IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4215

SARA: Smart Agent-Based Resource For Virtual Advising

John R. Durrett
Area of Information Systems and Quantitative Sciences, Texas Tech University, Texas
Tel: (806) 742-2994, Fax: (806) 742-3193, john@durrett.org

Lisa J. Burnell
Department of Computer Science, Texas Christian University, Texas
Tel: (817) 257-6378, Fax: (817) 257-7110, L.Burnell@tcu.edu

John W. Priest
Department of Industrial and Manufacturing Systems Engineering, University of Texas, Texas
Tel: (817) 272-3168, Fax: (817) 272-3407, jpriest@uta.edu

Copyright Idea Group Inc.

ABSTRACT

Few tools are available for managing distributed educational support processes, from answering "quick" student questions, to degree planning and course scheduling. The dual purpose of our work is to analyze the viability of synthesizing two well-established paradigms (Agent-based systems and Information Processing Theory) to create a distributed web-based support system for a virtual university. Currently we are developing a smart agent-based resource for advising (SARA) that will serve different departments at multiple universities. This effort allows the investigation of distributed advising systems in a truly virtual university, along with providing a platform for investigating utilization of brick-and-mortar research on coordination and control of human employees in a completely virtual world. In this article, we present SARA and argue for the benefits derived from integrating agent-based systems development and information-processing theory for creating rapidly customizable web-based support systems for virtual universities.

INTRODUCTION

Problem Motivation

Our motivation for this paper comes from two sources: first, to test the viability of distributed system design steps generated from a synthesis of research in Agent-Based Systems (ABS) and in Organizational Theory (OT); and second, to create a prototype virtual advising system. Given the limited space available in this article, our choice is to focus on the design environment and the design process process and not on the details of the individual agents, or other components, generated.

The growing popularity of distributed education has produced a variety of interesting research in the production and presentation of educational materials. Distributed education consists of much more than simply putting PowerPoint slides on a web site and receiving student homework or term papers by email [1]. At its best it consists of a dynamic environment where students and teachers can truly communicate and both can learn [2]. A viable virtual university also consists of the background coordination tasks necessary to successful educational efforts. Among these tasks is student advising.

Student advising at any university is an extremely complex and time-consuming process that is made up of many sub-tasks. A recent survey conducted at Texas Tech [3] showed that 60% of an advisor's time was spent helping students perform long-term degree planning and current semester scheduling (e.g., course approval). The remaining time was evenly divided among evaluating transcripts; responding to student requests for information concerning degree programs, course content, how to apply, kinds of jobs available to graduates; and mentoring activities. Human advisors are inundated with "quick questions" to the point that there is often little time remaining for other important duties. This limitation is as true in virtual universities as it is in traditional brick-and-mortar universities.

For an advising system of a virtual university to be effective, it must at the very minimum possess the same capabilities as brick-and-mortar universities. To provide this capability we are creating a smart agent-based resource for advising (SARA). Our research in the creation of SARA focuses on a synthesis of advances in software agents and OT. The creation of a truly usable advising system utilizing this synthesis will allow us to investigate the viability of distributed advising systems in a truly virtual university, along with providing a platform for investigating utilization of brick-and-mortar research on coordination and control of human employees in a completely virtual world.

BACKGROUND

In order to deliver effective, efficient services to students who are geographically and experientially diverse, university advising functions need to be accessible and adaptable. This particular task environment is complex, dynamic, and thus interesting for our research because of (1) the dynamic, complex, numerous rules that vary widely across departments and universities, making standard maintenance in conventionally designed systems difficult; (2) the wide diversity of course content; and (3) a heterogeneous user base ranging from brand-new college students to experienced professionals to instructors and advisors. Given this type of task environment most universities offering distance-learning programs still primarily utilize human beings for advising students [4]. To provide the flexibility necessary for our software to adapt to this task environment we are implementing SARA using smart agents guided by business rules. To provide the inter-agent coordination necessary in such a "virtual organization" we are utilizing guidelines from OT research, specifically Information Processing Theory (IPT). While not exclusively for Virtual Universities, SARA is particularly valuable for such environments. We are developing SARA to work in three different departments at three different universities.

