dumas, M., Hofstede, A. H. M., Russell, N., Verbeek, H. M. W., & Wohed, P. (2005). Life after BPEL? *Russell The Journal Of The Bertrand Russell Archives*, *3670*(2), 35–50. Retrieved from http://www.springerlink.com/index/p4743p8w27ktm5r5.pdf

Aalst, W. M. P. Van Der, Hofstede, A. H. M., & Weske, M. (2003). Business Process Management : A Survey. *Business*, *2678*(1), 1–12. doi:10.1007/3-540-44895-0_1

Aalst, W. M. P., & ter Hofstede, A. H. M. (2012). Workflow patterns put into context. *Software & Systems Modeling*, 1–5. doi:10.1007/s10270-012-0233-4

Abbasi, A. Z., Ahsan, M. U., Shaikh, Z. A., & Nasir, Z. (2010). CAWD: A tool for designing context-aware workflows. *2nd International Conference on Software Engineering and Data Mining (SEDM)*.

Abbasi, A. Z., & Shaikh, Z. A. (2009). A Conceptual Framework for Smart Workflow Management. In *2009 International Conference on Information Management and Engineering* (pp. 574–578). IEEE. doi:10.1109/ICIME.2009.95

Adams, M., Edmond, D., & Ter Hofstede, A. H. M. (2003). The application of activity theory to dynamic workflow adaptation issues. In *Proceedings of the 2003 Pacific Asia Conference on Information Systems PACIS 2003* (Vol. 369, pp. 1836–1852). Citeseer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.3874&rep=rep1&type=pdf

Adams, M., Hofstede, A. T. E. R., Russell, N., & Aalst, W. I. L. V. A. N. D. E. R. (2009). Dynamic and context-aware process adaptation. *Russell The Journal Of The Bertrand Russell Archives*, (ii), 104–136.

Adams, M., Ter Hofstede, A. H. M., Edmond, D., & Van Der Aalst, W. M. P. (2006). Worklets: A service-oriented implementation of dynamic flexibility in workflows. *On the Move to Meaningful Internet Systems 2006 CoopIS DOA GADA and ODBASE*, *4275*, 291–308. doi:10.1007/11914853_18

Allen, R. (2001). Workflow: an introduction. *Workflow Handbook*, 15–38. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Workflow+:+An+Introduction#0

ALLOW - Adaptable Pervasive Flows Project. (2008). Retrieved from http://www.allow-project.eu/

Amend, M., Ford, M., Endpoints, A., Keller, C., & Rowley, M. (2007). Web Services Human Task. *Business*, *8*(June), 206–213. doi:10.1007/s11121-007-0070-9

Amigo - Ambient intelligence for the networked home environment. (2008). Retrieved February 19, 2014, from http://www.hitech-projects.com/euprojects/amigo/index.htm

Apache JMeter. (n.d.). *The Apache Software Foundation*. Retrieved March 22, 2014, from http://jmeter.apache.org/

Architect, E. I., & Railways, D. (2006). How EDA extends SOA and why it is important. *Integration The Vlsi Journal*, *5*(December), 1–6. Retrieved from http://soa-eda.blogspot.com/2006/11/how-eda-extends-soa-and-why-it-is.html

Ardissono, L., Furnari, R., Goy, A., Petrone, G., & Segnan, M. (2007). A Framework for the Management of Context-Aware Workflow Systems. *Architecture*, 80–87. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.7648&rep=rep1&type=pdf

Arduino. (n.d.). Arduino. Retrieved March 21, 2014, from http://www.arduino.cc/

Arthur ter Hofstede, S. C. M. A. (2010). *YAWL - Technical Manual* (pp. 15–44). Retrieved from http://www.yawlfoundation.org/manuals/YAWLTechnicalManual2.1.pdf

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263–277. doi:10.1504/IJAHUC.2007.014070

Bali, M. (2009). *Drools JBoss Rules 5.0 Developer's Guide*. *Birmingham UK Packt Publishing*. Packt Publishing. Retrieved from http://www.amazon.com/dp/1847195644

Bellavista, P., Corradi, A., Montanari, R., & Stefanelli, C. (2003). Context-aware middleware for resource management in the wireless Internet. *IEEE Transactions on Software Engineering*, *29*(12). doi:10.1109/TSE.2003.1265523

Bikakis, A., Patkos, T., Antoniou, G., & Plexousakis, D. (2008). A Survey of Semantics-Based Approaches for Context Reasoning in Ambient Intelligence. *Proceedings of the IEEE*, *11*(1), 14–23. Retrieved from http://discovery.ucl.ac.uk/1321817/

Boari, M., Lodolo, E., Monti, S., & Pasini, S. (2008). Middleware for automatic dynamic reconfiguration of context-driven services. *Microprocessors and Microsystems*, *32*(3), 145–158.

