

Software Agent Architectures

Suresh Chande

chande@cs.helsinki.fi

Department of Computer Science,
University of Helsinki

Software Agent Architectures

"An Agent Architecture proposes a particular methodology for building an autonomous agent. It specifies how the overall problem can be decomposed into sub-problems, i.e. how the construction of the agent can be decomposed into the construction of a set of component modules and how these modules can be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions (effector's outputs) and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology"[1]

This article discusses about the architecture for Software agents as the main topic. We begin with a general discussion of the architecture for software agents that have been part of the efforts of the Agent research community past several years. We will then continue to discuss into the architectural approach that is required to be addressed in building a community of agent systems and the corresponding environment in which heterogeneous and multi-agents are to be hosted. We discuss these aspects by looking into the approach the Open Agent Architecture™ (OAA) has utilised. We finally conclude the article with a summary of the discussion and list the future directions of the agent based research efforts needs to address in-order to be deployed into the real world.

Introduction to Software Agent Architectures

There are different means of architecting an Agent based system. This architecture varies based on the approach we take up to vision the realization of an agent-based system. The traditional means is by viewing the agent-based system as knowledge based systems (Symbolic AI)[6]. Agent architectures are categorised into the following:

- Deliberative Agents
- Reactive Agents
- Hybrid Agents

The Agent architecture and correspondingly the implementation differ based on the type of the agent system we plan to realize. The above categories will be discussed below in more detail:

Deliberative Agent Architecture

This agent is based on physical symbol system hypothesis; this is based on the capability of representing the physically realizable world into physical entities, upon which processes can operate to perform a particular function. This architecture can be defined as explicitly represented, symbolic model of the world, and in

which decisions such as: what actions can be performed, are made via logical reasoning utilising pattern matching and symbolic manipulation [6].

Deliberation of such agent architecture usually leads to at-least two major problems:

1. The ability to translate the real world into symbolic representation, such that it can accurately represent the physical world, so that this representation can be utilised in a timely manner that it serves the purpose of the real world problems. Few of the efforts in this direction have been vision, speech understanding, and learning.
2. The ability to represent and reason about the complexities in the real world entities, specifically how to achieve reasoning and performing the corresponding action that needs to be taken up by an Agent within the relatively reasonable time utilising efficiently the available real world symbolic information. These have been previously addressed under the knowledge representation, automated reasoning, and automatic planning areas.

In fact none of these problems have been totally solved till date to a level that they can be utilised into real world solutions.

Few examples of the Deliberative Agents

Belief Desire and Intentional agents (BDI):

One example of the BDI based agents system is the Intelligent Resource-bounded Machine Architecture (IRMA). This agent contains the following key elements:

- Plan library and containing the beliefs, desires, and intentions.
- Reasoner: To reason about the real world
- Means-ends analyser: Determines which plans should be utilized in-order to achieve the agent's set intentions
- An opportunity analyser: Monitors the environment in order to determine further options for the agent, a filtering process and a deliberation process.

HOMER:

This is a simulated robot submarine hosted in a 2 Dimensional environment (Sea world). This agent takes its instructions from a human in a textual format and interprets from an 800 word based vocabulary by utilizing the linguistic capabilities. The agent plans how to achieve the human provided instructions and executes the plan accordingly in addition to handling the changes to the plan during the execution phase. This was developed by Vere and Bickmore and termed as HOMER[11, 6].

Planning Agents:

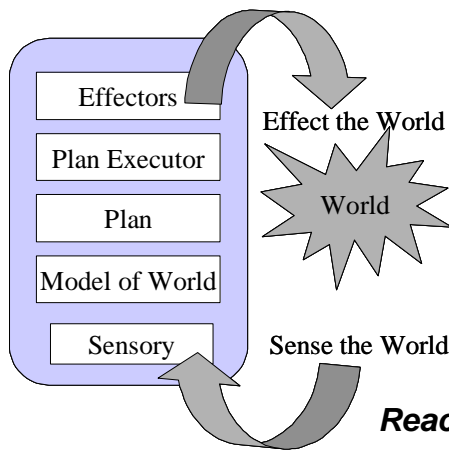
Agents such as STRIPS [7, 6], which takes a symbolic description of the world and the goal state including action descriptions, which characterise the pre- and post-conditions associated with various actions. This agent attempts to find a sequence of actions that will achieve the goal, by using a simple means-ends analysis, which essentially involves matching the post-conditions of actions against the desired goal. The STRIPS planning algorithm was very simple, and proved to be ineffective on problems of even moderate complexity.

