Cognitive Computing: Principles, Architectures, and Applications

Jerzy W. Rozenblit
Professor and Head
Dept. of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona 85721-0104, USA
Email: jr@ece.arizona.edu

KEYWORDS
Cognitive computing, agents, high autonomy systems, simulation-based design

ABSTRACT

This paper is a summary of the plenary presentation. The objectives of the presentation are threefold: a) to discuss conceptual foundations of cognitive computing, b) to demonstrate their impact on intelligent systems design, and c) to present a brief summary of relevant project experiences. An introduction to knowledge-based and cognitive systems, and the explanation of their origins and principles are given. Then, an agent metaphor is introduced as the basis for design of high autonomy, cognitive architectures. Examples of projects from both industry and research laboratories that leverage from the above concepts are discussed. Some recent work that focuses on decision making in complex, information rich environments, multi-agent gaming models, and implementation of symbolic representation techniques in highly flexible, reusable, object-oriented visualization systems is presented.

INTRODUCTION AND MOTIVATION

Cognitive computing is an emerging approach that builds upon a wealth of research and development work in Artificial Intelligence (AI). It strives to provide methods to construct and operate systems that "know what they are doing" (Brachman 2002). From a perspective of practicing modelers and systems engineers, the primary motivation behind adopting cognitive methods is to better support the design and deployment of complex, intelligent systems.

It is also the systems' complexity that motivates us strongly to develop new integration techniques that help achieve high levels of autonomy and intelligence. As modelers and designers, we are excellent at constructing system modules and subcomponents. However, we often falter at the integration of those components not just in a structural, but also in a functional sense. For several years now, the Defense Advanced Research Projects Agency (DARPA) has been driving an effort to build systems that are able to acquire and accumulate knowledge, reason, learn, explain themselves, and be aware of their own behavior (and be robust).

Clearly, these are very highly sophisticated objectives and, realistically, there is currently no artificial system that can exhibit that kind of a complex, integrative behavior.

From an engineering perspective, computer-aided support at the higher design level is urgently needed. Whereas excellent support exists at lower design levels — for instance, in circuit, or VLSI design — support for integrating hardware and software components at higher system levels is poor. Thus, our desire is to develop adequate modeling tools that support the development of complex heterogeneous systems, allow for reuse of models and modules, and help us in rapid prototyping.

In the following sections, we examine how the cognitive techniques could help us accomplish those goals. We begin with a discussion of cognitive systems and their underlying AI paradigms.

COGNITIVE SYSTEMS

The origins of cognitive systems work lie in cognitive science — a discipline that brings together researchers from the fields of psychology, linguistics, philosophy, computer science, and more recently, neurocomputing. We perceive "cognitive computing" as an approach that has emerged from, and attempts to subsume, the work done in AI. Given the computational power that we now have at our disposal, we are able to explore complex cognitive issues paradigms and supplement the often imprecise methods used in psychology by rigorous modeling. We can implement a lot of theories now in a computational mechanism that allows us to solve these problems computationally, not just necessarily analytically as has been tackled in the past. We could thus say that that cognitive systems are systems that understand, seek to understand how we perceive, how we think, remember, learn, and form models.

If we take a "computational approach", we can view cognitive systems in an information-processing context. More specifically, we might see them as a kind of input, output, and transition systems. Such systems are well described by an agent metaphor, i.e., a system that perceives its environment, processes information, and takes actions that affect the environment. The classical definition of an agent stipulates that it be an entity

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

capable of information processing at various levels of sophistication and able to affect the world in which it operates (Russell and Norvig 1995). This general metaphor is depicted in Figure 1.

An agent could be a robotic machine, a segment of software, etc. Several examples are given in Table 1. Agents are typically given specific goals and act in a purposeful manner. The goals drive the behaviors and allow us to generate metrics that assess how good these behaviors are. For instance, in a medical diagnosis system an agent would be perceiving symptoms, findings, and data that are gathered through interviewing the patient. The actions would be more questions, perhaps a deeper type of investigative technique, medical tests and treatments. The goal here would be a successful treatment outcome, that is a healthy patient with be a normal range of particular test values. The systems shown in the table can all be called agents. The question arises as to what degree of cognitive sophistication they exhibit. While we do not believe that computer programs or artificial systems that "know what they are doing" exist, we could argue that many systems do exhibit "knowledgeable behaviors". Thus the question that we want to answer is: "How can we tell that intelligence has been achieved or is being exhibited by an artificial system?

