Abstract: This paper presents a generic Semantic Web Service Framework (SWSF) as an agent-based Integrated Development Environment for Web Service activities. This framework can facilitate automatic web service delivery, discovery, selection, invoking, composition and interoperation. Current solutions like OWL-S and WSMO do not satisfy all basic requirements of SWSF. Our proposed framework has more extensible and autonomous distributed architecture, and it has more complete and effective web service automation. The semantic facilities in our framework are designed at a conceptual level to guarantee correctness and avoid inconsistencies among its internal modules. This framework is based on a stack of ontologies to describe the different parts of a Semantic Web Service and it contains a set of logic rules to form more intended request from the submitted user request. The framework is based on two paradigms, agent- and service-oriented, in a way that capitalizes on their individual strengths. We address several critical issues, including the appropriate architectural framework and its main elements structure design, supporting technologies, and modeling ontologies. This framework has been developed and evaluated.

Key-Words: Semantic Web Services, Ontology, Software Agents, SWSF.

1 Introduction
Distributed environments have shown to be a feasible solution for interconnection, integration and access to larger Web resources. The services become one of the most important Web resources. A service is a function or some processing logic or business processing that is well-defined, self-contained, and does not depend on the context or state of other services. Web Services are applications that can be published, located, and invoked across the Internet [1,2]. These Web Services may use other Web Services in order to perform their task (composite service). Web Services as have formally defined by WebServices.org: “Web Services are encapsulated, loosely coupled, contracted functions offered via standard protocols”. “Encapsulated” means the implementation of the function is never seen from the outside. “Loosely coupled” means changing the implementation of one function does not require change of the invoking function. “Contracted” means there are publicly available descriptions of the function’s behavior, how to bind to the function as well as its input and output parameters. Web Services are platform-independent and language-independent since they use standard XML languages [3] and most Web Services use HTTP for transmitting messages.

The Semantic Web Service Framework (SWSF) [4] spans the full range of service-related concepts where it provides a model for describing the various aspects related to Web services. The ultimate goal of SWSF is the Web services activities automation where a soft agent should be able to locate, select, employ, compose, and monitor Web services without a human interruption. Web resources are accessed by content rather than just by keywords, so computer-interpretable descriptions of the service and the means by which they are accessed are needed. The Web services contents are described by means of a markup language. An important goal of Web markup languages, then, is to establish a framework within which these descriptions are made and shared. A standard ontology, consisting of a set of basic classes and properties, for declaring and describing services, should also be available. WSMO [5] and OWL-S [6] are the most common such ontologies. There are two types of Web services: atomic and composite services. Atomic service is a single service invoked...
by a request while composite service is composed of more than one atomic service.

This research presents a semantic web service framework (SWSF) enabling automation of semantic Web services activities such as automatic Web service discovery, invocation, and composition. The proposed SWSF is designed to support the two service categories and it considers the notion of software agents in a distributed environment, so it is an Agent-Based Semantic Web Services Framework for distributed applications (PASWSF). E-learning environment is considered as a case study. PASWSF is an intelligent multi-agent environment and it is a context-aware and an owls-based system. It consists of four main elements: ontologies that provide the terminology used by other elements, request that defines the problem that should be solved by Web services, Web services descriptions that define various aspects of a Web service and mediators which bypass interoperability problems. Its architecture exploits the User’s context to determine the User intentions. Based on the identified intentions of the User, PASWSF determines the most relevant Web services, and then it selects a service to be invoked. After that, it provides the User with the results of the invoked service.

Automatic Web service discovery is to automatically locate Web services that satisfy the functional and non-functional requirements of the user request. OWL-S computer-interpretable semantic markup, a service registry and ontology-enhanced search engine are used to locate the services automatically. A service provider advertises its services in OWL-S within a service registry. OWL-S enables semantically declarative advertisements of service properties, capabilities, and arguments that can be used for automatic service discovery. Automatic Web service invocation is the automatic invocation of a Web service by an agent, given only a semantic description of that service. A soft agent should be able to interpret the input/output semantics to select and invoke a service. OWL-S, in conjunction with domain ontologies specified also in OWL, provides standard means of specifying APIs for Web services that enable the automation. Web service composition may be automated or sub-automated [7] performing a complex task given a high-level description of a request. Sub-optimality means a human involvement in the composition process. Service composition descriptions and data flow interactions are also given in OWL-S. The composer design is deferred to another paper.

The utility of Web services can be extended with autonomous control, re-activeness, and pro-activeness which are essential characteristics of agents. Service providing agents can deliberatively and proactively respond to changes in the application environment and this leads to leverage personalization and effective provisioning to the human user. Web services remain passive until invoked, whereas rational agents are not. In particular, Web services are not supposed to take any kind of initiative to deliberately deviate from their hard coded application-centric functionality, neither individually, nor in joint collaboration with other services-whereas agents can do. Autonomous agents are capable of pro-actively searching for, composing and negotiating services on behalf of its user, individually or in cooperation with other agents. Web services are supposed neither to negotiate their usage terms with potential customers, nor to flexibly adapt to stochastic changes in the environment.