GRATE:

This Agent was developed by Nick R. Jennings. The agent is based on a layered architecture consisting of namely:

- Domain Level System (DLS) layer: Solves the organizational level problems
- Co-operation & Control layer: Co-ordinates the DLS level activities with others in the community. This layer consists of the following three modules:
- Control Module – Interfaces with the DLS
- Assessment & Co-operation Module – These modules specify how the agent should act both internally and with other agents while in collaboration with them.

Basic Architecture of a Deliberative Agent

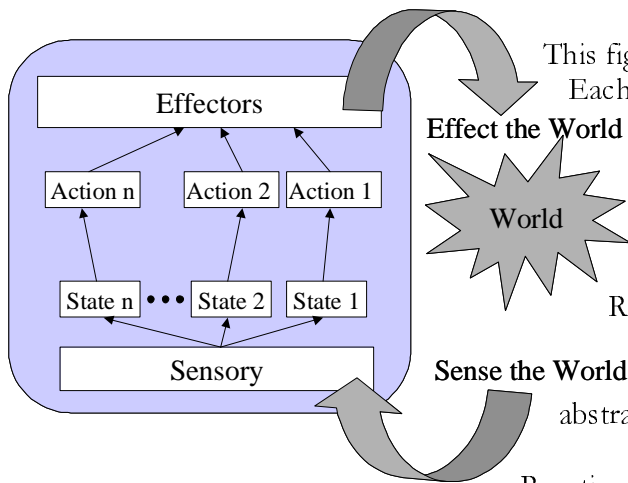


This figure[4] depicts a basic architecture of a Deliberative Agent. The key elements of this sort of agent are Sensory, which is responsible to collect input (sense) from the external world. The World Model, which is a representation of the physical world. Plan repository, basically containing a set of plans to achieve a specific intention. Plan executor and an effector, which ensure execution of a given plan and correspondingly affect the external world.

The evolution of the deliberative agents architectures led to the Reactive agents, which were as a result of learning's and experience gained in trying to solve the problems that existed in the deliberative agents.

Reactive Agent architecture proposes the agent based system from a pragmatic approach by removing the need any central symbolic world Model and the need to have any complex reasoning based on symbol hypothesis. Instead the approach has been to keep a very simple internal representation of the world and provide a tight coupling of this representation of the world to the perceptions and actions that the agent intends to take.

Basic Architecture of Reactive Agents



This figure [4] depicts the basic architecture of a reactive Agent. Each agent behaviour maps perceptual input to action output [4]. The figure depicts the different states of the environment and the corresponding actions, which the agent is capable to perform. Each change that occurs in the external world can result into one or more actions that the agent can perform. Reactive agents proposes that Intelligent behaviour can be demonstrated without the needs for Explicit Symbolic representation of the world and correspondingly explicit abstract reasoning of these Symbolic representations.

Reactive Agents at the same time also do not solve all the problems. They too have their share of complexity, such as:
The Developments takes a lot of time of such agent systems
Impossible to implement large systems
Can be utilised for the purpose of which it is built and is not flexible to be applied to a different domain or applications.

Few examples of the Reactive Agents

Brooks:

This is the architecture for a building agents developed by Rodney Brooks, an MIT researcher[12, 6]. This was proposed as result of frustration with the manner the symbolic AI were utilised to define Agent based system (centralized conceptualization of the world and abstract reasoning techniques).

This agent architecture was termed as subsumption architecture. This architecture demonstrates two key aspects, firstly that the real intelligence is out there in the external world and need not be represented within an internal system. This architecture consists of hierarchy of task accomplishing behaviours and each of this competes against each other to control the agent system. The behaviours are based on primitive behaviours,

which has high priority over the higher-level behaviours. The agent system built thus is extremely simple with no explicit reasoning or pattern matching but yet demonstrate impressive behaviour and capabilities.

