Design and Implementation of Intelligent Agent Shell

H. M. Mohamed¹, E. T. El Shenawy²*, N. M. Darwish³

¹Information Technology Department, National Research Center, El Behouth St., Dokki, Cairo, Egypt
²Solar Energy Department, National Research Center, El Behouth St., Dokki, Cairo, Egypt
³Computer Engineering Department, Faculty of Engineering, Cairo University, Cairo, Egypt

Abstract: It is of great interest to develop a framework containing a library of the main building blocks of common use to agent developers. This framework would save much of the routine work that is needed to build most types of intelligent agents. The present paper presents the design and the implementation of an agent development library. This library uses service oriented architecture, and exposes the functionality of the implemented blocks in the form of a set of XML web services. The use of XML web services makes agent development platform independent, and programming language independent.

The major block that is developed in this paper is the most common artificial neural networks structures. The developed system is used to the control the power generated from a photovoltaic module to deliver its maximum power as a test case. The system is also tested with simple clients of different types: web client, windows client, and mobile client.

Keywords: Agent shell, artificial neural networks, intelligent agent, maximum power, PV power.

Introduction

Intelligent agents are a new paradigm for developing software applications. An intelligent agent is software program that can perform specific tasks for a user and possesses a degree of intelligence that permits it to perform parts of its task autonomously and to interact with its environment in a useful manner [1].

The intelligent agent has many characteristics that make it different from traditional software programs. The characteristics of intelligent agents can be grouped into two large categories: external and internal properties. The external properties include all the characteristics that affect the agent interaction with other agents or human agent communications. The internal properties are those properties that form the internal being of an agent; internal properties determine the actions of the agent. The internal properties of the agent include: reactivity, pro-activity or goal orientation, reasoning or learning, autonomy, mobility, communication, and cooperation [2].

Agent Development Bottleneck

Expertise in designing and building agent applications is scarce. There is little in the way of production quality software support for building agent applications, and still less general understanding of the issues that need to be addressed when building such systems. Worse, most of today’s agent systems are built from scratch, using bespoke tools and techniques, which cannot easily be applied to other types of system. This is particularly worrying because a lot of infrastructure is required before the main components of an agent system can be built. At the moment, many developers rebuild this infrastructure from scratch in every new case; this process is clearly not sustainable. However, there is an increasing awareness that this situation is changing as agents are becoming of high importance for software development. Support is required throughout the agent system development process. It is of great interest to develop a library that can be used as ready made building blocks to be used by agent developers, to help them concentrate on the aim for which they are building their agent.

Model Architecture for the Intelligent Software Agent Shell

The agent shell system is designed using the service oriented architecture. Moreover, the architecture is modular in multi layer fashion. The layers of the proposed agent shell system architecture are shown in fig. 1. Those layers can be described as follows:

1. **Services Layer:** The services layer is the front end of the agent shell system. Agents interact with the agent shell system through the services layer. The services layer is composed from the following blocks:
2. **Business Façade:** This is the trust boundary that sits between middle tier business components and the service interfaces that call them. Moreover, it only applies in situations where there is two or more service interfaces associated with the middle tier. It provides a common interface for multiple service interfaces to interact with. For example, there is a web service that exposes the neural network, and another to expose only a part of the neural network e.g. summation functions, summation functions…etc. Those services expose the same business components but in a different view.

3. **Business Logic Layer:** This layer contains the business logic of the learning algorithms, and the interaction between the system components.

4. **Data Access Layer:** The system uses this layer to access the XML data representing the neural network topology, configurations, and training data.

- **Web Services Interfaces block:** This block contains the XML web services exposed to intelligent agents.

- **XML Schema Definitions (XSD) block:** contains the definitions of the standard data types used to interact with the system, described using the standard XML XSD. Additionally, it contains the schema used to define the network topology, configurations, and training data.

- **Contracts (WSDL):** The flavor of the service oriented architecture is that interaction between components and other system are based on contracts, not interfaces, nor classes that can be instantiated or inherited from specific language and platforms. Thus, contracts block contains the definitions of the methods, and classes that agents' use to interact with the agent shell system, described in the standard language WSDL [3].

Figure 1: Proposed architecture for the agent shell system.
5. **Data Source**: The system loads and saves data in the form of XML documents that follow certain XSD defined in the design of the system.

6. **Exception Handling**: This block interacts with all modules in the system to handle exceptional errors.

**Agent Shell System Component Diagram**

Fig. 2 shows the simplified component diagram for the Agent Shell System. The component diagram consists of the following components:

![Agent Shell System Component Diagram]
1. **Neural Network Component**: This component is a class library that encapsulates the neural network, and contains the main operations needed to create network, add new layer, and link all layers' neurons...etc.

2. **Network Configurations Component**: This library contains the business object that contains all run time settings needed to dynamically configure every thing of the neural network e.g. the type of summation function used, transfer function type, scaling function type...etc. This component is used by all other components in the system.

