2-15-2013 12:00 AM

Personal Smart Assistant for Digital Media and Advertisement

Ali Hussain
The University of Western Ontario

Supervisor
Dr. Hamada Ghenniwa
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science
© Ali Hussain 2013

PERSONAL SMART ASSISTANT FOR DIGITAL MEDIA & ADVERTISEMENT

Thesis format: Monograph

by

Ali Hussain

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

The expansion of the cyberspace and the enormous process in computing and software applications enabled technology to cover every aspect of our life, therefore, many of our goals are now technology driven. Consequently, the need of intelligent assistance to achieve these goals has increased. However, for this assistance to be beneficial for users, it should be targeted to them based on their needs and preferences. Intelligent software agents have been recognized as a promising approach for the development of user-centric, personalized, applications.

In this thesis a generic personal smart assistant agent is proposed that provides relevant assistance to the user based on modeling his/her interests and behaviours. The main focus of this work is on developing a user behaviour model that captures the deliberative and reactive behaviours of the user in open environments. Furthermore, a prototype is built to utilize the personal assistant for personalized advertisement applications, where this assistant attempts to recommend the right advertisement to the right person at the right time.

Keywords

Personal Smart Assistant, Personalized Advertisements, User Modeling, Intelligent Agent.

Acknowledgments

I would like to gratefully acknowledge the efforts of my supervisor Prof. Hamada Ghenniwa. This thesis would not have been accomplished without his valuable guidance, encouragement, and support throughout my master’s research journey.

I would also like to acknowledge the help and support that I have received from my colleagues in the Cooperative Distributed Systems (CDS) group. Working with all of them and being part of the CDS group is always a pleasure.

My deepest gratitude goes to my family, especially my parents, my sister, and my brother. Their ultimate kindness, love, support and encouragement made this achievement possible.

Last but not least, special thanks go to those who read the drafts of this dissertation and helped in improving it by giving their feedbacks and suggestions.

Table of Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

This chapter provides an introduction to the topic of this research work: the Personal Smart Assistant (PSA).

1.1 Smart Assistance

The core issue behind the need of Smart Assistance (SA), regardless of the domain in which it is applied for, is the user's lack of knowledge. The user does not have the complete knowledge that would help him/her to achieve his/her goals, and therefore, assistance is needed to update the user's knowledge to achieve these goals.

The user interface, an application-specific assistant, is an example of a smart assistant that reflects the different functionalities of the application, and update the user's knowledge about this application. The user interface models the user's preferences in a user model and dynamically accommodates them.

With the expansion of cyberspace, and the enormous process in computing and software applications, technology is covering every aspect of our lives, and therefore, many of our tasks and goals are now technology driven. Consequently, the problem of lack of knowledge has increased; as the user now might be required to work with many complicated applications to achieve his/her goal. Therefore, a form of smart assistance, beyond the user interface, is essential.

1.2 Personal Smart Assistant

The Personal Smart Assistant (PSA) is a software system that helps users to achieve their goals efficiently. The PSA can work on behalf of the user to achieve a task delegated by the user directly. Furthermore, it can provide a proactive assistance based on the current context of the user and the environment.

Unlike the user interface, which is application specific, the personal smart assistant models the environment in which the user is working, and it provides the functionalities of the different applications to the user based on the user's preferences and needs that are modeled in the user model.

The user model is an essential component in the personal assistant. Modeling the user's behaviours, interests and goals are required in order to provide an assistance that is relevant and personalized to the specific user which this assistance is intended to.

1.3 Personalized Advertising

By the advancement of digital media contents, advertisements have become more appealing and attractive. This enhanced the role of advertisements that they play in raising the awareness of the users about the various choices that are available to them, and therefore, influence user's decisions regarding these available choices. However, for these advertisements to be rewarding to businesses, they have to be beneficial for the users and help them accomplish their goals.

Personalized advertisements can be achieved through the personal smart assistant. In this context, the personal smart assistant provides the assistance of matching the user's behaviours and interests with the contents of the advertisements, hence, matching the right advertising message to the right person in the right context.

1.4 Scope of the Thesis

In this thesis, a generic personal smart assistant is proposed. The major part of the proposal is concentrated on the user model, which is an essential part of the personal assistant. Also, an implementation of a prototype of the proposed personal assistant is provided for the domain of personalized advertisements.

1.5 Thesis Organization

This thesis is organized as follows; Chapter 2 presents a literature review of concepts and models related to personal smart assistants, and various approaches proposed in literature. Chapter 3 defines and analyzes the problem addressed in this thesis, illustrates the motivation behind the research. Furthermore, it lists the major research issues and challenges in existing solutions and provides hypotheses to the thesis. Chapter 4 proposes our approach in building a generic personal smart assistant that overcome the challenges discussed in Chapter 3. Chapter 5 includes an implementation of the prototype of the proposed personal smart assistant in the domain of personalized advertisements, results and observations are given in this chapter to support the hypotheses in Chapter 3. Finally, conclusions, limitations and future work are given in Chapter 6.

Chapter 2

2 Literature Review: Background and Related Work

This chapter reviews background concepts as well as the related research work in the area of personal smart assistant.

2.1 Personal Smart Assistant

The personal smart assistant helps the user to achieve his/her goals. The personal smart assistant (PSA) can achieve tasks that are directly delegated to it by the user, the PSA can also monitor the user's actions and recommend different paths of actions for the user to take when needed, or, the PSA can predict the user's actions and provide a proactive assistance that helps the user when performing his/her next predicted actions. Many different approaches are proposed in the literature for building a personal assistant that can provide one or more of these types of assistance.

One of the most well-known personal assistants is Siri [1]. Siri is an intelligent personal assistant that help mobile users to access a wide variety of online resources as well as helping the users in their daily life activities. It allows the user to state his/her request in speech, typing, or selecting from options, and transfer this request into services to complete the user's task. The breakthrough of Siri is that it enables users to express their request in plain natural language, "Users interact with Siri in plain English, and Siri interprets the natural language in the context of dialog history, location, time-of-day and learned preferences to return personalized and action-oriented results [1]". To do so, Siri receives commands and instructions from the user, and try to learn and realize the concepts of these instructions and relate these concepts through underlying ontology [2]. Siri uses different active ontologies to narrow down the given instructions into a specific domain. An active ontology is a dynamic process of arranging entities based on ontology notions [2], [3]. Currently Siri uses basic ontologies for domains that address everyday life's tasks, such as Calendar, Maps, Weather, Friends, Messages, etc. [3]. Siri is a software agent that engages in conversations with users to get their tasks done. Siri is based on three main components [1]:

a) Conversational Interface: it enables users to submit their request in plain natural language. Siri applies an algorithm to interpret these requests and combine them with knowledge about the user's locations, time and preferences to determine the most probable understanding of the request.
b) Personal Context Awareness: Siri stores the user's preferences and locations, upon the approval of the user, to understand the user's request and fasten the process of accomplishing his/her requested task. Information stored by Siri in one domain can be automatically applied to other domains which helps accomplishing several tasks in a shorten period of time.
c) Service Delegation: Siri can reason about the services and resources that best accomplish the user's request. It uses an algorithm to filter the services based on their attributes (quality, speed, geographic constrains, etc.)

Siri intelligent assistant is based on the CALO personal assistant project, Cognitive Assistant that Learns and Organizes (CALO) [4]. The CALO project contains several frameworks and systems with different approaches and tools. However, its main domain is the office work, as it tries to help users accomplish their office work [4]. A framework was proposed in [5] to build a cognitive personal assistant agent that support the model delegation, in which the agent afford assistance in form of accomplishing the tasks that were delegated to it by the user. The major challenge targeted in this model is how to react with the infeasible delegations that could be assigned by the user. The framework extended the Belief-Desire-Intention (BDI) agency model to separate feasible and infeasible goals as well as incorporates the notion of advisability of an agent by the user [5]. The framework includes advice, goals and desires. The user delegates tasks to the agent, which become the agent's candidate goals. These goals are compared with the agent's beliefs about the current world and its capabilities and turned into adopted goals. From these adopted goals the agent derives intentions which are the committed goals and the means/plans to achieve them. Finally, these intentions are executed [5]. Although this is a very interesting approach in building a user assistant, however no details were given on how the user's decisions are made regarding delegating the tasks to the assistant, furthermore, no details were given on how to capture the agent's belief about the current world. Finally, the user is not modeled in terms of behaviours and preferences, and therefore, this agent could assist all the users in a similar way when delegated similar tasks.

Another system that is proposed as part of the CALO project is the Project Execution Assistant (PExA) [6]. PExA includes a Task Management component that organizes, filters, prioritizes, and oversees execution of tasks on behalf of the user. The task management is built on top of BDI framework called SPARK (SRI Procedural Agent Realization Kit) [6]. SPARK embraces a procedural reasoning model of problem solving based on process models. These process models are represented in a procedural language that is similar to the hierarchical task network [6]. Furthermore, a prediction model based on probabilistic modeling language (ProPL) is used to predict the successes and failures of processes and tasks [6], [7]. On the other hand, a time management component exists to help users organize their time and schedule their meetings according to preferences. These user's preferences are learned through passive learning, active learning and advice taking [6].

The CALO project has more focus towards delegated assistance; its proactivity comes mainly from determining how to solve these actions and what should be done in case of infeasibility in these actions [6]. In [8], a proactive behaviour for CALO assistant agent is discussed. A framework is proposed to provide a proactive assistant which is built on top of the BDI cognitive agent model, proposed in [5]. It defines a meta-level layer on top of the BDI framework (the based framework of CALO agent) to first, identify potential helpful tasks, and second, determine when those tasks should be performed. This proactive functionality depends on reasoning about the capabilities and commitments of the agent, as well as the user, and the state of the environment. The layer does not specify the mechanism for reasoning; rather this depends on the character of the agent as set by a proposed theory of proactivity [8]. The theory of proactivity describes the user desires, conditions under which it is safe to perform action, and a model of helpfulness. This approach also does not infer the user's goals and actions; rather, it takes these actions directly from an electronic "to do" list, and attempts to reason according to the belief and the desires of the user, and according to the proposed theory of proactivity [8].

2.2 User Modeling

In order to provide the needed assistance for the user, and help him/her to achieve his/her goals and objectives; the personal smart assistant should establish a user model/user profile that captures the user's goals, interests and behaviours [9]. The main focus of this research is on the user behaviour modeling. Capturing the behaviour of the user will enable the personal smart assistant to provide a proactive assistance to its user. According to the user's actions, the personal assistant should predict the next actions and therefore, provide a proactive help in performing these predicted actions. Since the behaviour of the user is dynamic, uncertainty is the nature of modeling the user's behaviours. Therefore, for a personal assistant to provide its proactive assistance it should be able to reason under uncertainty. The following section will briefly review the concept of uncertainty.

2.3 Uncertainty

Before addressing uncertainty, the term agent needs to be defined. According to [10], "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [10]."

Uncertainty is viewed as an inescapable factor that clouds the decision making process [11][12]. It is caused by partial observations; only some aspects of the world is observed (incomplete knowledge of the world), and therefore, we do not have a certain true state of the world [11]. Moreover, even the observed aspects of the world are often observed with some errors (incorrect knowledge about the world) [11]. Because of these limited observations, most of the relationships between aspects of the world are not deterministic; therefore, the true state of the world is rarely determined for certain [11].

As a result of uncertainty, the agent is not sure which state it is in, or in which state it will be after a sequence of actions [10]. Therefore, there exist several possibilities of the different actions that the agent could take in the world [10].

Although the agent's knowledge cannot guarantee any outcome for these different actions, the agent's knowledge could provide a degree of belief that a specific outcome will be achieved by an action. The main tool for dealing with degrees of belief is probability. Hence, a probabilistic agent can deal with uncertainty by giving a probability value (between 0 and 1) for each possible outcome of its actions, depends on this agent's degree of belief. Therefore, probability can be used to model uncertainty [10].

2.4 Probabilistic Modeling

Since the probability theory is used to model uncertainty, a brief introduction about the probability theory is necessary. This section outlines the major concepts in the probability theory [10].

2.4.1 The Basics of Probability Theory

Every possible world has a probability between 0 and 1 and that the total probability of the set of possible worlds is 1.

$$0 \leq P(\omega) \leq 1 \;\; for\ every\ \omega\ and\ \sum_{\omega \in \Omega} P(\omega) = 1 \qquad \textbf{(2-1)}$$

$\Omega \rightarrow$ Refers to sample space (the set of all possible worlds)

$\omega \rightarrow$ Refers to elements of the space

Unconditional probability (Prior): refer to degree of belief in propositions in the absence of any other information. Example: P (doubles) – the probability of rolling doubles, when throwing two dice [10].

Conditional probability (Posterior): refer to degree of belief in propositions given some evidence. Example: P (doubles | Die_1 = 5) – the probability of rolling doubles given that first die is 5 [10].

Conditional probabilities are defined in terms of unconditional probabilities as follows:

$$P(a|b) = \frac{P(a \cap b)}{P(b)} \quad \textbf{(2-2)}$$

2.4.2 Probability Distribution

Probability Distribution (P): probability is given as a vector of numbers for a specific random variable. Therefore probability distribution lists the set of possible values a random variable can take. It can be formulated as follows:

P(X) gives the values of P(X=x_i) for each possible i **(2-3)**

Discrete Probability Distribution: is a distribution in which random variables X have discrete values. Binomial distribution and Poisson distribution are examples of discrete probability distributions [13].

Binomial distribution is a discrete probability distribution for the number of successes in *n* independent trials. Each trial can take two values “success” or “failure”. Let π be the probability of success, and Y be the number of successful trials. Y can take on integer values 0, 1,…, n, the probability function of the binomial distribution can be given as follows:

$$\mathrm{f}(\mathrm{y}|\pi) = \binom{n}{y} \pi^{y} (1 - \pi)^{n-y} \quad \textbf{(2-4)}$$

Poisson distribution is another type of discrete probability distribution which counts the number of occurrence of probabilistic events over a specific period of time [13]. In Poisson distribution the values of the random variables can be countable infinite discrete values, hence, random variable X can have values of 0 ,1, 2, 3,… The mean in Poison distribution E(X) is very small compare to the maximum possible number of x values. Poisson probability function is given by [13]:

$$f(x|\lambda) = \lambda^{x} e^{-\lambda} / x! \qquad \textbf{(2-5)}$$

Where lambda λ is equal to the mean E(x)

Continuous probability Distribution is a distribution in which random variables X have continuous values. Gaussian Normal distribution is an example of continues probability distribution [13].

Gaussian Normal Distribution is continuous probability distribution which contains a mean μ, and the variance σ^2. The mean lies on the centre of the distribution, and the distribution is symmetric around the mean [13]. Given the mean μ and the variance σ^2 (and standard deviation σ), the probability density function of the Gaussian normal distribution is given as [13]:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \qquad \textbf{(2-6)}$$

Joint Probability Distribution: when probability is given as a table of numbers for multiple variables. It can be formulated as follows:

$\boldsymbol{P}(X, Y)\, gives\ the\ values\ of\ P(x_i, y_j) for\ each\ possible\ i,$ j pair as (i x j) table **(2-7)**

Full Joint Probability Distribution: specifies the probability of each complete assignment of values to all the random variables under consideration. Thus, the probability model is completely determined by the full joint probability distribution as it is able to calculate the probability of any proposition in the model [10]. However, full joint probability distribution is too large and not always available. Also, it does not scale up, as the number of considered variables increase, the complexity of the problem increases. Therefore, the full joint probability distribution is impractical and cannot be used for realistic problems. However, it is considered the theoretical foundation of more advanced approaches to building reasoning systems [10].

2.4.3 Bayes' Rule

Bayes' Rule: the probability of b given a can be calculated if we know the probability of a given b, probability of a, and the probability of b. The more general case of Baye's rule for multivalued variables can be written as:

$$\boldsymbol{P}(B|A) = \frac{\boldsymbol{P}(A|B)\boldsymbol{P}(B)}{\boldsymbol{P}(A)} \qquad \textbf{(2-8)}$$

The Bayes' Rule allows unknown probabilities to be computed from known conditional probabilities. This helps when we perceive as evidence the effect of some unknown cause, and we would like to determine the cause. However, applying Bayes' rule with many pieces of evidence run into the same scaling problem as the full joint distribution.

2.4.4 Bayesian Networks

This section explains the Bayesian networks, one of the network models used to reason under uncertainty based on the fundamentals of probability theory [10]. As discussed in Section 2.4.2, full joint probability distribution can answer any questions about the domain, but it tends to become impractical as the number of variables increases. Bayesian Network, which is a data structure that represents dependencies among variables, can represent any full joint probability distribution much more concisely.

A **Bayesian Network** is a directed acyclic graph (DAG) in which nodes represent random variables and arcs represent the dependencies between these variables [10]. Each node X_i has a conditional probability distribution **P** (X_i | Parents (X_i)) that quantifies the effect of the parents on the node, which suggests that, causes should be parents of effects. The conditional distributions of each node X in the network are shown as a conditional probability table (CPT) the conditional probabilities of each node value for every possible combination of values for the parents' node. Formally the Bayesian Network can be represented as follows [10]:

$$P(x_1, \dots, x_n) = \prod_{i=1}^{n} P(x_i \,|parents\ (x_i)) \qquad \textbf{(2-9)}$$

$P(x1, \dots, xn) \rightarrow$ Refers to the probability of a conjunction of a particular assignments to each variable

$parents\ (X_i) \rightarrow$ Refers the values of Parents (X_i) that appear in $x_1, \dots, x_n$

Equation 2-9 indicates that each entry in the joint distribution is represented by the product of the appropriate elements of the conditional probability tables (CPTs) in the Bayesian Network [10].

