Multi-Agent Framework for General-Purpose Situational Simulations in the Construction Management Domain

Eddy M. Rojas[1] and Amlan Mukherjee[2]

Abstract: The need for contextually rich education environments in construction management suggests the development of a general-purpose situational simulation framework that can be used by independent developers to build effective training environments. Design and implementation of such a framework involves an understanding of the reasoning processes underlying the construction management domain. These reasoning processes can be isolated using a conceptual classification of problems in the construction management domain into constraint satisfaction and planning. Such a classification allows us to distribute the different reasoning processes to autonomous agents that comprise the foundations of a multi-agent framework for building general-purpose situational simulations. The Virtual Coach is an implementation of the proposed framework. Experimental results from preliminary studies have shown the efficacy of the Virtual Coach as an educational tool.

DOI: 10.1061/(ASCE)0887-3801(2006)20:3(165)

CE Database subject headings: Models; Simulation; Construction management; Artificial intelligence.

Introduction

In traditional construction education, the learner and the learning context are detached. Concepts are presented as fixed, well-structured, independent entities and classroom activities are disconnected from authentic context, resulting in fragmentation and specialization of courses and educational experiences. McCabe et al. (2000) argue that current civil engineering coursework teaches only the theories of construction management (CM) and students encounter difficulties in applying theoretical principles when exposed to real-world situations upon employment. Sawhney et al. (2001) state that civil and construction engineering curricula do not allow the inclusion of issues of importance to construction or reflect the significance of hands-on experience and interaction with practitioners. Case studies, class projects that involve interaction with the industry, and even internships are useful for bridging such a gap. However, these approaches are limited, as they do not provide students the opportunity to explore the implications of management decisions.

This understanding has led researchers to explore alternatives in construction education using gaming and simulation environments, such as Superbid (AbouRizk 1993), STRATEGY (McCabe et al. 2000), and VIRCON (Jaafari et al. 2001). Some of these efforts have been inspired by earlier research undertakings in the area, such as CONSTRUCTO (Halpin and Woodhead 1970) and AROUSAL (Ndekugri and Lansley 1992). Simphony (Hajjar and AbouRizk 1999) and STROBOSCOPE (Martinez and Ioannou 1999) have also developed simulations that deal with construction operations such as tunneling and earthmoving. Simulation languages such as STROBOSCOPE and CYCLONE have also provided a general and special purpose framework for simulating construction operations and CM processes.

These efforts are a stepping stone toward creating participatory, contextually rich educational environments. However, their use has been limited to special purpose simulations of specific construction processes and general purpose frameworks for developing simulations of construction operations. All these implementations have limited or absent interactivity.

We claim that general purpose interactive situational simulations of CM processes can be used to develop effective learning environments for construction managers. Such environments allow participants to understand the interdependencies between various constraints that govern the CM environment, while developing better decision-making skills. The first step toward creating such general purpose situational simulations is to develop a scalable framework that can be programmed to emulate a variety of CM processes. In this paper we introduce a multi-agent framework for developing such general purpose simulations. We have also developed an implementation of a situational simulation using the framework, for a hypothetical construction project, and tested its usefulness on a selected group of CM seniors.

Situational Simulations as Educational Environments

In their simplest form, *simulations* of construction processes use a set of initial conditions and parameters and a well-defined model to project outcomes regarding the simulated operation. For example, given information regarding the availability of trucks and loaders, their unit costs and the amount of earth to be moved,

[1]Associate Professor, Dept. of Construction Management, Univ. of Washington, 116 Architecture Hall, Seattle, WA 98195-1610. E-mail: er@u.washington.edu

[2]Assistant Professor, Dept. of Civil and Environmental Engineering, Michigan Tech, 1400 Townsend Dr., Houghton, MI 49931-1295. E-mail: amukherj@mtu.edu

Note. Discussion open until October 1, 2006. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on August 26, 2003; approved on July 15, 2005. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 20, No. 3, May 1, 2006. ©ASCE, ISSN 0887-3801/2006/3-165–176/$25.00.

a simulation would be able to project the total time and cost for an excavation operation. *Situational simulations* are simulations that use temporally dynamic models, require user interaction, and use domain-specific knowledge to generate and reason about the context-sensitive scenarios. As the simulation proceeds, events are generated as a consequence of user interaction or to reflect real-life situational scenarios that might arise in the domain. How the simulation evolves in time is completely dependent on the model used, the way the events are generated, and user interaction.