A soft personal agent [8] adapts its actions to be based on the perceptual personal information. BDI (Belief-Desire-Intentions) agents as personal agents [9] receive continuous perceptual input and then take actions to affect their environment. The actions are also affected by their internal mental state. Beliefs, desires, and intentions are the three primary mental attitudes which capture the informational, motivational, and decision modules of an agent, respectively. In other words, BDI agent depends on sensed information as its inputs (beliefs), predefined knowledge base for the committing process (intentions), and User desires as its goals. BDI agents perform reasoning utilizing the existing Knowledge Base with the sensed User information to commit the User desires. The committed User desires may affect the user intentions. The user sensed information (including the user profile and the user contexts) are considered the BDI’s beliefs. Our BDI-agent provides logic rules inferring from the history events of the User to determine the User intentions. The intention is modeled as a composite concept indicating how much the agent is committed to the submitted choice.
This paper proposes a three-tier system architecture consisting of User Agent (UA) as front tier, Broker Agent as a Middle tier, and Semantic Web Services (SWSs) as a backend tier. The proposed architecture aims to bridge the gap between the SWS based systems and the personalization systems. The remaining of the paper is structured as it follows. Section 2 surveys other related SWSFs. Section 3 presents the proposed SWSF architecture and its core elements. Section 4 discusses implementation issues. Finally, section 5 summarizes our conclusions and outlines future works.

2 Related Works
In this section, related systems are illustrated and evaluated as agent-based personalization of the SWS.

2.1 Need-Aware Multi-agent System
Need-Aware Multi-agent “NAMA” interactively employs User internal and external context and SWS to identify User needs [10]. External context means a social or a physical environment that can have an impact on certain behavior while internal context means psychological context that does not needs external sensors. Time, Identity, Location, and Entity (TILE) represent the external context entities. Three ontologies are supplied: User profile, User preferences and a to-do list of activities without an upper ontology to describe the relationships between these three ontologies. The user preference concepts are defined in the system and the User desires are based on the to-do-list ontology. NAMA has no learning mechanism in its intention generation from the context and it depends only on the matching mechanism for identifying these intentions. In other words, there are no historical events of the, so AI mechanisms to learn user intentions cannot be applied. NAMA system is mainly composed of user agent and NAMA agent where the user agent is able to send the user resources to the NAMA agent and to get the results back from the NAMA agent. The NAMA agent manages the user context.

2.2 A Multi-agent Based Intelligent WWW Interfacer (MAII)
Most Web pages are syntactically described in HTML, so traditional search engines are keyword based. Without semantic description the user intension cannot be considered. This system consists of multiple domain specific self adaptive agents where each is concerned with a specific domain and a personal agent which is concerned of the user profile. Each agent has some parameters to be set according to the user intention or preference reflecting an intelligent and adaptive behavior. Interagent communications facilitate the information retrieval [11].

Through its associated character interface, the system adapts the user intentions. The user may modify an assumed intention by the agent and the agent updates its information about the user. In other words, the agent may learn from the user history actions and consequently the agent assumes the future user intentions more accurately.

The user submits domain specific keywords. Depending on the domain, an appropriate agent returns a set of WWW pages where subset of them is relevant. The most frequent keywords that appear in this subset are selected (keyword spice candidates). The appropriateness ratio A(c) of each keyword is computed as $A(c) = \frac{\text{the number of pages in relevant subset which contain the keyword}}{\text{the number of pages in whole set which contain the keyword}}$. A candidate with the largest $A(c)$ can be a keyword spice for the search.

2.3 Personalized semantic search engine (PSSE)
Personalized Semantic Search Engine (PSSE) [12] consists of multiple crawler-based search engines and document annotation agents working in parallel. The Web pages are clustered and ranked according to a computed score which is based on traditional link analysis, content analysis and the user profile. The process of PSSE is separated into offline and online parts. The offline part includes crawling and preprocessing processes. The online phase includes query processing and result ranking.

The PSSE uses Multi-crawlers (web spiders) that traverse World Wide Web, collect web resources and store them in a database. The preprocessor is used to maintain resources that are downloaded from Web sites which are graphed, then clustered and annotated. The main task of Indexer and link analyzer is to cluster the crawled web documents. A graph of the crawled pages is built and then authoritativeness of web pages is computed. The graph is clustered and these clusters are annotated by annotation agents. The weight annotations of the
clusters determine their relevancy to web resource using Term Importance Evaluator component. The user submits a query through a Google like search interface to be passed to the Query Analyzer component (QA). The QA makes use of an associated ontology performs mapping of query terms as well as query expansion. The user query is logged. The retrieved results are ranked based on personalization, query/document relevance, and authoritativeness factors.

2.4 Case-based reasoning technique for Arabic QA (CBRA)

El-Bayane is a case-based reasoning system conceived to give Fatawa for new situations, by using the Fatawa of past situations and not from scratch. It reuses the Fatawa, by imitating an imam. In order to be able to reuse past Fatawa, the system organizes its knowledge in cases which are collected in a memory, called a case-base. Then, an inference agent performs a reasoning process to find and reuse the appropriate Fatawa and its argumentation [13].

Case-Based Reasoning (CBR) is an approach to develop knowledge-based systems that are able to retrieve and reuse solutions that have worked for similar situations in the past. It has been argued that case-based reasoning is not only a powerful method for computer reasoning, but also a pervasive behavior in everyday human problem solving. Case-based reasoning has been formalized for purposes of computer reasoning as a four-step process: Retrieve, Reuse, Revise, and Retain. Given a target problem, relevant cases are retrieved from memory. A case consists of a problem, its solution, and, typically, the solution justifications (annotations about how the solution was derived). Map the solution from the previous case to the target problem. This may involve adapting the solution as needed to fit the new situation. Having mapped the previous solution to the target situation, the new solution is evaluated and revised if necessary. After the solution has been successfully revised to the target problem, it will be added to the knowledge base.