Inference system basically computes the conditional probability distribution of a set of query variables, given a set of evidence variables as follows:

$$P(X|\boldsymbol{e}) = \alpha\, \boldsymbol{P}(X|\boldsymbol{e}) \sum_{\boldsymbol{y}} \boldsymbol{P}(X, \boldsymbol{e}, \boldsymbol{y}) \qquad \textbf{(2-10)}$$

$X \rightarrow$ Refers to the query variables
$E \rightarrow$ Refers to the evidence variables
$Y \rightarrow$ Refers to the hidden variables
$\alpha \rightarrow$ Refers to the normalization constant

In Bayesian network, the terms $\boldsymbol{P}(X, \boldsymbol{e}, \boldsymbol{y})$ in the joint distribution can be written as products of conditional probabilities from the network. Therefore, a query can be inferred using a Bayesian network by computing sums of products of conditional probabilities from the network [10].

Exact inference algorithms such as the *enumeration algorithm*, *variable elimination algorithm* and *clustering (join tree)* evaluate the sums of products of conditional probabilities as efficiently as possible. In single connected (poly-trees) networks, the exact inference takes time linear to the size of the network [10]. However, in the general case (multiply connected networks), the problem is intractable, as it can be an N-P hard problem. Therefore, approximate inference algorithms are used in such cases, such as *likelihood algorithm* and *Markov Chain Monte Calro algorithm*. These algorithms provide a reasonable estimate of conditional probabilities distribution in the network; they can be used by much larger Bayesian Networks than exact algorithms [10].

2.4.5 Dynamic Bayesian Network

The Dynamic Bayesian Network is an extension of the Bayesian Network discussed in Section 2.4.4 that is applied to temporal models. Hence, the dynamic Bayesian network relates random variables in a series of time slices. Some of these random variables are observable (denoted as E_t) and some not (denoted as X_t) [10]. Using these variables, the transition model identifies how the world evolves by specifying the probability distribution over the latest state variables, given the previous values. Transition model is expressed as: $P(X_{t+1}|X_t)$. The sensor model identifies how the evidence variables get their values, it describes the probability of each percept at time t, given the current state of the world, and it can be expressed as: $P(E_t|X_t)$. Therefore, the specification of the complete joint distribution over all variables using equation 2-9 for any time t is given as [10]:

$$P(X_{0:t}, E_{1:t}) = P(X_0)\prod_{i=1}^{t} P(X_i|X_{i-1})\,P(E_i|X_i) \qquad \textbf{(2-11)}$$

$P(X0{:}\,t, E1{:}\,t)$ → Refers to complete joint distribution of variables from time = 0 till time = t
$P(X_0)$ → Refers to prior probability at time 0
$P(X_i|X_{i-1})$ → Refers to transition model
$P(E_i|X_i)$ → Refers to sensor (observation) model

Equation 2-11 can be used to construct the Dynamic Bayesian Network (DBN), similar to the construction of the Bayesian network from equation 2-9 in Section 2.4.4. The transition and sensor models are assumed to be stationary (they remain the same for all time t), therefore, it is convenient to simply specify them for the first slice and the complete DBN (with an unbounded number of time slices) can be constructed by copying the first slice [10].

The same algorithms that are used in static Bayesian network, mentioned in previous subsection, can also be used in DBN. Particle filtering algorithm, an enhancement of the likelihood algorithm, is an effective approximation algorithm in practice. A java library contains many of these algorithms can be found in [14].

2.4.6 Decision Theory

Decision theory is the combination of probability theory and utility theory. A decision-theoretic agent is an agent that can make rational decisions based on what it believes and what it wants [10]. The agent's preferences are captured by a utility function U(s), which assigns a single number to express the desirability of a state s. The probability of outcome s' is P (Result (a) = s'| a, e), where Result (a) denotes a random variable whose values are the possible outcome states, e is evidences observations [10]. The expected utility of an action *a* given the evidence *e, EU(a*|e), is the average utility value of the outcomes, weighted by the probability that the outcome occurs and it can be calculated as follows [10:

$$EU(a|e) = \sum \mathrm{P}\ (\mathrm{Result}\ (\mathrm{a}) = \mathrm{s}' \,|\mathrm{a}, \mathrm{e})\ \mathrm{U}(\mathrm{s}') \qquad \textbf{(2-12)}$$

The rational agent is the agent that choses the action that maximizes its expected utility [10]. Maximum Expected Utility (MEU) can be formally expressed as follows:

$$action = \mathrm{argmax}_a\, EU\ (\mathrm{a}|e) \qquad \textbf{(2-13)}$$

Decision networks are used as general mechanisms for making rational decisions. A decision network combines a Bayesian network with additional nodes of actions and utilities. It represents information about the agent`s current state, it's possible actions, the state that will result from the agent`s action, and the utility of that state. It contains three types of nodes [10]:

- Chance Nodes (Ovals) represent random variables (just as in Bayesian Network)
- Decision Nodes (Rectangles) represent points where the decision maker has a choice of actions.
- Utility Nodes (Diamonds) represent the agent`s utility function.

Actions are selected by evaluating the decision network for each possible setting of the decision node, and once this node is set, it acts as a chance node that has been set as an evidence variable [10].

Bayesian networks have been employed widely in user modeling. In [15], Horvitz et al. proposed the Lumière project which contains a decision network to model if the user needs help and what type of help the user might need. This is done based on calculating the utilities of providing an automated assistance, which is measured from the cost of the assistance due to interrupting the user, as well as, the user's acute needs for assistance based on the user's goals and actions [15]. Also the model includes dynamic Bayesian network to capture the changes in the user goals over time based on previous states.

In [16] Hue and Boutilier proposed a user model which infers the user's attributes and personalities, and decides accordingly if the user needs assistance. They used Dynamic Bayesian Network (DBN) to model their user's attributes which change over time, such as frustration (F) and neediness (N), as well as personalities, which they claim to remain static overtime such as tendency to get distracted (TD), tendency to work independently (TI), tendency towards frustration (TF), tendency towards neediness (TN). The users' acceptance to help depends on the quality of the available help, their tendency to work independently, and whether they consider help. A dynamic Bayesian network (DBN) is used, as shown in Figure 2-1, to model the system in a causal order. Variables such as the user's frustration (F), neediness (N) and consideration of help (COS) at a certain time is dependent on its value at the last time step, as seen in Figure 2-1[16].

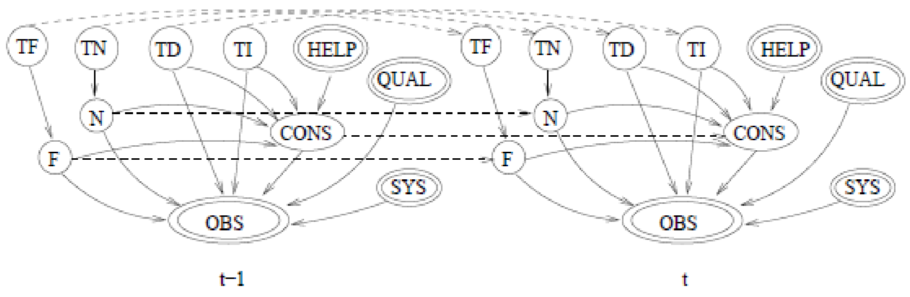


Figure 2-1: A two-steps Dynamic Bayesian Network Model [16]

The Dynamic Bayesian Network observers if the user accepts help (OBS), depending on the current state of attributes (F, N) and personalities (TF, TN, TD and TI). Furthermore, the user acceptance of the help (OBS) also depends on the quality of help (QUAL), whether or not help is available (HELP), and the direct request of help from the user (SYS) [16]. Variables such as F, N, TD and TI can be captured through observing the user's interaction with the system, for example, in their proposed text-editing system; frustration can be measured by the speed of the mouse pointer, the speed of clicks on the keyboard. While neediness to help can be measured by erasing many characters, the time of pausing, etc. [16]. Many interesting aspects have been proposed in this work, such as the use of dynamic Bayesian network to capture variable dependencies overtime, as well as, means of measuring different variables through observations of the user interaction with the system [16]. However, this work only decides whether a help is needed or not, but it does not go as far as predicting what type of help is needed. Therefore it only considers user's attributes and personalities rather than actions and interests.

Scott Brown et al [17], proposed a utility-based user model that is able offer assistance to the user not only based on what is probable, but also based on the benefit of this assistance for the user. The approach used Bayesian network to predict the user's actions and intents under uncertainty. It is used to model causal relationships between goals, environmental preconditions and actions. The random variables in the Bayesian network of this model are divided into three groups; goals, actions and pre-conditions. Each random variable that represents an action is associated with a utility function used to determine the expected utility of offering assistance to achieve the parent goal. Therefore, the expected utility of offering assistance for a goal is based not only on the probability of an action, but also on the utility of performing the action given the user is pursuing the goal [17].

In [18] a system is proposed which predicts the user's behaviour and then recommends the next behaviour, or suggests information that is relevant to the next predicted behaviour that is performed by the user [18]. The system was built on a mobile device that receives a GPS signal to detect current position and compare it with known positions in a local database, such as positions of the user's home, classroom, lab, etc. A prediction model based on Dynamic Bayesian Network is used which predicts the behaviour of the user using his/her past behaviour, the time of the day, day of the week, and the current position that is captured from the local database [18]. Based on this model the next behaviour is predicted and therefore, a recommendation is given to the user accordingly. For instance, if the next predicted behaviour of the user is eating, then the system can recommend several restaurants for the user to visit that are close to his current location, etc. [18]. One of the limitations of this work is that interests of the users are not considered. This solution only focuses on predicting the behaviour and giving assistance accordingly. Modeling the user's interests and preferences would make the recommendation more personalized. For instance, it is not enough to recommend restaurants when the next predicted behaviour is eating, rather, it is much more helpful to recommend restaurants that the user would like.

2.5 State-Based and Event-Based Modeling

As mentioned in Section 2.2, the major focus of this research is on modeling the user behaviour. State-based modeling and event-based modeling are used for behaviour modeling [19]. In state-based modeling, the system's states are emphasized and explicitly enumerated, and the behaviour is modeled in terms of the changes of states.

The user's behaviours under state-based model is deliberate, which means, the user is behaving under a specific plan to perform actions that can move the system from one state to another state till the goal is reached. Different paths of states might lead to the goal state, and hence, different sequences of actions could be possible to achieve the goal. In [20], the STRIPS (STanford Research Institute Problem Solver) problem solver is proposed to specify the different sequences of actions by generating a number of states and their dependencies. In STRIPS, a set of operators exist to transform a given initial state into a state that satisfies some stated goal conditions [20]. STRIPS problem space can be defined by an initial state, a set of operators with their preconditions and effects, and a goal [20]. This state-based partial order planning is then transferred into a directed acyclic graph which is inferred using probabilistic models to determine the likely path that the user is going to follow [21].

The state-based approach has adapted by many in the research field of user modeling. In [22] STRIPS is used to generate the activity-model for all users during a meeting. This activity model describes all combinations of the users' activities and the different sequences of actions that could achieve the goal [22]. Then probabilistic graphical model is applied to infer how probable a specific execution sequence is.

Another approach that is used to infer the user's behaviours and proactively assist the user is to identify patterns of sequences of actions that the user follows in the past to achieve his/her goals. In [23], a cooking support robot is proposed that use sensors to detect the current actions performed by the user. Furthermore, it extracts several previous sequences of actions through data mining techniques. Then, it matches the current sequence of actions performed by the user to a previous sequence of action extracted from a database. Finally, this approach considers the action that follows in the extracted sequence to be the action that the user is going to perform. This approach does not take the fact that users might follow new patterns or plans; it does not take the uncertain behaviour of the user into consideration. It only depends on existing patterns to decide the user's next action [23].

In [24], a state-based model is used to propose task assistant for persons with Dementia. A Partially Observable Markov Decision Process (POMDP) model is proposed which consists of a finite set of states, a finite set of actions, and a stochastic transition model to capture the probability of moving from one state to another when a specific action is taken [24]. The POMDP captures partial observability, noise, and the ability to customize behaviour to specific individuals by estimation of hidden user characteristics. Given the POMDP, the proposed solution then finds a policy that maximizes the expected discounted sum of rewards attained by the system [24].

Wu proposed a personal intelligent assistance that is specifically designed to help users in Collaborative Design Environment (CDE) [25]. CDE consists of virtual workspaces which provide interaction among users, resources, information, and tools that are necessary to perform specific tasks or operations. This collaborative environment enables users to seek assistance from other users, gather information that is needed to achieve their goals, and provide knowledge or services for other users. However, due to the integrations of platforms and complexity in human collaboration, Wu proposed a personal assistant agent for each user in this collaboration environment. This personal assistant reduces the workload required by the user, as well as, guides the user to ensure that he/she is following the correct steps to reach his/her goal [25]. As shown in Figure 2-2, the architecture of the Intelligent Assistant, proposed in [25], contains a user model, other users' models, application model, inference and knowledge components as well as application and collaboration assistance components.

The main focus of Wu's work is on building a cognitive state-based user model that can capture the user's interests and behaviour, and update them automatically. The context of the user in this model is captured in terms of his/her goal, interests and behaviours [25].

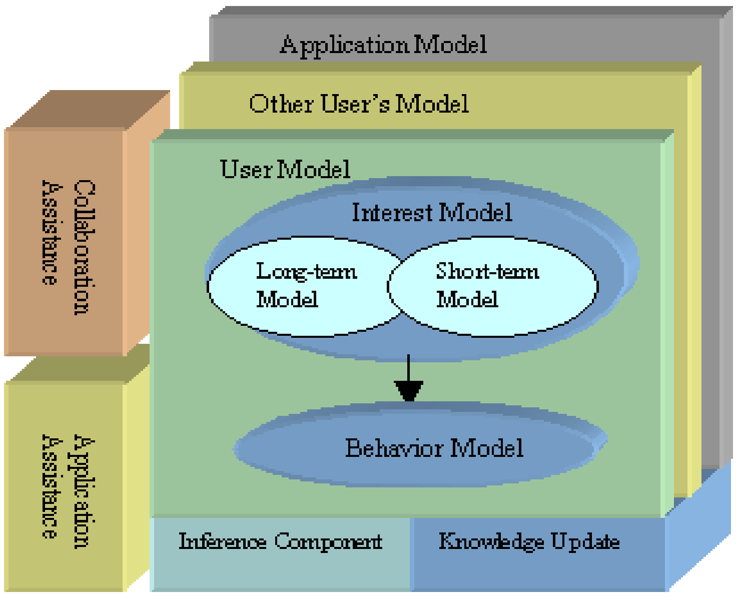


Figure 2-2: Intelligent Assistance Architecture [25]

Formally, a user model (UM) is a triple (G, UM_I, UM_B). Where G is a finite set of user's goals, UM_I is the user's interest and UM_B is the user's behaviour. This model assumes that the user's goals are explicitly specified, moreover, actions are viewed as the transition between different states, to achieve the goal [25].

The user interest model UM_I represents the portion of the domain context that is related to the user's goals. This interest model contains short-term and long-term interest models based on the user's goals [25]. The short-term interest model represents the portion of user's interests that is related to the user's current goals. This approach is dynamic in dealing with the user short-term interests, and provides relative stability in dealing with the user long-term interests [25].

Wu identified the user behaviour model UM_B as the sequence of actions a user will perform to achieve a goal within the context of the user's interest. UM_B determines the most likely path of actions the user will follow, which enables the personal assistant to provide the appropriate assistance proactively [25].

The inference component in Wu's proposal is designed to infer the user's intention so it can determine what type of assistance the user might need regarding his/her goal. Although the goal is explicitly determined, the inference component decomposes the goal into hierarchical structured sub-goals attached to specific applications. These sub goals are showed by conceptual graphs. Conceptual graphs are particularly suitable for formal representation of knowledge with the advantages of being logically precise and computationally tractable [26]. The user short-term interest model is inferred by the width-first search method. After identifying the goal, it searches the nodes that related to the interested concepts expressed in that goal and calculates the utilities along the search path. This ensures that the selected nodes are highly interested for the user [25]. Knowledge update for user short-term interest model happens only in the utilities of the nodes, simply, the utility increases when the user selects the concept contained in the node from the user interface [25].

The inference component of the user behaviour model is based on inferring the Bayesian network. The probability of each action is computed in order to find the branch with the maximum utility that represents the agent's belief of the user's behaviour preferences. Wu uses the Lauritzen-Spiegelhalter (LS) exact inference algorithm; it is part of the clustering algorithms mentioned in Section 2.4.4.

In [27], Zhang uses the user model proposed by Wu in [25], and she elaborated on several intelligent assistances that can be inferred using the proposed user model. The application assistance component in the model is used to help users in their interaction with the application. In [27] Zhang identified different forms of application assistances from Wu's proposed model:

- Assistance required by the User: Based on the user behaviour model, the personal assistant (PA) knows the path that achieve the goal, and therefore, it is able to provide assistance when the user does not know what to do next to achieve the goal. For example, through the user behaviour model, the PA identifies the current goal g1 and the actions to achieve the goal are {a1, a2, a3, a4}. Assume that the user is done action 3 and does not know which action should come next to achieve the goal, then he will ask the PA for assistance. The PA will identify the current action performed by the user (through monitoring the user's interactions) then suggest a4 to be performed next to achieve the goal.
- Proactive Assistance: When the PA identifies the user's current goal, on the basis of the knowledge in the user's behaviour model, it has the capability of predicting the user's next action, thus help to reduce the user's workload. Through the user's interaction, the PA will capture the user's actions and through the user behaviour model, it will be able to predict the user's next action, and provide proactive help accordingly.
- Monitoring Assistance: The PA will infer if the user acts on the right path towards achieving the goal. When the user's action is not on this path, the PA

would prompt the user with the right actions along the path to prevent the user from wrong actions that will harm or delay his work.