Hence, a situational simulation is part machine (computer software/hardware) and part human environment. The machine is responsible for simulating the CM environment using construction domain-specific knowledge while being sensitive to how human participants react to it. For example, given the knowledge that labor, when overworked, will tend to produce lower quality work, the machine would generate a "rework" event when the human participant tries to crash activities by making labor work overtime too often. It can also create a "bad weather" event that disturbs progress on outdoor activities based on statistical weather patterns of the participant's locale. The human participant is expected to try to finish the simulated project within time and budget constraints as they would in real life. Thus, their responsibility is to constantly take challenging decisions regarding resource allocation and time cost tradeoffs. As the simulation proceeds, there are a large number of ways to complete the simulated project. The project completion depends on the reactions of the human participant and the reactions of the machine.

The interactivity and the ability to adapt to context-specific scenarios make situational simulations useful learning environments. However, the interactivity and the ability to create context-specific events and scenarios require the simulation to be aware of construction management domain-specific structure and knowledge. This calls for the simulation to have autonomous reasoning capabilities and hence suggests the use of a multi-agent environment, where a group of autonomous agents collaborate to reason about and generate the simulation environment.

The use of situational simulation environments for learning is also supported by theories in situated cognition (Winn 2002). Such environments expose participants to clinical exercises that help them both explore future consequences of present decisions and the sensitivity of their contexts to such decisions and, over time, develop better decision-making skills. The Virtual Gorilla Project at the Atlanta Zoo (Allison et al. 1997), as well as the Virtual Puget Sound (Windschitl and Winn 2000) and the Surgical Simulator (Oppenheimer and Weghorst 1999) efforts at the Human Interface Technology Laboratory, University of Washington, are successful instances of such learning environments.

Extensive use of situational simulations has also been seen in the politico-military arena (Allen 1987; Goldhammer and Speier 1959) and in natural disaster relief management (Ritchie 1985). A general purpose framework for situational simulations dealing with CM processes could be useful for developing a very wide variety of education environments for the construction engineering and management domain.

Proposed General Purpose Multi-Agent Framework

A general purpose framework (GPF) provides a protocol that allows us to develop many different special purpose simulations. A common protocol allows communities of developers to share, extend, and build customized simulations while fostering collaboration in CM education. The participants of such communities can belong to both academia and industry, with the common goal of educating construction managers. In this section we discuss the conceptual foundations of the proposed GPF, the components that make it up, and the framework itself.

Conceptual Foundations of GPF

In developing the foundations of the GPF, we studied the CM domain to classify the preconstruction and construction phase processes into specific classes of problems. This abstraction is the first step toward creating the GPF.

During the preconstruction phase, the problem at hand is that of creating a resource-loaded activity schedule, also referred to as the "As-Planned" schedule. This can be classified as a constraint satisfaction problem (CSP). Such problems can be solved using a search-based constraint solver. A number of research efforts support this claim. Sucur and Grobler (1996) suggested a CSP formulation for construction project planning. They developed a structure that can represent precedence constraints (which they refer to as temporal constraints) and implicit resource constraints, and they provided a solution to the CSP using forward-checking constraint propagation algorithms such as pruning and conflict resolution. Hammond et al. (2000) suggested the use of a partitioned dependency structure matrix to represent information about a schedule, which on closer analysis proves to be a CSP in which each matrix is a state representation of precedence and resource dependencies in a schedule. WorkPlan (Choo et al. 1999) also used resource and precedence constraint satisfaction in the WorkPlan implementation. It is safe to claim that, given the appropriate constraints, the As-Planned schedule can be generated using search-based constraint solvers that return sequences of state transformations between an initial state representation of a schedule and a goal state representation (a resource-loaded As-Planned schedule) while assigning resources to all activities, in keeping with resource and precedence constraints.

During the construction phase, managers aim to complete the project within constraints of budget and time as encoded in the As-Planned schedule. However, in reality, circumstances seldom permit the "As-Built" schedule to be identical to the As-Planned schedule. Projects get derailed from the As-Planned implementation because of violations in resource and precedence constraints caused by unexpected events such as labor strikes, undelivered material, and bad weather. Construction managers face the challenge of completing the project while constantly making critical decisions that satisfy the constraints encoded in the As-Planned schedule by reallocating resources, rescheduling activities, and making time-cost trade-offs. Hence, the manager's job during the construction phase is akin to a planning problem.