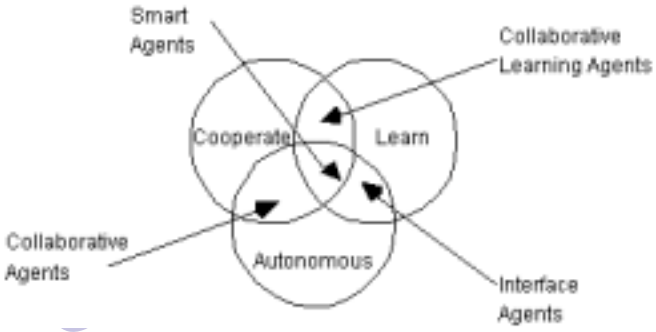
Smart Software Agents

An agent is an entity that represents a user by performing human-like tasks, such as gathering information known to be of interest to a user or responding to changes in the environment to maintain certain pre-defined user goals. Multiple agents cooperate in agent-based systems, performing tasks on the behalf of a community of users. These systems can include combinations of software, hardware, and humans. The development of ABS, particularly in complex, multi-user domains is a challenge. In part, ABS development is difficult because of software engineers' relative inexperience in developing such systems and the complexity of designing asynchronous, multi-user systems.

This paper appears in *Issues and Trends of Information Technology Management in Contemporary Organizations,* the proceedings of the Information Resources Management Association International Conference. Copyright © 2002, Idea Group Inc.

Most of the software available today has been designed primarily to solve problems that were suitable for a single program. Now the focus is on software that will be able to communicate and access multiple programs in order to solve a specific problem that is unable or very hard to solve by a single program. This concept of programming is called agent-oriented programming. The programs are to be able to interoperate. Genesereth and Ketchpel from the Computer Science Department of Stanford University define "interoperate" as the ability to exchange information and services with other programs[5]. The ability to communicate with several programs and platforms creates multiple design problems. Programs written in the several languages, at several time frames, and by multiple programmers are just a few of the difficulties. Designing agent-oriented programs is an attempt to bridge the gaps of these problem areas. Genesereth and Ketchpel[5] best summarize agent-based software engineering as an approach to software development, in which "application programs are written as software agents, i.e., software 'components' that communicate with their peers by exchanging messages in an expressive agent communication language."

Figure 1: Agent typology (adapted from Nwana & Azarmi, 1997)



The blackboard model has been used to approach "ill-defined" problems. As Butterfield and Cooprider (1994) elaborated, "the blackboard model was developed originally as a means of sharing distributed information. It assumes that for many ill-defined problems, solutions are partitioned among a number of different experts or knowledge sources. Rather than presenting a predefined problem and soliciting possible answers, blackboard models prompt the experts for information about the problem…." [Blackboard models] provide an appropriate framework of and system for helping to resolve conflicts in the requirements analysis process." [7]. The blackboard model served as an early model for distributed intelligent problem solving.

More recent approaches are based on distributed systems of rational agents using frameworks such as JADE [8], Jackal [9] and JATLite [10]. Individual agents are conceptualized as having beliefs, desires and intentions that can communicate with other agents to satisfy their goals. In these distributed systems, agents communicate using languages such as KQML or the FIPA ACL [11].

Just as human systems created to achieve complex goals are conceived of as organizations, ABS such as those described above can be reconceptualized as "software organizations". In both types of systems the individual components, human employees or software agents, need to be controlled, guided toward a constructive goal, and coordinated toward the completion of the necessary individual tasks. This conceptualization allows us to use well-established research from OT in creating guidelines for the design of our agent-based advising system.

Contingency-Theoretic Systems Development

OT is a field of study that examines an organization's structure, constituencies, processes, and operational results in an effort to understand the relationships involved in creating effective and efficient systems [12]. A major division of OT, Contingency Theory (CT) states that the structure of a successful organization is dependent upon the environment in which it operates [13]. IPT postulates that this environment-determines-organizational-structure dependency is the result of the coordination requirements among the basic elements and tasks that make up the organization [14] [15].

IPT states that the adaptations that organizations may utilize to solve the information processing requirements in an organizational structure can be generalized into two broad categories: planning and mutual adjustment. The more heterogeneous, unpredictable, and dependent upon other environmental resources a task is, the greater the information processing that the organization must be able to do in order to successfully accomplish the task. As diversity of resources, processes, or outputs increases, inter-process coordination requirements, uncertainty, and system complexity all increase. As uncertainty increases, information-processing requirements increase. These changes yield incomplete management information, which requires more mutual adjustment and cooperation in the organizational system. Conversely with more homogeneous, predictable, and independent tasks, management's ability to predict situations and plan for solutions increases. Thus standard-operating-procedures can be implemented and hierarchical control systems created to manage the organization effectively.