In the context of El-Bayane, the case contains the situation description (which is mainly a question and its contexts) and the Fatawa (judgment and justification). The system cases are stored in a case memory organized hierarchically in order to speed the extraction process and they are limited to the domain of drinking and smoking in Islamic legislation. The system offers strong guidance limiting the request expression. The system architecture consists of Indexing, Extraction, Adaptation, Validation, Storage modules. The Indexing module determines what case descriptors may be subsequently reusable and become indexes of the case. The Extraction module uses the indexes to retrieve the closest case to the input. The Adaptation module adapts the retrieved relevant case(s) if exceptions between old and new situations occur. The Validation is manually handled by a domain expert and the new solved case is stored after its validation.

2.5 Cross language IR Based on an Arabic Ontology in the Legal Domain (CLIR)

S. Zaidi et al. [14] proposed an approach to improve recall and precision of Arabic information retrieval in the legal domain thus minimizing the level of noise in the results. A query may be expanded using an Arabic ontology in the legal domain where the system navigates through ontology, looking for each term of the query; adding synonyms, relative derivatives and generic or specific concepts to the initial query for extending it. Each concept in the ontology associates its synonyms together with a restriction of derivatives set, which are the most strongly related to the legal domain. In addition to finding Arabic documents, the information retrieval process is enriched by enabling the user to retrieve English or French documents too.

The system architecture is composed of Arabic search engine, User interface, Query analyzer, Query expander, and Query translator. The Arabic search engine looks for documents written in Arabic, French or English. In the search process the new extended query is submitted to a search tool (or any search engine that supports Arabic query) which returns a list of documents classified according to their relevance algorithm. The preliminary results are quite promising and show that there is a significant improvement in the recall and precision. A simple Direct Manipulation Metaphor is implemented in the user interface where a query is formulated using a set of keywords introduced by the user. In Query analysis and normalization process stop words are removed and Arabic words are properly normalized by removing diacritics and replacing some characters. The Query expander expands the initial query by adding synonyms, relative derivatives and
generic or specific concepts to its terms. The optional Query translation step obtains French or English documents.

2.6 The WSMO Concepts
The Web Service Modeling Ontology (WSMO) is a specification to describe the various aspects related to Semantic Web Services (see http://www.wsmo.org). It is presented in WSML which is a language for formalizing Web Service descriptions. WSMO lays its foundation on knowledge representation and logical reasoning and it benefits from experiences gained in developing UPML and several case studies in different application domains. The main components of WSMO specifications are Goals, Web Services, Ontologies and Mediators, described in the following:

- WSMO describes Web Services by their Capabilities and Interfaces. The Interface description contains two closely related notions of Choreography and Orchestration. Choreography defines the information required to interact with a Web Service and Orchestration defines the information required to describe the composition of a Web Service.
- Goals represent the types of objectives to be achieved via Web Services. The goal definition describes a request for a service by means of defining the state of the desired information space and the desired state of the world after the execution of the required service.
- Ontologies provide the definition of the concepts and relations used in the other three component descriptions.
- Mediators specify interoperability mechanisms between all other components.

WSMO is a recent initiative in comparison to the OWL-S specification, consequently, still under development and subject to modification. It is worth mentioning that WSMO and OWL-S have different technical coverage. For example, the WSMO initiative presents specifications for architecture and execution environment as well as a language for ontology definitions, while these lay outside the scope of the OWL-S initiative.

2.7 The OWL-S
OWL-S consists of a set of ontologies designed for describing and reasoning over service descriptions. It consists of three main upper ontologies: the Profile, Process and Grounding Models. The Profile is used to describe services for the purposes of discovery. Service (and request) description is constructed from a description of functional properties (IOPes) and non-functional properties (human oriented properties) such as service name and other meta data parameters such as cost, reliability and quality. OWL-S Process models describe the composition or orchestration of a service in terms of its constituent processes. OWL-S Processes are classified into three subclasses: AtomicProcess, CompositeProcess and SimpleProcess. An atomic process is a stateless service that receives a set of inputs to commence its execution and produces a set of outputs after it is executed. A composite process is a stateful service composed of atomic processes. In other words, it may accept inputs and produce output messages at different stages during its execution. The SimpleProcess is not directly executable. It is an abstract description of a process that can be realized and performed by any of the other two types of processes. Assignment of a Simple Process to a concrete (executable) processes at run time is a discovery process which can be guided by exploring the realizedBy and expandsTo links.

A workflow expressing the composition of atomic services can be defined in the service ontology by using appropriate control constructs such as Sequence, Unordered,Choice, If-then-else, Iterate, Repeat-until, Repeat-while, Split, and Split+join.

The grounding facilitates the service execution. The process model is mapped to a WSDL description of the service, through a thin grounding. Each atomic process is mapped to a WSDL operation, and the OWL-S properties used to represent inputs and outputs are grounded in terms of XML data types. Additional properties pertaining to the binding of the service are also provided (i.e. the IP address of the machine hosting the service, and the ports used to expose the service).

The service.owl exists only to describe the semantic relationships among the other three ontologies and it is not used to annotate a semantic web, only subontologies are used for this purpose. The underlying ontology language OWL provides constructs and features to create an ontology document, while OWL-S, on the other hand, is just
one such ontology created by using OWL. The relations between OWL-S and the Web service activities are shown in Table 1.

