AN APPROACH TO OPEN INTELLIGENT INFORMATION SYSTEMS

RAJEEV KAUL† and OJELANKI K. NGWENYAM†
Knowledge and Information Systems Research Group, The Thomas J. Watson School of Engineering, State University of New York, Binghamton, NY 13901, U.S.A.

(Received 20 March 1989; in revised form 18 March 1990; received for publication 24 April 1990)

Abstract—The paper proposes a new approach to the design of open intelligent information systems based on artificial intelligence techniques. This approach is referred to as the open intelligent information systems architecture (OIISA) and proposes a mechanism for the effective partitioning of the knowledge base and a framework by which separate knowledge bases (which in themselves can be distinct knowledge systems) would cooperatively solve problems.

1. INTRODUCTION

Developments of fifth generation information processing techniques have prompted researchers to rethink the ways information systems are being developed and used. Attempts are being made to adapt knowledge-based systems concepts to information systems. Some researchers have argued for a new type of IS, a knowledge-based intelligent information system [1–7]. Research, however, has focused mainly on identifying similarities between knowledge based information system development and conventional information system development [8]. Except for Bubenko’s conception of an intelligent information system (IIS), there seems to be no clear conceptualization of how to apply fifth generation techniques to IS development and use. It is our opinion that business organizations could benefit from IIS if there were clear concepts and a model to guide conceptualization and development of IIS. This general belief has led us to develop and implement a prototype of an open intelligent information system (OIIS), which could serve as an initial model for practical OIIS application development in business organization. The starting point of our research is the idea of an open information systems architecture and Bubenko’s [2] conception of an IIS; we briefly outline these below.

1.1. Promise of open systems

The tacit assumption of traditional IS architectures is that the main task is to design one (large) system and to achieve total completeness and consistency. This is probably not the way we should continue to build systems as in most cases we have already reached the complexity barrier. Some researchers [9, 10] envision future information systems as consisting of a large number of independently developed systems, which communicate and cooperate by passing messages (knowledge, data, information). In this architecture there is no global conceptual schema, and no global objects. The “closed world assumption” does not hold—a system would not “know” if certain information existed before requesting it. Each system exists with its own “self knowledge”—own assumptions, constraints and rules. Every such system evolves independently and may at times pass portions of their knowledge to other systems, and acquire, by negotiations, knowledge from its communicating “partners.” Although some work is in progress on implementing this new architecture, the following problems still exist:

1. Handling of communication semantics. All message semantics must be known to the receiving system. In other words, there are no standard communication semantics, a system might be overly burdened with the handling of different procedures for different messages.

2. Defining a new approach for message transmission. Performance of the architecture will be affected if a message is sent to “whoever gets first” way.

1.2 Bubenko’s IIS

Bubenko [2] describes the future IIS, as an event driven system which has deductive capability, an extended time perspective, and never forgets. Events would signal the observation of a new situation (fact/assertion) or establishment of a new rule within the Universe of Discourse (UoD) of the application. New event-facts could be, for example, the arrival of a customer order, or a change in the price of an article. A new rule could, for instance be inserted, to prescribe a new way of dealing with reordering of articles and the time when the new procedure should...
come into force. According to this vision the system is monitored by a knowledge-base (KB), which contains the syntax of the formal language to be used for communication among the application systems. The syntax will (1) define what kind of assertions can be made and understood about the application and (2) include descriptions of the semantics of the concepts used in the formal language for describing phenomena, relationships, etc. over the UoD of the application. The KB would have an “application rule part”, which will contain a set of formal definitions of assumptions made and rules established concerning the structure and behavior of the application UoD. These rules would specify what assertions about the UoD are valid (or reasonable) and how certain assertions are related to (or can be derived from) other assertions. It would also include knowledge about transactions and operations on applications—“statics” and “dynamics” of the phenomena, relationships, etc. over the UoD of the application. The last part of the knowledge base contains a set of facts observed in, or asserted as true about the application UoD. Conceptually, it also “contains” the facts that can be derived from the observed/asserted facts and the rules in the “application rule part”. Further, the knowledge base would also view its domain in a time perspective and not restrict its knowledge only to “the current state” of the UoD. It would be possible to refer to variables or objects and their values or existence at any set of historical time points or intervals. Accordingly the KB should never “forget” anything.

The key features of Bubenko’s IIS may be summarized as follows:

- It has deductive capacity and embodies a theory of the application.
- It has an extended time perspective.
- It “never forgets” data and rules and maintains a full UoD history.
- It is driven by external events and includes triggering conditions for internal events which define the behavior of the system.
- It would permit system implementation in a truly incremental manner. There would be no need to apply rigid, hierarchical, top-down design approaches to problems which naturally are not hierarchical. Adding a new application part would imply the extension of its syntax and its set of rules.
- Using logic as the basis for the KB would also permit us to deal with incomplete and disjunctive information.