In our research we postulate that the environment-structure relationship outlined in the paragraphs above is also reflected in Software Systems designs, especially those utilizing intelligent agents. In previous research [16] [17] we have developed, and tested, the following CT-based guidelines for creating ABS:

1. *Describe Business Activity and Identify Tasks*: allow management and developers to refine the overall purpose of the software being designed.
2. *Determine Task Predictability*: Since a basic premise of CT is that the control structure of a business process must match the environment in which it operates, we must identify the predictability of each task
3. *Assign tasks to employees*: Once the level of predictability has been estimated for each task, the granularity of the employees being created can be determined and component designs finalized.
4. *Group employees into teams*: As with human organizations, our employees can be grouped along any of several dimensions, including task, workflow, product, manager, or communication requirements, as required by the operating environment.
5. *Identify communications needs*: Once teams are determined the communication, requirements of individual employees, and of teams, can be determined.
6. *Construct management groups*: In software systems operating in a dynamic environment, management is required only when employees are unable to handle events.

Our purpose in this research is to analyze the viability of utilizing this synthesis of two well-established paradigms (ABS and IPT) in the creation of a useable distributed student management system for a virtual university. We discuss the details of this project in the following section.

THE SARA VIRTUAL ADVISOR

Attempting to automate advising for virtual universities is a risky proposition, given the difficulty of advising in person. The traditional advising process, especially in larger departments, is fraught with miscommunications, misunderstandings, and misconnections. Policies and requirements are dynamic and difficult to maintain. Advisors are inundated with "quick questions" that often result in little time remaining for mentoring and truly advising students. Example questions are

- What courses do I still have to take?
- What's the fastest I can graduate (and is such a plan advisable given my life situation and past performance)?
- Do I have to take 4 hours, 4 classes, or 4 semesters of PE?

The SARA Virtual Advisor is a smart agent component of a planned Web-based information management system (a *knowledge*

Copyright Idea Group Inc.

portal) to support traditional and distributed education. SARA is intended to deliver many advising services to students, such as degree planning, course approval, transfer analysis, and general information regarding types of degree plans, and market outlook (types of jobs, location and pay). Moving beyond the common, inadequate strategy of making paper-based advising materials electronically available, we seek to provide a means by which a student (or potential student) may engage in a dialogue with a "virtual advisor." Students want to get answers quickly to specific questions, not wade through pages of on-line documents. As an example:

Elton wants to graduate as quickly as possible, subject to the realities of his specific situation. He can only take on-line or night classes, since he works full time during the day. In the summer, he could take a morning class, but he'd prefer not to. He does want to take Dr. Smith for his circuits class. He assumes that if the classes are not too hard, he could take 15 hours a semester. If he has one or more tough or really time-consuming classes, he should take only 9 or 12 hours. He wants to get a COSC degree, but if he can graduate much sooner, he would like to consider a CISC degree.

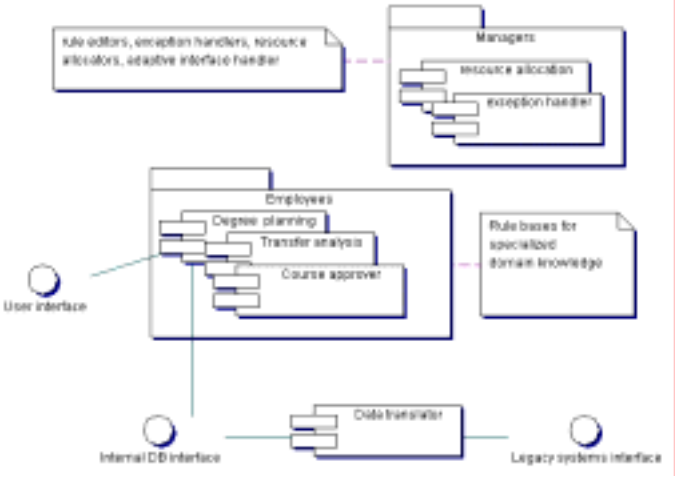
One function of SARA is the graduation planner, which identifies a degree's requirements, compares it to a student's information, determines courses that need to be taken, and then develops a schedule based on these requirements and user preferences. A partial list of the user options is shown below:

- o Graduation Plan
  - • Preferred or absolute graduation date – user selected
  - • System-suggested graduation date based on user selected number of hours, schedule and other limitations
  - • Shortest path of the fewest number of semesters to graduate
  - • Critical path of courses to meet graduation goal date (i.e., considering prerequisites – the order in which classes must be taken to meet desired graduation date)
  - • Can select certain classes at certain times/semester and system then develops the plan around these selections
- o Course Selection and Schedule
  - • Absolute or preferred schedule limitations by semester or overall (no exceptions) – max. number of hours per semester, limits including evening, day, M/W, web, no class before 9:30, no class after 5:30, etc.
  - • Average course difficulty rating over the remaining semesters (evenly distribute challenging courses over semesters)
  - • Maximum difficulty rating to not exceed a set limit
  - • Teacher preferences or teacher not-to-take list
- o Course Information
  - • Student comments, suggestions, and reviews of curriculum available for perusal

Representation issues — quantity, accessibility, quality, and dynamicism. The planner function requires much data, some of which is imprecise or missing, and that is subject to change. For example, planning a student's courses for Fall 2003 relies on many assumptions including student performance and course offerings.