3. **Summation functions component**: The summation functions class library encapsulates the implemented summation functions in the agent shell system. The summation function is applied to the weighted inputs to the neuron. The implemented functions are Sum, Max, Min, And, Or, Average, Majority, and Product.

4. **Transfer function component**: This library contains the implementation of the transfer functions that are needed by most of the neural networks. The transfer function is applied to the output of the summation function after optionally adding temperature noise. These functions are Sigmoid, Linear, Sin, Hard Limiter, hyperbolic Tangent, and Ramping.

5. **Scaling functions component**: A scaling value is optionally applied to the output of the transfer function. Additionally, an offset value can be added to the scaled value.

6. **Limiting Functions Component**: The output of the scaling function may go out of the correct range for the output limit of the actual output, thus the limiting functions can be used to limit the output of the neurons of the neural network to be in the correct range.

7. **Output Functions Component**: This component contains the implementation for the output functions that are applied to the outputs of the neural network's neurons. Neurons' outputs can be directly passed to the next layer; however, a competition can be made between the outputs of the layer to pick the maximum, or minimum outputs to represent the output of the layer.

8. **Temperature functions component**: A random noise (uniform, exponential, Gaussian) is added to the output of the summation function.

9. **Error functions component**: Contains the implementation for a list of the most commonly used error functions. That can be used to calculate the difference between the desired output signal and the output signal of the output layer; the implemented error functions are: difference, square error, error cube, and error square root.

10. **Transfer derivative functions component**: This component contains the implementation for a list of the most commonly used transfer functions derivates. That can be used to scale the error value of each neuron; these functions are: sigmoid derivative, hyperbolic tangent derivative, ramping derivative, linear derivative and sine derivative.

11. **Output functions XML web service component**: This is an XML web service that exposes the output functions component to other platforms to be platform independent, and to be extensible by agent developers.

12. **Scaling functions XML web service component**: This XML web service wraps the scaling functions component and exposes its functionality to other platforms.

13. **Limiting functions XML web service component**: This XML web service exposes the limiting functions component to other platforms.

14. **Temperature functions XML web service component**: Temperature functions XML web service exposes the temperature functions component to other platforms to be platform independent.

15. **Transfer function derivatives XML web service component**: The transfer functions derivatives component is exposed to other platforms using the transfer function derivatives XML web service.

16. **Error functions XML web service component**: This XML web service exposes the error functions component to other platforms.

17. **Summation functions XML web service component**: The summation functions component is wrapped by this web service.

18. **Transfer functions XML web service component**: This XML web service exposes the transfer functions component to other platforms.

19. **Neural Network XML web service component**: The main XML web service in the agent shell system is the neural network XML web service. This web service manages xml files representing the network topology, configurations, and training data. Moreover, it contains the operations needed to manage network training and calculations of the output for any given input. This web service contains every thing that is needed to manage the neural network.

20. **Windows Testing Application**: This is a simple windows application that is used to test the neural network. By uploading the network topology xml file, training data xml file, network configurations xml and then training the network, and finally getting the calculated output of the given inputs.

21. **Web Testing Application**: This application is used to test the ability of the system to interact with different types of clients, web client in this case.

22. **Mobile Device Testing Application**: The agent shell system is designed to be platform independent; this mobile device client is deployed on a smart device, and uses the agent shell system. The use of this client proved that it is very simple to use the agent shell system from smart devices.

23. **Photovoltaic Controller**: As a practical evaluation for the developed agent shell system, the photovoltaic controller application is developed to control the power generated from a photovoltaic module. This controller is implemented in the form of a desktop application.
The Experimental Results

1. Agent Shell System's Neural Network as a Controller for the Photovoltaic System

As a case study, the photovoltaic system consists of a PV module driving an electric fan load via maximum power point tracker based neural network. The data used to train the neural network is calculated by the conventional calculations. Fig. 3 shows a comparison between the PV module maximum power calculated by conventional method and the trained neural network at surface temperatures 40 °C. The performance of the neural network records maximum error about 0.3%, which indicates the high accuracy of the neural network.

Fig. 4 plots the power delivered by the photovoltaic system to the dc load, and the maximum power calculated by the neural network versus time, in a summer day. While, fig. 5 shows the corresponding solar radiation and surface temperature for the same day. It is clear that the solar radiation is high almost all the day and ranges from about 900 W/m² to 100 W/m², while the module surface temperature reaches about 60 °C during the day, and decreases to be about 30 °C near sunset. The effect of the solar radiation on the power of the photovoltaic system can be seen in fig. 6, which plots the power delivered to the load from the photovoltaic module and the maximum power calculated by the neural network module of the agent shell system.