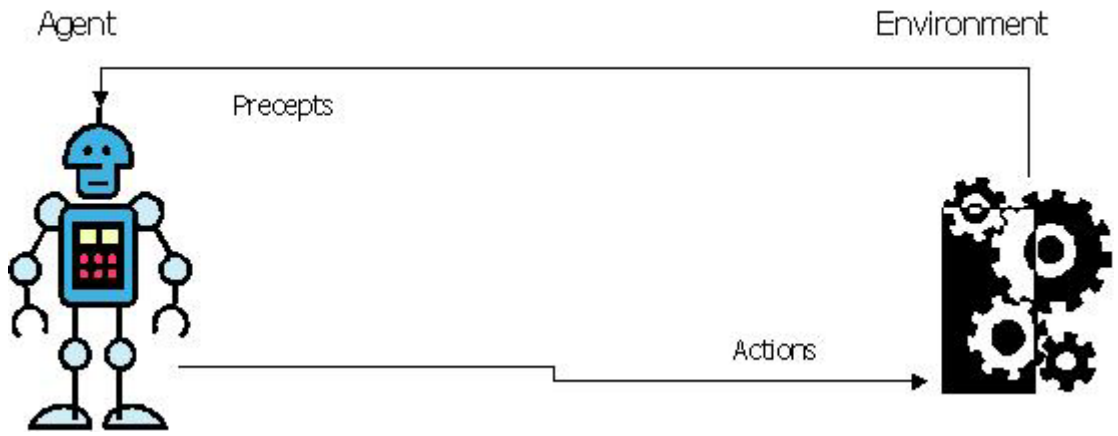

Figure 1: Agent Metaphor

Table 1 Examples of Agent Systems (adopted from "Artificial Intelligence: A Modern Approach", S. Russell and P. Norvig)

Agent Type	Percepts	Actions	Goals	Environment
Medical Diagnosis System	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize cost	Hospital, patient
Satellite image analysis system	Pixels of varying intensity, color	Categorization of scene	Correct categorization	Image processing computers/satellites
Part picking robot	Pixels of varying intensity	Pick parts and sort into bins	Place parts in correct bins	Manufacturing system
Reactor controller	Temperature, pressure readings	Open, close valves, adjust pressure, water temps.	Maximize safety, power	Reactor

Attributes of Intelligence

To determine if a machine is intelligent, classically the Turing test is carried out in which the machine is called "smart" if its performance cannot be distinguished from that of a human performing a task. The fallacy of this approach is that systems can be programmed that mimic human behavior without actually exhibiting any cognitive skills. (A good example was the Eliza system that emulated behaviors of a psychoanalyst by simply analyzing the syntax of patients' complaints (Weizenbaum 1966)).

Perhaps a broader test for discerning intelligence would be to ask what are the marks of intelligence. For instance, we might consider the following as representative attributes of intelligence.

We clearly have perception — our desire here is to build agents that are able to perceive. We perceive, we are able to recognize, we are able to classify and abstract certain common properties. We have mental states, in other words, we are thinking about something. We have certain beliefs and we could say that we believe something is true or false. We do learn (and so do animals). Here, we could argue that what clearly distinguishes us from other living beings is the ability to acquire knowledge, ability to improve that knowledge, and the ability to use it to solve new problems; that is something that machines do not do well.

We use language to communicate and disseminate knowledge in a purposeful way. And last but not least, we create models and use them to predict consequences of our actions and to explore our potential choices in an almost limitless way.

AI have so far achieved many of the above marks of intelligence in an isolated form. However, integrating those abilities in an artificial systems is a formidable goal. Ultimately this should be the objective behind the development of innovative cognitive computing architectures that can accomplish not necessarily the level of a created genius, but a level of a highly cognizant intelligent entity.

Tools for Cognitive Systems Design

A wealth of AI methods and tools exist to assist us in the design of cognitive systems. In the presentation, we will examine in detail a number of approaches. The fundamental areas from which we draw in our practice are state space-based search and problem solving, knowledge representation (KR), rule-, and model-based reasoning, genetic algorithms and co-evolution.