Another type of assistance that was discussed in [25] is the collaborative assistance. Collaborative assistance is used to support and enable user collaboration in the environment [25]. In [28], Kadri elaborated more on this collaboration assistance, he illustrated that in a collaborative environment, the personal assistant helps the user obtain his/her collaborator and share information with these collaborators. In order for the personal assistant (PA) to find the collaborators that could help the user in his/her tasks, the PA must find other collaborators with common domain concepts and have the same or a close degree of interest among these shared concepts. The similarity between two users is computed based on the Pearson Correlation equation [28]. Once the PA is informed with potential collaborations, the utilities from those neighbors are combined to compute a prediction about information to be shared [28].

The references [27] and [28] used the user modeling presented by Wu [25], along with the different types of assistances to develop a collaborative remote personal assistance framework for mobile users. In this framework, shown in Figure 2-3, engineers will be able to interact with each other in order to maintain equipment in manufacturing environments [28]. Field engineers are located at the factory where the equipment is located; they are represented by personal assistant agents. In-house engineers are a group of experts who could help in case field engineers failed to fix the problem, each in-house engineer is represented by a personal assistant. The diagnostic agent is connected to the equipment and monitors it from faults. If there is a fault, then the diagnostic agent sends request to the broker agent for a solution. The broker agent initiates a bidding process through communicating with the field engineer's personal assistant agents to locate capable field engineer to fix the problem. If the field engineer is unable to fix the problem, he/she will require further assistance from in-house engineers through the collaborative agent which collaborate between the personal assistances of the field engineers and the smart board assistance agents for the in-house engineers based on the collaborative assistance inference from the user model as mentioned above, in [27], smart board agents are not used, rather in-house engineers are represented by personal assistants just like field engineers. The following diagram shows the general framework of the manufacturing environment [28]:

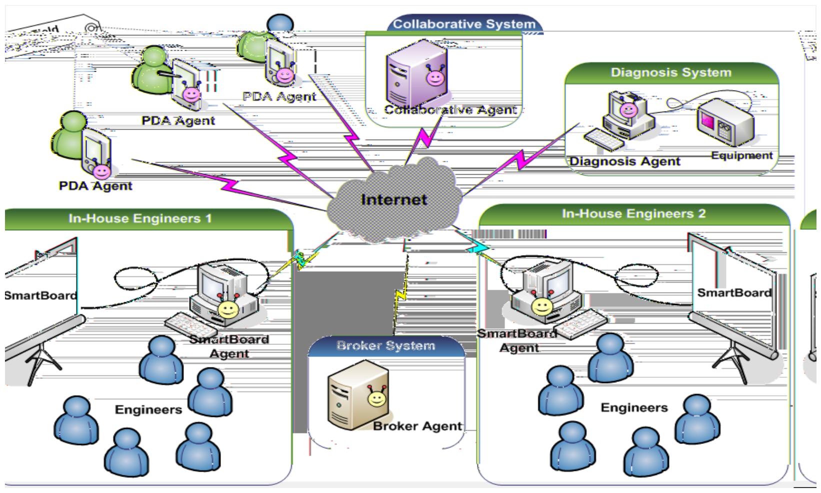


Figure 2-3: General framework of the proposed manufacturing environment [28]

The work discussed above, proposed in [25], [27] and [28] and summarized in [29], built an architecture based on agent technology to provide personal assistance to users in collaborative environments. It is the previous work of our research group in the field of personal intelligent assistance and it is referred to again in the upcoming chapters. The user model proposed in [25], acts as the foundation of the work proposed in this thesis.

While in the state-based modeling the focus is on the states as discussed above, the event-based model focuses on the set of possible events that occur in the system, and the behaviour is modeled through the system's response to these events [30], [31]. Event-based behaviour models are reactive, which means that the user's behaviour is a response to the events that occur in the environment. In other words, the events that occur in the environment derive the user's actions.

A tourism application is proposed in [34] that provides tourism information to the user on his/her PDA, or mobile device, based on matching the user's interests, location, and their travel history, which is a history of past events [32]. An Event Notification System (ENS) is used to store all the possible events that occur in the system. Events can be either location events, which are events that are connected to a specific user, time and location, or external events which are caused by external sources that send event messages to the system [32]. The combinations of the location events which identify the history of events that the user caused, as well as the external events derive the next information and/or recommendations that are going to be provided to the user in his/her tour.

In [33] an elderly health monitoring system is proposed based on the event-based modeling. Two categories of events are considered in this proposal; body-events and environment-events. The elder actions and the context of the environment trigger these events. For example, if the "lie down" event is triggered when the user performs the actions "standing" then "lying", while the event "fall down" is triggered with the user performs the actions "walking" then "lying" [33]. Modeling these events helps the caregiver in reasoning about the elderly's activities and hence can make better assessment about the elderly's health [33].

The combination of deliberate (state-based) and reactive (event-based) behaviour modeling is proposed in some software testing frameworks [31]. The software project's behaviour is modeled in terms of the different states which the software can move into as well as the response of this software to different external events and the testing progress validates the software based on both behaviour paradigms.

2.6 Personalized Advertisements

As discussed in Chapter 1, personalized advertisements is an essential approach to attract customers as it targets the products and services to users based on their preferences. In [34], a system is proposed that recommends advertisements to users based on their preferences. Through this system, a unifying user profile is proposed so that users can receive personalized advertisements on three platforms: on iDTV (interactive digital television), mobile phones, and internet. The user profile is created in two steps; first, the first time users have to provide some personal information explicitly (age, gender, region, etc) to determine basic favorite category of advertisements for the user. Second, all of the user's interactions, which measure his/her satisfaction with the advertisement, are logged (explicit requests for advertisements, bookmarking/forwarding advertisements, and requiring more information about the advertisement, etc.). In order to provide a system which recommends the desired advertisements to the users, the personalized content-based component is combined with a community content-based component, to use community preferences in the recommendation, as well as rule-based component which removes categories that are not related to user's profile (intended to different age groups, etc.). The content of the advertisement is represented by meta-data which contain simple information about the advertisement such as category it belongs to, and targeted audience, etc [34].

There are two major limitations for this work; first, this proposed solution depends on the user's interactions with the advertisements to capture the user's preferences and interests. However, recommending advertisements based on the user's goals and behaviours would be much more convenient, as the users usually do not look for advertisements; but, if these advertisements are proactively available to help the user achieves his/her goal, then they are much more appreciated. Also, a richer content of the advertisement is needed to recommend it to the user. Meta data about advertisements are not always available and they usually do not include much information.

A dynamic user modeling approach based on user’s long-term and short-term interests is proposed in [35]. This model is applied to the LUNA project, which is a project that provides free wi-fi access by a wireless communication company, Futur3, to its customers [35], in order to fund this free wireless communication, Futur3 uses the LUNA project to direct advertisements to customers based on their interests. Long-term interests capture the interests of the user in specific domain categories. The long-term interests are captured by directly asking the user to pick the categories that he/she is interested in the most when he access the network for the first time. While short-term interests are capturing the temporal interests during the current connection session by obtaining the feedback of the user about the several documents or webpages that he/she is currently accessing. Furthermore, the user's current location is captured to provide more accurate modeling for the user [35]. After capturing the user’s interests, these interests will be matched with advertisements, through their meta-data and a cosine similarity formula from the vector space model is used to compute similarity between advertisements and the categories that interest the user.

Long-term and short-term interest models are captured manually by getting direct information from the user about his interests [35]. Only the location of the user is determined automatically. The user gets to choose categories of a specific domain that he is interested in from a specific pool of categories, this pool contains limited number of categories that might not necessary reflect the user’s real interests. On another hand, each advertisement is pre-assigned for specific category manually, the meta-data of the advertisement, which contains the category that it is related to is created manually for each advertisement [35].

Argo is a system proposed in [36] that suggests advertisements to users based on their interests. The intuition is made that the users’ shared photos gives a good idea about their interests [36]. This work uses the collection of shared photos in flicker to gather the user’s interests. The proposed approach automatically annotates images using data-driven annotation approaches, and then a hierarchical topic space model is used to categorize these annotations and learn the user's interest [36]. Afterward, vector space modeling is used to match the contents of the advertisements with the user interests that are collected by the annotated images. Finally, a ranking model is applied to rank ads by their relevancy to the user's interest, and top-ranked ads will be suggested to the user as seen in Figure 2-4 [36].

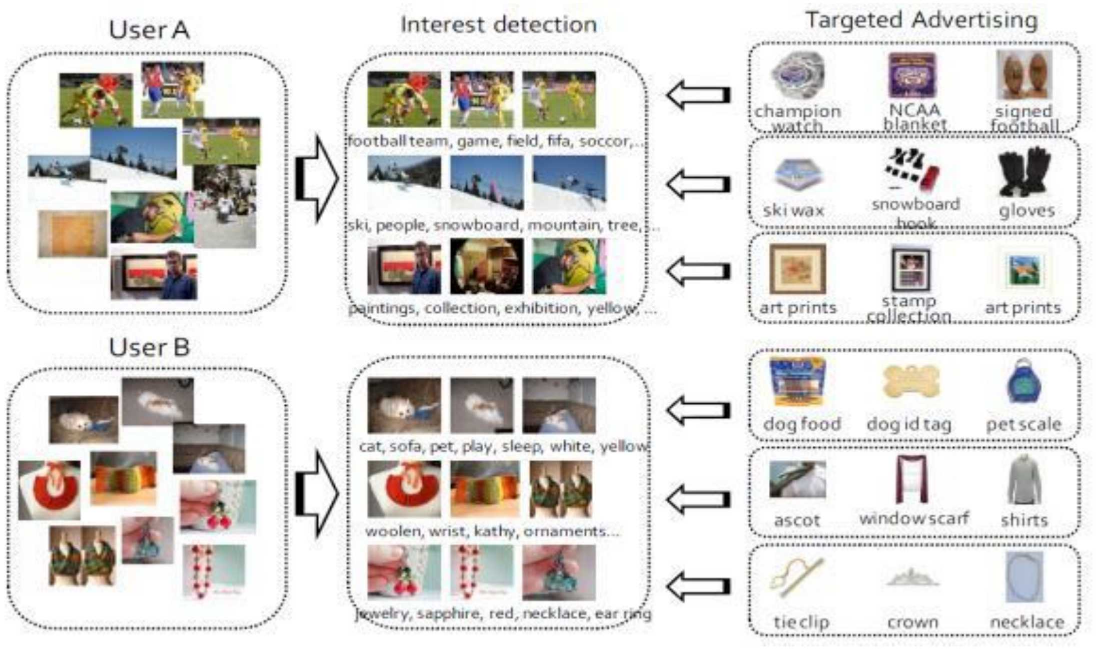

User A
Interest detection
Targeted Advertising
football team, game, field, fifa, soccor,...
ski, people, snowboard, mountain, tree,...
paintings, collection, exhibition, yellow, ...
champion watch
NCAA blanket
signed football
ski wax
snowboard hook
gloves
art prints
stamp collection
art prints
User B
cat, sofa, pet, play, sleep, white, yellow
woolen, wrist, kathy, ornaments...
jewelry, sapphire, red, necklace, ear ring
dog food
dog id tag
pet scale
ascot
window scarf
shirts
tie clip
crown
necklace

Figure 2-4: The Argo System [36]

2.7 Smart Multimedia

An important aspect of personalized advertisements is the actual advertisement. How is the advertisement presented? And how are the contents of this advertisement identified? The work presented by Moursi in [37], transfers the multimedia from a passive raw file into a smart entity that has computational and decision making capabilities. Therefore, the smart multimedia can recommend itself to the user based on the decision making capabilities that it performs after matching its itemProfile, which contains information about the media contents, with the user's preferences. This approach replaces the conventional recommendation system in which a centralized server stores the multimedia files and recommends them to users based on their contents and the user's preferences [37].

The proposed smart multimedia extends the conventional representation of multimedia as being a passive file, to include knowledge, computational capabilities, and behaviours, converting the multimedia into an active smart computational entity that can change its internal state as well as the state of the surrounding environment. It consists of four different layers as shown in Figure 2-5 [37].

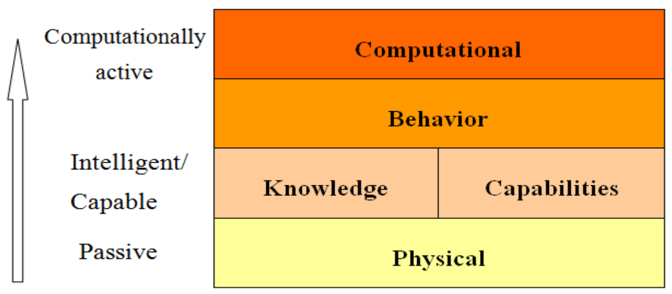


Figure 2-5: The four main layers of the Smart Multimedia Model [37]

Physical Layer: represents the actual multimedia file. It focuses on representing and encoding the physical attributes of the multimedia file such as sound, text, colors, etc.

Knowledge and Capabilities Layer: knowledge represents the intelligence of the multimedia, which is the ability of the media to reason about its contents' semantics; this knowledge component is built based on the model for intelligent multimedia proposed by Elsakhawy [38]. The capability structure enables the multimedia to have a set of actions which can change its internal and external state.

Behavior Layer: The behavior layer consists of the set of application-specific behaviours for the multimedia. Each behavior describes the sequence of actions to be performed by the multimedia computational entity in order to achieve the application requirements. It allows modeling of the different application-specific behaviors of the multimedia.

Computational Layer: This layer includes the computational engine which allows the multimedia to be computationally active. This computational engine is considered as the vehicle that performs the computation by driving and applying the sequence of actions of the multimedia behaviours. The proposed structure in Figure 2-5 [37], transfer the multimedia from passive element to a smart multimedia. Smart multimedia is "A multimedia computational entity which encapsulates knowledge about the multimedia internal content semantics (making it intelligent), in addition to computational capabilities, and behaviors allowing it to change its internal state as well as the state of the surrounding environment [37].”

Furthermore, the work in [37] proposed a recommender system using the smart multimedia. It is a decentralized recommender system in which the smart multimedia can recommend itself to users based on matching its contents which are modeled in itemProfile to the user’s preferences, which are modeled in userProfile. In order to match the itemProfile of the multimedia with the user preferences, the multimedia has to be able to describe its internal content semantics, and therefore, it has to be intelligent. A model for intelligent multimedia was proposed by Elsakhawy [38].

In [38], the proposed model is able to semantically model the multimedia content and match multimedia objects to their semantically equivalent domain concepts, thus enabling multimedia semantic level intelligence. Based on the work proposed in [38], the itemProfile is a model for the multimedia that provides the required intelligence by capturing a set of properties and relationships that describes all the concepts represented in the multimedia for a specific domain [37]. For example, the product sold in the Quick Service Restaurants (QSR) domain could be represented as a concept. A concept could have multiple instances such as the product sold by QSR could be a donut, hot chocolate, etc. Properties of this concept are represented by sweetness and flavor. Each of these properties have specific values; sweetness could be represented by high, medium or low, while flavor could be represented by chocolate or strawberry for instance. The userProfile, in [37], represents the user’s preferences for a certain concept in the domain by modeling all the properties describing this concept. The userProfile is initialized based on a specific stereotype, as the user interacts with the environment; the user’s preferences will be updated. Then the userProfile is matched with the itemProfile using preferences-based matching and business rules reasoning [37]. The smart multimedia model, shown in Figure 2-5 [37], can be utilized it in the proposed recommender system as follows:

- The knowledge component will include the itemProfile extracted from the intelligent multimedia model proposed in [38].

- The capabilities component will include the preference-based capability and the business rules reasoning capabilities.
- Behaviour layer will include recommendation behaviour which describes the behaviour of the smart multimedia computational entity. The recommendation behaviour of the multimedia can be described as follows:
- Multimedia receives a bid from the user agent which represent the user
    a. Multimedia performs preference-based matching.
    b. Multimedia refines its rating by reasoning about business rules.
    c. If the computed rating is above a certain threshold, multimedia sends the bid to the user.
    d. If the bid is accepted, the multimedia is presented on the user's device.

The work proposed in [37] is intriguing as it converts the passive multimedia into smart multimedia computation entity that can change its internal state as well as the state of the surrounding environment. Furthermore, this multimedia computation entity has been utilized to be able to recommend itself to users based on matching its contents to the user preferences [37]. However, users are not modeled based on their behaviours or interests; rather, they are stereotyped based on their age and preferences. The combined work of Moursi [37] and Elsakhawy [38] is summarized in [39].

Chapter 3

3 Problem Statement and Analysis

This chapter includes the problem statement, objectives and motivation of this research work. Also, the research issues of the stated problem are listed and analyzed. Furthermore, the hypotheses that the work is based on, is also given in this chapter.

3.1 Problem Statement and Objectives

In order to recommend assistances that are beneficial for a certain user to achieve his/her goals; the user's interests should be modeled. Furthermore, his/her current behaviour, which represents his/her current status, should also be modeled. When applied to the domain of advertisements, the recommended assistances are in the form of advertising messages. These advertising messages raise the user’s awareness about the various choices that are available to him/her, and can therefore, influence the user’s decisions regarding these available choices. For these advertising messages to be helpful for the user to achieve his/her goal, they need to be relevant to the user’s behaviours and interests.

The work here is aimed at:

1) Proposing a generic personal smart assistant that is able to model the user’s goals, interests and behaviours in an open environment based on the work proposed in [29], and be able to proactively assist the user in achieving his/her goal.

2) Enabling the personal assistant to match the user model with the contents of the advertising messages that are represented as smart multimedia agents, based on the work proposed in [39].

3.2 Motivation

The expansion of the cyberspace and the enormous process in computing and software applications enabled technology to cover every aspect of our life. Unfortunately, this caused people to be overwhelmed with technology, and they find difficulties in dealing with these different applications. Nevertheless, having a lot of aspects of people's life driven by technology also means that the personal smart assistant has an opportunity to assist users in different aspects of their lives. The personal assistant can utilize technology on behalf of the user to assist him/her in accomplishing his/her desired goal.