Agent Network Architecture:

This architecture [13, 6] was developed by Pattie Maes, in which an agent is defined as a set of competence modules. These modules are specified correspondingly with a pre- and post-condition. These modules in addition have an activations level, which basically specifies the relevance of this module given a particular situation in the execution of an agent system. The higher the relevance the likelier this module is linked with other modules containing the corresponding and relevant pre- and post-conditions. The execution of a set of such linked modules could result into a command to an effector or then to raise the level of relevance of the corresponding modules.

Rosenschein and Kaelbling:

This is an approach proposed by Rosenschein and Kaelbling, also known as situated automata. In this approach the agent is specified in declarative terms, which is then compiled to machine understandable format. The mechanism used to specify the declarative terms is essentially a modal logic of knowledge, basically interpretation of the world into states of the automaton.

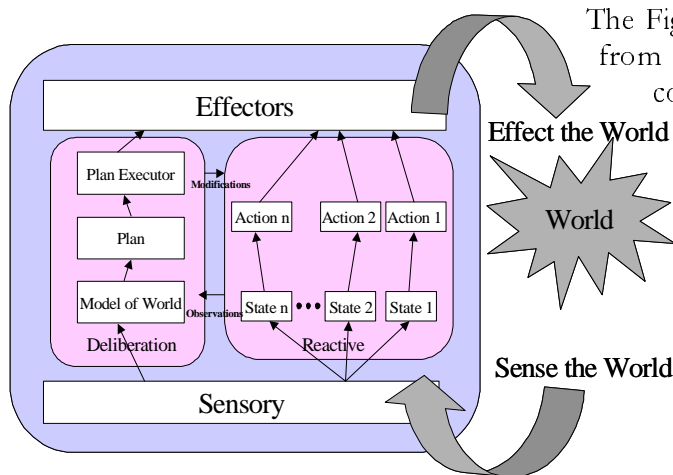
The agent is specified in terms of perception and action and correspondingly two agent elements namely:

- **RULER:** A system used to handle the perception of an agent. This system takes as input three inputs, namely: A set of input semantics; A set of static facts and the State transitions of the world. The systems is given desired outputs and then accordingly based on the provided input the compiler generates a circuit whose output will be the correct semantics
- **GAPP:** A systems used to handle the actions of agents. This systems takes as input a set of rules that contains information on how the goals can be achieved, a top-level goal. The systems then correspondingly generated a program that can be translated into a digital circuit in order to realise the identified goal.

Hybrid Agent

Agent researchers have suggested that neither of the deliberative or reactive agents architectures are completely suitable to build agent based systems. Instead they have suggested a case for hybrid systems, which utilises the learning's from research and development of both the deliberative and as well as reactive agents. Typically such an agent based systems consists of few subsystems, namely to plan and make decisions using symbolic reasoning and react quickly to the events occurring in its environment without complex reasoning.

Basic Architecture of Hybrid Agents



The Figure depicts the hybrid Agent architecture as can be seen from this figure that it contains both the deliberative component and as well as the Reactive components. The two components have also a means of collaborating by means of exchanging the observations of the real world and affecting the reactive behaviour of the agents systems by means of modifications triggered by the deliberation part of the agent system.

Procedural Reasoning System:

This is an agent architecture [14, 6] proposed by Georgeff and Lansky. This is based on a BDI agent architecture, which includes a plan library and a symbolic representation of Beliefs, Desires and Intentions. Beliefs, facts about the external world, are represented as first order logic; Desires are the systems behaviours and the Intentions are a set of active knowledge areas, which is active part of the partially elaborated plans belonging to a plan library. The Knowledge areas (KA) are associated with them an invocation condition, which basically determine if the KA is active or inactive. KA can also be reactive, resulting in immediate reaction to the external events. This agent system contains a system interpreter, which basically is responsible for updating beliefs, invoking the KAs and executing the corresponding actions.