In "the sciences of artificial" where most of the engineering systems are conceived and constructed, the state space approach is a rigorous method that allows us to represent the underlying problems and to solve them using efficient (often heuristic) methods. Many of the computational problems we face lend themselves to the following paradigm: the system that we build or analyze can be in a finite number of states. Then, the task at hand is to transition from an initial state to the goal state. Thus, solving the problem is to find a trajectory or a sequence of state transitions that would take the system to the goal state(s). This is a powerful paradigm, deeply rooted in the classical control and operations research problems. Many of these problems exhibit combinatorial and exponential behaviors with respect to the number of inputs we work with. AI has been extremely helpful in finding heuristic techniques that allow us to solve search problems efficiently.

Rule-, and model-based reasoning provide a repository of methods that give us introspection into the causes and effects when examining systems' behaviors. Traditionally, if-then productions (Russell and Norvig 1995) have been employed as a representational mechanism for encoding condition-action (premise-conclusion) pairs in expert and knowledge-based systems. Model-based reasoning allows for a higher level of cognition in which we built a repository of models which represent world states. Using such models (which have dynamic behaviors), we perform various diagnostic, prediction, and control functions (Zeigler 1984, Rozenblit 1992).

We extensively use genetic algorithms (GAs) and co-evolution (Peng et al. 2003, Suantak et al. 2001) as optimization tools that quickly generate suboptimal solutions when the numbers of solution possibilities are very large. GAs mimic the process of natural evolution where the fittest members of a population cross-over their best "genes", or adapt to environmental changes by mutating some of their gene sequences. GAs are also employed in learning – a process essential to building the agent's autonomy and its ability to improve how it determines its actions (Russell and Norvig 1995).

In our practical experience, we focus mainly on employing these, and other techniques, to design highly complex systems. Our design philosophy is firmly grounded in the simulation modeling enterprise. The following sections give an overview of our modeling approach, summarize some practical experiences, and propose a highly autonomous model-based system architecture.

MODEL-BASED DESIGN

In our previous work (Schulz et al. 1998, Rozenblit 2001), we have developed a process that uses stepwise refinement of simulateable models and abstracts system components at multiple levels of representation. In this methodology, a set of requirements and constraints is obtained for the system to be modeled. The system is then described as an abstract model that is a

combination of its structural and associated behavioral specifications.

Given a set of design objectives, requirements and constrains, we first build a simulateable model of the system under design (SUD). Modeling entails the specification of structure (object model) and behavior (dynamics). Object modeling (i.e., model structuring) typically leads to a specification of a structure instance. This is commonly done in a graphical language such as the Unified Modeling Language (UML), which has become a *de facto* tool for object modeling. However, rather than generating a *single* instance of an object model, we advocate the development of a generative object representation that underlies the entire family of possible design configurations for a problem domain at hand. Indeed, UML allows us to capture the multiplicity of design views and taxonomies (specializations) of components through its decomposition and specialization relationships. An enormous variety of decompositions and specializations in large scale systems leads to a combinatorial explosion of design choices. To harness this complexity, procedures are needed that prune out instances of design which best fit design objectives and requirements. Thus, we use heuristic search methods that convert design requirements into selection (for choices from among alternatives offered by taxonomic relationships) and synthesis (for aggregations from among decompositions) into production rules. Then, we search design spaces for best alternatives. The outcome of the search is a set of sub-optimal instances of design object models (Rozenblit and Huang 1991).

The dynamics (behavior) of model components is specified using various modeling formalisms such as the discrete event system specification (DEVS) (Zeigler 1984), finite state machines, Petri nets, etc. The choice of the specification formalism is based on the system's domain. Both the structural and behavioral specifications constitute a virtual representation of the system under design (SUD). This is a "design blueprint" from which a system will be realized. Model components remain implementation and realization (i.e., hardware or software) independent.

We verify correctness of models through computer simulation. A simulation test setup is called an *experimental frame* (Zeigler 1984). It is associated with the system's model during simulation. A frame specifies conditions under which the model of the system is observed. Simulation is then executed according to the run conditions prescribed by the frames. At the end of the simulation process the "best" (polyoptimal) virtual system prototype is obtained. The design is then partitioned into hardware, software and corresponding interfaces using a process that we call *model mapping* (Schulz et al. 1998). We have applied this framework to design a variety of highly autonomous systems by combining the above simulation modeling principles with the tenets of AI and cognitive systems. Examples are given below.