1.3. Objective and outline of the paper

Although Bubenko is quite clear about how IISs should function he does not offer advice on their design and development. It is our belief that current IS and KS architectures are inadequate for designing and implementing meaningful intelligent IS applications. We believe that an open architecture approach could help with the problem of complexity in IIS development and, take us closer to realizing the promised benefits of IS. The work reported in this paper describes an OIIS architecture based on fifth generation techniques.

In Section 2 we outline the basic concepts behind current information and knowledge systems architectures, and discuss why there is a need for a new architectural approach. The theoretical foundations for the OIIS architecture are outlined and discussed in Section 3, along with a description of the prototype. Section 4 concludes with implications for future research.

2. KNOWLEDGE SYSTEMS ARCHITECTURES

The way an information and knowledge systems are perceived, described and specified, largely follows the way their architectures have evolved. The original view of IS as a system of processes communicating by files (file-centered architecture) has been gradually augmented or replaced by database views (data-centered architectures). Knowledge systems, although more recent have been conceptualized and developed in similar manner to conventional monolithic databases.

The typical knowledge system consist of three main components: (1) knowledge base, (2) inference engine and (3) user interface. The knowledge base is a storehouse of all knowledge of the UoD. It may consist of facts (assertions) and rules (procedures) that use facts as a basis for decision making. It is a set of information what the computer currently “believes”. The inference engine called the “reasoning” processor contains an “interpreter” that decides how to apply the rules to infer new knowledge and a “scheduler” that decides the order in which the rules should be applied. The human interface is user-friendly front end, which serves as a cognitive window for the user [11]. The interface manages the human-machine dialogue during problem solving. Two common approaches for knowledge systems architecture, are (a) production and (b) blackboard. We will briefly review each of these with regard to their limitations for OIIS applications.

2.1. Production systems architecture

Production systems architecture use a single inference engine and a single knowledge base. The inference engine consists of problem independent control strategies which determine which knowledge rule/object should be executed next and executes the actions specified respectively. An extension of this architecture called the “structured” production systems” [12], has also been proposed. This architecture divides the knowledge base into knowledge chunks and uses meta-rules to determine which knowledge chunk should be accessed next. The production system was the initial architecture for knowledge system development [13], and has been the basis for many
pioneering applications for example, DENDRAL [14] and MYCIN [15]. In spite of its major deficiency of the single monolithic knowledge base, which degrades performance as its size increases, it is still the widely referred architecture [6, 16-23].

2.2. Blackboard systems architecture

The blackboard architecture offers another approach to KS. It was first developed in the context of the HEARSAY-II speech understanding project [24]. The system consists of a set of independent domain specific modules, called knowledge sources (or knowledge systems), which interact via a shared data structure—the blackboard. When a knowledge system is activated it examines the current contents of the blackboard and applies its knowledge either to create a new “hypothesis” and write it on the blackboard, or to modify an existing one. Hypotheses on the blackboard are ordered from low to high depending on the extent of their coverage of the problem under solution. The goal of the system is to generate a single hypothesis that represents a solution to the problem.

Although the execution of the entire HEARSAY-II system consists of the asynchronous execution of a collection of knowledge systems, the execution of an individual knowledge system is a sequential process. Once a knowledge system is activated, it executes without being interrupted until it is finished. The actual determination of which knowledge system should be activated next is done by a special knowledge system, called the “scheduler”, on the basis of its knowledge about how best to conduct the search in the particular domain. The scheduler uses ratings supplied to it by each of the independent knowledge systems.

The techniques developed in HEARSAY-II have since been generalized into HEARSAY-III [25, 26]. Blackboard architecture was the only architecture that allowed distributed knowledge system development, and has been the basis of some real world knowledge systems for example, CODER [27] and CRYSTALIS [28]. This architecture in spite of its provision for modularity, does not specify how the specific piece of knowledge has to be handled by the other knowledge sources—the blackboard does not “know” anything about the nature of communication semantics. Just posting of hypotheses is a very restricted form of communication when the overall purpose is to imitate human thinking processes. Also, decision about the further execution of a hypothesis is taken by an independent “scheduler” (another knowledge system), without any regard to a hypothesis background knowledge, which consequently reduces the coherency of the overall system. Further the idea of specialization KS was not implemented in the blackboard architecture. As a result a great deal duplication exists in these types of KS.