CT design guidelines tell us that domain policies must be specified declaratively and grouped according to predictability to minimize the impact of change. We use relational database tables to store raw data, downloaded from external systems. This design choice isolates legacy system interface tasks. Domain policies are categorized and stored as rules (currently in Jess). The degree planner manager, written in Java, interfaces with the rules and internal database to provide a degree of isolation of policy and implementation. The degree planner manager controls the mechanics of supplying data to and storing data supplied by, the degree planner rules. Thus, individual rule bases (which behave as employee specialists) are strictly focused on solving specific domain problems (such as creation of a degree plan based on student and university constraints). Managers handle the mundane tasks of acquiring and distributing resources (data) from the appropriate sources. A separate user interface again isolates domain policies and procedures from mechanics of implementation. The architecture for the system is shown below.

Figure 2: Architecture for the system



Prototype Example

The first prototype developed was for the Industrial and Manufacturing Engineering Department at The University of Texas at Arlington. Their curriculum and degree requirements are typical for an undergraduate program.

For this graduation planner example, 47 rules and 65 data requirements were identified. The rules could be classified by owner such as university, college, and department requirements or by functions such as transfer credit, minimum GPAs. An example rule (in JESS) that checks if a student can take a course is shown below:

```
(defrule can-take-course "check if student can take a course"
  ?goal2 <- (goal-is-to (action check-course) (argument-1 ?ID) (argu-
     ment-2 ?course))
    (course-prerequisites (dept ?Dept) (course ?course) (prereqs
     $?plist) (coreqs $?))
    (passing-grade $?passing-grades)
  (not (transcript ?ID ?course ? ? ?grade&:(member$ ?grade $?pass-
     ing-grades)))
;; check if student has passed every pre-requisite
    (courses (studentID ?ID) (course-list $?passed))
    (test (subsetp $?plist $?passed))
    ?fact <- (can-take (studentID ?ID) (course-list $?courses))  ;
=>
   (printout  t "  ==> " ?course " prereqs are satisfied." crlf)
    (bind $?newcourses (insert$ $?courses 1 (create$ ?course))) ;
     add valid course to list
    (assert (can-take   (studentID ?ID) (course-list $?newcourses))))
```

Many of the problems we found centered on 1) lack of data standardization, 2) data quality, 3) number of changes, and 4) algorithm complexity. An example of a data standardization issue is shown in course naming schemes (e.g., IE3315 versus Math321). The largest data quality concern was missing data that was usually caused by an inability of the department to commit to a long-term schedule, e.g., instructor teaching assignments.

The dynamic nature of the data is the largest challenge to system maintainability. As already noted, change must be dealt with in any virtual advising system. The most common changes for each semester are faculty changes, especially for Adjunct Professors and instructors teaching lower level classes. Another major problem is catalog changes that often occur every two years in which all three organizational units (university, college and department) can change requirements. These changes affect many of the rules. Finally, dealing with rules interaction and interfacing with the wide variety of legacy systems negatively affects algorithm complexity.

Copyright Idea Group Inc.

CONCLUSIONS

The SARA agent-based architecture was created by applying our contingency-theoretic guidelines to the analysis of the advising domain. By organizing rules according to unit (university, college, and department), setting up a conflict resolution strategy via a manager agent to resolve rule conflicts between units, creating specialist agents to deal with connections to legacy systems, and isolating the most dynamic data elements in the system, we have created a system that we believe will be adaptable to new advising situations, including those in virtual universities. Some of the special features of SARA include adaptability to changing requirements and new experiences via a knowledgebase of advisor policies and end-user customization for different departments and universities.

We expect to continue extending SARA, even as we move forward with integrating the system with other components necessary for virtual university support systems. Some of the specific enhancements we have planned include adding interactive personalities that adapt to different classes of users to provide specific answers to many common questions, exploring alternative representations to rules for the domain policies, for example by incorporating case-based reasoning to capture exception handling and influence diagrams to reason about preferences and uncertainty, and adding machine learning capabilities using data mining methods to improve advice given by the system.