Börger, E. (2011). Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. *Software Systems Modeling*, 1–24. doi:10.1007/s10270-011-0214-z

Buchmann, A., & Koldehofe, B. (2009). Complex Event Processing. *It - Information Technology*, *51*(5), 241–242. doi:10.1524/itit.2009.9058

Chan, A. T. S., & Chuang, S.-N. C. S.-N. (2003). MobiPADS: a reflective middleware for context-aware mobile computing. *IEEE Transactions on Software Engineering*, *29*(12). doi:10.1109/TSE.2003.1265522

Chappell, D. (2009). *The Workflow Way: Understanding Windows Workflow Foundation*. Retrieved from http://www.davidchappell.com/TheWorkflowWay--Chappell.pdf

Chen, H., Finin, T., & Joshi, A. (2004). Semantic Web in the context broker architecture. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications*.

Cho, Y., Choi, J., & Choi, J. (2007). A Context-Aware Workflow System for a Smart Home. *2007 International Conference on Convergence Information Technology ICCIT*, *6*(2), 95–100. doi:10.1109/ICCIT.2007.263

Cho, Y., Shin, C., Park, D., Cho, S., Cho, K., Park, J., & Yoe, H. (2010). A Workflow Service Scenario Based on uWDL for Smart Agriculture. In *2010 5th International Conference on Embedded and Multimedia Computing* (pp. 1–4). IEEE. doi:10.1109/EMC.2010.5575749

Cugola, G., & Jacobsen, H. A. (2002). Using publish/subscribe middleware for mobile systems. *ACM SIGMOBILE Mobile Computing and Communications Review*, *6*(4), 25–33. doi:10.1145/643550.643552

Dargie, W. (2009). *Context-Aware Computing and Self-Managing Systems (Chapman & Hall/CRC Studies in Informatics Series)* (p. 405). Chapman and Hall/CRC.

De Souza, L., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., & Savio, D. (2008). Socrades: A web service based shop floor integration infrastructure. *The Internet of Things*, *4952*, 50–67. doi:10.1007/978-3-540-78731-0_4

Digi XBee Wireless RF Module. (n.d.). Retrieved December 14, 2012, from http://www.sparkfun.com/products/8664

Drools Expert User Guide. (n.d.). Retrieved December 14, 2012, from http://docs.jboss.org/drools/release/5.4.0.Beta2/drools-expert-docs/html_single/index.html#d0e26

Drools Fusion User Guide Version 5.5.0.Final. (2012). Retrieved from http://docs.jboss.org/drools/release/5.5.0.Final/drools-fusion-docs/html_single/index.html

Eberle, H., Leymann, F., & Unger, T. (2011). Implementation Architectures for Adaptive Workflow Management. In *ADAPTIVE 2010* (pp. 1–6). Xpert Publishing Services. Retrieved from http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2011-03&engl=0

Ejigu, D., Scuturici, M., & Brunie, L. (2007). CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing. *Fourth International Conference on Information Technology (ITNG'07)*. doi:10.1109/ITNG.2007.49

Esper Reference. (2012). Retrieved from http://esper.codehaus.org/esper-4.7.0/doc/reference/en-US/pdf/esper_reference.pdf

Eugster, P., Garbinato, B., & Holzer, A. (2006). Pervaho: A Development and Test Platform for Mobile Ad hoc Applications. In *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems Workshops* (pp. 1–5). Ieee. doi:10.1109/MOBIQW.2006.361719

Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. *Artificial Intelligence*, *19*(1), 17–37.

