Implementation of hybrid software architecture for Artificial Intelligence System

B.Vinayagasundaram * *and S.K.Srivatsa*

Computer Center, MIT Campus Anna University Chrome pet Chennai-600044, Tamilnadu India

Summary

An intelligent system should have the ability to process complex information. This information processing is not possible as whole in principle. We need some form of partial computational strategy, which is flexible that can adapt to changes in the environment and requirements. This paper introduces software architecture for specification and verification of Artificial Intelligence system incorporating conceptual and formal framework. The focus is based on reusable components. The architecture consists of four components; Task specification layer, Problem solver layer, domain layer and an adapter layer. These four components are hierarchically organized in a layered fashion. In this architecture, the overall structure is decomposed into sub components, in a layered way such that adding new layer without changing the existing layers can change the behavior of the system. Hence, an AI system can be built in an evolutionary way by combining and adapting several reusable components with out changing the existing functionality of the system. The paper is organized in to two parts. The first part of the paper summarizes the various AI architectures proposed in the literature. The second part of the paper addresses the hybrid layered software architecture for Artificial Intelligence system. Based on the proposed layered architecture, various text categorization methods are implemented in the problem solver layer and the performance is also discussed.

Key words: *Artificial Intelligence, software architecture, Reusability, Software engineering.*

1. Introduction

The definition for Intelligence according to Hideyuki Nakashima is:

"Intelligence is the ability to process information in a properly complex environment in a partial way; nevertheless this partial processing mechanism yields maximally successful behavior." Even though AI is a sub field of information processing, there are some important differences, in information processing complete processing is a must, whereas in Artificial intelligence processing in most cases complete processing is not possible [1]. The illustrations for the above is, in information processing, the algorithm for searching a data must be complete, but in AI processing complex heuristics are used to reduce the search and in some cases the best options may be missed occasionally. Hence the AI systems should be architected in such a way not only for adaptations to new functionalities and environment but also to process complex information in partial way.

2. Software Architecture and its roles:

Software architecture of a system describes the structure, organization of components/modules and their interactions not only to satisfy the systems' functional and non-functional requirements but also provide conceptual integrity to the overall system structure. Software architecture concerns with the structure of large software intensive systems. [2] The architectural view is an abstract view that separates the details of implementation, algorithm and data representation and concentrates on behavioral aspects and interaction among the various components. In other words, the software architecture of the system provides an abstract description of the system by exposing certain properties and hiding others.[3] Hence the software architecture plays an important function with respect to following aspects in the development of large software intensive systems.

(i) Understandability: It helps to understand the large system by the appropriate level of abstraction. It also exposes the high-level design constraints, thereby providing way for making architectural decision.

(ii) Reusability: Architectural designs support the reuse of large components and provide frameworks into which components can be integrated.

(iii) Construction: An architectural description provides a blue print for the development of the system indicating the major components and the relationship among them.

* corresponding author ***Computer Center, MIT Campus Anna University Chrome pet Chennai-600044, Tamilnadu India***

(iv) Evolution: The architectural description of the system separates the functionality from implementation, thereby permitting to manage the concerns regarding performance, reusability and prototyping in an easy way.

(v) Analysis: The architectural description provides a new attribute for analyzing the system with respect to quality, performance, dependency etc,. Moreover analysis of architectures built with different styles can also be made to arrive at good architectural design decisions.

(vi) Management: Successful development of software addressing specific application depends on critical selection, analysis and evaluation of software architecture.

Artificial Intelligence systems are large and complex. The more powerful way of structuring the complexity lies in architecting the system. Hence an efficient method is needed to structure and handle the complexity of these systems. A good architecture of the system will not only satisfy the functional requirements but also the key non-functional requirements of the system such as performance, reliability, portability, maintainability etc,.

3. Architectures for Artificial Intelligence Systems

The architecture of a system describes its module capabilities and how these modules operate together as a whole. The Artificial Intelligence systems can be broadly classified into reactive systems, deliberative systems and interacting systems. Over the past, numerous architectures have been proposed in the literature for these systems addressing the important features of these systems. Reactive systems are built according to behavior-based paradigm. These systems have a very simple representation of world and provide tight coupling of perception and action. Deliberative systems have a symbolic representation of the world in terms of categories such as beliefs; goals or intentions and that possess logical inference mechanism to make decisions based on their world model. The interacting systems are able to coordinate their activities through communication and negotiation.