<table>
<thead>
<tr>
<th>Objectives</th>
<th>OWL-S Subontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic discovery of Web services</td>
<td>Profile.owl provides terms that can be used to describe what the service does, so this can be used to find the requested service</td>
</tr>
<tr>
<td>Automatic invocation of Web services</td>
<td>Grounding.owl provides terms that can be used to describe how to access the service, so this can be used to dynamically invoke a given service</td>
</tr>
<tr>
<td>Automatic composition and monitoring of Web services</td>
<td>Process.owl provides terms that can be used to describe how the service be used, so this information can be used to compose service descriptions from multiple services to accomplish a specific task. This information can also be used to monitor the execution of a given service</td>
</tr>
</tbody>
</table>

2.8 Malik [27]
A semantic Architecture for Islamic Fatwa System (Malik) was proposed based on set of cooperative agents. The proposed system depends on different types of Ontologies to classify Islamic Fatwa according to a predefined hierarchy that is built by an expert Sheikh. The proposed model effectively helps users to search relevant documents for a query. Semantic descriptions are injected into both user question and Fatwa database. An ontology-based matching algorithm is used to fetch semantically matched Fatwas. Each returned Fatwa is associated with a matching degree. The model is composed of five tiers, with each tier responsible for specific task. The model uses different types of agents, each with its own characteristics.

The user interface is a natural language interface allowing user to enter free Arabic text supported by an Arabic Word Net.

3. The PASWSF Architecture
The proposed framework is a three tier architecture consisting of the user agency (UA), broker middleware agency and SWS agency (repository tier). It adapts an agent-based approach. Agents are organized into categories (or agencies) where each category belongs to a tier: the user agency, the middleware agency, SWS agency. Agents play different roles such as agents’ aggregation, the activities of other agents’ coordination, work specifications, and support to other agents. The middleware agency receives the task specification and plan from the User Agency, and search for appropriate Internet-based Web services that can accomplish the tasks. Each agent may have its own profile which contains the responsibilities of the agent within a given context. The profiles of the various agents may be embedded throughout the system. The mapping to agent profiles identifies the points where agents are specified. Agents may interact in internal and external manners. The internal interactions are accomplished between agents within the same tier (intra-tier agents), while the external interactions are accomplished between inter-tier agents and between the agents and its external environment.

The broker coordinates the interactions between UAs and the SWSs. Each user has its own environment where it is aware of the user environment which includes the user profile, preferences, context, and the stored historical user events. The broker is the core of PASWSF and its matchmaker (MM) is an owls-based matcher. The broker authenticates the user, sends advertising data to UA, exchanges data with UA (authentication, user goals, relevant services), mediates mismatching, and matches goal with services. Fig.1 illustrates the internal structure of each tier and the inter-tier relationships.

The framework considers a belief-desire-intention (BDI) model. Beliefs correspond to the background knowledge of the user which may be updated by the user. Desires express the user goals (the user requests). The intentions are based on the user historical events and the user ontology which includes the user profile, preferences, and context. The user request may be reformulated based on the user beliefs and intensions. Protocol and data mediation [15,16] may be required to support the communication with the user and the SWS tier.

PASWSF provides a model for describing the various aspects related to Web services. Its main goal is to fully enable e-application (e-commerce, e-government, e-learning, e-health) by applying Semantic Web technology to Web services. It is centered on two complementary principles: a strong de-coupling of the various modules that realize an e-application; and a strong mediation service enabling
Web services to communicate in a scalable manner. Mediation is applied at several levels: mediation of data structures; mediation of business logics; mediation of message exchange protocols; and mediation of dynamic service invocation. The mediated interactions may be categorized into OO mediators mapping an ontology to another ontology, WW mediators mapping web service to another web service, WG mediators mapping web services to goals, and finally, GG mediators mapping goal to another goal.

PASWSF consists of four main elements: ontologies that provide the terminology used by other elements, goals that define the problems that should be solved by Web services; Web service descriptions that define various aspects of a Web service, and mediators that bypass interoperability problems.

3.1. The User Services Tier (The User Agency)
The main task of the User Tier is to specify a request (goal) and the server returns a list of relevant services to be ranked according to the preferences of the client. The preferences of the client are reflected into the user ontology. The User ontology includes the User profile, preference, and current context (such as the user locations and timings). The User tier enables the user to edit her request, update the associated user information such as the user ontology, historical events, Semantic Web Services (SWSs) and to interact with the broker agent. Such interactions include login and logout.

There is an associated Intelligent Database (IDB) with the tier. The IDB database relates each user with her historical events. The database includes a record for each request consisting of user ID (UID), request, location, timing, language, domain, sub-domains, and invoked services. The IDB database schema is illustrated in Fig.2. It also includes rules as a higher intelligent layer which constitutes the user intensions. The user may update the capabilities and domains of the published web services which belong to this user (Form-S). The user is also allowed to update her ontology instance (Form-O) and to access her associated IDB database entries (Form-D).

UA sends a user request to the broker which in turn invokes its MM to look for services satisfying all the user requirements. These requirements are functional properties (IOPE), non-functional properties (such as response time, cost and reliability), and User BDI. Assume there are two services in e-learning domain satisfying IOPE requirements: the serviceSessionInfo(subject) which takes a subject title as an Input and returns this subject course information. The course information (based on course ontology) includes the course session schedule such as instructor name, dates, timings, locations, and list of registered students. The serviceContentInfo(subject) which takes a subject title as an Input and returns course information. The course information (based on course ontology) includes the course contents. Both services may match the user request IOPE. Knowing the user BDI, one of the two services may be matched with higher matching degree than the other. For example, by knowing the user beliefs and intensions, we may figure out that the user is only interested in educational management process, so serviceSessionInfo may be more relevant. If the user beliefs and intensions are around the learning process, second service may be more relevant.