Planning problems make use of domain structures to generate relevant plans. Unlike search-based problem solvers, which are dependent on a specific set of successor functions to affect the environment, planners have a greater degree of autonomy and can create plans that are sensitive to context-specific information. Specifically, during the construction phase, managers are responsible for maintaining constraint satisfaction by taking corrective measures and dynamically updating the plan based on context-specific knowledge of the present and anticipated futures of the environment. A discrete state representation that is incapable of representing multiple dynamic relationships overlapping in time is not enough to describe such complex scenarios. Instead, a formal language is necessary to describe autonomous reasoning in such environments.

The first step toward developing the GPF was to agree on a "language" that could be used to represent and reason about activities, actions, and events in the CM domain. The semantics of such a language would have to be based in a concept that applies generally across all scenarios in the CM domain. Such a language was developed using the semantics of interval temporal logic (Mukherjee and Rojas 2003). The general concept is that an environment can be defined using a set of variables, each of which can take up values from specifically defined continuous or discrete ranges. Each such variable is also attached to an interval of time, which specifies the interval over which the value of the variable is valid. For example, the weather is represented by the variable *weather*, which can take values from the domain [*sunny, rainy, snowy*], and the predicate *weather*(*sunny*, *t*) signifies that the weather in the environment will hold sunny over the time interval *t*. Combinations of changes in the validity intervals of one or more such variables representing the environment signify actions in the environment. Postconditions of such actions signify events. The preconditions of such actions need to be fulfilled for the events to be triggered by the actions. The preconditions and postconditions for any action-event combination can be used to encode constraint information. The action-event combination thus represents constraint violations and the effect of such on the simulated CM environment.

From the preceding analysis, we can conclude that the CM domain can be abstracted to a planning problem during the project implementation phase and a constraint satisfaction problem during the preconstruction phase. It involves satisfaction of resource and precedence constraints and reasoning processes, which govern actions and events in the construction environment. The foundations of the GPF lie at the very heart of this general understanding. A language to represent and reason about CM constraints can provide the basis to simulate a diverse set of scenarios in the CM domain. We have created a GPF for situational simulations in construction using a multi-agent framework, based on the concepts discussed in this section.

Multi-Agent Framework

An agent is anything that can perceive its environment through sensors and can act through effectors (Russell and Norvig 2002). In the context of this paper, all discussions about agents refer to software agents. Software agents are programs that can autonomously create changes in their environments based on a perception of the existing conditions. The semantics of the *environment* have been formally defined in Mukherjee and Rojas (2003). Agents reasoning logically and acting autonomously (free of human control) toward a goal can be attributed a notion of intelligence. They are aware of the repercussions of their actions on the environment and dynamically integrate their experiences into existing reasoning mechanisms. In the suggested multi-agent framework, each agent handles a specific reasoning process.

Agents are responsible for simulating the real world. They can do so by generating current events as consequences of previous participant interactions, or by creating seed events using a random event generator. Secondly, agents can predict future consequences of present circumstances. Such predictions provide participants with guidelines for effectively planning future decisions. Finally, agents provide graphs and charts that visually capture the sensitivity of the simulated environment to critical participant decisions. For example, a graphical display of differences in the As-Built and the As-Planned trends over time allows participants to monitor progress and relate their decisions to delays and cost overruns. In order to accomplish the first two duties, agents need to be perceptive to changes in the simulation caused by participant decisions and react accordingly by effecting appropriate changes in it. Agents also need to have an awareness of the different contexts in which reason is required.

The literature provides us with a rich variety of agent-based frameworks that have been used in distributed environments. The generalized partial global planning (GPGP) (Lesser et al. 2002) framework and its associated TAEMS hierarchical task network representation were developed as a domain-independent framework for coordinating the real-time activities of small teams of cooperative agents working to achieve a set of high-level goals. Coordination between multiple agents running different algorithms has been exploited to develop efficient solutions to complex problems. In A-Team (Talukdar et al. 1996), a scale-efficient network of distributed computer agents was used to solve nonlinear algebraic equations in a shorter time than through individual processes, using the Newton-Raphson and genetic algorithms as agents. The M-RAM (Soibelman and Pena-Mora 2000), a multireasoning model, uses an agent-like approach to develop modules, each of which is specialized to perform particular tasks. The M-RAM model was used to support the conceptual phase of structural design and to study the applicability of agents to support the subprocesses of a divided structural design process. This paper focuses on the use of agent modules, each of which is specialized to perform a particular thread of reasoning pertinent to the implementation phase of a construction project. The autonomous reasoning and problem-solving capabilities of the agents allow us to efficiently design situational simulation environments for the construction domain.