Ferguson - Touring Machines:

This agent architecture was developed by Ferguson as his Doctoral thesis. The agent system consists of three key elements Perception and Action, which are interfaced with the agent's Environment. These three elements include a control framework which helps mediate between the layers as each of the layer is in an independent and concurrently executing processes. Each of these elements map directly to the layers: Reactive, Planning and Modelling Layers. The Reactive layer based on the events that occur within the agent's environment generates the corresponding set of quick actions that the agent should carry out. This is implemented based on a set of rules resulting into actions as defined in the subsumption architecture. The planning layer construct plan and accordingly selected the actions that are required to be executed in order to achieve the agent's set goals. The planning layer consists of the following two layers: *Planner*: Contains the plan generator and executor, and the *Focus of Attention*: This limits the amount of information that the planner receives by filtering out all the irrelevant information from the agent's environment. The modelling Layer contains the symbolic representation of the entities in the agent's environment. This layer of manipulated in order to identify and resolve any conflicts that may occur during the agent's execution

COSY:

This is a hybrid version of BDI architecture. This agent architecture consists of 5 key elements: The Sensors, Actuators, Communications, Cognition and the Intentions. Each of the elements are described as follows: Sensors, receive the perceptual input from the agent's environment; the Actuators, allows the agent to perform non-communicative actions on its environment; the communications element allows the agent to send out communicative messages out to the agent's environment; The cognition element is responsible to analyse the intentions of the agent and its beliefs and correspondingly choose the appropriate action that the agent should perform; and finally the Intention contains the long term goals, the agents attitude and responsible with regard to its environment.

INTERRAP:

This is a layered architecture, where each successive layer representing a higher-level abstraction of the below layer. These layers are further divided into vertical layer, while containing the knowledge base and the other containing the control layer that interact with the knowledge base contained in that specific layer. The control in the INTERRAP is both data and goal driven

The lowest of the layers is the World interface layer, containing the Control component and the World model, the world control component as its name suggests it manages the interface between the agent and its environment namely in acting, communicating and perceiving its environment, while the model serves as the knowledge base of the world (agent's environment). The Behaviour based component is the next layer above the World interface layer. This layer is responsible for controlling the basic reactive capability of the agent. The various conditions are evaluated and correspondingly resulting into call to the lower layer World interface or then to a higher layer to generate a plan. Above this layer is a plan based component layer, which contains a Planner responsible to generate the single agent plan in response to the signals delivered from the lower layer. This layer contains a set of plans and a plan library. The layer above this one is the co-operation component layer, which is basically responsible for generating joint plans which should satisfy the goals of a set of agents, this is done so by elaborating plans selected from a plan library, initially generated by the lower layer.

Architecture for Distributed and Heterogeneous Agent System

In the previous section we have discussed about the different Agent architectures. The discussions are basically of agent architectures, which are simple and single agent based system. The architecture very soon reaches its complexity as we address multi-agent systems and specifically increases exponentially as we vision a heterogeneous multi-agent systems, leaving aside the distributed nature of such a heterogeneous multi-agent system.

In this section we will discuss about the architectural issues in addressing such a distributed heterogeneous Multi-Agent system, by specifically discussing about Open Agent Architecture (OAA, hereafter) [8], which addresses this in a very pragmatic approach.

Overview of the Open Agent Architecture

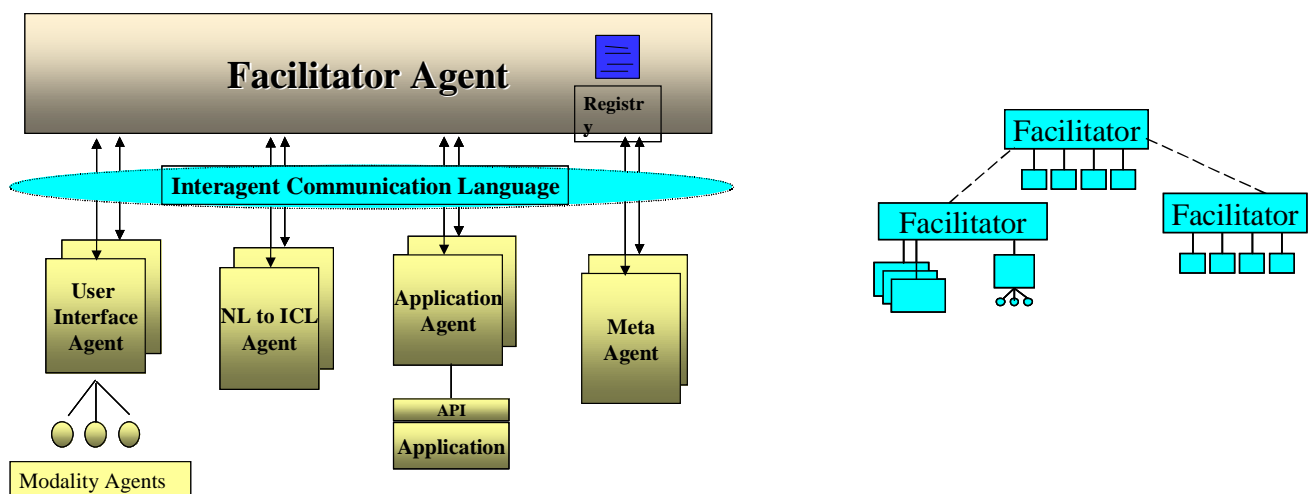
OAA is a framework for integrating a community of software agents in a distributed environment. It is developed in Artificial Intelligence Centre at SRI International, California. OAA facilitates flexible adaptable interactions among distributed components through delegation of tasks, data requests and triggers. OAA enables the natural, mobile and multimodal user interfaces to distributed services.