Based on BDI, a service relevancy (and so its ranking) may be affected by the user location, timing, and preferences. A service which returns learning material in Arabic language may be more preferred to the user who has Arabic as a native language. The services which return learning material in certain language, certain shape, authored by certain instructor, and/or published in certain period may be the most relevant. This tier consists of five modules: User Client “UC” module, Publisher Client (PC) module as a Service Annotation manager Interface, authentication module (AUTH), Intended Request module (IR) and Client Interactions Maker module (CIM). Fig.1c shows the User Agent architecture. The UC has a GUI interface which allows the user to edit her ontology instance, goal, beliefs. The composite request can be described as a business process graphically. While modeling a business process, she can search for services, browse the results and incorporate the appropriate services for the business process. The composite request will be deferred to a next search paper. The PC allows managing the repository where it allows editing or deleting already existing service descriptions, or insert service descriptions into SWS repository. It allows the user to semantically annotate a new service and then calls the publisher module of the broker, so the service is added to the SWS repository. The PC may be restricted to the system administrator which is authenticated by “sa” as user ID. The user may be authenticated using the AUTH module. All the interactions between UA and the broker are done
through the CIM module. It enables the User to start/end the interaction with the broker, send the goal and other information, and get the relevant services which match the sent goal considering the current context and preferences. The interaction between the CIM module and the broker layer is done by using messages which pass User requests to the broker and get its responses (deterministic communications). When the User triggers the Start() message, the CIM module invokes the broker agent with the User-id “UID” data to login into the system. The UID is validated. The End() message is used to end the session and disable all the other messages.

```
1. Receive a Request from the user.
2. Semantically Search into the IDB for the same Request. If found, skip to 7.
3. Consult the IDB database to retrieve a history of the user.
4. Consult the current user belief to retrieve related profile, preferences, and context.
5. Collect these user data (current and historical) in the form: attribute name and value/certainty factor (attribute = value/CF).
6. Apply the rules to infer the intended Request with a Request certainty factor (RCF).
7. The Request and its intended Request (with RCF > threshold value) is browsed to the user to be approved or to be modified.
8. The Request and its intended Request are saved into the IDB.
9. The intended Request is passed to the broker agent.
```

A Request may have more than one intended Request with its own RCF. In this case all are browsed to the user to choose one or the highest RCF Request is browsed. This option as one of the system parameters may be set by the user at the beginning of the session.

### 3.1.1 The User Ontology

Ontologies describe basic concepts and its relationships in a domain. Basic building blocks of an ontology include classes (concepts), properties of each concept describing various attributes of the concept (slots), and restrictions on slots (facets). The ontology together with a set of individual instances of classes constitutes a knowledge base. The ontology provides a common vocabulary for all parties in its domain, so it helps to share common understanding of the structure of information among
people or software agents, and to enable reuse and analyze the domain knowledge.

This user ontology has been adapted from SEE Ont which is e-learner ontology [18]. It holds concepts and attributes that are needed for the e-learning network. In addition to that it holds social concepts such as those exist in the social networks’ ontologies. It is consisted of twelve categories that are based mainly on the needs of the e-learning systems. The user ontology main concepts are summarized in Table 2.

Table 2: The User Ontology Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity information</td>
<td>It describes information that identifies the user such as age, name,</td>
</tr>
<tr>
<td></td>
<td>gender, mail address.</td>
</tr>
<tr>
<td>Health status and emotion</td>
<td>It describes the health status of the user. The user could be hearing</td>
</tr>
<tr>
<td></td>
<td>impaired, visually impaired or had any physical disabilities and</td>
</tr>
<tr>
<td></td>
<td>problems. User emotions can be recognized by analyzing her facial</td>
</tr>
<tr>
<td></td>
<td>expressions, voice, or observable behavior but, the physiological</td>
</tr>
<tr>
<td></td>
<td>sensors may be used as auxiliary mechanisms to infer emotions more</td>
</tr>
<tr>
<td></td>
<td>accurately. Some examples of evidence detected by physiological</td>
</tr>
<tr>
<td></td>
<td>sensors are skin conductivity and heart beat.</td>
</tr>
<tr>
<td>Relationship</td>
<td>It describes the relationships of the user with other people in the</td>
</tr>
<tr>
<td></td>
<td>same social network.</td>
</tr>
<tr>
<td>Surrounding Environment and</td>
<td>It describes the characteristics of the country where the user lives.</td>
</tr>
<tr>
<td>Residential constraints</td>
<td>Everyone is affected by the culture, the rules and the policies of the</td>
</tr>
<tr>
<td></td>
<td>country.</td>
</tr>
<tr>
<td>Goals and Objectives</td>
<td>It describes the aims, goals and objectives of the user.</td>
</tr>
<tr>
<td>Qualifications and Knowledge</td>
<td>It describes the qualifications, certificates or licenses obtained</td>
</tr>
<tr>
<td>Background</td>
<td>along the user life such as the educational history, training and</td>
</tr>
<tr>
<td></td>
<td>skills.</td>
</tr>
<tr>
<td>Preferences</td>
<td>It describes the preferences of the user. It is related to the things</td>
</tr>
<tr>
<td></td>
<td>that the learner will be comfortable if it is realized. This may</td>
</tr>
<tr>
<td></td>
<td>prioritize and filter the result of the search such language</td>
</tr>
<tr>
<td></td>
<td>and activities.</td>
</tr>
<tr>
<td>Interests</td>
<td>It describes the social activities and hobbies that the user performs.</td>
</tr>
<tr>
<td></td>
<td>It may include social games, sports, reading books and novels.</td>
</tr>
<tr>
<td>Affiliation</td>
<td>It describes the memberships of the user to institutions, universities</td>
</tr>
<tr>
<td></td>
<td>and any other groups/organizations.</td>
</tr>
<tr>
<td>Security</td>
<td>It describes any security settings for the user. It may be security</td>
</tr>
<tr>
<td></td>
<td>features and credentials.</td>
</tr>
<tr>
<td>Learning styles</td>
<td>It describes the learning style that is suitable for the user. There</td>
</tr>
<tr>
<td></td>
<td>are three primary learning styles (methods) which are visual, auditory</td>
</tr>
<tr>
<td></td>
<td>and tactile. Through identifying the learning style, it will be able</td>
</tr>
<tr>
<td></td>
<td>to capitalize on his/her strengths and improve his/her skills.</td>
</tr>
<tr>
<td>Temporal Conditions</td>
<td>It describes the time factor that affects the decisions of the user.</td>
</tr>
<tr>
<td></td>
<td>The user preferences and schedules are affected by the time.</td>
</tr>
</tbody>
</table>

