Contingency Theory, Agent-Based Systems, and a Virtual Advisor

John R. Durrett
Texas Tech University, USA

Lisa Burnell
Texas Christian University, USA

John W. Priest
University of Texas at Arlington, USA

INTRODUCTION

In this article, we investigate the potential of using a synthesis of organizational research, traditional systems analysis techniques, and agent-based computing in the creation and teaching of a Contingency Theoretic Systems Design (CTSD) model. To facilitate understanding of the new design model, we briefly provide the necessary background of these diverse fields, describe the conceptualization used in the integration process, and give a non-technical overview of an example implementation in a very complex design environment. The example utilized in this article is a Smart Agent Resource for Advising (SARA), an intelligent multi-agent advising system for college students. To test all of the potential of our CTSD model, we created SARA utilizing a distributed instructional model in a multi-university, multi-disciplinary cooperative design process.

Just as a dynamic task environment forces an organization to compress its management structure and to outsource non-core activities in order to become flexible, a dynamic software development environment forces designers to create modular software. Until now, cooperative development paradigms were too complex to facilitate inter-organizational cooperative development efforts. With the increasing popularity of standards-based Web services, the development of pervasive computing technologies, and the advent of more powerful rapid application development languages and IDEs, this limitation has been removed. Our purpose in this research is twofold: first, to test the viability of using Contingency Theory (CT), a sub-discipline of Management Organizational Theory (OT), in an agent-based system; and second, to use these new technologies in creating a distributed instructional model that will allow students to interact with others in diverse educational environments. As an example implementation, we create a virtual advisor that will facilitate student advising in distributed environments.

In the following sections, we outline the background theories involved in the conceptualization of our design model. We start with the shifts in systems design techniques and how CT can be applied to them and to various Multi-Agent Systems (MAS) to allow Contingency Theoretic Systems Design (CTSD). Once the necessary background is in place, we briefly discuss our new eLearning approach to cooperative distributed education. Finally, the structure of the SARA is discussed.

BACKGROUND

Multi-Agent Systems

Agents and communication protocols form the basic components of a multi-agent system. Agents exchange messages according to a protocol of expected messages delivered in a communication language in which the message content and format adhere to a shared standard. Individual agents make decisions, which may include contacting other agents for information, and perform processing to satisfy their goals

An agent is commonly defined as a program or collection of programs that lives for some purpose in a dynamic environment and can make decisions to perform actions to achieve its goals. In other words, agents are goal-based programs that must deal with changing access to resources, yet run continuously. Like the best administrative assistants, agents know and adapt to their master. Individual agents may be conceptualized as having beliefs, desires, and intentions that can communicate with other agents to satisfy their goals. Multi-agent systems are those in which multiple agents (usually) cooperate to perform some task. Agents may be independently developed and allow the decomposition of a complex task into a collection of interacting agents that together solve some problem. It is not necessary that an individual agent "understand" the overall system goals or structure.

Agent communication can be viewed at four distinct levels. The first level is the expected protocol for exchanging sequences of messages, like a script. For example, when negotiating, the parties expect bids to be offered, rejected, and counter-offered. The second level relates to the content or meaning of the messages. To enable inter-agent communication, an ontology is created. Examples of such concepts are things, events, and relationships. At the third level, a representation language defines the syntax for structuring the messages; The Knowledge Interchange Format (KIF) (Gensereth & Fikes, 1992) is one example. At the fourth level, an agent communication language (ACL) such as the Knowledge Query and Manipulation Language (KQML) or the Foundation for Intelligent Physical Agents (FIPA) ACL (Labrou, Finin, & Peng, 1999), defines message formats and message delivery. An example KQML message, in Sandia Lab's Java Expert System Shell (JESS) (Owen, 2004), that shows how an agent registers a service is shown below:

```
(register :sender student :receiver advisor :reply-with msg1
:language JESS :ontology SARA  :content '(MajorCourses:
Compliance Check Hours))
```

Just as human systems created to achieve complex goals are conceived of as organizations, multi-agent systems can be conceptualized as "organizations of agents". Individual components, whether human employees or software agents, need to be managed, guided toward a constructive goal, and coordinated toward the completion of the necessary individual tasks. In "empowered organizations", lower-level employees have the knowledge and authority to perform many tasks without the intervention of superiors. This conceptualization allows us to use well-established research from management organization theory (and Contingency Theory in particular) in creating guidelines for the design of agent-based systems.

ORGANIZATIONAL THEORY (OT)

While much of the background concerning OT is explained in the main chapter below, the following is a brief overview of the relevant research trends. OT examines an organization's structure, constituencies, processes, and operational results in an effort to understand the relationships involved in creating effective and efficient systems. A major division of OT, Contingency Theory (CT), postulates that no organization operates without constraints from environmental, personnel, technological, and informational influences (Andres & Zmud, 2001). This relationship is explained by the information processing theory (IPT) (Galbraith, 1973). IPT postulates that 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 it. As complexity and unpredictability increase, uncertainty increases due to incomplete information. As diversity of processes or outputs increases, inter-process coordination requirements and system complexity increase. As uncertainty increases, information-processing requirements increase. The basic premise of IPT is that the greater the complexity and uncertainty in the tasks in an organizational system, the greater the amount of information that the system must process (Galbraith, Downey, & Kates, 2001). A basic premise of our research is that this relationship is also true for information systems (Avgerou, 2001).

MAIN THRUST OF THE ARTICLE

Multi-Agent System Architectures Using CTSD

Contingency-theoretic system development (CTSD) adapts CT and IPT to the development and maintenance of software systems (Burnell, Durrett, Priest et al., 2002; Durrett, Burnell, & Priest, 2001, 2003). A business can organize employees in a number of different ways, for example by function or by project, and reorganize as the business environment changes. Software systems can benefit from this flexibility as well. The CTSD design approach is focused on design for maintainability, a crucial requirement for complex, dynamic systems.

Agent-based architectures are a means for structuring software systems that adhere to Contingency Theoretic principles. Each agent is viewed as an employee that has specific capabilities, responsibilities, and knowledge within an organization. Agents, like employees, are grouped into departments, as needed, to best satisfy the goals of the organization. Agents can communicate peer-to-peer within and across departments, and manager agents resolve conflicts and make resource allocation decisions.

Tightly interrelated tasks are grouped into one or more agents. Each of these groupings is referred to as a "software team", and parallels a department of employees that perform roughly equivalent jobs. For example, a set of agents that each handle one type of course requirement (e.g., lab, art appreciation) may be grouped into a team, where communication can occur quickly between these agents and with a "manager" agent that can resolve conflicts, exceptions, and course-independent tasks. An example agent in our system is encoded using JESS rules to check that student preferences (e.g., for afternoon courses) and constraints (e.g., no more than 12 hours per semester) are satisfied. Another agent offers heuristic advice as an actual advisor might. For example, a student may be able to enroll in 17 hours of math and science courses, but this may be strongly advised against, depending on the student's GPA and perhaps other factors.

Each agent in a multi-agent architecture has specific tasks to perform and communications requirements. Once an ontology and agent communication language has been specified, agents can be designed independently and integrated into the system to progressively add capabilities. Using CTSD principles, tasks that are dynamic and shared are grouped into support agents to enhance maintainability of the system. The