From The figures, it can be seen that the power delivered to the load is almost tracking the maximum power of the photovoltaic module at different operating radiation and temperatures. This indicates that the neural network module of the agent shell system is well trained and guarantees that the load is delivered the maximum possible power from the photovoltaic module.
2. Advantages of the Agent Shell System as a Controller for the Photovoltaic System

The major advantages of using the neural network module of the agent shell system to control the power of the photovoltaic system can be summarized as follows:

**a- Increased power utilization of the photovoltaic module:** when the load is directly connected to the photovoltaic module it operates at a certain voltage and current, regardless this is the optimal point or not. Whereas, when the neural network of the agent shell system is used, the load tracks the maximum power from the photovoltaic module as shown in fig. 4, thus it is clear that the power consumed by the load is increased.

**b- Reduced cost of the photovoltaic system:** the neural network of the agent shell system enhances the utilization of the photovoltaic module, which reduces the number of photovoltaic modules needed to drive a certain load. Thus, the cost is minimized.

**c- Maximized operating periods:** the load stops working when the feeding power falls below certain level about 3 watt as shown in fig. 4. This level is reached faster when the load is directly connected to the photovoltaic module, than when using the agent shell system's neural network. The later system tracks the maximum power of the of the photovoltaic module, thus,
the load power does not fall below this level as long as the maximum power of the photovoltaic module covers the minimum operating power level.

Agent Shell System in Miscellaneous Platforms

The agent shell system is designed to be platform independent. To demonstrate this feature; miscellaneous simple clients have been developed: windows client, web client, and mobile client. The following section describes the results of using the agent shell system with these types of clients and even with other types of clients.

Requirements to use the agent shell system from any client

All that is needed for any client to be able to use the agent shell system library is to support the following:

- Support the web standard http protocol to establish the connection to the agent shell system, the http protocol is responsible for requesting and transmitting data over the internet to and from the agent shell system [4].
- The WSDL protocol which represents the metadata, the client needs to interact with the agent shell system services [5].
- The client needs to support the SAOP protocol that defines the rules for message format of the transmitted data to and from the agent shell system.
- XSD needed to define the network topology, network configurations, and training data. The schema of those elements must be recognized by the client.

Form the above requirements it is clear that all the implemented clients needed nothing more than above set of simple web standard protocols to interact with the agent shell system, Which proves that the agent shell system is a true platform independent shell.

Advantages of the Developed Agent Shell System

The current agent shell system uses the service oriented architecture and implements its front end as a set of XML web services. This architecture has the following advantages:

- **Programming Language Independent**: Agent developers can use the agent shell system to build new agents using any programming language. Without any limitations or constrains on the programming language.
- **Platform Independent**: The agent shell system uses the web standard protocols (e.g. HTTP, XML, SOAP..etc.) to communicate with its clients. These protocols are worldwide standard protocols, easy to use and understand. The use of these protocols put no limitations on the operating system and the hardware).
- **Extensible Architecture**: The agent shell system can be extended from any platform. New blocks can be developed in the form of XML web services using any programming language, and then those services can be hosted on any platform. Thus these services are ready to be used by agent developers.
- **Firewall Friendly**: The agent shell system uses the standard HTTP ports by default, which requires no extra open ports in the system firewalls, thus minimizing the security threats.
- **No Data Marshalling**: Use of worldwide standard data type through the use of XSD, removes the need for marshalling, which causes delay and possibility of loss of data.

Conclusions

The intelligent agent shell system designs and implements the service oriented architecture. This architecture enables agent developers to develop their agents from any web enabled platforms, through the use of standard web protocols (HTTP, XML, SOAP, XSD, and WSDL). Thus the agent shell system is true platform independent, and programming language independent agent development tool.

The developed intelligent agent shell encapsulates the most common artificial neural networks structures, in a set of XML web services, which can be used by agent developers as ready made building blocks. These services are designed to have the following design objectives: programming language independent, platform (operating system and hardware) independent, extensible architecture, firewall friendly, and need no data marshalling.

To verify that the developed system satisfies the design objectives, it has been tested with many simple test clients and a photovoltaic system controller. The developed simple clients are: web client, windows client, and mobile client. The
photovoltaic system controller uses the neural network module of the agent shell system; to derive the photovoltaic system to deliver the maximum possible power to the load. The neural network used in this controller is constructed from three XML files: Network topology, network configurations, and training data. These files follow the XSD schemas defined by the contracts layer of the agent shell system. The network topology file defines the number of layers and the number of neurons in each layer and optionally provides initial weights for links of the network. The network configurations file defines the set of run time parameters of the network (e.g. type of summation function, noise function, transfer function, scaling function, limiting function, output function and error function), if any parameter is not supplied it is given a suitable default value. Moreover, a controller program is developed to use this neural network to control the output power of the photovoltaic module at run time.

The results of the training indicated that the system is well trained to track the maximum power of the photovoltaic module. The measurements recorded during running of the photovoltaic system gave the following enhancements in the photovoltaic system: power consumed by the load is maximized, the operating periods of the system is maximized, and the total cost of ownership of the photovoltaic system is reduced.

References