3.1 Architectures for reactive systems

These systems make decision at run time based on limited information and simple situation action rules. These architectures were often called behavior based, situated or reactive. Some researchers especially, Brooks with Subsumption architecture denied the need for symbolic representation of the world, instead the systems make their decisions based on the inputs. The decision of reactive architectures are partly guided by Simon's hypothesis which states that that the complexity of the behavior of the system can be a reflection of the complexity of the environment rather than the reflection of the system's complex internal design.

3.2 Brooks'subsumption architecture

The basic concept of subsumption architecture can be characterized as follows:

(i) Each module in the system is connected in parallel between the input and output in contrast to the conventional serial processing

(ii) Modules form layers in which higher layers can subsume lower layer functionalities and hence the name subsumption.

(iii) The lower level layers governs the basic behaviors whereas the higher level layers provide control mechanism

(iv) Augmenting a new layer at the top level without disturbing the existing layers can change the total behavior of the system.[4][5]

Steels behavior based system architecture: The basic concept behind this architecture is that, the complex behavior of the system as a whole can emerge by the interaction of simple individuals with simple behavior. This describes the feature of swarm intelligence [6]. This approach foregoes any planning but refers to emergent functionality. The performance of the system can be greatly improved by reactive cooperation method.

3.3 Architectures for deliberative system

Most AI systems maintain the internal representation of their world, which can be modified by some form of symbolic processing. AI planning system can be regarded as the predecessor for this deliberative system architecture.[7] Recently the AI system architectures are modeled based on the Beliefs, Desires, Intentions (BDI architectures). The world is modeled using a temporal structure with branching time future and linear past

called time tree. Situations are defined as particular time points in the world. Time points are transformed in to one another by events, which can be primitive or non primitive. The non-primitive plans are useful to model partial and hierarchical plans that are decomposable into sub-plans and finally in to primitive actions. In this architecture there is a distinction between choice and chance. That is ability to select its actions from a set of alternatives and the uncertainty of the outcome of the actions where the decision is made by the environment rather than by the system.

4. Hybrid AI architectures

Purely reactive systems have a limited scope as they can hardly implement goal directed behavior where as most deliberative architectures are based on general purpose reasoning mechanisms which are not tractable and which are much less reactive. These disadvantages can be overcome by layered architectures. Layering is a powerful way of structuring the complexities in general and functionalities in particular.[8] The layering approach supports several properties such as reactivity, deliberation, cooperation and adaptation. The main idea is to structure the functionalities into two or more hierarchically organized layers that interact with each other to achieve coherent behavior. Layering also offers the following advantages:

(i) It supports modularization. Different functionalities can be clearly separated and linked by well-structured interfaces.

(ii) The design of the architecture is compact, increases the resolution and facilitates debugging.

(iii)It is possible to run different layers in parallel there by increasing the computational speedup.

5. Design of Hybrid layered architecture

Most AI systems display a rigid separation between the standard computational components of data, operation and control. That is, if these systems are described at an appropriate level one can identify three important components viz. *knowledge database* that is manipulated by well-defined *operations* all under the control of global *control* mechanism. Even though at machine code level, any neat separation in to distinct components can disappear, it is important to specify them at appropriate level of description. One difficulty in using conventional software systems with hierarchically organized programs for AI applications is that modifications to knowledge base might require extensive changes to various existing programs, data structures and subroutine organization. In the proposed hybrid architecture for Artificial Intelligence system, components are defined in a modular way. The system design is more modular and changes to any of the components can be made relatively independently. Apart from describing the system with distinct components, the critical issue in the construction of AI system is its architecture; that is its gross organization as a collection of interacting components. A good architecture can ensure that a system will satisfy not only the key functional requirements but also ensures the non functional requirements such as reliability, portability, modifiability etc,. In the proposed hybrid architecture, four distinct components are defined at appropriate level as shown in fig.1. A task definition layer defines the problem that should be solved by the system. The second component viz., Problem solver layer defines the method for reasoning and a domain model layer describes the domain knowledge of the AI system. These three components are described independently to enable the reusability The fourth component Adapter layer adjusts the three other components to each other and to the specific problem.