3.2 The Broker Services Tier (The Middleware Agency)

The main goal of this tier is to coordinate the interactions between the other two tiers. The Broker as a middleware is composed of modules that perform mainly publishing, discovery and mediation. Its main modules are matchmaking (MM), Publisher (PUB), mediator, Broker Results Handler (BRH), SWS Interaction module (SWS-IM), Matching utilities (MU), Ontology Manager (OM), Composing Manager (CM), Negotiator (NEG) and Reasoner. While the role of the matchmaking agent is to determine which services match the request, the mediator mediates mismatching between different parts. The broker provides synchronization mechanisms among autonomous agents. An analysis of the requirements of a Broker that performs both discovery and mediation between agents and Web services is presented in [19,20]. The reasoner interprets the capability advertisements, the request, how the intended request is composed, how the services are chosen, how the matching degrees are computed, and the invoked service effects. It also reasons about ontologies. The reasoner provides a method for checking whether a service has a certain formula model. Only if it has, the set of preconditions and assumptions is relevant. The reasoner is basically a logic reasoner where its methods are provided as Web services to be consistently located by the available discovery algorithm.
The broker agent may interact with the chosen provider as necessary to fulfill the Request where once the provider is discovered; the Broker may negotiate the provider for some nonfunctional properties. The ontologies are used to semantically interpret the request, the Web services IOPE, their interaction patterns and the domain on which they operate. The broker protocol can be divided in two categories: the advertisement protocol, and the mediation protocol. In the advertisement protocol, the Broker first collects the advertisements of available Web services. The mediation protocol provides the mediation process. The requester sends the intended request to the Broker and it waits for a reply while the Broker uses its discovery capabilities to locate a provider that can answer the Request. Upon receiving the Request, the provider returns the reply to the Broker and finally the Broker replies to the requester.

The Publisher (PUB) as a Service Description Manager provides methods for creating, editing and deleting service descriptions. Thru this module, the Publisher Client (PC) of the User client is able to upload, delete and edit repository entries and to browse the repository contents. The BRH module filters the returned results. The “BRH” module handles the retrieved SWSs returned from the matchmaker, to be more readable and usable by the User agent. To do so, it applies some filters to extract only the useful fields, extracts the data for each field in the User result record, reconstructs these data into XML-string, and sends it back to the UA to be stored into the IDB. The SWS-IM module facilitates the communications with the third tier repository. The SWS Interaction Maker “SWS-IM” module has the basic methods that can be used to interact with the Matching Utilities (MU). MU is a web service that composes a set of functions which are considered as extensions of the matchmaker. The MU manipulates the matchmaker resources such as the published OWLS Services, domain ontologies in OWL, and queries in OWLS. The MU provides some functional extensions of the SWS matchmaker. The OM contains methods for creating and managing ontologies. While the domain ontologies and mappings among them can be directly managed within the OM, the CM is created to manage the behavior (choreography and orchestration) of a service as ontology instances. The NEG is an agent-based web service negotiation facilitating the terms negotiation between the service requester and the service provider. Self-interested requester and provider agents can (semi-) automatically negotiate the terms and conditions of using individual or compound candidate services. The candidate has been selected according to their semantic relevance to the given query (service selection as part of service discovery). There is a wide range of negotiation protocols (mechanisms) that can be used by agents for service negotiation such as voting, contract nets, auctions, bargaining, general equilibrium market mechanisms, and coalition forming. In any case, all agents have to agree in advance on the used negotiation protocol, procedure of contracting and its scope of enforcement including terms and condition of penalty payments, the general payment scheme, xml message templates and vocabulary (XML namespaces) for (semantically) interoperable interaction. After reaching a searching a service level agreement (SLA) at the end of the negotiation phase, the SLA gets transformed into a legally binding contract which has to be signed by both service providers and requesters (contracting). This contracting phase is separated from but often considered being part of service negotiation.

The queries of the matchmaker should be in OWLS format having the requested WS Inputs and the requested WS Outputs, and optional parameters such as Preconditions, Effects, and other non-functional parameters. Messages are passed between the UA and the Broker such as:

- Login (UID)
- GetRepositoryResources(UID)
- SendUserResources(UID, Profile, Desire, Context)
- GetUserResults(UID)
- SendIntendedRequest(UID)
- SendSWSDescription(UID, ServiceDescriptions)
- GetReasoning (UID, features) messages.