Agent architectures have also been used in synthetic and software environments. However, as Tambe et al. (1995) explain, though closely related, the concept of using agents for synthetic environments differs distinctly from both software (Etzioni 1993) and robotic (Brooks 1991) and test bed (Hanks et al. 1993) environments. The most significant difference between software and synthetic environments is that the latter requires real-time behavior in dynamic, limited information worlds and therefore cannot be strongly dependent on traditional planning. Unlike robotic environments, synthetic environments do not need to deal with low-level motor control and perception. Test-bed environments differ from synthetic environments often because of the domain of problems they handle. Synthetic environments tend to handle real-life domains (like construction, in this case), while test-bed environments tend to deal mostly with domains that often tend to have greater complexity than test-bed domains, where developers can "prestructure the environment, choose aspects of behavior, or instrument the domain for experimental purposes" (Tambe et al. 1995).

There has been a great deal of investigation in the use of agents in interactive simulations, which are very similar to situational simulations. The obvious benefit of using agents is that they can replace humans when a large number of entities are needed to populate a virtual world (Tambe et al. 1995). Notably, Cremer et al. (1994) suggest the use of intelligent agents in traffic simulators, to simulate scenarios involving slowing and speeding of vehicles, pedestrians, traffic jams, and other complex traffic patterns. Tambe et al. (1995) explore the use of intelligent automated agents for battlefield simulation environments. Their environments are based on distributed interactive simulation

(DIS) technology, in which large-scale interactive simulations are built from a set of independent simulators linked together by a network. They developed independent, intelligent automated pilots in the environment based on the underlying Soar integrated architecture for general intelligence (Laird et al. 1987). Soar has an explicit symbolic representation of its tasks, which it manipulates by symbolic processes. Domain-specific knowledge is also symbolically coded and used as a guideline for behavior. Intentions are represented by a general scheme of goals and subgoals. Goal formulation is achieved by finding a desired state in a problem space, which is defined as a space with a set of operators that apply to a current state to yield a new state (Laird et al. 1987). Thus, all goal formulation tasks can be completed using some heuristic search technique. If knowledge to immediately formulate a goal (say, select an operator) is insufficient, then a subgoal is created which in turn can further create subgoals. Hence, the behavior of Soar involves a tree of subgoals and problem spaces. The ability to recursively create subgoals allows Soar to learn continuously and automatically by storing the "results of its subgoals as productions." For example, if at any point more than one operator can be chosen, a subgoal is created to break the tie, and the final result of problem solving within this subgoal creates a preference that resolves the tie. The operator sequence is stored as a production and is delivered as the preferred solution in a relevantly similar situation. In this way the architecture uses a production system for single memory organization of all long-term knowledge. Laird et al. (1987) illustrate the Soar architecture using the Eight Puzzle, among other problems.

The reason why the Soar architecture is of great interest to us is that Tambe et al. (1995) successfully use it to create situational simulations for the air-combat domain. They created pilot agents that participate in battlefield simulations using ModSAF (Calder et al. 1993), a distributed simulator that has been commercially developed for the military. Taking advantage of DIS technology, copies of ModSAF are used to simulate a number of different fighter aircrafts, across a network of workstations. The aircrafts can participate in simulated combat with or against each other. ModSAF runs the simulation by sequentially invoking each agent. The simulation model is affected by the action of all agents across the network and allows predictions regarding future states of the environment. Depending on the predictions and the actions of the agents, the simulation is updated at the end of each cycle. The states in the Soar architecture represent situations, and operators represent actions that can be in the form of simple primitive actions that modify internal states or arbitrarily complex ones.

At this point, it is important to compare the Soar multi-agent architecture with the multi-agent architecture for situational simulations in construction presented in this paper. The Soar architecture represents situations as states and operators facilitate state transformations. This means that, in such an architecture, time is represented as a sequence of states. In addition, the operators, which represent actions in the real world, will tend to be instantaneous. Intervals can be defined as a sequence of states, but this would make representation of multiple overlapping events difficult. As in the case of the interactive simulation developed for the air-combat domain by Tambe et al. (1995), the simulation of multiple interacting simulated aircrafts is achieved using DIS technology, which involves running multiple copies of the simulation over a network and coordinating them in parallel. In the construction domain, this would entail running multiple copies of the simulation for each construction activity. However, in the absence of DIS technology, the agent framework introduced in this paper uses temporal reasoning based on an interval representation of time (Allen and Ferguson 1994) to represent parallel activities within the construction domain. Our architecture ascribes operations to agents. However, the operations are not defined to create transitions between states. Instead, the agent operators change the attribute values of entities, which are logical aggregates of variables. Each variable defines some aspect of the environment. The time interval reasoning allows the description of an aspect of the environment as an assertion about a variable attribute over a time interval. Different entities are affected at different times by different agent operations, and at any time it is possible to have persistent states of variables or multiple operators acting on multiple entities, each specific to a particular context or activity.