Garlan, D., Siewiorek, D. P., Smailagic, A., & Steenkiste, P. (2002). Project Aura: toward distraction-free pervasive computing. *Ieee Pervasive Computing*, *1*(2), 22–31. doi:10.1109/MPRV.2002.1012334

Getting Started with Java Message Service (JMS). (2004). Retrieved March 22, 2014, from http://www.oracle.com/technetwork/articles/java/introjms-1577110.html

Gu, T., Pung, H. K., & Zhang, D. Q. (2004). A middleware for building context-aware mobile services. *2004 IEEE 59th Vehicular Technology Conference VTC 2004Spring IEEE Cat No04CH37514*, *5*(1), 2656–2660. doi:10.1109/VETECS.2004.1391402

Han, J., Cho, Y., & Choi, J. (2005). Context-aware workflow language based on web services for ubiquitous computing. *Computational Science and Its Applications–ICCSA 2005*, 1008–1017. Retrieved from http://www.springerlink.com/index/t588qyd8t6n8218g.pdf

Han, J., Cho, Y., Kim, E., & Choi, J. (2006). A Ubiquitous Workflow Service Framework. *Computational Science and Its ApplicationsICCSA 2006*, 30–39. Retrieved from http://www.springerlink.com/index/1125k33326572g76.pdf

Hermosillo, G., Seinturier, L., & Duchien, L. (2010). Using Complex Event Processing for Dynamic Business Process Adaptation. *2010 IEEE International Conference on Services Computing*, 466–473. doi:10.1109/SCC.2010.48

Herrmann, K., Rothermel, K., Kortuem, G., & Dulay, N. (2008). Adaptable Pervasive Flows - An Emerging Technology for Pervasive Adaptation. *2008 Second IEEE International Conference on SelfAdaptive and SelfOrganizing Systems Workshops*, 108–113. doi:10.1109/SASOW.2008.25

Hill, E. F. (2003). *Jess in Action: Java Rule-Based Systems*. *Environment*. Manning Publications Co.

HornetQ - putting the buzz in messaging. (n.d.). Retrieved March 22, 2014, from http://www.jboss.org/hornetq

Hsu, H.-J., Wu, S.-Y., & Wang, F.-J. (2010). A Methodology to Developing Context-Aware Pervasive Applications. In *2010 Fifth IEEE International Symposium on Service Oriented System Engineering* (pp. 206–213). IEEE. doi:10.1109/SOSE.2010.63

Huang, Y., & Garcia-Molina, H. (2004). Publish/Subscribe in a Mobile Environment. *Wireless Networks*, *10*(6), 643–652. doi:10.1023/B:WINE.0000044025.64654.65

jBPM Documentation Version 6.0.1.Final (No. Chapter 23). (2013). Retrieved from http://docs.jboss.org/jbpm/v6.0.1/userguide/

Kawsar, F., Kortuem, G., & Altakrouri, B. (2010). Supporting Interaction with the Internet of Things across Objects , Time and Space. *Design Issues*, 1–8. doi:10.1109/IOT.2010.5678441

Kiani, S. L., Riaz, M., Zhung, Y., Lee, S. L. S., & Lee, Y.-K. L. Y.-K. (2005). A distributed middleware solution for context awareness in ubiquitous systems. In *11th IEEE International Conference on Embedded and RealTime Computing Systems and Applications RTCSA05* (Vol. 0, pp. 451–454). IEEE Computer Society. doi:10.1109/RTCSA.2005.9

Kindberg, T. I. M., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., … Serra, B. (2002). People , Places , Things : Web Presence for the Real World. *Mobile Networks and Applications*, *7*(5), 365–376.

Kortuem, G., Kawsar, F., & Altakrouri, B. (2010). Flow-driven ambient guidance. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)* (pp. 796–799). IEEE. doi:10.1109/PERCOMW.2010.5470544

Krumm, J. (2009). *Ubiquitous Computing Fundamentals*. Chapman & Hall/CRC. Retrieved from http://dl.acm.org/citation.cfm?id=1803789

Lee, C., Ko, S., Lee, S., Lee, W., & Helal, S. (2007). Context-Aware Service Composition for Mobile Network Environments. *Korea Science Engineering Foundation*, *4611*, 941–952.