The OAA tries to address the following three issues:

- The increasing requirements to support networked computing model, forces development of new approaches in software design to flexible utilise the distributed processing elements in addition to taking into consideration the dynamically changing and unstable environments.
- Users and software designers are expecting the deployment of smarter, automised, and autonomous services and solutions, correspondingly facilitating access to personalized services.
- As the services increase in complexity users and software systems expect to provide highly intuitive user interfaces, which keep the human-computer interaction as close to human-human interactions. (Multi-Modal interactions, Natural Language Understanding (NLU), Pen, Gestures, handwriting)

This architecture makes it possible to enable distributed collections of autonomous agents to co-operate and communicate with each other including the human users to deliver intelligent services to end users. The communication and collaboration between multiple agents is made possible with help of intermediaries, namely the Facilitators, which do the match making of the received requests from users and agents with the corresponding descriptions of the agents it has within it. Hence allowing the requesting agents/users to not necessarily be aware of the serving agents. This mediator based architectural approach is advantageous as it allows a flexible means of introducing new agents and still keeps a loose coupling between the requesting and the serving agents. The OAA is designed to support simpler means of developing new agents and wrap existing applications written in different languages and running on different operating systems and integration of them with the OAA based systems. The keys differentiator of this agent architecture is that:

- It deploys a mediator based approach to delegate complex request serving mechanism
- It supports Multi-Modal user interfaces and includes humans as the first class citizens of an OAA based agent systems.



The Figure (Reference 9) above depicts the Open Agent Architecture. This figure shows the User Interface Agent, Application Agent, Natural Language Agent and Meta Agent grouped into an agent community by being related to a single Facilitator Agent. The Facilitator agent is a mediator or intermediary agent between the agents within a single community. The main purpose of the Facilitator Agent is to provide:

- Global Data store: Allowing the agents in the community to adopt a blackboard style of interactions
- Coordinating Agent Communication
- Co-operative problem solving

This facilitator agent can be interfaced with other similar facilitator agents hence forming inter-agent Society. The Application agents typically provide collection of services which domain specific /neutral ones, such as: speech recognition, natural language understanding, emails, and travel planning or reservation agents. Meta-Agents assists the Facilitator agents in the co-ordinating the activities of other agents and specifically provides assistance from specific domain or application specific knowledge. The UI Agents is in some cases implemented as a set of micro-agents called as Modality Agents, each of which is monitoring a different mode of collaboration with the end user (pen, gestures, speech, etc).

Client Agents (UI, Application & Meta Agents) at the start-up register their capabilities and the services they offer with the parent facilitator to which they belong. The communication between the client agents and the Facilitator is based on the Inter-Agent Communication Language (ICL). The Facilitator, when it identifies that the service from a particular agent is required, it send a request to the agent using the ICL. The client agent on receiving the request processes the Request and correspondingly delivers a Response if required to do so. The Client agent during the processing of the Request can utilise the services of other agents and as well as the Facilitator by utilising the ICL based communication with the Facilitator.