3. CONCEPTUAL FOUNDATIONS FOR THE OIIS ARCHITECTURE

Although many researchers believe that the modularity provided by the production rules [17, 29, 30], is sufficient to make the design of knowledge system structured, the nature of the organizational problems/applications argue for a clearer conceptualization of IIS architectures. We have taken the position that a formalism which supports interactive communication among small autonomous knowledge systems is a viable alternative for IIS development.

In our approach each system is capable of communicating and negotiating with others. There are no global objects, the only common feature among the systems is their ability to communicate. It follows then that the fundamental element of our approach is the systems and syntax for communication.

3.1. Communication syntax and semantics in the OIISA

The theoretical foundations of our approach is based in a speech act theory proposed by Searle [31]. A discussion of this theory would lead too far afield, the reader may however wish to examine research on the utilization of speech acts in information systems [13, 32, 33], in artificial intelligence [34, 35], and in knowledge systems [36, 37].

Message acts [MACTS] as we choose to call our objects of communication derived from speech acts [SACTS]. They embody all the necessary protocols for communication among intelligent information systems. Each act has a semantic connotation depending upon the associated illocutionary act (1) and its intended perlocutionary effect ((2) IPE) [38], which distinguishes it from other computer communication protocols. There are two aspects involved in the message act development.

1. A set of possible MACTS underlying the use of particular kinds of SACTS, and conditions under which MACTS of those types are appropriate, i.e. proper classification of MACTS. This classification would state conditions that involve the speaker’s beliefs and the hearer’s beliefs, goals and inferential process.

2. How an ideal speaker/hearer chooses one or perhaps more than one MACTS out of a possible set of MACTS. This means how a speaker decides which MACTS to perform, thereby enabling a specific behavior from the hearer.

The general format of a message act is:

OPERATOR (SPEAKER, HEARER, ACTION)

Where OPERATOR is a particular revised SACT or a classified illocutionary act: SPEAKER is the sender (KS) of the message; HEARER is the receiver (KS) of the message; ACTION is the specific action(s) that SPEAKER wants either HEARER to perform or know about.
Fig. 1. Conceptual structure of OIISA; DSKS, domain specific knowledge system; CONCEPT-D: concept dictionary; COMM-D communications dictionary.

For a successful communication to take place among the systems it is important that each has a knowledge of the existence of the other. This means that the speaker should be aware of the various hearer(s) with whom it is possible to communicate. Also "when" to call with depend upon the requirements of the speaker. It is the issue which should be resolved at the knowledge acquisition stage. The speaker and hearer are interchangeable, depending upon the requirement. The various message acts and the action(s) proposed or undertaken by them are as follows.

3.1.1. Asserts (S, H action). This corresponds to the assertive illocutionary act and its IPE. S presents some state of affairs (Action). It tells to H how the world is, and orders it to accordingly change its world. H will have to act on Ss belief expressed through this Action, e.g. the price of product X is Y.

3.1.2. Inform (S, H action). This corresponds to the commissive and declarative illocutionary act and their IPEs. S commits itself to a certain course of action (Action) and wants H to be aware of it. Here S is doing some work (Action) and H is just informed about it. H will record the Action, but may also react to the information provided by S.

  e.g. the order shall be fulfilled two weeks;
  e.g. we classify you as a credit customer.

3.1.3. Requests (S, H, Action). This corresponds to the directive illocutionary act and its IPE. S requests H to do some course of action (Action) reply back

  e.g. we can order product X in quantity Y.

3.1.4. Gratitude (S, H, Action). This corresponds to the expressive illocutionary act and its IPE. Here S is expressing some gratitude or disappointment to H on some work done by H. H does not have to record the operation. Action could be thanks, sorry, etc.

  e.g. thanks on the occasion of having orders increased by 20%.

3.2. The OIIS architecture

The basic approach to the open intelligent information system architecture is to partition the entire IIS into domain specific knowledge systems DSKS (cf. Fig. 1). These DSKS communicate and collaborate with each other to solve general business information management problems. Message acts from the basis for communication and collaboration among the DSKS. The MACTS concepts offers a powerful, easy to implement architecture for the design and development of OIIS.

The characteristics of the OIIS architecture are as follows:

1. Instead of having a single knowledge base [(a) la Bubenko], the entire KB is partitioned into different and smaller DSKS. This would help reduce complexity and being modularity into the development process.

2. The overall information system may have one or several user-interfaces in such an OIIS depending on the needs of the organization. The architecture could support IISs in a single or geographically distributed environments.
3. The user-interface will interact with the relevant DSKSs inference engine before accessing the knowledge base. The user interface is also DSKS.