Leutnant, V., Schmalenstroeer, J., & Poortinga, R. (2007). *Context Management Service, IST Amigo Project*. Retrieved from https://www.google.com.tr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCUQFjAA&url=https://gforge.inria.fr/frs/download.php/3229/CMS-Tutorial.pdf&ei=MmcjUufMB8KGhQeL3YDwDg&usg=AFQjCNEySW2DnO-qu1-bgNKcqiOcN64KpA&sig2=Bi4r69OlnBtktGYE7BGYXQ&bvm=bv.51495398,d.ZG4

Leymann, F., Unger, T., & Wagner, S. (2010). On designing a people-oriented constraint-based workflow language. In C. Gierds & J. Sürmeli (Eds.), *Proceedings of the 2nd CentralEuropean Workshop on Services and their Composition ZEUS 2010 Berlin Germany February 2526 2010* (Vol. 563, pp. 25–31). CEUR-WS.org. Retrieved from http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2010-13&engl=0

Mans, R. S., van der Aalst, W. M. P., Russell, N. C., & Bakker, P. J. M. (2009). Flexibility Schemes for Workflow Management. *BPM 2008 Workshops*, *17*, 361–372.

Marinovic, S., Twidle, K., & Dulay, N. (2010). Teleo-Reactive workflows for pervasive healthcare. *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops PERCOM Workshops*, 316–321. doi:10.1109/PERCOMW.2010.5470648

Michelson, B. M. (2006). Event-Driven Architecture Overview. *Architecture*, 8. doi:10.1571/bda2-2-06cc

Naming on JBoss. (2004). *JBoss Inc.* Retrieved February 12, 2014, from https://docs.jboss.org/jbossas/jboss4guide/r1/html/ch3.chapter.html

Orton, J. D., & Weick, K. E. (1990). Loosely Coupled Systems: A Reconceptualization. *Academy of Management Review*, *15*(2), 203–223. doi:10.2307/258154

Ouyang, C., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., & La Rosa, M. (2007, October 20). Service-oriented processes : an introduction to BPEL. *Semantic Web Services : Theory, Tools, and Applications*. Information Science Reference (IGI Global). Retrieved from http://eprints.qut.edu.au/15271/1/15271.pdf

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems Man and Cybernetics Part A Systems and Humans*, *30*(3), 286–297. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11760769

Pesic, M., Schonenberg, H., & Van Der Aalst, W. M. P. (2007). DECLARE: Full Support for Loosely-Structured Processes. *11th IEEE International Enterprise Distributed Object Computing Conference EDOC 2007*, 287–287. doi:10.1109/EDOC.2007.14

Ranganathan, A., & McFaddin, S. (2004). Using workflows to coordinate Web services in pervasive computing environments. *Proceedings IEEE International Conference on Web Services 2004*, 288–295. doi:10.1109/ICWS.2004.1314750

Red Hat JBoss BRMS. (n.d.). Retrieved March 22, 2014, from http://www.redhat.com/products/jbossenterprisemiddleware/business-rules/

RedBee RFID Reader. (n.d.). Retrieved March 22, 2014, from http://www.sparkfun.com/products/10073

Reichert, M., Rinderle, S., & Dadam, P. (2003). ADEPT workflow management system: flexible support for enterprise-wide business processes. *Information Systems Journal*, 370–379. Retrieved from http://portal.acm.org/citation.cfm?id=1761169

Rellermeyer, J., Riva, O., & Alonso, G. (2008). AlfredO: an architecture for flexible interaction with electronic devices. *Middleware 08 Proceedings of the 9th ACMIFIPUSENIX International Conference on Middleware*. Retrieved from http://portal.acm.org/citation.cfm?id=1496950.1496953

Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., & Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *Ieee Pervasive Computing*, *1*(4), 74–83. doi:10.1109/MPRV.2002.1158281

Russell, N., & Ter Hofstede, A. H. M. (2009). Surmounting BPM challenges: the YAWL story. *Computer Science Research and Development*, *23*(2), 67–79. doi:10.1007/s00450-009-0059-7

Rüdiger Pryss, Julian Tiedeken, & Manfred Reichert. (2010). Managing Processes on Mobile Devices: The MARPLE Approach. In *CAiSE'10 Demos*. Retrieved from http://dbis.eprints.uni-ulm.de/663/

Salatino, M., & Aliverti, E. (2012). *jBPM5 Developer Guide* (p. 364). Packt Publishing. Retrieved from http://www.amazon.com/jBPM5-Developer-Guide-Mauricio-Salatino/dp/1849516448