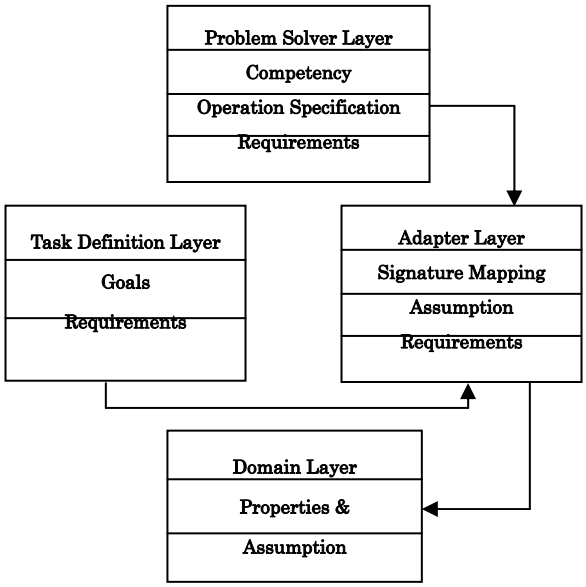

Figure 1. Hybrid Layered Architecture for The Artificial Intelligence System

6. Formal Specifications of Components

6.1 Task description Layer

The task description layer consists of two elements viz., the goals that should be achieved to solve a given

problem. The second element of the task description layer is the definition of the requirements on the domain knowledge. Usually, axioms are used to define the requirements. The task definitions are done by algebraic specifications that provide a signature consisting of types constants functions predicates that defines the property of this component.

6.2 Problem solver layer:

The concept of problem solver layer is present in many AI frameworks. The problem solver describes the reasoning steps and which types of knowledge are needed to solve the given problem. Even though, there are some differences in the approaches, the following features are common in almost all problem solvers:

(i) The problem solver should decompose the entire reasoning process into primary set of elementary inferences.

(ii) The problem solver should define the type of knowledge that is required by the inference steps to be done.

(ii) The problem solver should define the control and flow of knowledge between the inferences.

The description of problem solver layer consists of three elements viz. Competency description, operational specification and requirements on domain knowledge. The competency of a problem solver is a logical theory that characterizes the solution process. It is similar to the specification of functionality in software engineering. Selection of a solution method for the given problem and the verification of whether the Problem solver fulfils the requirement for the solution process can be done independently from the details of the internal reasoning behavior of the method by proving that the Problem solver has some competency. The operational specification defines the dynamic reasoning of the problem solver, which explains how the competency is achieved. The third element introduces the requirements on domain knowledge. All the inference steps and competency description of the problem solver requires the specific types of domain knowledge. These requirements on domain knowledge distinguish a problem solver from conventional software. Competency description specifies the actual functionality of the AI system where as the task description specifies the problem that should be solved by the AI system. Distinction between the competency description and task description is made because of the following reasons.

(i) The problem solver introduces requirements on domain knowledge in addition to the task description. Even though this knowledge is not necessary to define the problem it is required to describe the solution process.

(ii) It is not always assumed that the functionality of the AI is strong enough to completely solve the problem. Hence the problem solver should introduce some important assumptions to reduce the problem size.

This is similar to the distinction between functional specification and the design- implementation of software in software engineering. The functional specification deals with *what* and design specification deals with *how*. This separation is often not practically possible even in the domain of software engineering, would not possible in the development of artificial intelligent system because a large amount of problem solving knowledge is required, that is knowledge about how to meet the solution requirements is not a question of algorithm and data structures, it exists as a result of experience over the years and heuristics. Even though, some problems are completely specifiable but it is not necessarily possible to derive efficient algorithms from such specifications. Hence it is not sufficient to have knowledge about what is the solution to the problem, but also have knowledge about how to derive such a solution in an efficient way.

6.3 The Domain Layer

The description of the domain layer introduces the domain knowledge required by the problem solver and the task description layer. To represent domain knowledge ontologies are proposed so that the knowledge can be reused. In the domain layer of the architecture three elements are defined viz., properties, assumptions and the domain knowledge itself. The domain knowledge is necessary to define the task in the given application domain to carry out the inference steps of the problem solver. Properties are derived from the domain knowledge and assumptions are the attributes that have to be assumed to be true. Hence, the properties and assumptions are both used to characterize the domain knowledge.

6.4 The adapter Layer

Adapters are of general importance in component-based software. In the hybrid architecture, adapter layer is introduced to relate the competency of the problem solver to the functionality given by the task description

layer. Further, the adapter layer introduces new requirements and assumptions because the most problems tackled with AI systems are in general complex and intractable. Hence, a problem solver can solve such tasks with reasonable computation by the way of partial processing by introducing assumptions that restrict the complexity of the problem or by strengthening the requirements on the domain knowledge. The other three layers viz., Task description, problem solver and domain layer can be described independently since the adapter layer combines these layers in such a way that meets the solution requirements of the specific application. The consistency in the relation and the adaptations to the specific aspects of the problem makes it reusable.