4. There will be a module called the concept dictionary, with which all DSKSs and the user-interface will interact. This dictionary is a compendium of the various concepts handled by the different DSKSs of the IIS.

5. There will also be a module called the communications dictionary, which will contain the generalized procedures for the execution of MACTS.

6. Usage of specific MACTS by any DSKS in the OIISA will be embedded in the knowledge-rules of its knowledge base. Whenever specific work has to be done from the other DSKSs, the calling DSKS will generate the specific MACTS to the called DSKS.

7. Communications between the DSKSs will always be via the communications dictionary.

8. Whenever any DSKS desires to know which concept is being handled by which other DSKS, it will access the concept dictionary.

9. The inference engine of each DSKS will determine the control strategy for the handling of a DSKSs knowledge base and finding solutions.

10. Users can specify their interaction with the OIIS through the user-interface in the following ways:
   (a) Specify a precipitate (concept) as a basis for question or confirmation of some work.
   (b) Specify the MACT directly as a basis for updating the knowledge base of any DSKS.
   (c) Deleting any knowledge from the knowledge base of a DSKS.

11. As each DSKS is a knowledge system in itself, it can have its own knowledge representation scheme for representing knowledge. In such a multiple knowledge representation environment, the inference engine of the DSKS will have the responsibility of transforming the external knowledge format to its internal format and vice-versa when communicating externally. Alternatively one can have a standardized knowledge representation scheme for communication purposes which the inference engines of the different DSKSs understand.

3.3. The description of the prototype

The scenario has three knowledge systems viz. marketing, accounting and inventory. These DSKSs contain all knowledge pertaining to their respective domains. How for instance, these DSKSs interact for the processing of a customer order in an OIIS architecture is described below (cf. Fig. 2).

1. The user-interface receives an order for some product $P$, from customer $C$, for some quantity $Q$.
2. The user-interface checks with the concept-dictionary and transfers order-processing to marketing DSKS.
3. Rules for order-processing in marketing-DSKS are activated, and the marketing DSKS:
   (i) issues a request (MACT) to accounting DSKS, regarding the financial standing of the customer $C$, if $C$ is not a new customer;
   (ii) does some internal processing and then checks with the concept-dictionary to find the DSKS handling product $P$, i.e. inventory-DSKS, and generates an assert (MACT) to inventory DSKS to ship the order. Also details of the order are intimated by inform (MACT) to the accounting-DSKS.
4. The inventory-DSKS does some internal processing, and then "ships the order" and intimates by inform (MACT) the accounting-DSKS and marketing-DSKS about it.
5. Now, upon receiving intimation (from inventory-DSKS) of the shipping of the product $P$, the accounting-DSKS verifies whether it had an earlier information about this order. Since marketing-DSKS had already informed accounting-DSKS about the order, the invoice is generated.

3.4. Implementation specifics

Reproducing all the rules involved in the prototype would be unduly burdensome to the reader. However the following features written in prolog (the language used to implement the prototype) should give the reader an idea of how the OIIH architecture is supported. We conceptually organize the discussion as follows:

1. Rules incorporating MACTS within the DSKSs.
2. Entries in the communications dictionary.
3. Entries in the concept dictionary.

3.4.1. Rules incorporating MACTS. Whenever MACTS have to be used, they should be incorporated in the rule during its development. In other words, MACTS should be used as a natural communication entity within the logic of the specific rule. Below are sample rules having MACTS within themselves. All clauses have a prefix specifying their DSKS identification.

%Sample Rule in Marketing Knowledge Base%

```
mkt:order(Cust,Product,Qty,Dt) if
  mkt:gen_custno(Cust,Cust_No),
  com:com(cust-fin_status,DSKS1),
  mact:request(mkt,DSKS1,custfin_status(Cust-No,Cust-Status)),
  CustStatus = good,
  mkt:order_entry(Cust_No,Product,Qty,Dt,New_Ord_No),
  mact:inform(mkt,acct,custrd(New_Ord_No,Cust,Product,Qty)),
  com:com(Product,DSKS2),
  Temp = ..[Product,Cust_No,Qty,New_Ord_No],
  mact:assert(mkt,DSKS2,Temp).
```

The above rule is invoked if the user specifies an “order” with personal name (Cust), product name (Product), quantity (Qty) and date (Dt). The clauses within the body of the “order” rule are then invoked in order. Initially the clause “gen_custno” generates the customer number (Cust.No) associated with the customer (Cust). Then the rule interacts with the concept dictionary to find the DSKS handling the financial status of the customers (cust_fin.status). Once this DSKS has been identified a “request” message act is issued to it, and depending on the reply (Cust.Status) further processing is done. If the reply is “good”, the order details are entered in the existing DSKS i.e. mkt by the “order_entry” clause. Thereafter an “inform” message act is issued to the accounting DSKS (acct), informing about the customer order (cust_ord). Now once again the rule interacts with the concept dictionary to find the DSKS handling the shipping of the specific product (product1), whence an “assert” message act is issued for the actual execution of the order.