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *Ieee Personal Communications*, *8*(4), 10–17. doi:10.1109/98.943998

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., & Van Der Aalst, W. (2008). Towards a Taxonomy of Process Flexibility. *Information Systems Journal*, 81–84. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.6746&rep=rep1&type=pdf#page=87

Sirin, E., Parsia, B., & Hendler, J. (2004). Template-based Composition of Semantic Web Services. *AAAI Fall Symposium on Agents and the Semantic Web*, *FS-05-01*(1), 85–92. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.2584

Smanchat, S., Ling, S., & Indrawan, M. (2008). A survey on context-aware workflow adaptations. *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia MoMM 08*, *13*(1), 414. doi:10.1145/1497185.1497274

Snyder, B., Bosanac, D., & Davies, R. (2011). *ActiveMQ in Action*. (J. Bleiel, Ed.)*Online* (p. 406). Manning Publications Co.

Strang, T., & Linnhoff-Popien, C. (2004). A Context Modeling Survey. *Graphical Models*, *Workshop o*(4), 1–8. doi:10.1.1.2.2060

Taylor, H., Yochem, A., Phillips, L., & Martinez, F. (2009). *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise* (1st ed.). Addison-Wesley Professional. Retrieved from http://www.citeulike.org/user/Scis0000002/article/6512654

Tiedeken, J., Kreher, U., & Reichert, M. (2010). Towards flexible process support on mobile devices. *Forum American Bar Association*, *72 LNBIP*, 150–165. doi:10.1007/978-3-642-17722-4_11

Truong, H., & Dustdar, S. (2009). A survey on context-aware web service systems. *International Journal of Web Information Systems*, *5*(1), 5–31. doi:10.1108/17440080910947295

Tüysüz, G. (2013). *A Workflow-based mobile guidance framework for managing personal activities*. Middle East Technical University.

Unger, T., Eberle, H., & Leymann, F. (2010). Research challenges on person-centric flows. In C. Gierds & J. Sürmeli (Eds.), *Proceedings of the 2nd CentralEuropean Workshop on Services and their Composition ZEUS 2010 Berlin Germany February 2526 2010* (Vol. 563, pp. 97–104). CEUR-WS.org. Retrieved from http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2010-12&engl=0

Unger, T., Eberle, H., Leymann, F., & Wagner, S. (2010). An event-model for constraint-based person-centric flows. In *2010 IEEE International Conference on Progress in Informatics and Computing* (pp. 927–932). IEEE. doi:10.1109/PIC.2010.5687886

Unger, T., Eberle, H., Marconi, A., & Sirbu, A. (2010). *Declarative language for goals, constraints, adaptation and evolution*. Retrieved from http://www.allow-project.eu/deliverables/allow_d3-2_v1_0final.pdf

Unger, T., & Hanna, E. (2008). *Basic flow-model and language for Adaptable Pervasive Flows*. Retrieved from http://www.allow-project.eu/deliverables/allow_d3-1_v1_0final.pdf

Van Der Aalst, W. M. P., Adams, M., Hofstede, A. H. M., & Pesic, M. (2009). Flexibility as a Service. *Technology*, *5667*, 319–333. Retrieved from http://www.springerlink.com/index/m630826617341mq0.pdf

Van Der Aalst, W. M. P., Aldred, L., Dumas, M., & Ter Hofstede, A. H. M. (2004). Design and Implementation of the YAWL System. *Proceedings of the 16th International Conference on Advanced Information Systems Engineering CAiSE04*, *3084*, 281–305. Retrieved from http://www.springerlink.com/index/CPA194XBMAUDUUWN.pdf

Van Der Aalst, W. M. P., Pesic, M., & Schonenberg, H. (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science Research and Development*, *23*(2), 99–113. doi:10.1007/s00450-009-0057-9

Van Der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, *14*(1), 5–51. doi:10.1023/A:1022883727209

Van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). Process mining: a research agenda. *Computers in Industry*. Retrieved from http://www.sciencedirect.com/science/article/pii/S0166361503001945

Vigneras, P. (2008). InfoQ: Why BPEL is not the holy grail for BPM. Retrieved March 22, 2014, from http://www.infoq.com/articles/bpelbpm