Some Practical Experiences

Our laboratory conducts research in systems design and analysis, engineering of complex systems, and software engineering. Detailed principles for designing such systems will be shown including a testing methodology that ensures conformance to project's requirements. In the presentation we will show several instances of complex systems. Examples will include a *unified sensing system* model in which configuration, management, and tracking algorithms are implemented over a wireless, multi-sensor network (Vaidya et al. 2005), and a large scale object-oriented system for decision making in complex, information rich situations (such as military, peacekeeping, or disaster relief operations).

The purpose of this latter work is to provide visualization capabilities to decision makers using advanced computer technology that symbolically abstracts the most important features of the information space. The technology facilitates rapid creation of tailored, low resolution, high semantic content visualizations of complex operations. Recent extensions (Peng et al. 2003) include a hybrid software/hardware that builds on the symbolic, object-oriented visualization software.

TOWARDS A COGNITIVE, HIGH AUTONOMY ARCHITECTURE

We postulate that simulation modeling could play the key role in designing highly autonomous, cognitive architectures. High autonomy, defined here as the ability to function with little or no intervention from the "operator", is a mark of cognitive sophistication.

The postulated architecture shown in Figure 2 consists of three major elements: a) the executive layer that comprises the planner and simulation, b) the coordination layer that includes the diagnoser, model base, monitor, and executor, and c) the execution layer that acts upon the real world through the effector, and collects observables through the perceptor.

The planner's function is to generate nominal action plans, given a task or mission description and the world states obtained from the models which reside in the model base. The simulator provides model-based

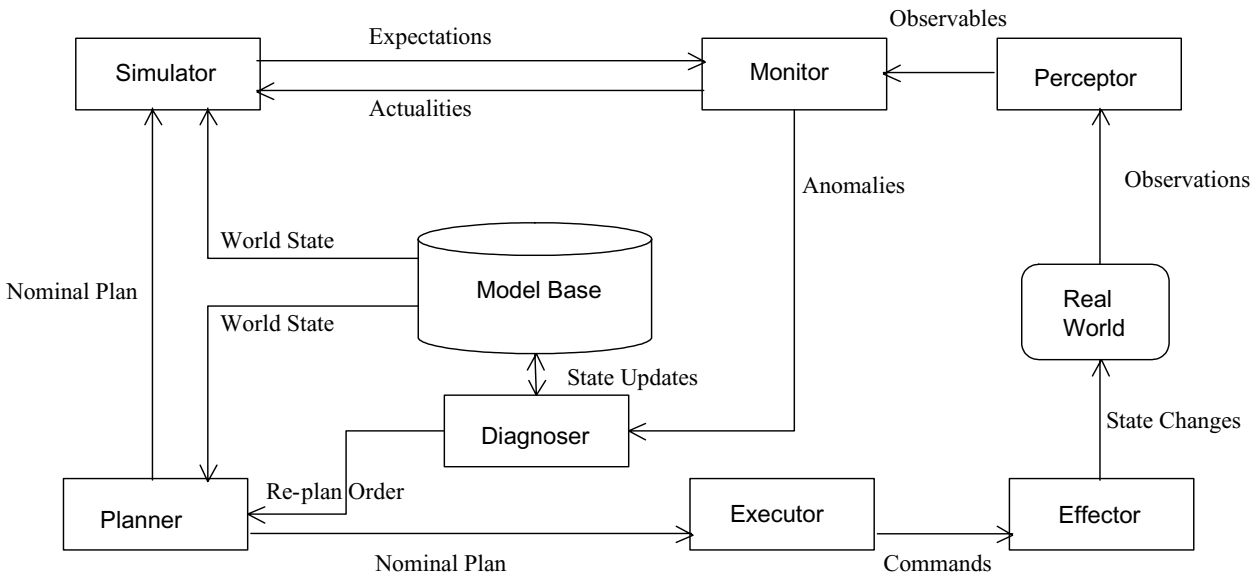

Figure 2: High Autonomy Cognitive Architecture

expectations that are compared with the actual observables in the monitor. Any discrepancies are reported to the diagnoser which, in turn, orders replanning directives.