REFERENCES

1. Dumont, R., *Teaching and Learning in Cyberspace*. IEEE Transactions on Professional Communication, 1996. **39**(4): p. 192-196.
2. Chellappa, R., A. Barua, and A.B. Whinston, *An electronic infrastructure for a virtual university*. Communications of the ACM, 1997. **40**(9): p. 56-58.
3. Durrett, J.R., *Area Report on College and Departmental Advising*. 2001, Texas Tech University, College of Business: Lubbock. p. 1-5.
4. CHEA, Distance Learning in Higher Education, in Council for Higher Education Accreditation. 1999.
5. Genesereth, M.R. and S.P. Ketchpel, *Software agents*. Communications of the ACM, 1994. **37**(7): p. 48.
6. Nwana, H.S. and N. Azarmi, eds. Software Agents and Soft Computing : Toward Enhancing Machine Intelligence. 1997, Springer.
7. Butterfield, J., J.G. Cooprider, and S. Rathnam. Resolving Cognitive Conflict in Requirements Definition: a Blackboard Based Model and System Architecture. in Computer Personnel Research Coference on Reinventing IS: Managing Information Technology in Changing Organizations. 1994. Alexandria, VA USA.
8. Bellifemine, F., A. Poggi, and G. Rimassa, *Developing multi agent systems with a FIPA-compliant agent framework*. Software - Practice And Experience, 2001(31): p. 103-128.
9. Cost, R.S., et al., Jackal: A Java-Based Tool for Agent Development, in Working Notes of the Workshop on Tools for Developing Agents. 1998, AAAI Technical Report.
10. Jeon, H., C. Petrie, and M.R. Cutkosky, *JATLite: A Java Agent Infrastructure with Message Routing*. IEEE Internet Computing, 2000. **4**(2): p. 87-96.
11. Labrou, Y., T. Finin, and Y. Peng, *The current landscape of Agent Communication Languages*. Intelligent Systems, IEEE Computer Society, March/April 1999. **14**(2): p. 45-52.
12. Scott, W.R., *Organizations: Chapter 1*. 1992, New York, NY: Prentice Hall. 3-26.
13. Van de Ven, A.H.D., Robert, *The Concept of Fit in Contingency Theory*, in *Research in Organizational Behavior*, L.L.S. Cummings, B., Editor. 1985, JAI Press Inc: New York, NY. p. 333-365.
14. Cyert, R.M., J. G., *Behavioral Theory of the Firm, A*. 1963, Englewood Cliffs, NJ: Prentice-Hall.
15. Galbraith, J.R., *Designing Complex Organizations*. 1973, Reading, Mass: Addison-Wesley. 149.
16. Durrett, J.R., L.J. Burnell, and J.W. Priest. Organizational Metaphors for Software Architectures. in IERC-2001: 10th Annual Industrial Engineering Research Conference. 2001. Dallas, Texas.
17. Durrett, J.R., L.J. Burnell, and J.W. Priest. Contingency theoretic methodology for agent-based, web-oriented manufacturing systems. in SPIE: Photonics East 2000. 2000. Boston, MA USA.

Copyright Idea Group Inc.
Copyright Idea Group Inc.
Copyright Idea Group Inc.
Copyright Idea Group Inc.

0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/proceeding-paper/sara-smart-agent-based-resource/31788

Related Content

Choosing Qualitative Methods in IS Research: Lessons Learned
Eileen M. Trauth (2001). *Qualitative Research in IS: Issues and Trends (pp. 271-288).*
www.irma-international.org/chapter/choosing-qualitative-methods-research/28267

Social Computing
Nolan Hemmatazad (2015). *Encyclopedia of Information Science and Technology, Third Edition (pp. 6754-6761).*
www.irma-international.org/chapter/social-computing/113139

Mutation Testing
Pedro Delgado-Pérez, Inmaculada Medina-Buloand Juan José Domínguez-Jiménez (2015). *Encyclopedia of Information Science and Technology, Third Edition (pp. 7212-7221).*
www.irma-international.org/chapter/mutation-testing/112419

A Domain Specific Modeling Language for Enterprise Application Development
Bahman Zamaniand Shiva Rasoulzadeh (2018). *International Journal of Information Technologies and Systems Approach (pp. 51-70).*
www.irma-international.org/article/a-domain-specific-modeling-language-for-enterprise-application-development/204603

Teaching Media and Information Literacy in the 21st Century
Sarah Gretterand Aman Yadav (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 2292-2302).*
www.irma-international.org/chapter/teaching-media-and-information-literacy-in-the-21st-century/183941