Vincent, P. (2007). Differences between a BRE and a rule-driven CEP engine (Part 1) | The TIBCO Blog. Retrieved October 15, 2013, from http://www.tibco.com/blog/2007/06/26/differences-between-a-bre-and-a-rule-driven-cep-engine-part-1/

Von Ammon, R., Emmersberger, C., Greiner, T., Springer, F., & Wolff, C. (2008). Event-Driven Business Process Management. In *2nd International Conference on Distributed Event-Based Systems*. Retrieved from http://epub.uni-regensburg.de/6829/1/edBPMDEBS2008.pdf

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. doi:10.1109/PERCOMW.2004.1276898

Wieland, M., Kaczmarczyk, P., & Nicklas, D. (2008). Context Integration for Smart Workflows. In *6th IEEE International Conference on Pervasive Computing and Communications PerCom (2008)* (pp. 239–242). Ieee. doi:10.1109/PERCOM.2008.27

Wieland, M., Kopp, O., Nicklas, D., & Leymann, F. (2007). Towards Context-aware Workflows. In B. Pernici & J. A. Gulla (Eds.), *CAiSE07 Proc of the Workshops and Doctoral Consortium* (Vol. 2, pp. 1–15). Citeseer. Retrieved from http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2007-18&engl=0

Wieland, M., Nicklas, D., & Leymann, F. (2008). Managing technical processes using smart workflows. *Towards a ServiceBased Internet First European Conference ServiceWave 2008 Madrid Spain December 1013 2008 Proceedings*, *5377*, 287–298. Retrieved from http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2008-111&engl=0

Wolf, H., Herrmann, K., & Rothermel, K. (2009). Modeling Dynamic Context Awareness for Situated Workflows. *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, 98–107. Retrieved from http://link.springer.com/chapter/10.1007%2F978-3-642-05290-3_19

Wu, C.-L. W. C.-L., Liao, C.-F. L. C.-F., & Fu, L.-C. F. L.-C. Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology. , 37 IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews 193–205 (2007). IEEE. doi:10.1109/TSMCC.2006.886997

Yang, S. J. H., Zhang, J., & Chen, I. Y. L. (2008). A JESS-enabled context elicitation system for providing context-aware Web services. *Expert Systems with Applications*, *34*(4), 2254–2266. doi:10.1016/j.eswa.2007.03.008

Yao, W., Chu, C.-H., & Li, Z. (2010). Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications*, *34*(3), 799–810. doi:10.1016/j.jnca.2010.04.020

YAWL - User Manual Version 2.2. (2011). *The YAWL Foundation* (p. 256). Retrieved from http://www.yawlfoundation.org/manuals/YAWLUserManual2.2.pdf

VITA

Bilgin Avenoğlu was born in İstanbul, Turkey on March 10, 1978. He received his B.S. degree in Computer Education from Gazi University in June 2001 as a valedictorian. He received his M.Sc. degree in Computer Education and Instructional Technology from Middle East Technical University (METU) in 2005. He worked more than five years in several public and private companies as a Software Developer and Senior Software Developer. After then, he worked as a Research Assistant in The Department of Information Systems, METU between 2007 and 2012. His main research areas are Pervasive/Ubiquitous Computing, Intelligent Environments, Context-aware Systems, Event-driven Architectures, Rule-based Systems and Complex Event Processing and Business Process Management.

TEZ FOTOKOPİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü [ ]

Sosyal Bilimler Enstitüsü [ ]

Uygulamalı Matematik Enstitüsü [ ]

Enformatik Enstitüsü [X]

Deniz Bilimleri Enstitüsü [ ]

YAZARIN

Soyadı : ... AVENOĞLU .....
Adı : .... BİLGİN .....
Bölümü : .. BİLİŞİM SİSTEMLERİ .....

TEZİN ADI (İngilizce) : A CONTEXT-AWARE AND WORKFLOW-BASED FRAMEWORK FOR PERVASIVE ENVIRONMENTS

TEZİN TÜRÜ : Yüksek Lisans [ ] Doktora [X]

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın. [ ]
2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullancılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) [ ]
3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) [X]

Yazarın imzası ..... Tarih .. 27/02/2014 .....