Increasing levels of autonomy could be defined as: a) the ability of the system to achieve its objectives, b) the ability to adapt to environmental changes, and c) the ability to develop its own objectives. We believe that the model-based approach allows for building such functionality into the architecture presented above (perhaps with the exception of item c.).

CLOSING REMARKS

The notion of cognitive systems and computing is not new. Well established AI-based methods have existed for several decades. However, to a large extent AI has not delivered an integrative capability to build complex systems that combine many of the intelligent features found in isolation in simpler components. We postulate that a simulation modeling approach to design of highly intelligent, autonomous computing architectures is a powerful tool in accomplishing this integration at both structural and functional levels.

ACKNOWLEDGMENT

I am grateful to Ms. Rozanne Canizales and Mr. Jeffrey Peng for their assistance in editing this manuscript.

RERERENCES

Brachman, R.J. 2002. “Systems that know what they're doing”. *IEEE Intelligent Systems and Their Applications,* 17(6) 67 – 71.

Cunning, S.J. 2000. “Automating Test Generation for Discrete Event Oriented Real-Time Embedded Systems.” PhD Dissertation, The University of Arizona.

Peng, J., Rozenblit, J.W. and L. Suantak. 2003. “A Hybrid Architecture for Visualization and Decision Making in Battlespace Environments”, *The Tenth IEEE Conference on Engineering of Computer-Based Systems*, 207-213.

Rozenblit, J.W., 1992. “Design for Autonomy: An Overview”, *Applied Artificial Intelligence*, 6(1), 1-18.

Rozenblit, J.W. and Y.M. Huang. 1991. “Rule-Based Generation of Model Structures in Multifacetted Modeling and System Design,” *ORSA Journal on Computing,* 3(4), 330-344.

Rozenblit, J.W. 2001. “Systems Design: A Simulation-Based Modeling Framework.” In Discrete Event Modeling and Simulation: A Tapestry of Systems and AI-based *Theories and Methodologies.* (Eds. F. Cellier and H. Sarjoughian), Springer Verlag, 107-127.

Russell, S. and P. Norvig. 1995. “Artificial Intelligence: A Modern Approach”. Prentice-Hall.

Schulz, S., Rozenblit, J.W., Mrva, M. and K. Buchenrieder. 1998. “Model-Based Codesign.” *IEEE Computer*, 32(8), 60-68.

Suantak, L., Momen, F., Rozenblit, J.W., Barnes, M. and T. Fichtl. 2001. “Intelligent Decision Support of Support and Stability Operations (SASO) through Symbolic Visualization.” In *Proceedings of the 2001 IEEE International Conference on Systems, Man, and*

Cybernetics, 2927-2931.

Vaidya, D and J. Peng, L. Yang, J. W. Rozenblit. 2005. "A Framework for Sensor Management in Wireless and Heterogeneous Sensor Network." In Proc. of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 155-162.

Weizenbaum, J. 1966. "ELIZA--A Computer Program For the Study of Natural Language Communication Between Man and Machine." *CACM* 9(1), 36-45.

Zeigler, B.P. 1984. "Multifacetted Modeling and Discrete Event Simulation." Academic Press, 1984.

BIOGRAPHY

JERZY W. ROZENBLIT is Professor and Head of the Electrical and Computer Engineering at The University of Arizona, Tucson. He holds the PhD and MS degrees in Computer Science from Wayne State University, Michigan and the MSc in Computer Engineering from the Technical University of Wroclaw, Poland. His research and teaching are in the areas of complex systems design and simulation modeling. His research in design has been supported by the National Science Foundation, Siemens AG, Semiconductor Research Corporation, McDonnell Douglas, and the US Army Research Laboratories.

Dr. Rozenblit serves as Associate Editor of ACM Transactions on Modeling and Computer Simulation, Associate Editor of IEEE Transactions on Systems, Man and Cybernetics, and Executive Board Member of IEEE Technical Committee on Engineering of Computer Based Systems. He was Fulbright Senior Scholar and Visiting Professor at the Institute of Systems Science, Johannes Kepler University, Austria and has held visiting professorship appointments at the Technical University of Munich, Central Research Laboratories of Siemens AG, and Infineon Technologies AG, in Munich. His e-mail is <jr@ece.arizona.edu>.