The communication and collaboration (CC) under the OAA is based on message exchanges using the ICL specified formats. The CC is established based on the following three steps: The Service providers register the service details and the capabilities with the Facilitator; The service requester Agents formulate the ICL based message specifying the goals of the request and relay them to the Facilitator; The Facilitator then co-ordinates with the appropriate service provider agent able to accomplish this goal and reflect the results back to the service requesting Agent.

The core elements of the OAA are:

- **Agent Library:**
This contains the necessary software and infrastructure to construct and deploy an Agent based system. This can be done using multiple programming languages based on various operating systems. One key focus is to minimize the efforts to deploy such an agent based system. The libraries available will facilitate development of agent based systems which can collaborate and communicate with the Facilitator, with other agents and also internally within the implementing agent system, which uses the OAA.
- **Inter-Agent Communication Language:**
This is an interface, communication and a task coordination language that is utilised between all elements within an OAA based Agent community. ICL can be used to perform queries, execute actions, exchange information, set triggers and manipulate data in Agent community. The ICL contains the conversation protocol (Similar to the one provided by KQML) and a content layer (Similar to KIF)
- **Capabilities Declaration:**
This provides a high level interface to the agent, basically describing a set of Agent capabilities specified using ICL. This interface is utilised by the Facilitator to communicating with the agent and more specifically to delegate the service requests. The two main sorts of capabilities that an Agent can express are procedural, where the agent can perform some action and Data, where an Agent can provide some access to Data.
- **Service Requests:**

An Agent requests the services of other Agents by encapsulating the tasks and the goals which the request agent wants to achieve into a request message and then delegating the Facilitator to achieve them. The request contains the calls to one or more than one agent capabilities. The request not necessarily contains the agents that should process the request; this is optional as the Facilitator is responsible to delegate the request to the serving Agent. There is an option for the requesting Agent to specify the serving Agent/s.

- **Facilitation:**

This is the core functionality of the Facilitator. The Facilitator maintains the knowledge base containing record of a collection of the agents and their capabilities within the agent community and accordingly utilises this knowledge to assist the requesters and the service provider to find each other. The facilitator demonstrate the following four functionalities, Firstly Transparent Delegation, secondly Handling of compound goals, which includes, delegation, optimization and interpretation, thirdly utilization of strategies and instructions provided by the requesting Agent and Finally the Distribution of both the data update requests and the installation of triggers request.

- **Management of Data Repositories:**

The information about the Agent capabilities are stored, maintained and queried. This provides a means to maintain the information about the agent's capabilities within the community. The facts associated with each capability is stored and maintained by the Agent Library. The Agent Library also maintains the facts about the incoming requests containing queries about the agent capabilities.

- **Autonomous monitoring using Triggers:**

Triggers are a means to specify that certain actions are to be performed when some set of conditions are met. Agents can install such triggers locally or remotely on a Facilitator or even on their peer agents. The triggers can be of four types, namely:

Communication: Monitor different (in/out) messages

Data: Monitor state of a data repository

Tasks: To identify when a specific agent capability becomes available or for checking the internal status of a specific task

Time: Monitor time and accordingly fire scheduled actions

The Trigger contains the type, a condition and an action all of which are expressed using ICL. The condition specifies under what condition can this trigger be fired and the action specifies what should be done when the trigger is fired. Triggers can be utilised within the OAA for different purpose for example to monitor the state of different executions within the OAA, communication coordination of different agents to synchronize relevant tasks, etc.

We have just discussed about the approach the Open Agent Architecture efforts has taken in addressing the issues of an autonomous, distributed and heterogeneous Multi-Agent system. This basic architecture, the components and their functionality, including the core elements that the OAA consists of were discussed. The OAA presented in the above discussion highlights the complexity of issues, which are necessary to be solved in order to enable wider application of agent-based systems into the real world solutions. It also identifies that the key to a Multi-Agent systems in real world situation is to deal with integration of Agent systems with the existing legacy applications and inclusion of the Human as a key participant in the complete solutions. These aspects have not been widely addressed in the past and seem that it has been a key issues which has contributed for the late deployments of Agent based systems into the real world.