7. Proposed architecture applied to text Classification:

To illustrate the approach the task of text categorization is used. The objective of the text classification is to assign a number of appropriate categories based on the content. To carry out this task manually, a large amount of human resources are needed. Several features of text classification task make it different from other artificial intelligence problems. First, the problem spaces of text classification involve a high dimensional space.[9] Second even though the problem space is high the each document contains only a small number of features which are sparse. Some of the standard methods for text classifications are K-nearest neighbor algorithm, Bayesian algorithm and back propagation algorithm. The main disadvantage of K-nearest neighbor algorithm is it makes use of training examples as instances for computing similarity. To overcome this k-nearest neighbor is enhanced such that it makes use of generalized instances for computing the similarity, which is called as Generalized Instance Set algorithm. The main idea of this enhanced version is to construct the generalized instances to replace the original training examples. Given a particular category, it can be observed that the regularity among the positive examples is usually more than that of the negative examples. The classification knowledge induced from a pool of similar examples is relatively accurate. But, on the other hand negative examples close to such a pool are likely incorrect negative instances or noise. Based on this idea the Generalized Instance Set algorithm focuses on refining the original instances and constructs a set of generalized instances. First it selects a positive instance and conducts a generalization process using k nearest neighbors. After a generalized instance is formed it is used as a new starting point and the process is repeated based on nearest neighbors. This search is repeated until no positive instance remains.

8. Results and discussion:

The performance of various algorithms implemented in the problem solver layer is measured and the results obtained have been compared. The performance measurement for text classification is usually done by macro averaged recall/precision break-even point measure (F measure). In this scheme precision and recall are two important parameters used in calculating the F measure. For a category I, the precision is defined as the ratio of number of documents classified according category I by the classifier. Recall is defined as the ratio between the number of documents correctly classified to category I to the total number of documents actually belonging to I.

Macro averaged $F= \sum F(I)/ m$
Where m is the total number of categories and
$F(I)= (2* \text{precision}(I)*\text{recall}(I))/(\text{recall}(I)+\text{precision}(I))$

The fig. 2 shows the performance of the problem solver implemented with Generalized Instance set(enhanced K-nearest neighbor) and K-nearest neighbor algorithms. The recall level at which K-nearest neighbor starts is lower than the recall level of Generalized Instance set. For the recall level between 0.5 and 1 it is seen that the precision level of Generalized Instance set method is less than k- nearest neighbor. This is because k nearest neighbor method operates well on this data without noise. Hence, there is a slight increase in the performance for this recall level. But when the macro averaged precision F is compared it is inferred that the average precision of GIS is more than that of K-nearest neighbor and also the macro averaged break-even point measure for GIS is better than K nearest neighbor. To over come this problem feature set pertaining to each category can be extracted and generalized instance can then be formed. When this is done GIS method will show better performance for all recall levels.

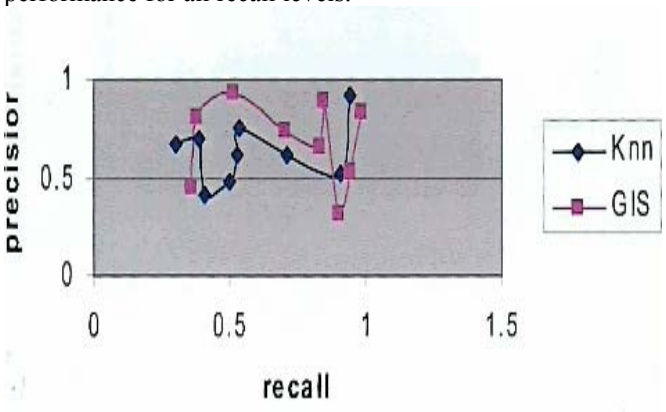

Figure 2. Relative performances of GIS and K-Nearest Neighbor Algorithm

The fig. 3 shows the performance of the problem solver implemented with enhanced K-nearest neighbor method and Bayesian method. Except for recall level at 0.4, the precision is more for enhanced K-nearest neighbor method than that of Bayesian. The precision is more for all other recall levels. The macro-averaged break-even point measure is also higher for GIS method.