Research in personal assistant has been around for years; researchers always try to improve the personal assistant because it is being used in more domains each day. Whether it is in the medical field, the manufacturing field, the digital advertising field, or just to help users in their daily office work; personal assistant needs to be aware of the user's goals, interests and behaviours and provide the necessary assistant accordingly. Therefore, it is necessary to design a general personal assistant framework that could be applied to any domain.

The outstanding success of Siri as a personal assistant, discussed in Chapter 2, provides a great motivation in terms of building a personal assistant that can reduce the workload of the user's daily activities. The approach by which siri was developed; mainly the personal-context awareness component provides a great motivation to build a personal assistant which can similarly provide help to the user by modeling the user's activities and preferences.

The widespread of digital signage, online advertisements, digital TV and mobile applications has enabled advertisers to deliver their advertisements and messages in a more interactive way to the public all the time. However, although advertisements are all around, people tend to ignore them and avoid them all together if possible through the means of pop-up blockers, video recorders (in which they can control what to watch and what to skip), etc. Therefore, the challenge for advertisers now is to convince their clients that these advertisements worth viewing and should not be ignored. One possible way is to personalize these advertisements and therefore, advertise to users only what interests them.

As a prove of concept for the proposed personal assistant, a prototype is built to recommend advertising messages to users based on their behaviours and interests. The contents of these advertising messages are represented by smart multimedia agents, proposed in [39]. The content of each multimedia is matched with the user's interests and behaviours which are modeled in the personal assistant. In this way, users will see advertisements that interest them, and therefore, interact with them positively.

3.3 Problem Analysis

A personal assistant needs to model the user's interests and behaviours in order to recommend the "right" assistance at the "right" time to help the user achieve his/her goals.

A main objective is to model the user and provide a proactive assistant according to this modeling. As discussed in Chapter 2, for a personal assistant to model the user, it has to capture the user's contexts, in particular his/her goals, interests and behaviours.

In order to model the user, the nature of the environment which the user is working in should be determined. Considering open environments is essential in building a generic personal smart assistant that can be used to assist users in different domains. Open environment is an environment at which entities can enter and exit any time [40]; here entities are assumed to refer to humans, software applications or features, resources, etc. Therefore, when the user is modeled, changes in the environment from other entities should be considered, and the user's responses to these changes should also be modeled.

To provide a proactive assistance, the personal smart assistant needs to predict the user's next action and try to help him/her to achieve that action. As discussed in Chapter 2, the user's behaviours are uncertain, and therefore, only a degree of belief that a specific action is going to be performed next, can be captured. This degree of belief can be built through capturing the user’s behaviours and the context of the environment.

Another objective of this work is to recommend and advertise the smart multimedia based on its contents that are matched with the user model. To achieve this objective, the content of the multimedia should be realized, this can be done using the smart multimedia which contains an intelligent component which can represent the semantics of the multimedia file [39]. The personal assistant should provide a matching mechanism that identifies the closest match between the user’s goals, interests and behaviours in one side, and the contents of this smart multimedia, which contains the advertising message in the other side.

Based on the user model, the personal smart assistant should extract an output that can be matched with the contents of the smart multimedia. In other words, the results of modeling the user’s goals, interests and behaviours should be interpreted in terms of products or services that can be matched with the contents of the advertising message, presented by the smart multimedia.

3.4 Research Issues and Challenges

A lot of work has already been done in the field of personal assistant and user modeling. However, there are still issues and challenges that are not fully addressed in the field. This section lists and discusses the open research issues that are relevant to the research problem stated and analyzed in the previous sections of this chapter.

3.4.1 Modeling the User in Open Environments

As discussed in the analysis above, the personal smart assistant needs to model the user in an open environment. The user’s responses to changes in the environment should be modeled and captured as part of his behaviours. In Chapter 2, state-based modeling and event-based modeling were discussed as the two main user behaviour modeling approaches. The state-based is the deliberative approach as oppose to the event-based which is the reactive approach. The deliberative approach is essential as it is a goal-driven approach that sets a plan of actions to achieve the desired goal by changing the states of the world. However, in open environment, unexpected events can occur which forces the user to responds to, and therefore might alter his/her goal-driven plan. Therefore, reactive, event-driven responses should also be modeled as part of the user's behaviour. This will ensure more accuracy in modeling actual user behaviour in an open environment. As a result, the provided assistance from the personal smart assistant will be more relevant, and can help the user achieve his/her goal efficiently, even if the environment is constantly changing. Furthermore, the user's interests can also be altered based on some events that occur in the environment, hence, modeling reactive behaviours can also reflect a more accurate modeling of the user's interests.

Hence, combining the deliberative and reactive approaches in user behaviour modeling is essential to adequately model the user behaviours in an open dynamic environment. To the best of our knowledge, this combination in user behaviour modeling does not exist in the literature. The proposal of this thesis, given in Chapter 4, delivers a model that can provide this combination of deliberative and reactive user behaviour modeling.

3.4.2 Correlating the User's Behaviours and Interests

Modeling the user's behaviours helps in understanding how the user is acting to achieve the goal, while modeling the user's interests defines the concepts that the user is interested in and their relations within the environment. Most of the existing work in user modeling focus on explicitly modeling either the user interests or behaviours and ignore modeling the other one. Few efforts were given to model the user behaviours and interests together, however, in these efforts it is not clear how these models reflect each other, and how observing the user's behaviours for example, would reflect the user's interests. Most of the reviewed literature captured the user's interests through explicit input from the user rather than a reflection of his/her behaviour.

The challenge is to provide explicit, yet correlated, models for the user's interests and behaviours. The user does not need to explicitly indicate his/her interests, or consider the interests to be initially the entire domain's concepts and relations [25]; rather, the interests should be a reflection of his/her behaviours.

3.4.3 Capturing the Context of the Environment

The existing research in the field of personalized advertisements concentrates on providing digital advertisements to users based on their behaviours in the cyberspace. However, with the advancement of the smart environments, personalized digital advertisements need to be provided based on the user's interactions in these smart environments as well as the cyberspace.

As discussed in Chapter 2, personal assistants have already been proposed to utilize smart environment's sensors in providing relevant assistance to the user's in smart homes or smart meeting rooms. The challenge is to propose a personal assistant that utilizes the smart environments to provide personalized assistances to the user. In other words, we need to sense and interpret the user's physical actions, which he/she performs in these smart environments, and provide an assistance that is relevant to these actions and behaviours.

3.5 Hypotheses

Based on the problem definition, problem analysis and the research issues that are identified for this problem, three hypotheses are made that will have to be verified in the proceeding chapters.

First, modeling the user's interests and behaviours, will ensure more relevant recommended assistance, and therefore, a more positive interaction from the user.

Second, recommending a relevant assistance will help the user to achieve his/her goals more efficiently. In other words, these recommended assistances reduce the number of actions required to achieve the goal.

Third, observing the user’s reactive as well as deliberate behaviours in open environments will ensure that the Personal Assistant is able to provide recommended assistances to help the user deal with sudden changes in the environment more efficiently, again efficiency here is measured by the required number of actions.

Chapter 4

4 Proposed Solution

As discussed in previous chapters, the personal smart assistant is able to provide a relevant assistance to the user based on his/her goals, behaviours and interests. In this chapter, a generic personal smart assistant architecture is proposed. The proposal is emphasized on the user model, which is a major component of this architecture.

4.1 Personal Smart Assistant Architecture

The proposed architecture of the personal smart assistant is given in Figure 4-1. This architecture is an expansion of the previous work proposed within our research group in [25], [27] and [28], hence it is similar to the architecture given in Figure 2-2 [25]. As seen in Figure 4-1, the proposed architecture integrates several components that are needed to provide a relevant assistance to the user. The main component of the personal assistant is the user model. In this architecture, the user model contains user behaviour model, which is the centre focus of this proposed work, and the user interest model. The user behaviour model extends the proposed model in [25] to model the user's deliberative and reactive behaviours as will be discussed in the following sections of this chapter. The user's interests in this architecture are derived from his/her behaviours. The inference component infers the user's needs based on the given goal and the user model. The environment assistance component provides the assistance that matches the needs of the user, identified through the inference component, with the most relevant assistance available in the environment. The environment model component is responsible for modeling the context of the environment. In addition, the knowledge update component is responsible for updating the knowledge captured in the user's interest and behaviour models. The collaboration assistance utilizes the other users' models, which reflects this personal assistant's belief about other users in the environment. The collaboration assistance component is not covered in this proposal. However, in [28] the collaboration component is proposed and can be adjusted and integrated in this proposed architecture.

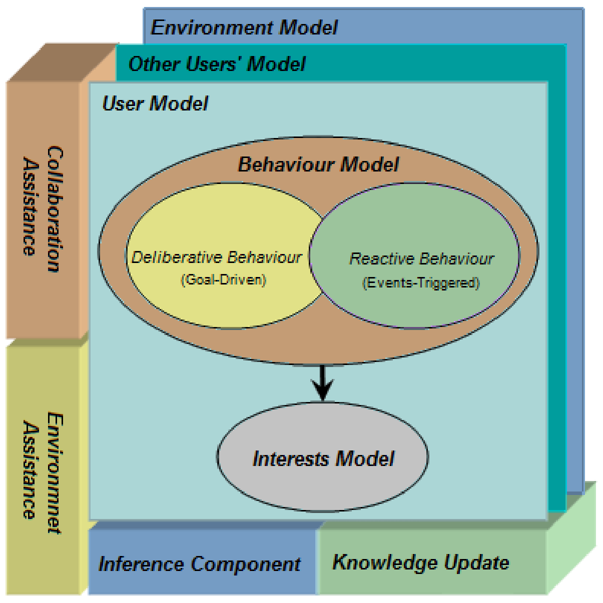


Figure 4-1: The Personal Smart Assistant Architecture

4.2 The User Model

As discussed previously, the user model is an essential component of the Personal Smart Assistant (PSA) as it models the user and captures his/her interests, behaviours and goals which impact the assistance that the PSA provides.

The user model (UM) consists of user's Goal (G), the User Behaviour model (UB), and the User Interest model (UI), The User Model can be expressed as a triple:

$$UM = <G, UB, UI>$$

4.2.1 Domain Representation

Before modeling the user, the domain at which the user is modeled should be formally represented. This work assumes that the domain is modeled using extensional conceptualization [41]. The entire collection of everything that exists is the **world W**, it is a set of the possible worlds, $W = \{w_1, w_2, \ldots, w_n\}$. A **state s** can be defined as an instantaneous observation (snapshot) of the world, several states can exist in one possible world w_i , $w_i = \{s_1,\ldots,s_n\}$. An **Event E** is defined as the occurrence of a change that triggers a response in the world. The things that exist in the world are called **Concepts C,** C= $\{c_1,\ldots,c_n\}$, where c_i is a single concept. The relations between these concepts are called **conceptual relations R**, R= $\{r_1,\ldots, r_n\}$, where r_i is a single conceptual relation**.** Therefore a state S consists of concepts C, and their Relations R.

$$\text{State} = <\text{Concepts, Relations} >$$

4.2.2 The User's Goal

The first component of the user's model is the user's goal. We need to model the user's goal in the domain represented in Section 4.2.1. A goal G is the desired end, which, if achievable, is represented as one of the possible states of the world W. This work assumes that the goal is given explicitly by the user.

A goal is reached when desired relations are achieved between desired concepts in a state of the world. Hence, a state is considered a goal state, or the desired state s_d, when the desired concepts and relations (C_dR_d) are a subset of that state. However, if s_d is not one of the possible states of the world, then the user's goal is not achievable.

Formally:

$$C_dR_d \subseteq s_d$$

$$G = \{s_i \mid C_dR_d \subseteq s_i, s_i \in w_i, w_i \in W\} \text{ IMPLIES G is achievable, } s_i = s_d$$

$$G = \{s_i \mid C_dR_d \subseteq s_i, s_i \notin w_i, w_i \in W\} \text{ IMPLIES G is not achievable, } s_d \notin w_i$$

4.2.3 User Behaviour Modeling

An assumption is given that the world stays at one state until an action occurs. Therefore, if the current state of the world is not the desired state, then the user needs to perform some actions to move the world to the desired state and achieve his/her goal. Hence, state-based modeling can be used to model the user behaviour as shown in Figure 4-2. Each node in Figure 4-2 represents a state of the world, the first node is the current state of the world S_c, and the last node represents the desired state S_d at which the user's goal is achieved. The arrow represents a transition between the states of the world; a_i is an action associated with this transition.

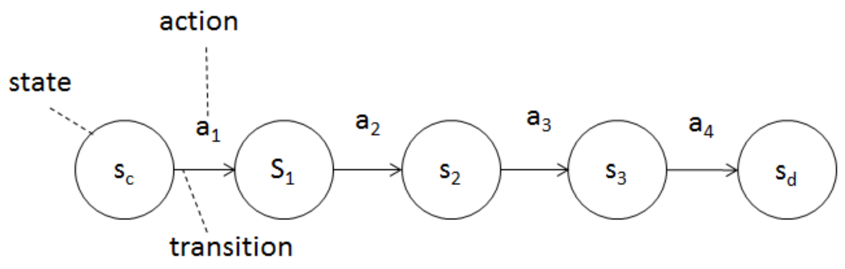


Figure 4-2: State-based User Behaviour

4.2.3.1 Extending the State-Based Modeling with Events

The user needs to perform actions to transform the world into the desired state in order to achieve his/her goal. State-based modeling models only the transition of the states of the world. In [25], Wu proposed state-based user behaviour and assumed that modeling the transition of the states is sufficient to model the user behaviour. However, we argue that in order to model the user behaviour, we also need to explicitly model the changes in the environment, and how the user is reacting to these changes, rather than just the transitions of the states. This is essential in an open environment where entities can enter and exit and any of them can change the states of the world any time. State-based modeling is a goal-driven deliberative approach which needs to be extended to include event-driven reactive approach. In this work, because we consider that the personal assistant is supposed to assist its user in open environment, we propose to extend the state-based modeling through modeling the events that happen in the environment explicitly as the states of the world are changing. Therefore, this work combines the deliberative approach and the reactive approach to model the user's behaviour.

An event is defined as a change in the environment that triggers a response, or "an external observable phenomenon, such as an environmental or a user stimulus, or a system response punctuating different stages of the system activity [31]”. Hence, modeling the events explicitly will enable us to understand the behaviour of the user, as he/she acts not only to achieve the goal, but also, to react to these changes that might require the user’s responses.

4.2.3.2 Modeling the Actions

The user’s behaviour is the sequence of actions that the user performs. Hence, in order to model the behaviour of the user with states and events, we need to model the actions.

An action can be represented by i) an identification, ii) preconditions that should be satisfied for the action to take place, and iii) post conditions that are the impact of the action. As example is shown in Figure 4-3, the action is an operation that causes the transformation of the world from one state, which is the preconditions of the action, into a different state, which is the post conditions of the action, the states of the world are assumed to remain stationary unless an action occurs.

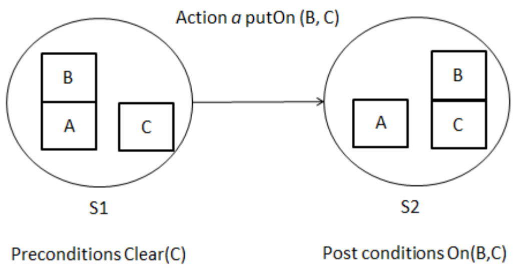


Figure 4-3: Modeling an Action

Formally,

Action A = <ID, Pre-Conditions, Post-Conditions>

4.2.3.2.1 Modeling Pre-Conditions and Post-Conditions

In this proposal, the user behaviour is modeled by events as well as states. Hence, for an action to occur, its preconditions are not only states, rather, events could also be required before an action can take place. Therefore, in this work, preconditions are modeled by states and events.

Pre-Condition = <State, Event>

Similarly, Post conditions, which are the impact of an action, are not only modeled as states, rather, events could also be an impact of an action. Therefore, in this work, post conditions are modeled by states and events.

Post-Condition = <State, Event>

4.2.3.3 States and Events to Model the User Behaviour

The user behaviour is a sequence of actions that will enable the user to achieve his/her goal. Figure 4-4 shows how the user's action is modeled based on both states and events.

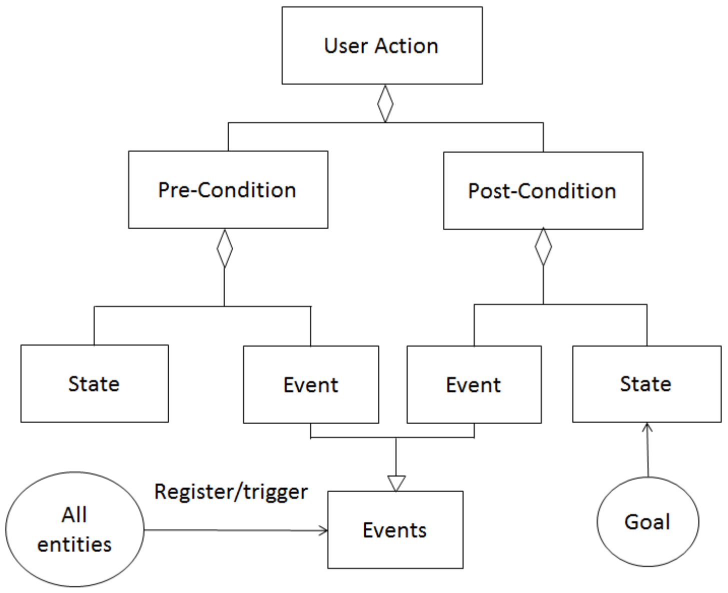


Figure 4-4: Proposed Model of Actions Based on States and Events

As shown in Figure 4-4, state-based model is extended with events to model the user's actions. Each of the user's actions are either derived by the user's goal, and therefore the action is to transform the states of the world to the desired state s_d, or triggered by events, and therefore, they are responses to the events that are occurring in the environment. As discussed before, each action has preconditions and post-conditions. In this model, these preconditions and post-conditions are not only states of the world, but also, events in the environment. So for an action to occur, specific states of the world should be reached as well as specific events should be triggered. Furthermore, if some events are triggered, which are preconditions of an action, these events causes an action to take be performed as a response to this event, and therefore, the states of the world need to change to satisfy the rest of the preconditions of the action. Therefore, actions could also occur because of events, even though these actions might not help the user to achieve his/her goal. On the other hand, post-conditions are the impact of the action on the environment. In this model this impact is not strict to the states of the world; rather, each action could trigger an event in the environment, in other words, the environment responses to these actions by producing these events. The significance of extending the state-based model with events is to further analyze the user's behaviour in environments which he/she might not be the only actor. We do not model the actions occurring by other entities but we model the impact of their actions through the events that occur in the environment. So events can be produced by all entities in the environment and this does derive the user's behaviours.

In state-based model, we have finite number of possible states of the world. For events we also have finite number of possible events that could occur in the environment, these are the events that are registered in the system either by the user or by other entities. Once an entity registers an event, the system captures all the registered events in the events' list. An example of the events' list is shown in Figure 4-5.

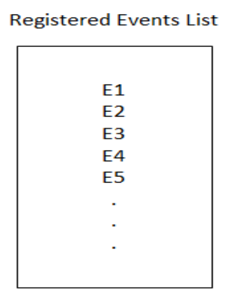


Figure 4-5: A list of events that are registered to the system

Modeling events alone does not represent the details of the system and how the possible states of the world are changing. Therefore, both states and events need to be combined to model the user's behaviour. Figure 4-6 shows the extension of the modeled events to the state-based modeling for the user behaviour.

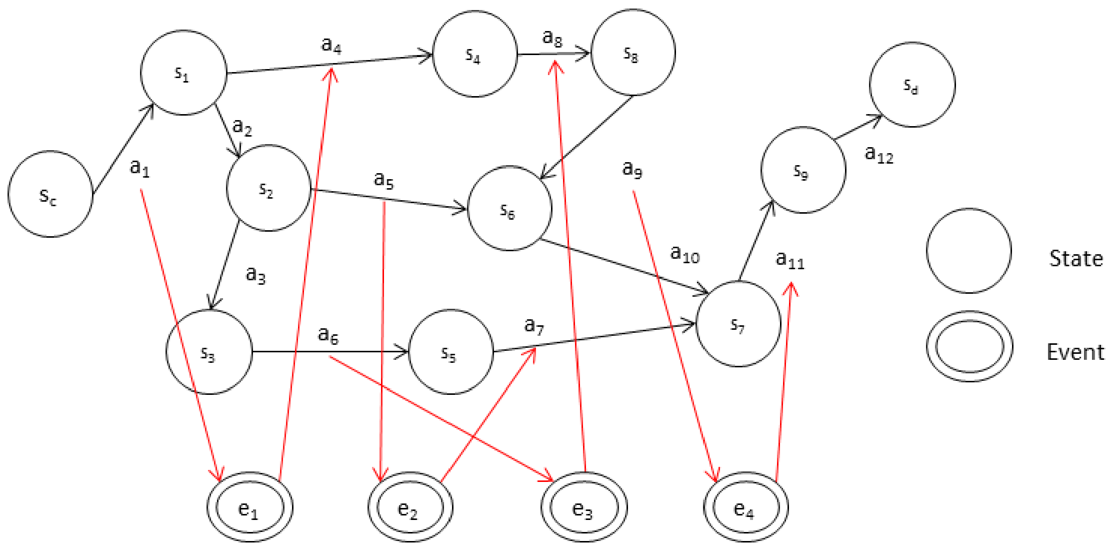


Figure 4-6: States and Events to represent sequence of user's actions

As shown in Figure 4-6, the triggered events in the environment derive the actions of the user as he/she is trying to move the states of the world from the current state of the environment till the desired state. The top part of Figure 4-6 is the state-based model that shows the sequence of states that are required to move from the current state of the world

s_c till the desired state s_d. However, as the actions are performed by the user, and because events are included in the preconditions or post conditions of these actions, then these events are triggered in the environment either as an impact of an action (i.e. in case of post condition), or as a stimulus for another actions to take place (i.e. in case of precondition) as shown in bottom of Figure 4-6. The red arrows in the diagram represent the actions' preconditions and post conditions in terms of events, i.e. an arrow from a_1 to e_1 suggest that; as the user performed a_1 to change the state of the world, e_1 occurs in the environment as a result of a_1, because e_1 is a post condition of a_1. When e_1 occurs, the user needs to perform a_4, because e_1 is triggering a_4 and its part of its preconditions, hence the arrow directed from e_1 to a_4, therefore, the occurrence of e_1 is deriving the user behaviour although the desired state could have been reached without performing a_4, furthermore, some events could be triggered because of others' actions, but they still influence the user actions as he/she is trying to reach to the desired state.

In conclusion, the sequence of the user's actions, that is either derived by the goal or triggered by the events, is the user's behaviour. Therefore, in this model, we are able to capture more realistic user behaviour by including actions that the user could perform in response to the environment rather than just the actions that he/she is performing to achieve the goal. All of these user actions are defined as the behaviour:

$$UB = \langle Actions \rangle$$

4.2.4 User Interest Modeling

In this work, the user's interests are derived by the user's behaviours, in other words, the user's behaviour is an indication of the parts of the world that the user is interested in. As discussed above, the user behaviour is the sequence of actions that the user performs in the environment. These actions include preconditions, which are the requirements for the action to take place. Therefore, to move towards the desired state, and to achieve the goal, the requirements for every action are part of the user's interests. Furthermore, to realize what these actions are achieving, the impacts from these actions are also interesting to the user, these impacts are represented as post-conditions, the post-conditions of the last action performed by the user include the goal, and hence the user interest of a user is the combination of preconditions and post-conditions of the user's performed and predicted actions.

Formally,

UI = <Preconditions, Post Conditions>

4.3 Capturing the User Model

After modeling the user in Section 4.2, this section illustrates capturing this proposed user model. Capturing the user's behaviour, capturing the user's interests and combining the behaviours and interests are all explained in this section.

4.3.1 Capturing the User's Behaviour

The user behaviour is modeled as the sequence of actions that the user performs in the environment. These actions are either driven by the user's goal or triggered by events that occur in the environment. These actions could be represented in a causal network because each action contains preconditions which can be satisfied by other actions; therefore, actions are causing other actions to take place.

Figure 4-7 gives an example of the causality of a_0 to several other actions that the user needs to perform in the environment. In this example, a_0 is triggered by the event E5. However, for a_0 to take place, all of its preconditions should be satisfied, hence they should be part of the current world. a_0 contains three states as part of its preconditions (s1, s2 and s3), these three states either exist in the current state of the world in which no more actions are required, or they are included as post conditions of some actions. Therefore, to achieve action a_0 we need to achieve these actions that satisfy s1, s2 and s3, in this example; these actions are a_1, a_2 and a_3. Furthermore, these new actions also contain preconditions that should be satisfied before these actions occur, and so on till all the required preconditions are part of the current world, and hence, no more actions are required to satisfy them. So after E5 is triggered, the user might be forced to perform several actions just as a response to E5, these actions are not related to the user's goal, however they are triggered by the events of the environment and require the user's reaction. All of these actions need to be captured and included in the possible sequences of actions that the user is performing to achieve his/her goal, even though they are not directly related to this goal. However, these actions need to be captured to reflect more accurate user behaviour, and thus, provide a more relevant assistance.

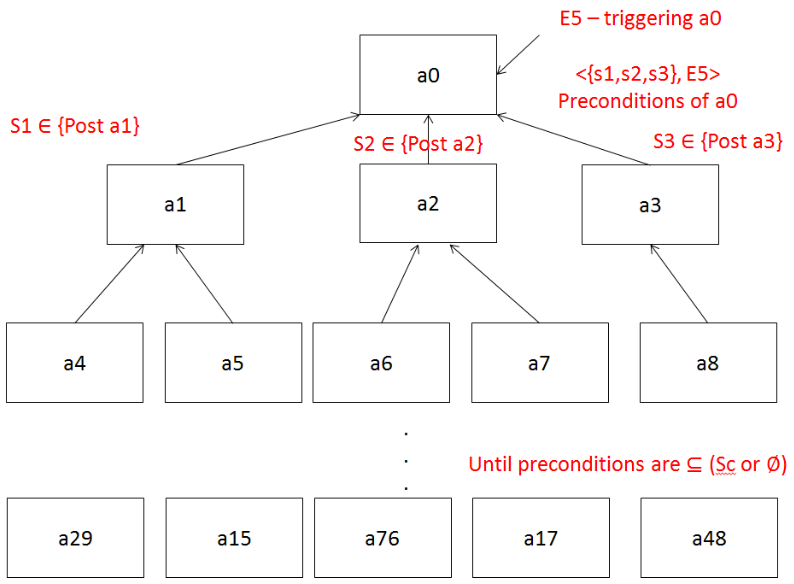


Figure 4-7: Causal Actions Network

It is necessary to capture the possible paths of actions that the user can follow and be able to predict the next likely action, and therefore, provide a proactive assistance that would help in achieving this action. In Figure 4-8, a flowchart is given that outlines a proposed algorithm, the path determination algorithm, that; given a specific action, will determine the different paths of actions that could be taken to perform this action, and measures the degree of belief that the actions' preconditions are satisfied. As stated above, for an action to be performed, we could need several other actions to ensure that this action's preconditions are satisfied. This algorithm considers all of the user's actions, whether they are derived by the goal or triggered by the events.

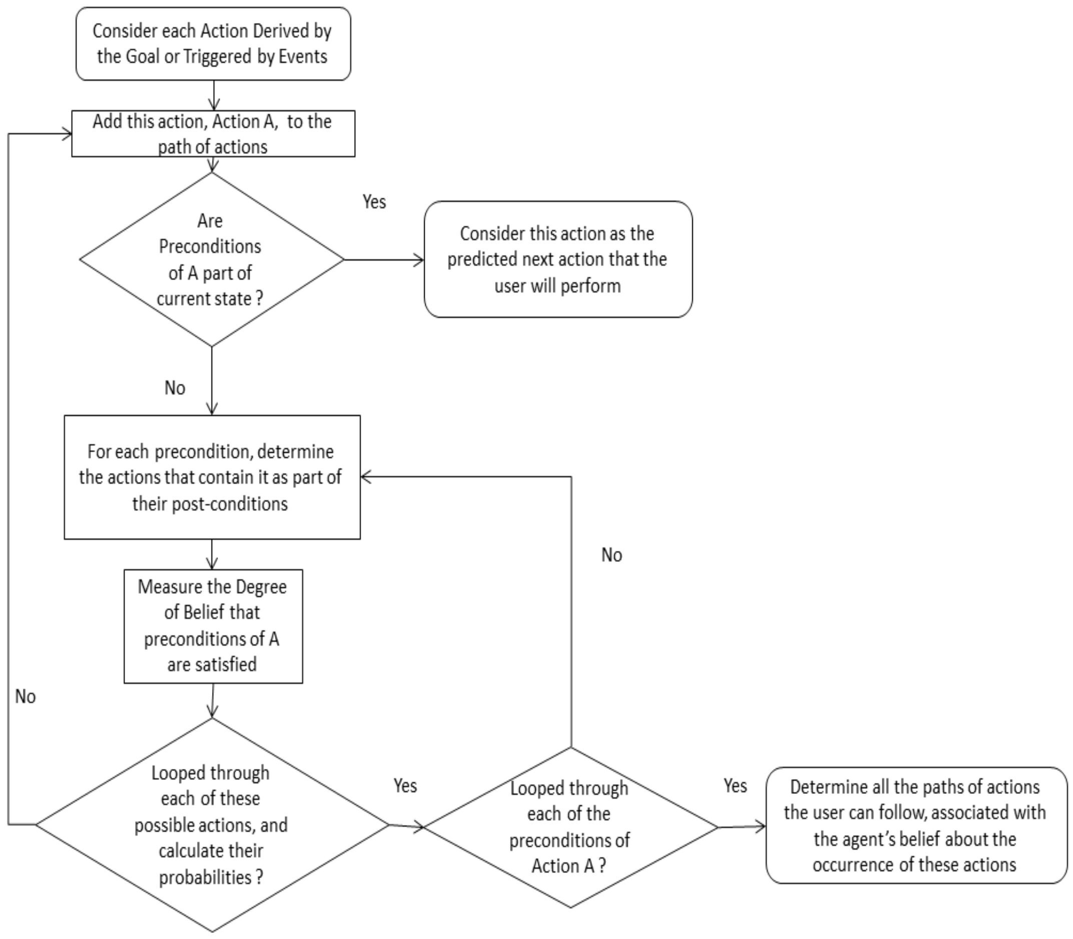

Consider each Action Derived by the Goal or Triggered by Events
Add this action, Action A, to the path of actions
Are Preconditions of A part of current state ?
Yes
Consider this action as the predicted next action that the user will perform
No
For each precondition, determine the actions that contain it as part of their post-conditions
No
Measure the Degree of Belief that preconditions of A are satisfied
No
Looped through each of these possible actions, and calculate their probabilities ?
Yes
Looped through each of the preconditions of Action A ?
Yes
Determine all the paths of actions the user can follow, associated with the agent's belief about the occurrence of these actions

Figure 4-8: Path Determination Algorithm

As shown in the flowchart, if the user's action has its preconditions as part of the current state of the world, then this action will be predicted as the next user's action, whether this action is part of the user's plan to achieve his/her goal, or it is an action that should be performed as a response to an event. If the action's preconditions are not part of the current world, then this algorithm determines the paths of actions that the user can take to make these preconditions part of the current world, and then perform the action. These paths of actions contain all the possible actions that could be performed to satisfy a certain precondition, hence, these possible actions are part of the plan of actions that will eventually achieve the goal or respond to an event. These possible actions, noted by n, are a subset of all actions in the environment N. Hence, there are only n number of actions that can be satisfy a certain precondition, where $n \subseteq N$. Three main points to consider regarding this algorithm:

- This algorithm defines all possible sequences of actions the user can take to achieve his/her goal, or to respond to events.
- The algorithm starts with the action that can achieve the goal (a_g, where $g \subseteq$ of post-conditions(a_g))
- The algorithm finishes when all the paths are determined along with their probabilities from desired state s_d back to current state s_c

One of the algorithm's steps is to measure the degree of belief that the preconditions of an action are satisfied. As mentioned in Chapter 2, uncertainty is considered when capturing the user behaviour because of the dynamic nature of this behaviour. Furthermore, since we are modeling the user in open environment, it is not deterministic which actions and events could occur in the environment, and therefore, we are not certain that a specific action's preconditions are satisfied. Rather, probability distribution is used to measure the degree of belief that these preconditions are satisfied given the possible actions in the environment. An assumption is given, such that the probability of satisfying the preconditions of an action is the same as the probability of performing that action. This assumption is appropriate because since an action is driven by the goal or triggered by events, once the preconditions are satisfied, then the action will be performed.

Binomial distribution is used to calculate the probability at which the preconditions of the Action are satisfied based on other actions in the system; these actions are represented by the random variable A. (a_1, a_2, a_3… are possible values for the random variable A).

$$P\ (\text{preconditions (Action)}|\ a_1, a_2, a_3\ldots) = ? \qquad \textbf{(4-1)}$$

Since all events are registered in the environment, and all actions that the user can perform are known, it is assumed that the overall number of possible actions is known, furthermore, the preconditions and post conditions of all the actions are also known.

Therefore, it is possible to know the number of actions that can satisfy a specific precondition, in other words, the number of actions that have this specific precondition as their post condition. If n represents the total number of possible actions, and k represents the number of actions that contains the preconditions of the action as part of their post condition, then the binomial distribution calculates the probability that one of the k will occur next given n, hence it calculates the probability that the next action will be an action that can satisfy the precondition.

The probability of satisfying the preconditions of an action is equal to the probability that these preconditions (states and events) are part of the post-conditions of other actions in the environment as given in the following equation:

$$P(\text{preconditions}(ai)) = P(\{s_1, \ldots, s_n\} \in \text{PostConditions}(a_k, k \; \{1\ldots n\}, k \neq i) \cap \{e_1, \ldots, e_n\} \in \text{PostConditions}(a_k, k \; \{1\ldots n\}, k \neq i)) \quad \textbf{(4-2)}$$

The probability that a certain state s or event e is an element of post-conditions of other actions in the environment is calculated through the conditional probability function of binomial distribution

$$P(s \text{ (or } e) \in \text{PostConditions}(a_k, k \; \{1\ldots n\}, k \neq i) = f(k|\pi) \quad \textbf{(4-3)}$$

As mentioned in Chapter 2, f (k|π) is the conditional probability function for binomial distribution, given as:

$$f\,(k|\pi) = \pi^{k}(1-\pi)^{n-k} \quad \textbf{(4-4)}$$

Where n is the number of actions, k is the number of successes, and π is the probability of success. The probability of success, π is equal to 0.5 since the probability of a state (or event) being in a post condition of an action is 0.5, either true or false. Equations 4-2 to 4-4 represent the approach followed to calculate the prior probability of the preconditions of the action given all the other actions in the environment, using binomial probability distribution.

In order to predict the next action that the user is going to take, a probabilistic graphical modeling is used. A Bayesian network is constructed using the different paths of actions that are generated from the path determination algorithm, as shown in Figure 4-9. Each color in Figure 4-9 represents a unique sequence of actions, and each action is associated with a prior probability that is calculated using equations 4-2 to 4-4 as explained above.