OAA still only addresses inclusion of Agents, which can talk ICL, hence requiring existing other agent oriented languages to talk in ICL to communicate with an OAA agent community. This is addressed in the OAA by introduction of new Agent, which can function as a language interpreter.

Conclusion

Software Agent architecture plays a very important role in realizing the agent based systems in the real world. The Software Agent architectures are to be addressed in two fold hand in hand, one Internally to a single Agent architecture, as discussed in the Section 1 and secondly as discussed in the Section 2, focusing on the external interfaces which includes the agent itself and the agent execution environment where agents requires to interact and collaborate with the external world. The external world includes with equal importance one or more than agent based system, external agent environment representing the world and finally very importantly the human user as a key participant in the complete agent based architectures and solutions. The distributed nature of the complete Agent based system will bring in and demonstrate greater value to automatised services.

This is too huge a problem domain to be addressed by small communities of experts bringing forwarded their own domain specific architectural views to provide a solution. In order to see wider application of Agent based system we need to align the Agent based architectures to existing architectures such as J2EE, .NET and the Web World and integrate with loose coupling approaches as for example suggested by OAA. There is need to standardized unified Agent communication language, which should act as a bridge between heterogeneous agent communication languages.

There are several architectural and technical aspects that are still required to be addressed by the agent research community, few of the important once being security and privacy models, Agent-Mobility, Inter-operability with the mobile World (PDA, mobile device (phone), etc) technologies and Heterogeneous Agent Communication Language, Automation of user interactions with services, Inter-operability with Web Infrastructure including web technologies such as XML & Web Services, Management and Remote monitoring, Reliability and Accountability.

We have seen that the Agent research community is based on several years of experience and as result we currently understand the complexity to develop an Agent based system. Taking a pragmatic approach in developing and deploying Agent based systems into the real and distributed world with minimal efforts in establishing an agent environment is very critical. This is the direction the community is and should be focusing on. The Developer and the End user community must be the main focus in the agent research efforts. The developers must be supported with good tools to architect, design, develop, deploy, monitor and manage, while the end user's should be provided with intuitive user interfaces to interact with Agent system and be able to in a semi/fully Automatised manner be able to receive the services of an agent system which in turn should ease the utilization of the current/future local and distributed services.

References

1. Maes, P. (1991). The agent network architecture (ANA). *SIGART Bulletin*, 2(4):115-120.
2. <http://pattie.www.media.mit.edu/people/pattie/>
3. Foundation of Intelligent Physical Agent(FIPA), <http://www.fipa.org>
4. Agent Architectures & Agent Oriented programming, Hekki Helin, Spring 2003, Software Agent Technology, a course at University of Technology, URL: http://www.cs.helsinki.fi/u/hhelin/opetus/oat3/architectures_10022003.pdf
5. Software Agent Architectures, Suresh Chande, Spring 2003, This document submitted as part of the course refered to in reference 4, URL: http://www.cs.helsinki.fi/u/chande/courses/PhD/Sem1_2003/SAT/SureshChande_Foundation_of_IntelligentAgents_Assignment3.doc
6. Intelligent Agents: Theory and Practice, *Knowledge Engineering Review* Volume 10 No 2, June 1995, Michael Wooldridge and Nick Jennings
7. Fikes, R. E. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189-208.
8. D. L. Martin, A. J. Cheyer, and D. B. Moran, "The Open Agent Architecture: A framework for building distributed software systems," *Applied Artificial Intelligence: An International Journal*. Volume 13, Number 1-2, January-March 1999. pp 91-128. URL: <http://www.ai.sri.com/~cheyer/papers/oaa.pdf>

9. "Building and Using Practical Agent Applications", PAAM98 Tutorial, David Martin Adam Cheyer, URL: <http://www.ai.sri.com/~oaa/distribution/doc/paam98-utorial/index.htm>
10. Overview of Open Agent Architecture, URL: <http://www.ai.sri.com/~oaa/oaaslides/>
11. Vere, J. and Bickmore, T. (1990). A basic agent. *Computational Intelligence*, 6:41-60.
12. Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14-23.
13. Maes, P. (1991). The agent network architecture (ANA). *SIGART Bulletin*, 2(4):115-120.
14. Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. *In* Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), pages 677-682, Seattle, WA.