%Sample Rule in Accounting Knowledge Base%

```
acct:cust.fin.status(Cust.No,good) if
  acct:unpaid.balance(Cust.No,0).

acct:cust.fin.status(Cust.No,bad) if
  acct:unpaid.balance(Cust.No,1).
```

The above rule is invoked if somebody wishes to know the financial status of a customer (Cust.No). The status is “good” if the customer has no unpaid financial balances (unpaid.balance) pending. The status is “bad” if the customer has any unpaid financial balances.

%Sample Rule in Inventory Knowledge Base%

```
inv:productl(Cust-No,Qty,Ord_No) if
  inv:productl_balance_STOCK(Qty),
  inv:load_productl_shipment(Cust_No,Qty,Ord_No,Dt_Send),
  mact:inform(inv,mkt,mkt_ord_ship(Ord_No,Cust_No,Dt_Send)),
  mact:inform(inv,acct,acctord_ship(Ord_No,Cust_No,Dt_Send)).
```

The above rule is invoked if “product1” is to be shipped. Initially the product1 stock is updated by the “product1_balance_STOCK” clause. Then the clause for product shipment is invoked, i.e. “load_productl_shipment”. As the product is shipped, “mkt” and “acct” DSKs are informed about this shipment by issuing the respective “inform” message acts.

3.4.2. Entries in the communications dictionary. Communications dictionary will contain the necessary procedures for implementing the message acts. Each procedure is in the form of a prolog rule.

%ASSERT Message Act Structure%

```
mact:assert(Speaker,Hearer,Action) if
  mact:check-action1(Action,Check_Status),
  Check_Status = good,
  call(Hearer:Action).

mact:assert(Speaker,Hearer,Action) if
  mact:check-action(Action,Check_Status),
  Check_Status = good,
  not(call(Hearer:Action)),
  write(‘improper ASSERT message act - concept non-existing’).
```
The function of "assert" message act is to initially check if all the arguments in the "Action" predicate are non-variables by the "check_action1" clause. If this results favorably (Check_Status = good), the appropriate Hearer DSKS is either:

(i) appended by the "Action";

or

(ii) a concept similar to the "Action" invoked.

Again the reason for having non-variables as arguments in the "Action" predicate is to ensure that the knowledge is exact in its structure and the "Action" fully reflects the Speaker's world. Basically the Speaker is intimating to the Hearer some piece of knowledge (the background of which is part of the Speaker's world), which the Hearer may or may not use in future.

The function of "inform" message act is to convey the Speaker's feelings. Since in a computerized environment, there are no human feelings involved, this message act has limited significance.

3.4.3. Entries in the concept dictionary.

Concept dictionary is an index which contains entries specifying the DSKSs which deal with various concepts in the overall system. It provides a basis for integrating the disparate DSKSs.

%Concept Dictionary Structure with sample entries%

com:com(order,mkt),
com:com(cust_fin_status,acct)
com:com(product1,inv).

All entries are clauses with predicate "com" and two arguments. The first argument specifies the concept and the second argument specifies the DSKSs handling the particular concept. It is possible that the same concept be handled by more than one DSKS: in which case the Speaker will have to decide as to which Hearer DSKS will best serve its purpose.

4. CONCLUSIONS AND IMPLICATIONS FOR FUTURE WORK

We have presented an open architecture for IIS and KS which is based on communication protocols rooted in ordinary language theory. The primary contribution of the work reported here are: (1) It offers the possibility to standardize open communication which had been a problem for IIS/KS: in so doing it has brought the concept of open systems closer to reality. (2) The conceptual structure of the
architecture provides a basis for partitioning IIS/KS in to smaller subsystems. This helps to solve two problems (a) complexity in large knowledge-bases and (b) it allows flexibility in design and development of IIS/KS. Applications can be developed and implemented as a set of unique modules without regard to order. (3) The conceptual structure also allows for the specification of varying levels of intelligence within the same system. This has implications for graceful degradation of systems during operation.

Currently research is in progress to: (a) extend the conceptual framework of the architecture [37], (b) develop methods and techniques to support structured IIS/KS design and development [39], and (c) the development and testing of other prototypes.

REFERENCES