To process these messages, the Broker defines many actions such as Validate/Identify User, Return all the published services belong to the user, receive the user resources, return the matched services, receive the intended request, Receive SWS descriptions to update the SWS repository, and Return the reasoning report about the supplied features on witch reasoning process is applied.
3.3. The Semantic Web Service Tier (The Services Agency)

The tier offers multiple domain specific self adaptive crawling agents where each agent is concerned with a specific domain and it has some parameters to be set according to the user preference as shown in Fig.1c. The user submits domain specific keywords. Depending on the domain, an appropriate agent returns a set of relevant Web pages to the requester and to be stored in the associated IDB.

This tier acts as a repository of semantic service descriptions, domain ontologies and ontologies mappings. The Crawling agent extracts the SWSs Domains and Concepts by navigating all the published OWLS SWSs using one of the OWLS parsers and searches the Internet for semantic Web services. It collects the Inputs and Outputs concepts and their related Ontologies, into an XML string. Repeated concepts are removed.

The WSDL guarantees syntactic interoperability, but it fails to provide semantic interoperability. The semantic interoperability is crucial for Web services to allow composition of Web services to achieve a more complex service and to know preconditions that are required to use the service and effects of having invoked the service. Several approaches do exist to provide explicit semantics that enable automatic Web Service location, composition and invocation.

At present, there are two major types of approaches to find service descriptions: search-oriented and storage-oriented. The search-oriented approach employs a crawler to collect description on the Web. The storage-oriented approach uses storages to store and organize the service descriptions published by service providers actively. The storage of service descriptions can be constructed as a registry, an index, or a peer-to-peer system. One of the most eminent business service registries is the Universal Description, Discovery, and Integration specification [http://www.uddi.org/specifications].

The repository is an XML-based registry (directory) of web service interfaces (described by WSDL) and it is specifically designed for discovering Web services. It cashes the Web services descriptions located by the crawler agent. This caching speeds up the Web services searching for later requests where the crawler searches the repository before surfing the Internet. This repository may be centralized or distributed. The repository is a platform-independent framework for describing, discovering, and integrating business services throughout the Internet.

To get access to the repository, it exposes a set of APIs in the form of a SOAP-based Web Service. The API is divided into two logical parts which are the Inquiry API and the Publishing API. The Inquiry API lets us search and browse services information. The Publishing API is used for the business registration through an XML file specifying Web Services. The specification defines a SOAP-based programming protocol based on HTTP for registering and discovering Web Services. Using the UDDI discovery services, businesses individually register information about the Web Services that they expose for use by other businesses. Our repository follows the UDDI XML schema.

3.4 Goals

A goal specifies objectives that a client may have when he consults a Web Service. A complete declarative description of the goal consists of elements: pre-conditions, post-conditions, the goal name, textual description, assumptions, effects, non-functional properties and imported ontologies. The goal name serves as an identifier, and the textual description provides a human-understandable description of the goal. Pre-conditions describe what a Web Service expects for enabling it to provide its service. Post-conditions are the complete information that the service provides to the requester after its execution. Assumptions describe conditions over the state of the world that must hold previous to the service request. They express conditions over information that is not directly contained in the information given to the service. Notice that the border line between assumptions and pre-conditions is defined by the nature of the objects they refer to. Preconditions refer to information given directly by the requester, and assumptions refer to information not made directly available to the service. In both cases, the Web Service will fail to provide its service if the conditions do not hold. Effects describe conditions over the state of the world that holds after the service execution. They reflect the changes in the state of the world due to the service execution. The relation between effects and post-conditions is analogous to the relation between assumptions and pre-conditions. We distinguish a post-condition from an effect because the latter reflects a change in the state of the world. That implies that some effects cannot be invoked if a constraint over these effects is placed by the requester. We also separate pre-
conditions from assumptions because the conditions they reflect must be treated separately. At the discovery step, the requester can know if she can fulfill the pre-conditions of a given service by checking the available information. The truth value of the assumptions cannot be known without accessing a knowledge base or invoking a different service. Therefore, the requester can either assume that the pre-conditions hold or use appropriate sources to determine their truth value. Pre-conditions, post-conditions, assumptions and effects define all together the functionality of the service. Other relevant properties of the goal, such as security, time of response, keywords, or quality of service are defined using non-functional properties. The last element in a goal definition is the specification of the imported ontologies that provide the vocabulary used to define the goal.

3.5 Mediators
Mediators can link goals to Web Services or Web Services to Web Services. Mediation types include data mediation and process mediation. Web Service/Goal mediators relate the Web Service capability to the goal, expressing possible weakening of the Web Service functionality with respect to the functionality declared in the goal. The cases in which the service is usable may be less than the applicable scenarios for the goal and the Web Service may provide less information or world-altering results than the goal. Web Service/Web Service mediators link two Web Services, establishing the necessary data and process mediation necessary to enable the communication between the services. Enabling communication between heterogeneous services is essential in a domain and in other domains where different parties want to cooperate in a dynamic fashion without redefinition or re-implementation of their respective interfaces. Mediators import ontologies to be used as their vocabulary and to declare non-functional properties.

A mediator is given a name and a textual description, it has a declarative nature and it is done by an external agent. The definition of a mediator includes a source which is always a Web Service and a target which may be a Web Service or a goal. Therefore, an essential property of the mediator is the Web Service that actually performs the mediation between the source and target. This mediation can be limited e.g. if there is not exact matching between two ontologies, so a reduction is also declared, expressing the limitations of the mediation service.