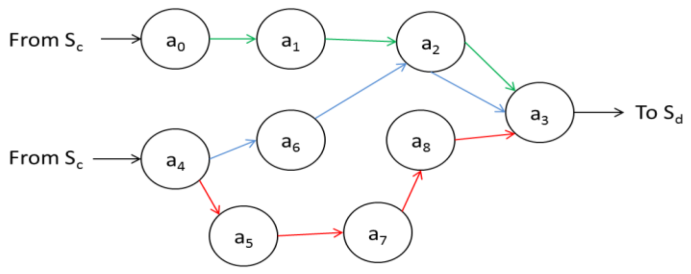


Figure 4-9: Bayesian Network for User's Actions

$$P(a_1, \ldots, a_n) = \prod_{i=1}^{n} P(a_i \mid parents\,(a_i)) \qquad \textbf{(4-5)}$$

As discussed in Chapter 2, the Bayesian network is a probabilistic graph that calculates the conditional probability of one node based on the observed values of its parents. Using the Bayesian network in Figure 4-9 and equation 4-5, the predicted next action can be calculated. When the next action needs to be predicted, equation 4-5 is used to calculate the probability of each child based on the observed value of its parents. From each possible sequence, the child is the final action in that path, and the last parent is the current action that the user performed. The path with the highest probability will be considered as the predicted path and the first action of that path, after the current action that is already performed, is considered as the next predicted action to be performed by the user. This Bayesian calculation is only used if more than one possible action can be performed next, i.e. there is more than one possible sequence of actions to follow.

4.3.2 Capturing the User's Interests

The user interests are modeled as the preconditions and post-conditions of the actions that the user performs. These preconditions and post-conditions are the required and impacted states and events in the environment respectively. The events are captured in the registered events' list; on the other hand, a knowledge representation tool is necessary to encode the states of user interest model into a format understood by the computer. A Conceptual Graph (CG) is one of the formal knowledge representation tools that are used to represent the knowledge of the system. A Conceptual Graph (CG) is a directed bipartite graph of two kinds of nodes, concepts (C) which are represented as square nodes, and relations (R), which are represented as circle nodes, an arc between them which is an ordered pair <r,c>, that links a conceptual relation r to a concept c [42]. Each concept node has a type label and referent, while each relation node has a type associated with it. A type label designates the type of the concept or the relation, while referent specifies the entity that the concept node is referring to [42]. Therefore, CG is able to represent domain model expressed above, with concepts of the domain represented by concept nodes and relations between these concepts are represented by relation nodes. The user's interest model, which consists of concept types (such as products, properties, etc.), concept and relations types such as (instance, is_a, etc) can be modeled by CG as shown in the example of Figure 4-10. Furthermore, CG can represent the user interest in a format that is logically precise, computationally tractable, and humanly readable [26].

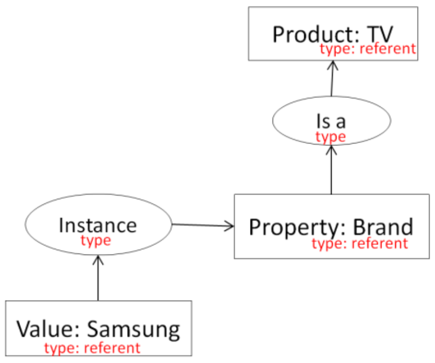


Figure 4-10: CG to represent part of the User Interest Model

4.4 The Inference Component

All possible sequences of the user behaviour's actions are captured through the path determination algorithm. Then a Bayesian network is used to predict the next action that the user is going to perform. The process of querying the probability of a node from a Bayesian network is called probabilistic inference. Several algorithms are available to infer the Bayesian networks and they are divided into two main categories, exact inference algorithms and approximate algorithms as mentioned in Chapter 2. The result of inferring the Bayesian network is to identify the next action that the user is predicted to perform as explained in Section 4.3.

Once the next predicted action is identified, the inference component identifies the user interests associated with this predicted action. The predicted actions, as well as the user's interests are forwarded to the environment assistance's component to match them with the appropriate assistance that is available in the environment.

4.5 The Environment Assistance Component

The environment assistance component retrieves the associated actions and interests from the inference component as discussed in the previous section. The environment assistance component then matches these actions and interests to the available assistances in the environment. The assistance that has the highest matching result, and passed a certain threshold will be recommended to the user.

4.6 The Environment Model

The environment model represents the context of the environment and captures the actions that are actually performed by the user, as well as the events that are triggered in the environment. This is done through several sensors or through following the user's interactions with his/her computer or mobile device that this personal assistant is implemented in. Additionally, the environment model identifies any potential assistance that exists in the environment, and forwards their information to the environment application component to be matched with the contents of the user model.

Chapter 5

5 Implementation, Results and Observations

In this chapter, a Personal Smart Assistant's (PSA) prototype is developed for the domain of personalized advertisements, based on the proposed generic PSA architecture given in Chapter 4. The PSA is utilized to provide relevant advertising messages as a form of assistance to the user that enables him/her to achieve his/her goal efficiently.

5.1 The Prototype: PSA for Digital Advertisements

The personal smart assistant (PSA) is used to deliver relevant advertising messages to the user. The PSA exists in an environment with other entities as shown in Figure 5-1. The user has a goal that he/she is trying to achieve in the environment; this goal is provided to the PSA as a direct input from the user. The user performs several actions in the environment to achieve his/her goal, and response to several events that could be triggered in the environment. The PSA observes the user's actions, both the deliberative (goal-driven) and reactive (event-driven) actions, and model these actions into a behaviour model. Furthermore, the PSA models the user's interests, which are reflections of his/her behaviours. The environment also contains smart multimedia agents that represented advertising messages that are promoted to the users of the environment.

Upon modeling the user, the PSA's problem-solver matches the user's model with the available advertising messages within the environment. The PSA then recommend the advertising messages that are relevant to the user and filters out the rest. Hence, only those advertising messages that are relevant and can help in achieving the goal are delivered to the user through his/her PSA.

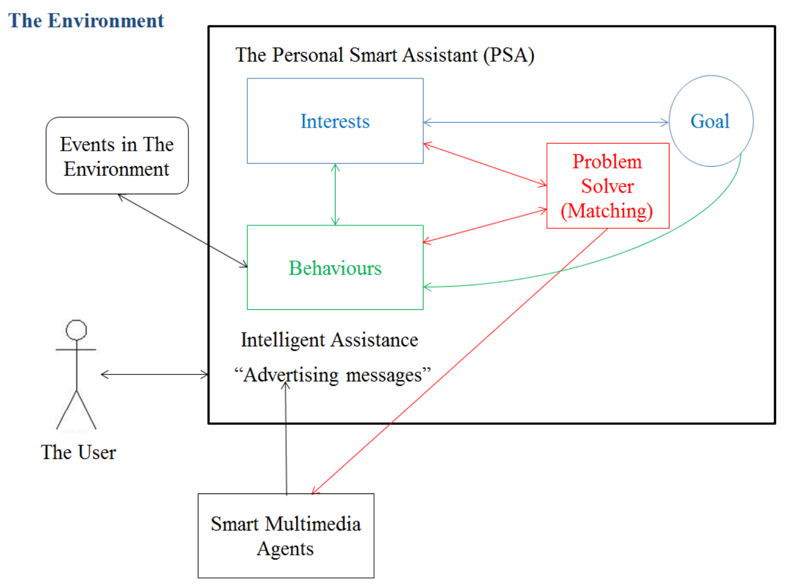


Figure 5-1: System’s Overview

5.1.1 The System Architecture

The system’s architecture is given in Figure 5-2. The first component of this architecture is the personal smart assistant PSA, which models the user, and provides relevant advertising messages accordingly in order to help the user in achieving his/her goals. The PSA is represented as an agent that needs to reason about the user and the environment and make decisions to assist the user, more details about modeling the personal assistant as an agent is given in the next section.

The assistance that the PSA is delivering to its user is in the form of a multimedia file. It is represented as a smart multimedia agent, proposed in [39], that can reason about the content of its multimedia file and make decisions on where to display this file. The environment also contains sensors that captures the user’s actions and trigger events in the environment. The sensors utilized in this prototype are the RFID [43] and the Kinect [44]. The components of the environment are integrated through the Real-Time Integration Layer (RTIL). The RTIL controls the services, resources, events and data that exist in the environment. Services and events have to be registered in RTIL to be accessible to any of the components of the environment [37]. The personal assistant agent and the smart multimedia agents are built on top of the Java Agent DEvelopment framework (JADE). Jade ensures that the necessary communication is achieved between the agents as they are interacting within the environment [45].

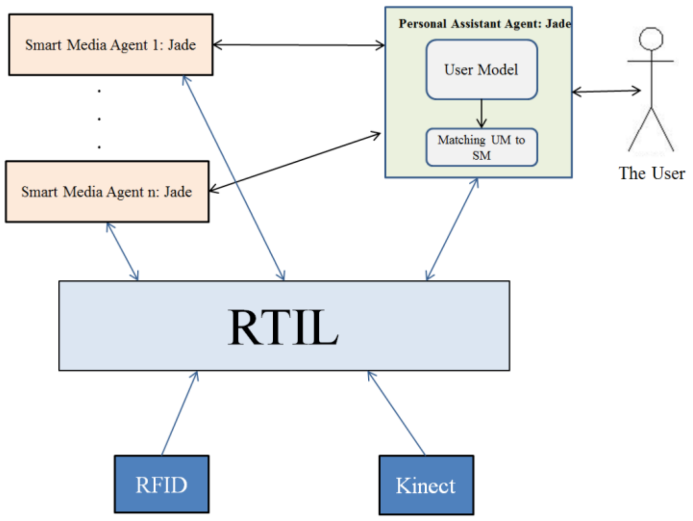

Smart Media Agent 1: Jade
Smart Media Agent n: Jade
Personal Assistant Agent: Jade
User Model
Matching UM to SM
The User
RTIL
RFID
Kinect

Figure 5-2: System's Architecture

5.2 The Personal Smart Assistant Model

The personal smart assistant (PSA), proposed in Chapter 4, needs to model the user and make decisions in terms of suggested assistance based on this modeling. It should act autonomously to reason about the user's behaviours and interests and provide the relevant assistance based on its problem solving ability. Furthermore, the PSA needs to communicate with other entities in the environment to provide the assistance to his/her user. Consequently, the PSA needs to have capabilities such as autonomy, cooperation and intelligence to be able to provide its assistance to the user; these capabilities are ensured in the CIR-Agent Model [46]. The Coordinated Intelligent Rational (CIR) provides a generic model for an agent in cooperative distributed systems. The CIR-Agent contains capabilities such as autonomy, cooperation, intelligence and rationality, and hence, it's able to model the proposed personal smart assistant.

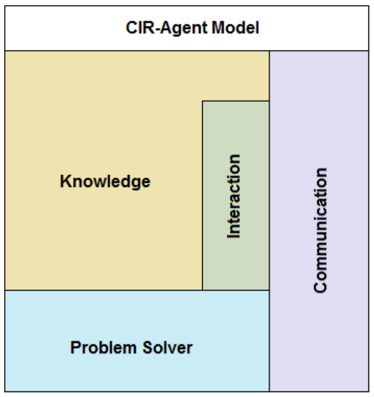


Figure 5-3: CIR-Agent Architecture

As shown in Figure 5-3, the CIR-Agent architecture includes four main components [46]:

- Problem-solver: provides the agent the capability to reason based on its knowledge and make decisions to achieve its tasks accordingly.

- Knowledge: provides the knowledge required for the agent to reason and make decisions. It contains local knowledge about the agent itself as well as external knowledge about the environment that the agent is performing in.

- Interaction: solves interdependencies problems through providing different interaction protocols.

- Communication: enabling the agent to send, receive and interpret messages.

The personal smart assistant architecture can be mapped to the CIR-Agent components as indicated in Figure 5-4. The user model and the environment models are included in the CIR-agent’s knowledge component. The knowledge update component is part of the CIR-agent’s interaction component. The inference component, collaborative assistance component and environment assistance component are all included in the CIR-agent’s problem solver component.

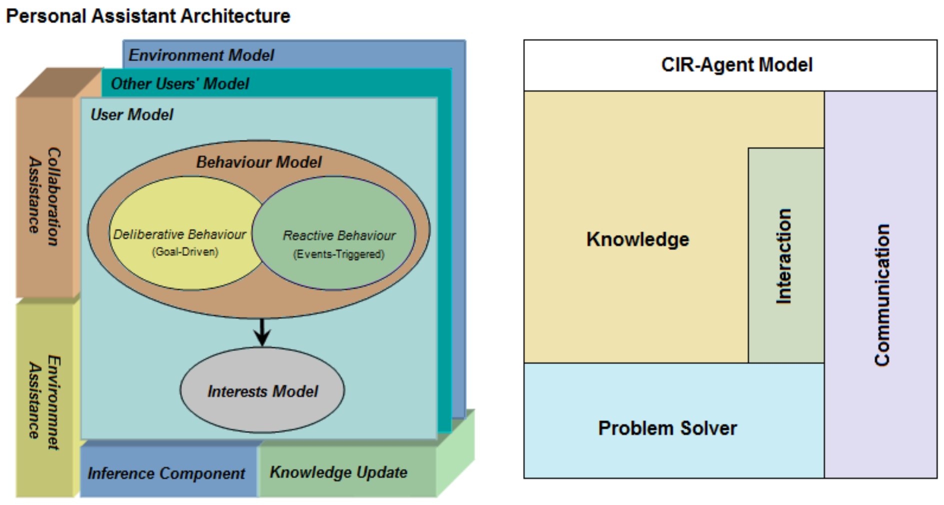


Figure 5-4: Modeling PSA as a CIR-Agent

5.3 Implementation of the Personal Smart Assistant

In this section, the major implemented components of the Personal Smart Assistant (PSA) are discussed. Other components from the system's architecture, such as the RTIL and the Smart multimedia agents are already developed [37]; their implementation is not part of this work.

With the expansion of cyberspace and smart environments, people are interacting with technology all the time through their mobile devices. Therefore, it is not sufficient enough anymore to build a personal smart assistant (PSA) that is able to assist the users only while they are using their computers. Rather, the personal smart assistant should be built in a mobile device that can accompany the user anytime and therefore provide relevant assistances at every opportunity. Therefore, the platform that is chosen to build the PSA is Google's Android Platform [47]. Android provides a java SDK that can be utilized to build java-based application on the mobile platform [48]. The major components of the implemented android-based PSA's prototype are discussed in the following subsections.

5.3.1 Conceptual Graphs

A conceptual graph (CG) is a knowledge representation approach that can represent concepts and relations in a format that is logically precise, computationally tractable, and humanly readable. Therefore, the conceptual graphs are chosen to represent the different states of the world as well as the user interests. In this prototype, Notio Java Library [49] is used to create the required conceptual graphs for the PSA. Notio is a set of java classes designed to create and manipulate major components of conceptual graphs such as; concepts, relations and types.

5.3.2 Path Determination Algorithm

An important component of the implemented PSA is the path determination algorithm, shown in Figure 4-8 in Chapter 4. This algorithm is needed to determine the different possible paths of actions the user can take to achieve his/her goal. Furthermore, it determines the possible paths of actions that the user should perform when an event is triggered in the environment. The PSA implementation contains several key objects such as Action, Preconditions, and Post-conditions that are utilized in this algorithm as shown in the algorithm's pseudo code in Figure 5-5.

```
Input: Goal or Event Actions (Actions which contains the desired state or triggered event in their post-
conditions)

Function Determine Path (ArrayList of inputActions)
1) For each Action A in the input ArrayList
2)      Measure Prior Belief of Action A
3)      Add to actionsInPath ArrayList
4)      If the Current States of the World contains all Preconditions of action A
5)              Return actionsInPath ArrayList
6)              End Algorithm.
7)      Else
8)              For All Actions in Environment do:
9)                      If Action A's preconditions contains an environment's actions post-conditions
10)                             Add this environment action to requiredActions ArrayList
11)             Recursively Call Determine Path (ArrayList of requiredActions)
```

Figure 5-5: Path Determination Algorithm's Pseudo Code

As shown in Figure 5-5, the result of this algorithm is an ArrayList of actions *actionsInPaths* which contains all the actions that the user could possibly perform to achieve his/her goal or to respond to a triggered event. ArrayList was chosen because of its simplicity and effectiveness in this case. More complicated data structures such as linked lists or trees are not necessary as this produced ArrayList will be further utilized by a Bayesian Network Graph data structure as explained in the next subsection. On the other hand, ArrayList was a better option than arrays because the maximum size of the array is not known in advance, therefore ArrayList provides a more expandable option.

The complexity of this algorithm depends on two variables; the first one is the total number of actions in the environment, let it be N. The second variable is the total number of possible actions required to achieve the goal or to respond to an even, let it be n, this variable contains the number of actions that are within the plan of actions, and hence, it expands as recursion occurs in the algorithm. Since there is a nested for loop of two lists of elements (N and n) then the complexity is O(Nn), however, this function is recursively called based on the number of required actions n. Hence, the overall complexity of the Path Determination algorithm is O (Nn^2).

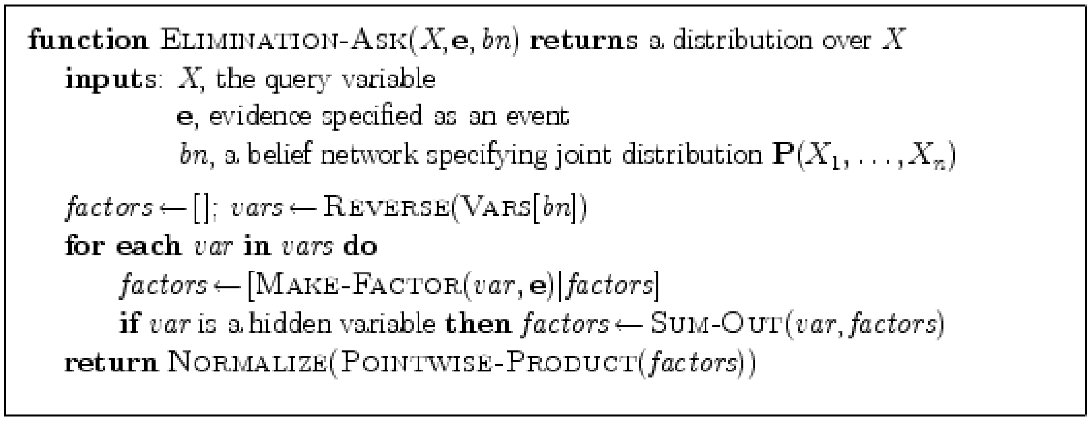
5.3.3 Belief System

The degree of belief that a certain action will be performed next by the user is calculated at two levels. First a prior belief is measured that a certain action will be performed using binomial distribution. As indicated in the Figure 5-5, this belief is measured during the path determination algorithm. To calculate binomial distribution, as explained in Chapter 4, an apache binomial distribution library [50] is used. Every action in the *actionsInPaths* ArrayList that is produced from the path determination algorithm contains a prior probability attribute that its value is measured through the binomial distribution.

The next stage in measuring the belief that a certain action will be performed next is to consider posterior (conditional) probabilities; the probabilities of performing the action given that other actions are performed. As mentioned in Chapter 4, Bayesian Networks are directed graphs that calculate the posterior probabilities of the child node, given the observed values of the parents' nodes. JavaBayes [51], along with some helper functions provided in [52], are used to provide a java library to construct a Bayesian network based on the Graph data structure.

Each action "a" in the *actionsInPaths* ArrayList is transformed into a node in the Bayesian Network graph. The parents of the node of action "a" are the nodes that represent actions which their post-conditions are preconditions of action "a". Therefore, the probability whether action "a" is going to be performed or not depends on whether its parents are performed or not. The utilized Bayesian network from JavaBayes library uses the *variable elimination*, an exact inference algorithm, to determine the next possible action to perform that will result in higher probability of achieving the user's goal.

The complexity of the belief component in the PSA is driven by the complexity of the algorithm used to infer the Bayesian Network. In this prototype we are using the variable elimination algorithm. Basically, variable elimination algorithm eliminates all the hidden variables (variables that are neither in the query or evidence), one at a time, by summing out all the factors related to these variables. Factor is a function that takes arguments and gives a value for every assignment to these variables [11]. The probability of the query variable is then calculated by normalizing over the product factors of the remaining variables (which are the query and evidence variables) [10]. The pseudo code of this algorithm is given in Figure 5-6 [10].



```
function ELIMINATION-ASK($X$, $\mathbf{e}$, $bn$) returns a distribution over $X$
   inputs: $X$, the query variable
           $\mathbf{e}$, evidence specified as an event
           $bn$, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

   $factors \leftarrow []$; $vars \leftarrow$ REVERSE(VARS[$bn$])
   for each $var$ in $vars$ do
       $factors \leftarrow$ [MAKE-FACTOR($var$, $\mathbf{e}$)|$factors$]
       if $var$ is a hidden variable then $factors \leftarrow$ SUM-OUT($var$, $factors$)
   return NORMALIZE(POINTWISE-PRODUCT($factors$))
```

Figure 5-6: Variable Elimination Algorithm's Pseudo Code [10]

The complexity of this algorithm is driven by the total number of values in the largest generated factor in the algorithm. The total number of values in the largest factor is measured as the possible values for all variables in the Bayesian network, d, raised to the power of the number of variables in that factor, k. Therefore the complexity of the variable elimination algorithm in a general Bayesian network is O(d^k), where d is the possible values of the variables in the Bayesian network, and k is the number of variables in the largest factor used in the algorithm. In our Bayesian network, the values of all the different variables are 2 (the action is either performed or not). Therefore, the complexity of using the variable elimination algorithm is O (2^k) where k is the number of variables in the largest factor in the algorithm. In other words, k is the largest degree among the degrees of variables in the network. Hence, as the Bayesian network gets more complicated and each node get associated with several other nodes, this variable elimination algorithm complexity increases, therefore, it should not be used in complicated Bayesian networks. Other approximate algorithms are available to reduce the complexity of a Bayesian network. However, for our prototype, and for a Bayesian network with small number of variables, then variable elimination algorithm is an acceptable choice.

The complexity of the belief component in the PSA is based the complexity of the variable elimination algorithm used, which is O (2^k), where k is the largest degree among the degrees of variables in the network.

This work assumes that the next predicted action that the user is going to perform is the action that will result in higher probability to achieve the goal; hence, it is the action that falls in the path that contains the less number of actions required to achieve the goal. If more than one path is applicable, then the next predicted action will be picked randomly from these paths.

5.3.4 Matching Algorithms

Once the next action is predicted, the user’s interest is determined as the preconditions and post conditions of the current and next predicted actions. Then the user’s interest is matched with the contents of the advertising messages provided by the smart multimedia agents. As discussed before, through the work proposed by [37], the content of the smart multimedia is already determined as concepts and their relations. Hence, a direct matching is possible between the contents of these advertising messages and the user interest model.

To ensure good performance for the Personal Smart Assistant (PSA) two stages of matching are performed. First, once the user give his/her goal, the PSA matches the concepts of type “Product” that are included in the goal, such as tea, milk, sugar, etc. with all the available advertising messages in the environment. The advertising messages that contain concepts of type “Product” which are not the same as the goal concepts are eliminated from the matching. Hence, once the user gives his/her goal, the PSA eliminates the advertising messages that are irrelevant to the products of the goal even before the user start performing any of his/her actions to achieve the goal. This stage of the matching component is performed by a proposed algorithm which is called the filtering algorithm, as it filters out the advertising messages that are not relevant to the user’s goal. Figure 5-7 shows the pseudo code of this algorithm.

```
Function filterOutAds (User Goal, All Advertising Messages)
1)  For each concept "c" in the user goal.
2)          If the type of concept c is "Product"
3)          For each advertising Message "ad"
3)                  If ad's concepts contain c
4)                          Add "ad" to potential_match ArrayList
5) Return potential_match
```

Figure 5-7: Filtering Algorithm

The complexity of this algorithm depends on the number of concepts in the user's goal as well as the number of overall advertising messages. Let the number of the goal concepts be *G*, and the overall number of advertising messages be *A*, since the algorithm contains a nested loops for the entire goal's concepts, *G*, and the entire number of advertising messages *A*, then the complexity of the filtering algorithm is O (*AG*).

The second stage is to match the user's interest with the remaining "potential_match" advertising messages. This is called the recommending algorithm as it recommends the most relevant advertising message. The concepts of each advertising message is compared with the concepts of the user's interest (for simplicity, only the concepts are matched in this prototype, relations are ignored upon matching), this matching stage happens as the user's behaviours and interests are captured. If the advertising message contains concepts of types "product", "property" and "value" that are similar to the concepts of types "Product", "property", "value" in the user interest, then the matching result variable increases accordingly. So the more properties and values that are similar the higher the matching result number is. However, if the "Product" type in user interest is different than the "Product" type in the advertising message, then the matching result of this advertising message is 0 regardless of the similar properties and values. For example, a green, low-priced, lemon is not relevant to a green, low-priced tea. The matching results of all "potential_match" are sorted and the top ranked advertising message is recommended to the user if it passes a certain threshold. This threshold decided the required relevancy of the advertisement. The higher the threshold, the more relevant this advertisement is to the user's interests. Figure 5-8 shows the pseudo code of recommending algorithm.

```
Input: User Interest, advertising message from potential_match list

Function recommendingAds (User Interest, potential_match Advertisements)
1) Set threshold
2) For each concept "c" in the user interest
3)        For each advertising Message "ad"
4)                If the type of concept c is "Product"
5)                        If ad's concepts contain c
6)                                ad_matchingResult = ad_matchingResult+50
7)                        Else
8)                                ad_matchingResult = 0
9)                                Break
10)               If the type of concept c is "Property"
11)                       If ad's concepts contain c
12)                               ad_matchingResult = ad_matchingResult +10
13)               If the type of concept c is "Value"
14)                       If ad's concepts contain c
15)                               ad_matchingResult = ad_matchingResult +20
16)               Add ad_matchingResult to ResultsArray
17) Sort ResultsArray
18) If first element in ResultsArray > threshold
19)       Recommend the Ad associated with the ResultsArray's first element
```

Figure 5-8: Recommending Algorithm

The complexity of this algorithm depends on the number of concepts in the user's interest's model, the number of potential advertising messages, as well as, the complexity of the sorting algorithm. Let the number of the user interest's concepts be *I*, and the number of potential advertising messages be *a*. A simple quick sort is used to sort the matching results of the matched advertisements with complexity of O(*a*.log (*a*)). Furthermore, since this recommending algorithm contains a nested loop with the entire number of interesting concepts *I*, and the entire number of potential advertising messages *a*, then O (*Ia*) is also an added complexity in this algorithm. Therefore the overall complexity of the recommending algorithm is O (*a*.log (*a*)) + O (*Ia*).

5.3.5 Utilizing Sensors

To capture the user's physical actions a smart environment has to be utilized. The environment in which the implemented PSA is developed contains two sensors as discussed before, the RFID and the Kinect. The user's actions were divided into two types, the ones that require some time to perform; these actions are detected by the Kinect. The Kinect contains a depth camera that enables us to measure the distance of a person to the Kinect. This was utilized to sense the user's actions such as "moving towards", "moving away", "standing in front of", and "looking for". All these actions require some time and by measuring the distance of the user from the Kinect, these actions can be sensed. As the distance decrease, the user is assumed to be "moving towards", as the distance increases, the user is assumed to be "moving away", if the distance is close to the Kinect (less than 1m for example), then the action "standing in front of" is detected. If the user remains standing for some time, then "looking for" action is assumed, etc.

The second type of user actions are the ones that are instant actions and require instance sensing, such as "picking up item", "placing item in cart", etc. These actions are sensed through the RFID reader and the tags that are associated with every product. Two RFID readers are utilized; the first reader represents the shelf that contains the products. If that reader was tagged, then the product that is represented by the tag is assumed to be "Picked up". The second reader represent the user's cart, if reader 2 reads the tag, then the product that is represented by the tag is assumed to be "put in the cart".

5.4 Results and Observations

In this section, results from the implemented prototype of the Personal Smart Assistant (PSA) is gathered and observed to verify the hypotheses given in Chapter 3.

5.4.1 Only Relevant Assistance is recommended through the PSA

Modeling the user's behaviours and interests ensures that the Personal Assistant recommend relevant advertisements. In order to verify this hypothesis, two different cases are compared.

In the first case, no personal assistant is used and the environment promotes advertising messages, through a simulator, to the user's device randomly. A simple application is built in the user's device that simply receives these advertisements which are represented by the smart multimedia agents. A goal is given by the user; however, in this case, this goal is just used to compare how many of the received advertising messages are related to the goal. The goal is not used to filter advertisements in this case. The complexity of this simple simulator is O(A), where A is the overall number of available advertising messages. Every time an advertising message is promoted, it is actually picked out randomly from the entire number of advertising messages. As the overall number of advertising messages increases, the time it takes to promote a random advertising message increases.

In the second case, the personal smart assistant's prototype is used to filter out the irrelevant promoted advertisements, and only recommend the ones that are relevant to the user's model. This filtering algorithm, with complexity of O (AG) only runs once after the goal is given to the PSA by the user. Afterwards, the number of matched advertisements are reduced to a, which are those advertisements that are relevant to the user's goal, and the recommending algorithm, with complexity of O(Ia) is called every time the captured user interests need to be matched with these match able advertisements. An increase in the overall number of advertising messages will increase the time it takes to perform the filtering algorithm. However, assuming that the number of relevant advertising messages remains the same, then the time to run the recommending algorithm will not be affected.

Furthermore, regardless of the overall number of available advertising messages, the PSA guarantees a 100% relevance percentage between the promoted advertising messages and the user's model. On the other hand, since the simulator promotes advertising messages randomly, the relevance percentage depends on the ratio of relevant advertising messages to the overall number of available advertising messages, as this ratio decreases, the average relevance percentage decreases.

Both the PSA and the simulator were tested in a simple scenario. In this scenario, the environment contains 100 advertising messages; only 5 of them are relevant to the user's goal. The goal of the user in this scenario is to purchase a black tea and a low-priced sugar. Table 5-1 shows five different trials for case 1. It is clear that the relevance percentage is very low as there are only 5 relevant ads out of the 100.

Table 5-1: Promoted Ads without PSA

	NO Personal Assistant Case		
Trials	Total Ads Promoted	Ads Relevant to Goal	Relevance Percentage
1	42	3	7.14%
2	32	2	6.25%
3	28	2	7.14%
4	19	0	0%
5	15	0	0%

On the other side, in case two, the relevance percentage is always going to be 100%, as the PSA always promote advertising messages that are relevant to the user, as shown in table 5-2. Because of the limitation of the implemented environment, as discussed in Section 6.2 of this thesis, the results in table 5-2 contains trials for very few numbers of promoted advertisements. However this simple scenario is still able to show that all promoted advertisements are relevant to the user and hence it supports the first hypothesis and illustrates that once the personal assistant models the user, then only relevant advertisements will be recommended to the user.

Table 5-2: Promoted Ads with PSA

	Personal Assistant Case	
Trials	Total Ads Promoted	Ads Relevant to Goal
1	2	2
2	2	2
3	2	2
4	3	3
5	3	3

5.4.2 Relevant Assistance Increases Efficiency

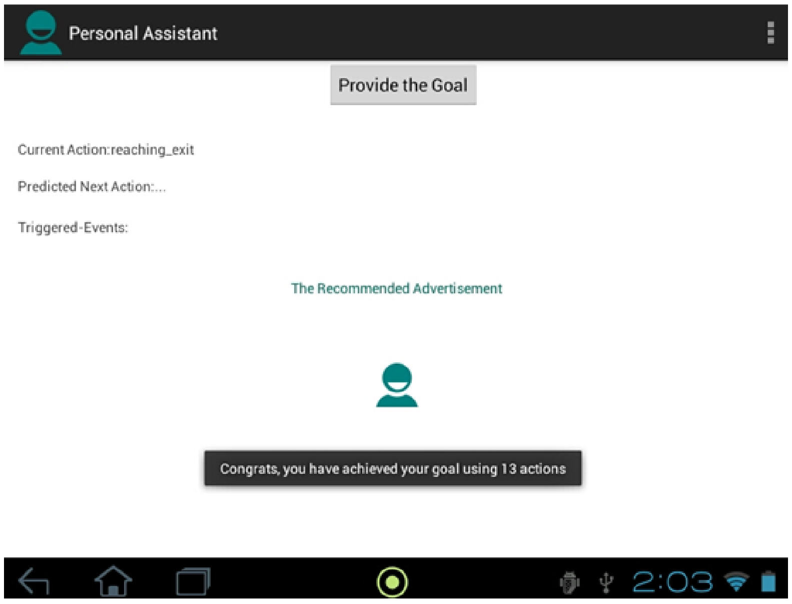
The second hypothesis that is given in Chapter 3 states that recommending a relevant advertising message helps the user to achieve his/her goals more efficiently. Efficiency in this work is measured by the number of actions the user needs to perform to achieve a certain goal. To verify this hypothesis, two cases were observed.

In the first case, without any assistance, a user unfamiliar with the environment will perform actions randomly until his/her goal is achieved. Therefore, an exhaustive, brute-force search with complexity O(N), where N is the overall number of actions in the environment is taking place each time the user is performing an action. In the upper bound, the user might need to perform all the actions in the environment to achieve his/her goal, hence the complexity is O(N^2). An increase in number of actions in the environment leads to a quadratic increase in the number of actions that the user is going to perform randomly before achieving the goal.

In the second case, the personal assistant provides recommended assistances to the user to help him/her achieve the goal using the shortest possible path of actions. The next predicted action is always the action that achieves the goal using the shortest path of actions, and hence, the PSA recommends assistances that can help the user achieve this next predicted action that will help him/her achieve the goal using the shortest possible path. The provided PSA relevant assistance keeps the user on that shortest path, and ensures that the user performs the least number of actions to achieve his/her goal. Although an increase in the overall number of actions in the environment will increase in the complexity of the PSA, however, the shortest path to achieve the goal is not going to change, and the number of actions required for the user to perform, if the assistance is considered, is going to remain the same.

The implemented PSA's prototype is used to observe these two cases. A simple scenario is built where the user's goal is to purchase a black tea and a low-priced sugar. Several actions are required for the user to achieve this goal such as; moving towards the line (or aisle), standing in front of the product's shelf, looking for a product, holding (or examining) the product, putting product in cart, etc.

Figure 5-9 shows the user interface of the implemented prototype. The interface is displaying the result of the first case, the number of actions required purchasing a black tea and a low-priced sugar without any recommended advertising messages is 13 actions.



Current Action:reaching_exit

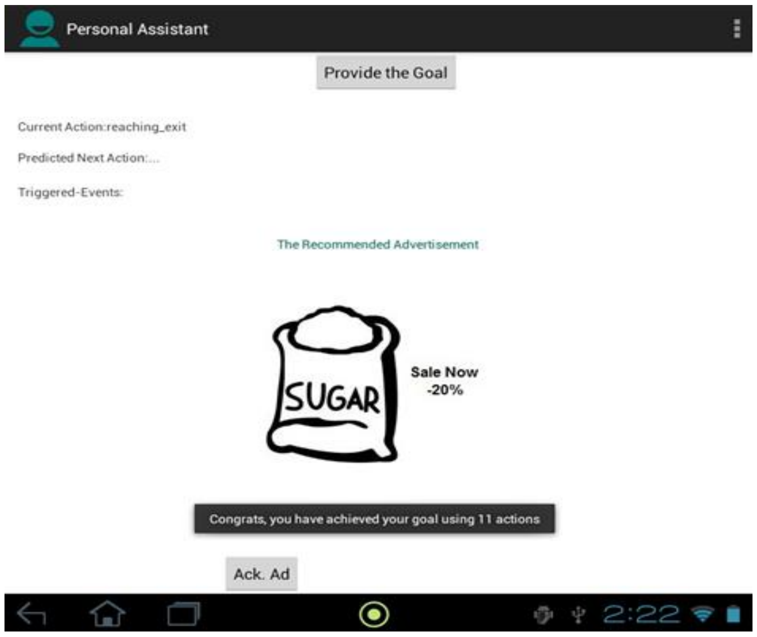
Predicted Next Action:...

Triggered-Events:

Congrats, you have achieved your goal using 13 actions

Figure 5-9: The Goal is achieved without Assistance

In the second case, the PSA recommends relevant advertising based on the user's behaviours and interests. As shown in Figure 5-10, the number of actions required to perform the goal has now been reduced to 11 actions instead of 13 in Figure 5-9.



Current Action:reaching_exit

Predicted Next Action:...

Triggered-Events:

Congrats, you have achieved your goal using 11 actions

Figure 5-10: The Goal is Achieved More Efficiently with Assistance

In this scenario the difference between the numbers of actions performed with and without assistance is not large. This is because the environment contains small number of overall actions, and most of these actions are required to achieve the goal, yet the PSA was able to reduce the number of performed actions even in this simple environment. In environments with larger number of possible actions, the difference between actions performed with and without assistance expands. Therefore, given a relevant assistance, the user was able to achieve his/her goal more efficiently (by performing fewer actions). By providing relevant advertising messages to the user, the user will be guided to perform certain actions that fall in the shortest path of actions to achieve the goal.

5.4.3 Observing Triggered Events Increases Efficiency

Events can be triggered in open environments that require the user to take reactive actions in response to these events. The third hypothesis that is given in Chapter 3 states that observing the user’s reactive as well as deliberative actions ensures that the user response to these events more efficiently. This hypothesis can be verified similar to the analysis given in Section 5.4.2.

If an event occurs in the environment, then the user needs to perform actions as responses to this event, without assistance, these actions could be randomly performed, and again in worst case, the user might perform all the actions in the environment to respond to an event. However, the PSA, being able to model dynamic changes in the environment, and finds the shortest paths of actions to response to these changes; it will provide assistances that will guide the user to follow the shortest path.

For example, if the same scenario is used from Section 5.4.2; the user is trying to achieve the goal of purchasing a black tea and a low-priced sugar. However, this time, as the user is moving towards line (aisle) 1 to pick up the recommended tea product (that was recommended to the user through the PSA as shown in the previous section); an event is triggered indicating that this recommended product is now out of stock. If the PSA does not model events, and does not know the actions that the user needs to perform as a response to this event, then the user will have to perform actions randomly to respond to this event. However, if the PSA is able to model these changes of the environment, then it will realize that because of this event, the user needs to look for another product, and hence, it will provide the assistance to ensure that the user responses to this event with the least number of possible actions. It could recommend another advertising message that promotes an alternative product which is still relevant to the user’s model.

Chapter 6

6 Conclusions & Future Work

The prime objective of this work is to propose a generic personal smart assistant that is able to model the user's behaviours and interests and recommend a relevant assistance accordingly. This personal smart assistant is applied in the domain of personalized advertisements to deliver advertising messages that assist the user in achieving his/her goal. This chapter concludes this thesis and discusses the limitations and future work.

6.1 Conclusions

With technology covering every aspect of our lives, the need of intelligent assistance has increased. However, for this assistance to be beneficial for users, it should be targeted to them based on their needs and preferences. The Personal Smart Assistant (PSA) agent is a promising approach for personalization and user centric assistance. This thesis proposed a PSA that is able to model the user, and the environment he/she is interacting with, in order to provide the relevant assistance that the user needs to achieve his/her goal efficiently. The work in this thesis combines a deliberative, goal-driven, approach with a reactive, event-driven approach to model the user's behaviours. Such a combination guarantees a more accurate modeling for the user behaviours in an open environment that is constantly changing with unexpected events. The modeled user behaviour is then used to predict the user's next actions using probabilistic approaches. Also, a user interest model is proposed that captures the interests of the user based on his/her behaviours. Furthermore, the proposed PSA is able to provide a proactive assistance by matching the user's interests and the predicted behaviours with relevant assistances.

Personalized advertising is essential for ensuring that advertisements are being beneficial for users and rewarding for businesses. The proposed PSA is used in the domain of personalized advertising in an attempt to match the right advertising message to the right person in the right context. A prototype of the PSA is implemented, as a mobile application, that is able to model the user, and proactively recommend advertising messages that help the user to achieve his/her goals efficiently.

6.2 Limitations and Assumptions

Due to time restrictions, a feasible scope of the work has to be drawn. Hence, few limitations and assumptions have been introduced in the proposed work. First, the user’s goals are assumed to be explicitly and clearly given to the proposed personal smart assistant (PSA) by the user.

Second, it is assumed that the overall number of possible actions in the world is given, and these actions, along with their preconditions and post conditions are known. Furthermore, it is assumed that for an event to be triggered it has to be registered in the environment, and hence, the number of possible events that can be triggered in the environment is also known.

Third, there is no prior knowledge of the user’s interests. Rather, it is assumed that the user behaviour is a reflection of his/her interests. Hence, the user’s interests are captured merely based on the actions that this user is performing.

Forth, due to the time restrictions and the limitations of resources (i.e. sensors), the prototype of the PSA is implemented within a limited environment. Therefore, the complexity of the user’s goals, the possible actions, and the scenarios built within the implementation does not serve as a validation tool for the proposed work. In Chapter 5, simple observations were mentioned, and results were captured from the simple implementation to support the thesis’s hypotheses.

6.3 Future Work

This section illustrates the future work that can be done to expand the proposed Personal Smart Assistant (PSA).

- Propose a collaboration component that can be used for the PSA to collaborate with other PSAs in the environment in order to better understand the environment's context and therefore provide a better assistance to the user.

- The proposed user behaviour model can be extended to include patterns of the user's actual pervious behaviours as a prior knowledge that can help to make more accurate predictions. Sometimes, users have their own preferred patterns and sequences of actions for achieving their goals, so predicting that the user is going to follow the shortest path to achieve his/her goal might not always be accurate. Rather, the personal assistant should be able to extract the usual patterns of behaviours from statistical data through data mining techniques and then predict the next actions based on those patterns. This is similar to the cook support robot given in [23], as discussed in Chapter 2.

- A decision network could be added to the proposed PSA that calculates the expected utility of offering assistance. Therefore, the provided relevant assistance is not only based on which action is the most probable to occur next, but also, based on the actions that maximize the user's expected utility if performed. Similar to the approach proposed in [17].

- Privacy control is an important feature to include as an extension to the proposed PSA. Although the user would welcome an advertising message to assist in achieving the goal, he/she might be cautious about sharing his/her behaviours and interests with other entities, such as the vendors that are providing these advertising message. Therefore, future proposals of the PSA should guarantee privacy measures.

- Create an enriched environment that the proposed implemented PSA's prototype can be integrated with. This environment should reflect real life environments that contain many different actions for the user to do, as well as more enriched goals that the users are trying to achieve. Such environments can provide several scenarios and hence many different realistic trials can be recorded. Furthermore, the environment should be able to capture the user’s feedback about the assistances provided by the PSA. Building such enrich environments provide richer results to validate the proposed PSA.

References

[1] S. Joyce. (2010, March 29). TLabs Showcase – Siri. *Tnooz tm* [online]. Available: http://www.tnooz.com/2010/03/29/tlabs/tlabs-showcase-siri-intelligent-local-search-with-voice/

[2] T. Geller. "Talking to Machines", *Magazine Comm. of the ACM*, vol. 55 (4), pp. 14-16, April 2012.

[3] A. Cheyer and D. Guzzoni. ACTIVE Ontologies. *SRI International's Artificial Intelligence Center* [online], Available: http://www.ai.sri.com/software/ACTIVE

[4] W. Roush. (2010, June 14). The Story of Siri, from Birth at SRI to Acquisition by Apple-Virtual Personal Assistants Go Mobile. *Xconomy* [online]. Available: http://www.xconomy.com/san-francisco/2010/06/14/the-story-of-siri-from-birth-at-sri-to-acquisition-by-apple-virtual-personal-assistants-go-mobile/

[5] K. Myers and N. Yorke-Smith, "A cognitive framework for delegation to an assistive user agent," *in Proc. of Association for the Advancement of Artificial Intelligence Symposium on Mixed-Initiative Problem-Solving Assistants*, 2005, pp. 94-99.

[6] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe, "An intelligent personal assistant for task and time Management". *AI Magazine,* vol. 28, pp. 47-61, 2007.

[7] A. Pfeffer, "Functional Specification of Probabilistic Process Models," *in Proc. 20th Annual National Conf. Artificial Intelligence,* 2005. pp. 663-669.

[8] K. Myers and N. Yorke-Smith, "Proactive Behavior of a Personal Assistive Agent," *in Proc. Autonomous Agents and Multiagent Systems*, volume 7, 2008.

[9] S. Wu, H. Ghenniwa, W. Shen, and K. Ma, "Intelligent User Assistance in Collaborative Design Environments," *in Proc. 8th International Conference on Computer Supported Cooperative Work*, 2004, p. 259-266.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* 3rd ed., Pearson Inc., 2010.

[11] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques,* 1st ed., MIT Press, 2009.

[12] E. Smith "Uncertainty analysis", *Encyclopedia of Environmetrics*, vol. 4, pp. 2283–2297, 2002.

[13] W. Bolstad, *Introduction to Bayesian Statistics,* 2nd ed., John Wiley & Sons Inc., 2007.

[14] "Aima-Java - Java Implementation of algorithms from Norving and Russell's "Artificial Intelligence – A Modern Approach 3rd Edition", *Google Code,* [Online] Available: http://code.google.com/p/aima-java/

[15] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere Project: Bayesian user modeling or inferring the goals and needs of software users," *in Proc. 14th Conf. on Uncertainty in Artificial Intelligence,* Madison, WI, 1998, pp. 256–265.

[16] B. Hui and C. Boutilier, "Who's asking for help? A Bayesian approach to intelligent assistance," *in Proc. 11th international conf. on Intelligent User Interfaces,* 2006, pp. 186–193.

[17] S. Brown, E. Santos, and S. Banks, "Utility Theory-Based User Models for Intelligent Interface Agents," *in Proc. 12th Canadian conf. on Artificial Intelligence,* Vancouver, Canada, 1998. pp. 378-392.

[18] Y. Kim Y. and S. Cho, "A Recommendation Agent for Mobile Phone Users Using Bayesian Behavior Prediction," *in Proc. of the 3rd international Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2009, pp. 283-288.

[19] A. McNeile and N. Simons, "Methods of Behaviour Modelling: A Commentary on Behaviour Modelling Techniques for MDA". *Metamaxim Ltd*, 2004, DRAFT Version 3

[20] R. Fikes and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence* vol. 2(3/4), pp. 189-208, 1971.

[21] C. Burghardt and T. Kirste, "A Probabilistic Approach for Modeling Human Behavior in Smart Environments," *Lecture Notes in Computer Science,* vol. 5620, pp. 202–210, 2009.

[22] C. Burghardt, M. Giersich and T. Kirste, "Synthesizing probabilistic models for team activities using partial order planning," *in Proc. of Ambient Intelligence Workshop,* Germany, 2007.

[23] T. Fukuda, Y. Nakauchi, K. Noguchi and T. Mastubara, "Human Behavior Recognition for Cooking Support Robot," *in Proc. of the international Workshop On Robot and Human Interactive Communication*, 2004, pp. 359-364.

[24] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. "A decision-theoretic approach to task assistance for persons with dementia," i*n Proc. 19th international Joint Conf. on Artificial Intelligence*, Edinburgh, Scotland, 2005, pp.1293–1299.

[25] Shumin Wu, "Intelligent Assistance In Collaborative Design Environments," MESc. thesis, Dept. of ECE, University of Western Ontario, London, ON, Canada, 2003.

[26] J. Sowa, "Conceptual Graphs Summary," in *Conceptual Structures,* Ellis Horwood, 1992, pp. 3-51.

[27] Y. Zhang, “Agent-Oriented Assistance for Collaborative Mobile Environments,” MESc. thesis, Dept. of ECE, University of Western Ontario, London, ON, Canada, 2005.

[28] Abdou Al-Rahim Kadri, “Agent-Oriented Personal Assistant For Mobile Environment”, MESc. thesis, Dept. of ECE Department, University of Western Ontario, London, ON, Canada, 2007.

[29] S. Wu, H. Ghenniwa, W. Shen and Y. Zhang, “Personal assistant agents for collaborative design environments”, *Computers in Industry,* vol. 57 (8/9), pp. 732–739, 2006.

[30] G. Allen and S. March “The Effects of State-Based and Event-Based Data Representation on User Performance in Query Formulation Tasks,” *MIS Quarterly,* vol. 30, pp. 269-290, June 2006.

[31] F. Belli, C. Budnik, and W. Wong, “Basic Operations for Generating Behavioral Mutants,” *in Proc. 2nd Workshop Mutation Analysis*, 2006, pp. 9-18.

[32] A. Hinze and A. Voisard, “Location- and Time-Based Information Delivery in Tourism,”. *in Proc. of 8th international Symposium in Spatial and Temporal Databases*, Santorini Island, Greece, 2003.

[33] Y. Cao, L. Tao, and G. Xu, “An event-driven context model in elderly health monitoring,” *in Ubiquitous, Autonomic and Trusted Computing, Symposia and Workshops* (2009), pp. 120–124.

[34] T. Pessemier, T. Deryckere, K., Vanhecke and L. Martens, “Proposed Architecture and Algorithm for Personalized Advertising on iDTV and Mobile Devices” *IEEE Transactions on Consumer Electronics,* vol. 54, no. 2, pp. 707-713, 2008.

[35] L. Paolino, M. Sebillo, G. Tortora, A. Martellone, D. Tacconi, and G. Vitiello, “Dynamic User Modeling for Personalized Advertisement Delivery on Mobile Devices”, *Lecture Notes in Business Information Processing (LNBIP),* vol. 20, pp. 508-513, 2009.

[36] X. Wang, M. Yu, L. Zhang, R. Cai, and W. Ma, “Argo: Intelligent advertising by mining a user's interest from his photo collections,” i*n Proc. 3rd international Workshop on Data Mining and Audience Intelligence for Advertising*, 2009, pp. 18-26.

[37] S. Moursi, “Agent Oriented Smart Multimeida and Its Application on Personalized Advertisement”, MESc. thesis, Dept. of ECE, University of Western Ontario, London, ON, Canada, 2011.

[38] M. Elsakhawy, “Agent Oriented Intelligent Multimedia”, MESc. thesis, Dept. of ECE, University of Western Ontario, London, ON, Canada, 2011.

[39] S. Moursi, M. Elsakhawy and H. Ghenniwa, “Agent Oriented Media Recommender System Utilizing Smart Multimedia,” *in Proc.15th international Conference on Computer Supported Cooperative Work in Design*, 2011, pp. 437-444.

[40] E. Shakshuki, H. Ghenniwa and M. Kamel, “Agent-Based System Architecture for Dynamic and Open Environments,” *Journal of Information Technology and Decision Making*, vol.2 no. 1, pp. 105-133, 2003.

[41] M. Genesereth, and N. Nilsson, *Logical Foundation of Artificial Intelligence*, Los Altos, California, Morgan Kaufmann Publ., 1987.

[42] J. Sowa, *Information Processing in Mind and Machine*, Addison-Wesley Publ., 1984.

[43] S. Shepard, RFID, *Radio frequency Identification*, New York, McGraw Hill, 2005.

[44] Kinect, *Microsoft*, [online], Available: http://www.xbox.com/en-GB/kinect/

[45] Java Agent Development Framework (JADE), [online] Available: http://jade.tilab.com/

[46] H. Ghenniwa and M. Kamel, “Interaction devices for coordinating cooperative distributed systems,” *Automation and Soft Computing*, vol. 6, no. 2, pp.173-184, 2000.

[47] Android, *Google*, [online], Available: http://www.android.com/

[48] Android Software Development Tool (SDK), *Google*, [online], Available: http://developer.android.com/sdk/index.html

[49] F. Southey, and J. Linders, “Notio – A Java API for developing CG tools,” *in Proc. of 7th international Conf. on Conceptual Structures*, 1999, pp. 262-271.

[50] Binomial Distribution Implementation, *Apache*, [online], Available: http://commons.apache.org/math/api-1.2/org/apache/commons/math/distribution/BinomialDistributionImpl.html

[51] F. Gagliardi, JavaBayes - Bayesian Networks in Java, [online], Available: http://www.cs.cmu.edu/~javabayes/Home/

[52] J. Schweitzer. (June 7). Simple Bayesian Network Inference Using Netica and JavaBayes. *DataWorks* [online]. Available: http://www.dataworks-inc.com/site/blog/simple-bayesian-network-inference-using-netica-and-javabayes

Curriculum Vitae

Name:	Ali Hussain
Post-secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2004-2008 BESc.
	The University of Western Ontario London, Ontario, Canada 2011-2013 MESc.
Related Work Experience	Research and Teaching Assistant The University of Western Ontario 2011-2012
	Software Engineer and System Administrator Abu Dhabi Distribution Company Abu Dhabi, United Arab Emirates 2009-2010