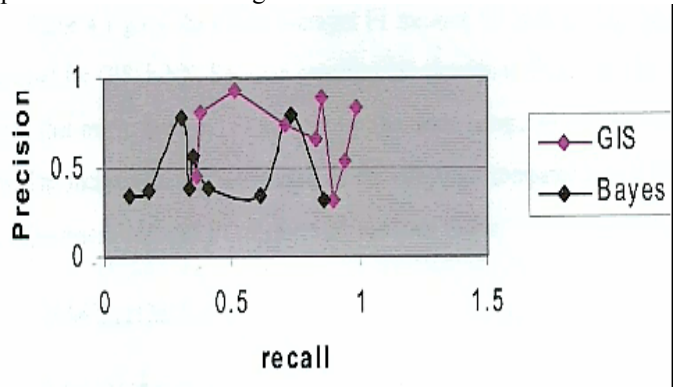

Figure 3. Relative performances of GIS and Bayesian Algorithm

The table given below gives the macro averaged F measure for various categories in the data set for enhanced K-nearest neighbor, k-nearest neighbor and Bayesian classification method. From the table it is inferred that the macro-averaged precision is also higher for enhanced K-nearest neighbor. In this paper, Bayesian method-nearest neighbor method and the enhanced version of it were implemented in the problem solver layer of the hybrid layered software architecture in a reusable way. It is also found that the F measure for enhanced version is 12 % higher than primitive version.

Table: 1 Macro averaged F measure for various methods

Category	Generalized Instance Set Method	K-nearest neighbor Method	Bayesian Method
corn	0.47368421052631 58	0.49090909090909 09	0.15447154471544 72
Crude	0.68243243243243 25	0.65909090909090 91	0.43537414965986 39
Earn	0.90890990542558 49	0.93741307371349 08	0.76190476190476 20
Ship	0.65853658536585 37	0.49799196787148 60	0.43533123028391 16
Interest	0.51485148514851 50	0.41626794258373 20	0.35842293906810 03
Trade	0.73267326732673 28	0.62499999999999 99	0.42567567567567 56
Acq	0.87068381855111 71	0.71932773109243 70	0.46450304259634 88
Wheat	0.72602739726027 39	0.57142857142857 15	0.23214285714285 71
Money-fx	0.40201005025125 63	0.40909090909090 91	0.39534883720930 23
∑ F(I)/ m	0.66331212803200 90	0.59183557730895 84	0.40701944869514 10

9. Conclusions and future work:

Software architecture for artificial intelligence system has been is developed based on conceptual and formal framework based on reusable components. Future work may be focused on the development of a semantic search layer, which can be augmented to the existing system without changing the architecture that can change the behavior of the system. The consistency checker in the adapter layer cannot make changes directly to the knowledge base in the domain layer. Hence the present work can be extended to perform this task by suitably modifying the task description layer. By specifying suitable quality metrics, the quality of the software architecture can also be measured. In this architecture, the overall structure is decomposed into sub components, in a layered way such that adding new layer without changing the existing layers can change the behavior of the system as a whole. Hence an AI system can be built in an evolutionary way by combining and adapting several reusable components.

References:

[1]. Hideyuki Nakshima, "AI as Complex processing" Minds and Machine vol 9. pp57-80 1999.

[2]. David Garlan, "Software Architecture: a Roadmap". ACM press 2000

[3]. Rikard Land ,"A brief survey of software architecture" MRTC report Dept.Computer Science Malardalen University Sweden 2002

[4]. Rodney A. Brooks, " Intelligence without representation" Artificial Intelligence 47:139-160 1991.

[5] Hideyuki Nakashima Itsuki Noda "Dynamic Subsumption architecture for Programming Intelligent agents" Proc. International Conference on Multiagent Systems" AAAI press pp190-197 1998.

[6]. Steels. L" Cooperation between distributed agents through self organization" Decentralized AI pp 175-196. 1976.

[7] Newell A.Simon" Computer Science as empirical enquiry: Symbols and search" Communications of ACM 19(3) pp113-126. 1976

[8]. J.F Sowa "Architectures for Intelligent systems IBM Systems Journal vol41 No.3 pp331-349. 2002

[9].Dolores Del Castillo Jose Ignacio Serrano " A Multistrategy Approach for Digital Text Categorization from Imbalanced Documents" Sigkdd Explorations vol.6. pages 70-77.

[10]. Wai Lam Yiqui Han " Automatic Textual Document Categorization based on Generalized Instance sets and Metamodel" IEEE Transactions on Pattern Analysis and Machine Intelligence vol.25 no.5 pp 628-633

.

B.Vinayagasundaram received his B.E. and M.E degrees in 1985 and 1994 from Madurai Kamraj University and Anna University respectively. He is working as Senior Lecturer in the computer center, MIT campus Anna University. His research interest includes Software Engineering, AI and

Dr.S.K.Srivatsa, a Ph.D from IISC Bangalore, Professor(Retd.) of Electronics Engineering in 2005. Currently he is Senior Professor at St. Joseph College of Engineering, Chennai. He is author of more than 200 publications. His research interest includes Computer Networks, Logic