4. Implementation and results
Islamic Fatawa is considered as a case study to implement and evaluate the system against other related systems such as NAMA, MAII, PSSE and CBRA. Islamic Fatawa is a formal legal opinion concerning Islamic affairs (usually response to a question). The Islamic Fatawa may be categorized according to its topic (Fatawa Domain) such as Praying, Fasting, Business Treatments, and other daily life matters. It is mainly issued by an Islamic legal authority like Dar al-Ifta al-Misriya [Web Site] where questions and their corresponding Fatawa responses are stored in a database. Each Fatawa has its issuing time and place and its provider (publisher) besides its contents (question and answer). The database may be searched syntactically which is based on submitted user keywords. To be accessed semantically, the questions and their corresponding answers have to be annotated. Five hundred Fatawas of a certain domain have been annotated in RDF.

The system is written in JAVA. The IDB schema is illustrated in Fig.2. It is implemented in MySql. The user agent has more one form. Form-R allows the user to edit a request. Form-O allows the user to modify an ontology instance. Form-D allows the user to access IDB. The IR module fires rules (stored in the IDB database) which are based on the user request, profile, preferences, context (from the user ontology instance), and history of the user actions (from the IDB database). These logic rules may be updated by the user. A sample of the rules are given below.

Rule 1: If the preferred language is Arabic and the number of Arabic language iterations is greater than a threshold value, Set Arabic language as a nonfunctional property.
Rule 2: If the timing is day, Set learning style equal to Visual.
Rule 3: If the number of a certain provider occurrences is greater than a threshold value, set the provider as the best one.
Rule 4: If the user is manager, the managerial services are preferred.
Rule 5: If the user is a learner, the contents retrieval services are preferred.
The proposed framework is compared to the other related systems mentioned in Section 2 as shown in Fig. 3. The comparison is based on different attributes such as development methodology, user interface, query format, data storage capability, availability of user intension and modularity. The system is tested for 100 requests and 500 published Fatawas fetched from Dar al-Ifta al-Misriyya database. The average response time of the IR module (request revising), average relevancy (precision), and average recall are shown in Table 4 against the other related systems. The comparison attributes are justified as it follows. The genericity attribute means relative independence from particular applications. The User Interface attribute defines the interface type such as GUI or text based interface. The development methodology defines the system ability to be deployed, updated, and adapted. Web based applications need to be installed only once and to be accessed anywhere while desktop applications need to be installed separately on each computer from which to be accessed. Updating desktop applications is more cumbersome than web applications. Only Web application can be accessed from mobile phones. On the other hand, web application development and its maintenance cost higher than desktop application, but it is more vulnerable to security risks. The Query format attribute defines weather the query format is restricted or free text. While the query intension generation increases the result precision, it exerts an overhead. The Data storage attribute defines the backend data storage parameters such as access time. Modularity defines the degree to which a system module may be updated without affecting other modules. Some systems are monolithic, while others are modular. Granularity attribute defines the extent to which a system is broken down into modules to be manageable. Search attribute means whether the search is semantic or syntactic. Domain attribute means whether the system is restricted to a specific domain, or it can switch from domain to another. Matching degree calculation defines weather the implemented matching algorithm computes the matching degree for each returned relevant service. SWS repository attribute defines the system supports SWS repository since the SWS repository reduces search time. Ontology attribute means whether the system considers ontologies or not. Ontology is an important component for any semantic environment where it provides a meaning for each concepts and relationships between the concepts. Word Net attribute defines weather the system consider an ontology for the supported language. The Word Net access time may affect the overall system response time. Implementation means how much effort will be taken to implement the system which depends on different feature such as modularity, granularity, and used tools.

Table 4: Average IR Response Time and Relevancy Evaluation

<table>
<thead>
<tr>
<th></th>
<th>NAMA</th>
<th>MAII</th>
<th>PSSE</th>
<th>CPRA</th>
<th>Malik</th>
<th>PASWSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.3</td>
<td>0.27</td>
<td>0.32</td>
<td>0.4</td>
<td>.28</td>
<td>0.21</td>
</tr>
<tr>
<td>Response Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>0.8</td>
<td>0.75</td>
<td>0.82</td>
<td>0.77</td>
<td>.93</td>
<td>0.94</td>
</tr>
<tr>
<td>Precision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>0.75</td>
<td>0.93</td>
<td>0.84</td>
<td>0.81</td>
<td>.93</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3: Comparison between different systems
5. Conclusions and Future work
A Multi-agent Personalization of the SWS framework is presented in this work consisting of three tiers: Users Agents Tier, Broker Agent Tier and SWS Tier. The internal structure of the framework, tiers and their interactions have been designed. Personalization adapts User-to-SWS interactions where the system output is based on the user goal (user desire), user belief, and user context. Consequently, this improves the search feasibility for the required web services. The core of the PASWSF is the broker which coordinates the interactions between the user and the SWS tiers. The user agent has the capability of identifying User intentions dynamically depending on the user context and the user historical events. A BDI approach is considered to generate the intention for a Request exploiting the user context. A OWLS-Request is built for each identified intention. A matchmaker is built to semantically match the generated OWLS Request with the published OWLS-based services. Parallel processing of multiple domain (sub-domain) dependent crawling agents reduces the overall response time. Considering resource content relevance as well as user preferences enhances search result and increases user satisfaction.

References:


