STATIC AND DYNAMIC INTEGRATED EXPERT SYSTEMS: STATE OF THE ART, PROBLEMS AND TRENDS

Galina Rybina, Victor Rybin

Abstract: Systemized analysis of trends towards integration and hybridization in contemporary expert systems is conducted, and a particular class of applied expert systems, integrated expert systems, is considered. For this purpose, terminology, classification, and models, proposed by the author, are employed. As examples of integrated expert systems, Russian systems designed in this field and available to the majority of specialists are analyzed.

Keywords: integrated expert systems, real-time, simulation modeling, object-oriented model, rule, complex engineering systems, software tools, task-oriented methodology.

ACM Classification Keywords: I.2.1 Artificial Intelligence: Applications and Expert Systems

Introduction

In the mid-1980s and early 1990s, the scientific and commercial success of expert systems (ES), one of the rapidly developing directions of artificial intelligence (AI), showed that, in actual practice, there is a sufficiently large class of problems that cannot be solved by the methods of conventional programming (for example, it is impossible to formulate the solution of the problem in mathematical terms in the form of a system of equations). This class is rather important, since the overwhelming majority of problems are very important in practice. In other words, if, 20 years ago, 95% of all problems solved by computer methods were problems that had an *algorithmic solution,* (hen, under modem conditions, especially in the so-called *describing application* domains (medicine, ecology, business, geology, design of unique equipment, etc.), the situation is completely different—*nonformalized problems* (NF-problems) make up a considerable amount of problems. These problems have one or several of the following characteristics: they cannot be given in a numerical form; the goals cannot be expressed in terms of a strictly defined goal function; and an algorithmic solution exists, but cannot be used due to the limited resources (time and/or memory).

However, NF-problems are not isolated from formalized problems (F-problems) as it is supposed in the basic concept of ESs (KNOWLEDGE + INFERENCE = EXPERT SYSTEM), but are a constituent of problems of real complexity and importance that are solved by conventional methods of programming. In practice, especially taking into account the constantly growing complexity of modern technological and management-technological systems and complexes, this results in combining in the framework of a 'joint programming system such diverse components as ESs and DBs comprising engineering, manufacturing, id management data; ESs and applied software packages (ASP) with developed computing, modeling, and graphical tools; ESs and learning systems including ring and learning systems; ESs and simulation sys-s for dynamical applications; ESs and developed hypertext systems; etc.

It is worth noting that *integration processes* that initiate important new classes of software systems are characteristic not only of ESs, but of all AI. They considerably expand the theoretical and technological tools are employed in designing *integrated intelligent stems* of various types and destinations, which was stressed in the papers of Pospelov, Popov, Erlikh, Tarasov, Kusiak and, later, Val'kman, Emel'yanov and Zafirov, Vittikh and Smirnov, Fominykh, Kolesnikov, Jackson, as well as other Russian and foreign specialists that have considerably contributed to the investigation and concept formulation of integrated intelligent systems.

It is worth noting that the trend towards integration of the AI paradigm with other scientific-technological paradigms when designing applications of intelligent systems has activated processes of so-called *hybridization* connected with designing *hybrid methods of knowledge representation*. The ideas of designing various hybrids and hybrid systems have been known for a fairly long time in AI. However, real practical applications have

only recently emerged. For example, the design of *neural expert systems,* that combine neural network methods and models for searching for solutions with mechanisms of ESs based on expert (logical-linguistic) models of knowledge representation and models of human reasoning became possible due to modern platforms and tools like Neur On-Line (Gensym Corp). However, despite successful individual theoretical investigations and implementations, there is no adequate theory and technique *of integration* and *hybridization* in AI, which partially explains the terminological confusion in using such terms as *integrated system* and *hybrid system* [Rybina,2002].

Thus, the trend towards integration of investigations in different fields in recent years has led to the necessity of combining objects, models, concepts, and techniques that are semantically different. This has inevitably generated both completely new classes of problems and new *architectures of software systems* that cannot be implemented by a straightforward usage of the methodology of simple ESs and tools like the *shell* of ESs that supports this methodology, which is inefficient, labor consuming, and expensive [Rybina,2004].

However, despite the obvious advantages of the design of IESs that can solve real practical problems in the framework of a unified architecture of a software system ensuring efficient data processing based on the interaction of logical-linguistic, mathematical, informational, simulation models, etc., this problem has not left the stage of problem statement.

Isolated attempts to solve the mentioned problems for IESs have been made both abroad and in Russia. However, most of the investigations and developments encompassed only part of these problems and cannot' pretend to propose a particular methodology for designing IESs. Among the Russian fundamental investigations in the field of producing elements of the IES concept, the papers of Erlikh and, from the standpoint of the methodology and technology of designing the software of conventional ES, the results of Khoroshevskii are the best known. Among the foreign works, we note the G2 tool system (Gensym Corp.), which can be used for designing dynamical IES. However, this system does not support a number of important tasks connected with computer knowledge acquisition and automatic generation of a knowledge base (KB). It also does not deal with the problems of designing applied IESs for various classes of tasks typically used in the IES concept.

For the authors of this paper, static IESs became an object of research and development in the late 1980s [Rybina,1997]. Later, in the mid-1990s, I started my research in the field of real-time dynamical IESs (RT IESs) [Rybina,Rybin,1999]. During this period, based on the experience of designing a number of IESs (from simple static IESs and others to dynamical RT IESs for such important and resource-consuming tasks as diagnostics of technological objects, management of industrial complexes and control of technological processes, ecological monitoring, etc.), a scientific direction that has its own object of investigation, conceptual, theoretical, and methodological basis has been formed. A set of methods and software tools (the AT-TECHNOLOGY system) that enables us to support the complete life cycle of automatic designing IESs for particular classes of tasks of solved problems and types of application domains on the basis of a particular task-oriented methodology of designing IESs in static and dynamical application domains has been developed [Rybina,2004].

Characterizing the proposed methodology in short [Rybina,1997], it is worth noting that it is based on a multilevel model of analysis of the integration process in IESs, modeling of particular types of tasks, the relevant techniques of conventional ESs (a task-oriented approach), and the methods for designing a model of the architecture of IESs and the corresponding software components at each integration level. This experience allows one to analyze rather systematically and visually the basic types of integrated intelligent systems (that comprise components similar to simple ESs) irrespective of the specific features of the particular application domain employing only a principle conditionally called "from the standpoint of the architecture."

However, the goal of this paper is not to attempt to make an *inventory of* all types of existing systems of the IES class. The survey of functional, structural, and, partially, programming specific features of various architectures of IESs (using examples of Russian systems) proposed below, in addition to presenting the state of the art for the field, is mainly directed towards finding ways for overcoming the technological lag in designing ESs described above as compared with modern technologies of designing and programming informational systems. This survey may also be useful in order to help to solve the problems of mutual semantic adaptation of scientific terms and notions that can be used by specialists in different fields in the development of applications on the basis of ESs.

1. Classification of IESs, the Interrelation of the Integration and Hybridization Processes in IESs

At present, we can observe two main processes in AI. One of them is connected with the expansion of architectures of intelligent systems by supplementing them with new tools from other scientific fields (the *integration process).* The other process is associated with the initiation of new subfields from known fields and the development of special new tools for them (the division process). No doubt, *integration processes* are of the most practical importance, since they join different approaches and paradigms, and, as a consequence, totally new classes of software systems.

An IES is a software system whose architecture, together with a conventional component, an ES that, as a rule, uses the ideology of simple production ESs in order to solve NF-problems, comprises certain components "N" that *extend functional capabilities* of ESs (e.g., a database, packages of application programs, learning systems). Therefore, all IESs can be split into two subclasses: IESs with *shallow component integration* and IESs with *deep component integration*.

Then, in the case of shallow integration of the components of ES and N components, the interaction between, e.g., the DBMS and ES can be carried out at the level of sending messages, and the results of the operation of each component are the source data for the other. In a more complex variant of the shallow integration, one of the software components (ES or N) calls the other in order to refine data or to solve some tasks. In this case, a closer interface is implemented, e.g., with the help of special programs that provide a bridge between the DBMS and ES for messaging. Examples of IESs with shallow integration are presented in Section 2.

The deep integration of ES and N components means that these components are improved by functions that are not characteristic of the conventional types of these components. For example, if we solve the integration problem improving the shell of an ES by including certain functions of a DBMS, this may mean that we set a goal to incorporate a generalized DBMS inside a knowledge base (KB). This approach is the most interesting if the KB is too large to be stored in the computer memory and when efficient mechanisms of data access are required. On the other hand, in relation to an improvement of the DBMS by incorporating basic functions of ESs, we should keep in mind that the DBMS has a developed specialized mechanism for executing such tasks as data storage, maintenance of data integrity, knowledge base navigation, and efficient servicing of complex queries. In addition, this mechanism can also be supplemented by functions of the ES. In particular, one of the approaches to solving this problem is to extend the DBMS in order to transform it into a generalized inference mechanism. It is worth noting that, for designing an IES with a deep integration of components, the approach associated with improvement of tools for ESs by including unconventional functions seems to be more preferable, which, in particular, was employed in the development of the task-oriented methodology for designing IESs.

Complete integration is the highest level of integration for IESs and joins the best properties of the ES and N components. For example, the complete integration of a DBMS and an ES consists in selecting the best specific features and mechanisms for designing the DMBS and the advantages of the ES and in designing new data and knowledge models in order to develop completely new systems with new capabilities. The undoubted advantage of this approach to integrating DBMS and ES components is that all components belong to one system with a unified structure for modeling facts and rules and uniform data and knowledge processing.

The so-called *multilevel model of integration processes* proposed in the framework of the task-oriented methodology allows one to analyze the trends towards integration and hybridization in modern IESs. This methodology considers the integration processes from the standpoint of the following key aspects:

- integration in the architecture of IESs of different components that implement both F-problems (the N component) and NF-problems {the ES component) and determine the specific features of functioning of the IES *(high level of integration);*
- integration *(functional, structural, and conceptual)* connected with the concepts and methodologies of designing and developing particular classes of IESs and their components *(middle level of integration);*
- integration (informational, programming, and technical) connected with the technologies, tools, and platforms (low level of integration).

For at least three levels of integration in IESs, this model allows us to trace the interrelation of particular methods of integration of some components of the IES (N or ES) with the necessity of internal integration,

i.e., *hybridization* of some models, methods, procedures, algorithms, etc. In this connection, the following conclusions and suggestions have been made:

- the notion of integrated intelligent systems differs from the notion of hybrid intelligent systems;
- an integrated intelligent system must not be hybrid and vice versa;
- an integrated intelligent system is always hybrid only in the case of complete integration of components.

Therefore, the term "integrated expert system" should be used for complex programming systems that join methods and facilities of ESs with the techniques of conventional programming. The term "hybrid expert system" is advisable to use both for ESs and tools for ESs with hybrid methods of knowledge representation (i.e., joining inside one tool different models of knowledge representation and mechanisms of functioning).

These recommendations are apropos and useful, since the lack of semantic unification leads to misunderstandings in the usage of terms by specialists. Some problems in using the terms "hybrid" and "integrated" may be explained by historic reasons connected with the early paper, where IESs were defined as "hybrid ESs."

It is worth noting that the consideration of problems of shallow and deep integration of the base component ES with a component N (where N is a DBMS, ASP, etc.) at the *high* level allows one to construct in general the so-called generalized model of the architecture of the designed system that reflects its structure. The *intermediate level* is of the greatest interest, since, in the framework concepts and methodologies of IESs, it provides an opportunity to consider the distribution of functions among the ES and N components when developing an application, as well as some design decisions over all components of the designed system. In particular, we can consider possible variants of the joint functioning of the ES and N components. There are several possibilities: the ES and N can be included in each other; they can be used in parallel to solve the same problem; they can be partially included in each other; and an intelligent interface between them can be organized. The integration at the *lower level* is provided by the technology and capabilities of the tools used in the course of development.

Let us explain the terms "static IES" and "dynamic IES" beginning from the notions "data domain" and "application domain." It is well known that "data domain" means a domain of human activity that is specially selected and described (i.e., a set of entities that describe the domain of expertise). The notion *application domain* includes the *data domain* plus a totality of problems solved in it. If an IES is based on the assumption that the source data about the data domain that provides a background for solving the particular problem does not change in the course of solving the problem or the problems solved by the IES do not directly change knowledge about the data domain in the course of their solution, then this domain is *static* (i.e., the domain has a static representation in the IES), while the tasks solved by the IES are static. In other words, a *static IES* is supposed to operate in a static *application domain* if a static representation is employed and static tasks are solved. Otherwise, we deal with a *dynamic IES* that operates in a *dynamic application domain.* This means that a dynamic representation is used and dynamic problems are solved.

Below, based on the introduced terms, notions, and classifications, examples and specific features (functional, structural, and, partially, programming) of the design of a number of Russian and foreign IESs are considered corresponding to the types of N components.

2. Analysis of the Specific Features of the Design of an IES

The analysis of foreign and Russian developments of IESs has shown that almost all of them are devoted to the description of applications of IESs that combine in their structure a set of components that implement both F-tasks and NF-tasks; i.e., they are systems with a *high integration level* (an integration within the scope of the general architecture of the IES). Below, we consider some examples.

2.1. Integration of an ES and a DB

It is worth noting that, despite the fact that, historically, ESs in artificial intelligence and DBs have been developed separately, nevertheless, there are many application domains where, simultaneously, both access to an industrial DB and the use of an ES for decision making on the basis of experience or expertise are required. In this connection, the problems of integrating *static* ESs and DBs have been investigated the best. Therefore, at present, there are many publications devoted to a specialized application of IESs that combine the techniques of KBs and DBs. However, most authors have concentrated their efforts on the description of ways of implementing peculiarities of one or another application domain and have paid substantially less attention to the principles,

methods, and models of integration in the framework of ESs and DBs. In this connection, we place the main emphasis only on the papers that are of interest from the standpoint of methods of joining ESs and DBs (DBMSs).

There are two approaches to joining ESs and DBs—*weak and strong coupling*. Strong coupling is used when there is an opportunity to split the operation of an ES into a sequence of steps that require definite predictable and limited information from the DB (e.g., in tasks of medical diagnosis, the information about a particular patient can be requested before the beginning of a consultation). As a result, the opportunity to efficiently arrange the operation of the ES is a merit of this approach, since, in the course of consultations of the ES, there is no need to query the DB.

However, in other application domains, this approach may be impossible and a strong coupling of the ES and DB may be required. This coupling allows one to multiply read and modify the content of the DB. It is quite obvious that a strong coupling imposes more serious requirements on the system; in particular, the rate of transactions between the ES and DB supposes an efficient implementation of this functional component. Among the first Russian systems of this class, the shell for the DI*GEN ES that has a mechanism of strong coupling of the ES and DB is worth mentioning. To implement this mechanism, DI*GEN contains a set of functional primitives that allow one to position the DB, to read and modify data from an arbitrary field of any record of the DB, to add/delete any records in the DB, retrieve data using a condition, etc.

A radically different approach to integrating ESs and DBs was implemented in one of the earliest Russian applied ESs, the integrated intelligent system of the molecular design of physiologically active substances, which was also *hybrid,* since it employed several reasoning strategies and models of knowledge representation. From the standpoint of the technique for joining an ES and a DB, the characteristic specific feature of this system is the use of an original relational DBMS for generating training samples and storing data, since the decision strategy in this IES is based on two complementary concepts of inductive learning by examples.

Among other developments in the field of static IESs of the early period, a large group was presented by the systems, in which the integration between the ES and DB, ES and ASP, and ES and other components was obtained by using integrated tools like the well-known GURU (INTER-EXPERT) system that contains facilities for designing static ESs, built-in DBMSs, electronic worksheets, and graphical packages. In GURU, the *premise* of each rule in the KB can contain working variables with one value, multivalued fuzzy variables, static variables, cells of electronic worksheets, fields of the KM AN relational DB, numerical functions, operators of relations, Boolean operators, operators over numbers, operators over strings, and symbols, thus providing the integration. Thus, using GURU, from the standpoint of the *lower level of integration,* numerous components N can be joined with components of the ES in the scope of one operation, which allows one to efficiently arrange processing of various data in applied IESs both together with the ES and irrespective of it. This also eliminates the necessity to write special bridges for the IES.

In later *dynamical* IESs that operate in real time (RT IESs), the integration of the ES and DB at the high level is based on designing special bridges, e.g., when using G2 (Gensym Corp.), the G2 Database Bridge is used. For example, when designing subsystems for data acquisition and storage for a system of on-line monitoring, the capabilities of integrating G2 and DBMS Oracle through the G2-Oracle (Oracle Bridge) interface were used. In addition, the tools of the DBMS Oracle and the system software support data backup and long-term storage. In G2, the transmission of online data from external sources was supported by the standard interface of G2 (GSI).

2.2. Integration of ESs with Learning Systems

The next large group is represented by IESs whose architecture integrates ESs with various kinds of learning systems. In general, in connection with the problem of designing computer learning systems, which emerged earlier than ESs and have gone a long way from laboratory programs to power commercial systems, we cannot avoid mentioning the following important specific feature. There are two main processes of learning—"learning" and "tutoring"—that substantially affect the approaches of the soft implementation of IESs.

The direction connected with "learning" process *(learning systems)* includes self-learning, learning with a teacher, adaptation, self-organization, etc. Therefore, when designing learning systems, sufficiently strict models (perhaps, hybrid) are investigated and constructed. These models show the ability to adapt to the external world by storing information. However, from the standpoint of this survey, the direction connected with "tutoring" process *(tutoring systems)* is the most interesting. In this direction, models of the transmission of information from a teacher with

the help of a computer are investigated. This is especially interesting, since, in pedagogic, there is no commonly accepted learning theory or formal models of a trained person, tutoring, teaching effects, explanations, etc. Due lo these facts, specialists depend on expert models and tools of their implementation in the scope of IESs.

Tutoring ESs and expert-tutoring systems are the most popular, as well as the entire field called "intelligent tutoring systems" (ITS), among others. It is worth noting that the interpenetrating of integration processes in AI and pedagogic was reflected in ITSs, as well as in tutoring ESs (which can be considered as a subclass of ITSs). This interpenetrating stresses the need for additional components N that can support a *model of a trained person.* Based on this model, the teacher specifies a current subgoal at the strategic level and the components that realize a particular *tutoring model* in the form of *tutoring actions* at the tactical level. These components must also provide an opportunity to the teacher to observe the actions of the trained person and to help him by using the developed interface facilities.

However, none of the existing systems is able to completely implement the ideas of deep and, especially, complete integration of ESs and N components that support the specified models. In the best case, successive functioning of autonomous components that implement the simplest model of the trained person and a number of learning actions (one of which was "training" based on an ES) was provided.

2.3. Integration of ESs with Hypertext Systems

A wide range of papers is connected with the description of various IESs whose architecture includes integration of components of ESs with not only DBs and tutoring components, but components with hypertext facilities (HT-facilities) as well. The appearance of hypermedia and multimedia systems and interactive leaching environments in recent years has led to the necessity of developing a new type of integrated system—*intelligent teaching environments*—that combine the capabilities of ITSs, hypermedia systems, and teaching environments.

Modern investigations devoted to the problems of integrating ESs with HT-systems provide another sufficiently large range of investigations. Their practical orientation becomes more obvious under the mass distribution of Internet technologies, electronic encyclopedias, Internet shops, etc. The deep integration of the components of ISs and HT-tools actually supplement the capabilities of ESs for hypertexting and allow one to make logical inferences in searching for relevant text fragments, especially, in the fields of activity recommended by standard documents and in electronic business. For future applications, IESs of this type can evolve to "intelligent texts" or "expert-texts," and adaptive models of a user can be designed.

2.4. Integration of an ES and an ASP

Let us consider the analysis of processes of integrating an ES and an ASP in an IES. The global problem of designing *intelligent computer-aided design systems* (ICAD) has been of interest for a long time for researchers and designers. The number of publications devoted to this topic is very large. It is the scope of these papers where the problems of deep integration of ESs with developed mathematical, computational, modeling, and graphical tools, implemented, as a rule, in the form of ASP, were first posed. In such complex software systems, a conventional ES should play the role of an *intelligent user interface,* while the decision maker of the ES should service not only the KB, but also be a *monitor* for the ASP, which can be supplemented not only from the external world, but as a consequence of the operation of the ES.

However, despite the large number of papers that describe applied IESs for various but rather narrow applications, the problems of integrating ESs and ASPs are mainly solved in the scope of IESs with *shallow integration,* i.e., at the level of message exchange between components. The PRIIUS system that joins in its architecture an ES, a data retrieval system, and programs in C, Pascal, and FORTRAN based on the principles of complicated shallow integration is a typical example.

A large number of Russian IESs of this type use foreign ASPs and shells of ESs. For example, in surveys, a large group of IESs applicable in problems of analysis, design, and optimization of the structure of industrial structures and objects of building was described in detail. Among these systems are SACON, designed based on a shallow integration of the well known shell EMYCIN and the MARC package of structural analysis; HI-RISE, in which an ES functions in the DICE environment of the graphical interface; PROTEI, in which the ES plays the role of a monitor that provides the organization of the interaction of all subsystems and components of the IES; and BTEXPERT, in which an interactive FORTRAN program is combined with a ES shell written in Pascal/VS and the processing of symbolic data and the interface are implemented by a pair of mutually complementing tools produced by IBM (ESCE and ESDE). For other classes of problems (control problems), such systems as

CACE-III implemented with the help of DELPHI and having access to an ASP of numerical analysis (CLADP, SSDP, SIMMON, and SEPACK) and CASCADE implemented with the help of the DECIDE shell and working with an ASP library that comprises 78 programs of numerical analysis written in FORTRAN are well-known developments.

However, as was shown in this paper, to describe the set of *typical situations* for each type of aircraft, a totality of mathematical models is used in the OOAES. To support these models, an autonomous functional unit, connected with a rule base and a mechanism for making decisions in a rather simple way (like that of GURU) by including in the working memory (data base) of the OOAES elements of the corresponding mathematical models and using them in the left-hand sides of rules-productions, was developed. In (his case, the valuated elements of the working memory correspond to the actual values of the output signals from the unit for supporting models; the mechanism of logical inference in the OOAES—processing rules—initiates the execution of autonomous units of the support of mathematical models and, then, continues the search for a solution in the rule base. Thus, mathematical models are called from the decision-maker at the level of external programs. In this connection, early OOAESs can be referred to as IESs with *complicated shallow integration* of components of ESs and ASPs (in this case, by mathematical models and the facilities for their support) with further perspectives for deep integration of components and, perhaps, hybridization on the basis of hybrid models of knowledge representation and reasoning.

It is worth noting that the ideas of *integration and hybridization of diverse models* in constructing applied intelligent systems (including IESs, stipulated by modeling real objects, setting and solving problems on certain models, integration of models, problems, and solutions obtained) were initially in the background of AI, which has been mentioned several times. The design of a unified informational environment and formulation of nine features of integration allow one to analyze the interaction of a user, complex product, and data resources and processes from the standpoint of design, theoretical, and experimental investigations on the basis of synthesis and analysis of models (and their junction). Nevertheless, despite the general principles proposed and particular examples of implementation, the problem of designing IESs with complete integration of components, i.e., hybrid systems, remains the most complex problem.

2.5. Integration of ESs with Simulation Systems

The developments that join conventional ESs and simulation systems (SSs) in the framework of *dynamic* IESs are the most widespread. This is the case in which the necessity and possibility of integrating methods for solving NF- and F-problems reveal themselves in the most complete way. In particular, these two techniques are considered, their similarities and differences are analyzed, the expediency of their mutual complementing and interaction is substantiated, and classifications of possible approaches and integration of the ES and N components are proposed from the standpoint of the *intermediate level of integration* (built-in systems, parallel systems, cooperating systems, and intelligent interfaces). A dynamic real time IES that comprises N components represented by a subsystem for modeling the external world is a typical example of an IES designed on the principles of integrating ESs and SSs.

It should be noted that dynamic IESs that are used to support solutions of NF-problems and AI problems are the most complex systems from the standpoint of implementation. As a rule, either *complicated shallow integration* of certain components of the IES or *deep integration* is used. The problems of complete integration are partially discussed only at the conceptual level.

The facilities of simulation are incorporated not only in the architecture of IESs. Very frequently, simulation systems are included in CAD systems, since, due to the complexity of the designed system, only simulation methods allow one to use relevant information of various kinds, including exact data and quantitative data, as well as expert, heuristic knowledge-based experience and assessments. In this case, the structure of the CAD system, together with SSs, contains a certain number of ESs and a common DB.

2.6. Integration of ESs with Knowledge Acquisition Systems

The advances in the ES field have led to the growing importance of special methods and software tools for knowledge acquisition (eliciting, extracting), especially in designing large knowledge bases. In other words, the direction of knowledge acquisition, which is conventional and key for AI, is separated out into a particular research field and attracts methods and approaches from other scientific fields such as psychology, linguistics, neural networks, machine learning, mathematical statistics, regression analysis, data visualization, etc.

An analysis of papers devoted to new lines of AI investigation shows that, beginning in 1994, many specialists specify software tools for knowledge acquisition into a separate category, directly connect it with data mining that rapidly develops knowledge acquisition in DBs, and predict great prospects for this field. In some cases, the systems that combine the capabilities of conventional ESs with a component for knowledge acquisition (and, sometimes, even with a DB) are proposed to be called "partner systems". However, there is no information about the methods and ways of integrating the ES and N components in the architectures similar to IESs in that paper.

2.7. Integration of ESs with Other Software Components

Consider other examples of integrating the ES and N components in the architecture of static and dynamic IESs that are not so characteristic. Under modem conditions, the integration processes manifest themselves most visibly in the architecture of systems aimed at reengineering business processes of enterprises. As is known, the main task of business process reengineering is to find a new way of constructing the existing business on the basis of employing the most modern data technologies, including the methods and tools of ESs. Here, it is important to take note of two aspects.

The first is associated with the general methodology of designing ESs, OOA, and business process reengineering. All of these methodologies are based on the same principles and the same life cycle and the models of processes of a new business that are designed in the course of reengineering correspond to analogous models that are constructed when designing software systems for supporting business and can serve as models of application domains developed at the stages of "identification" and "conceptualization" when designing ESs.

The other aspect is the close relation of the conventional paradigm of ESs of the paradigm oriented to the rules with the so-called business rules. The languages for knowledge representation of production types by and large have the required power for the representation of business rules, in contrast to, e.g., the tools of diagram techniques of structural analysis or object data models of OOA. We can describe all language constructions proposed, in particular, by Ross for describing business rules using the facilities of the language for knowledge representation used in the famous ReThink system. However, only preliminary studies have been undertaken in this area.

Conventional decision support systems (CDSS) have great prospects in the field of integrating the ES and N components, as well as in the hybridization of models of knowledge representation and methods of reasoning. This is especially true in connection with their evolution toward designing *intelligent* CDSS, which, as was mentioned by the authors, can be fully implemented only if the modem techniques for designing intelligent systems based on the concepts of distributed AI and dynamic knowledge models and the methods of plausible reasoning are harnessed. It is also necessary to employ powerful computational platforms and corresponding tool systems similar to G2.

Completing the survey, we should note that, in real practice, together with the considered cases, it is possible to find other examples of the use of models, methods, and tools of ESs in combination with conventional approaches and programs, e.g., integration of ESs with geoinformation systems oriented toward processing cartographic data. And, no doubt, the most modern agent-oriented paradigms in AI are based on the further evolution of intersystem integration processes.

2.8. New Architectures of Distributed IESs

From the standpoint of further prospects in the field of development and evolution of IESs, we should notice (he rapid progress in the field of Web-oriented IESs (Web-IESs), which, on the whole, fits the modem trends in designing intelligent systems connected with the transition from isolated autonomous systems with centralized management to distributed integrated systems with network control structures.

Up to now, the methods of interchange between a Web browser and a network has considerably complicated the design of Web-IESs, in particular, the functioning of the inference engine of an ES. However, currently, the technologies that allow one to solve these problems have begun emerging. At present, there are two main types of architectures of Web-IESs, whose specific feature is completely determined by the methods of activating the inference engine of the ES in the network. The first is the "server-master" architecture that stipulates the ES to be run on the server. The second architecture—"client-master"—stipulates the ES to be run on the client computer (local user computer). Let us evaluate the merits and demerits of each of these methods from the standpoint of designing the architecture of a distributed IES.

Despite the certain access rate connected with the availability of all programs required for operating an ES on the server (which allows us to eliminate the loading of (his program on the user computer), the server-master

approach implies a number of inconveniences. First of all, this is associated with the impossibility of meeting a number of conditions of normal functioning of an ES such as the impossibility of saving the history of the system's states and of tracing the steps of inferences made. Moreover, when several clients simultaneously work with the ES, the load of the server and response time to the user queries increase. When using the client-master method, data are processed on the user computer and thus eliminate the problem of storing the states and trajectories of the inference in the ES.

Consequently, each of these approaches has it advantages and disadvantages, and the future will show which variant will be preferable. It is clear that the server approach requires considerable resources on the server, while the design of Java-client ESs shifts all processing problems to the user computer. In addition, in order to prefer some of the considered variants of designing a Web-IES, it is necessary to know the amount and rate of data transmitted by the corresponding N component.

ES and N components are combined at the level of shallow integration by special bridges that can be implemented by standard techniques (COM, DCOM and CORBA), which makes the processes of interaction of an ES with other N components transparent and simple to implement.

At present, a number of foreign Web-IESs an known that are implemented by either the tools of conventional programming such as C++, Visual Basic, Active Server Pages, and Active X or using special tools like KROL (an object language for knowledge representation) or others.

Conclusion

The questions of to what extent modern expert systems and, in a broader sense, *knowledge-based system* are expert and what are the future trends and prospect: of the development of this class of intelligent systems the most popular from the beginning of the middle of the 1980s are of interest for researchers and designer; of ESs, as well as potential users. The euphoria from the first successes in designing conventional ESs for limited application domains was marred by the difficulties and technological problems when trying to solve problems of real practical complexity and importance.

This survey was devoted to one of the most important problems connected with the change and complication of the architecture of modern ESs because of the dominating processes of integration and hybridization (the emergence of integrated, hybrid, and Web-ESs). Naturally, tile limited length of this survey cannot give a complete representation of the variety of types of the existing architectures of IESs and approaches to their implementation, since it is the ES field where the greatest experience has been accumulated. This experience has become an essential part of almost all *integrated intelligent systems.* Note that we were primarily in the analysis of Russian developments in this area using sources available to any specialist that might be interested in the theory and techniques of the entire process of designing intelligent systems.

Acknowledgments

This work was supported by the Russian Foundation for Basic Research, project no. 03-01-00924.

Bibliography

[Rybina,1997] G.V.Rybina. Task-Oriented Methodology for Automated Design of Integrated Expert Systems for Static Application Domains, Izv. Ross. Akad. Nauk, Tear. Sist. Upr., 1997, no. 5.

[Rybina,2002] G.V.Rybina. Integrated Expert Systems: State of the Art, Problems, and Trends, Izv.Ross.Akad.Nauk, Teor.Sist.Upr., 2002, no. 5.

[Rybina,2004] G.V.Rybina. The new generation of software tools for application intellectual systems development, Aviakosmicheskoe Priborostroenie, 2004, no. 10.

[Rybina,Rybin,1999] G.V.Rybina, V.M. Rybin. Real-Time DynamicJB Expert Systems: The Analysis of the Research Experience and Developments, Prib. Sist. Upr, 1999. no. 8.

Author's Information

Galina Rybina – Moscow Engineering Physics Institute (State University),Kashirskoe shosse, 31, 115409, Moscow, Russia,Email: galina@ailab.mephi.ru

Victor Rybin – Moscow Engineering Physics Institute (State University),Kashirskoe shosse, 31, 115409, Moscow, Russia,Email: rybin@aie.mephi.ru