Tambe et al. (1995) argue that finite-state machine (FSM) languages are too restrictive to represent human-like intelligence. Similarly, situational calculus, an FSM language, is inadequate for representing information about the parallel nature of events in the construction domain. Even though the situational calculus approach was used in the air-combat domain, parallelism and simulation of multiple fighter planes could be achieved through DIS technology. By running multiple copies of ModSAF, they were also running multiple FSMs. The framework introduced in this paper runs multiple finite-state machines (for each activity context) within a single simulation model. This has been explained in detail by Mukherjee and Rojas (2003).

Representation and Agent Reasoning

The situational simulation environment modeled in this paper is based on the conceptual framework of a process model, a product model, and an information model. This framework is discussed in detail by Rojas and Mukherjee (2003). In general, the process model comprises accurate representations of the different processes being simulated. The product model represents the constructs, which make up the interface and the physical embodiment of the simulated environment. The information model encompasses information about the project that is being simulated. The information is coded into databases and knowledge bases. While the database has information about the As-Planned execution of the project, the knowledge base has knowledge about actions and events specific to the context of the construction project being simulated.

The main threads of reasoning underlying the situational simulation system can be listed as reasoning about actions and events in the environment and reasoning about the dynamics of the simulated system. Relationships between different aspects of the construction environment are mathematically modeled in Rojas and Mukherjee (2003a). Reasoning about actions and events is based on axiomatic semantics as described in Mukherjee and Rojas (2003). The agent definitions are based on this functional distinction. The Mathematical Agent (MA) is guided by the mathematical model and the Logical Agent (LA) is guided by the logical model.

Actions are triggers that create events and situations. Some examples of outcomes of actions include bad weather, material delivery delay, reallocation of resources, and labor strikes. In the simulation environment, actions occur instantaneously in time at the starting time point of the interval of the event they trigger.

Events reflect the effects of real-life episodes on resource and precedence constraints within the construction domain. All events span over time intervals. Each event is associated with three sets of variables, the *Precondition* set, the *Event Condition* set, and the *Consequence* set, and is triggered by a unique action. Member variables of the event condition and precondition sets are identical. However, the variables in the two sets must have different attribute values. The change in attribute values is triggered by actions. The event is reflected by the event condition set of variables. Future effects of the event are captured in the consequence set, which is a set of assertions about values of variables in the future. The compound predicates *Pre_Cond(t)*, *Event_Cond(t)*, and *Consequence(t)* are conjunctive clauses of simple predicates that assert attributes of the member variables over the time interval *t*, during which the conditions specified by the precondition, event condition, and consequence sets hold, respectively. They are also homogeneous over the time intervals in which they hold.

Consequences of an event are assertions about the future that are direct outcomes of the event. The consequence set is a set of variables that assert attributes of entities in a future time interval, which is directly affected by the occurrence of the event. Information about actions and events is stored in a knowledge base and is based on the event and action definitions.

Situations are events that result in constraint violations that demand immediate user intervention to continue with the simulation. All events may not create immediate constraint violations and hence may not create situations.

Participants interact with the environment by changing values of the variables. However, participants can interact only with variables within their jurisdiction. By changing the contexts of resource variables, participants can reallocate resources between activities. Global variables are beyond their control (e.g., the participant cannot change the weather). Access is limited to context-specific variables, which describe the resource requirements of the activities.

Agents have greater access to variables than participants. Agents can access all global and context-specific variables. However, in taking actions that affect the environment, agents are not allowed to change eternal truths about the environment (e.g., the agent cannot change the attribute of an excavation activity from outdoor to indoor).

All agent actions are essentially operators, which transform a set of preconditions to a set of event conditions. Because participant interactions are limited to resource reallocation and replacement within the environment, they cannot directly create events in the environment. However, reallocation of resources might result in resource constraint violations that, when perceived by the agent, will indirectly create events. Hence, participant interactions can only create the precondition set, but only agent actions can transform the precondition set to the postcondition set.

Agents operate on *entities* of information in the simulation environment. Logical classification of simulation information is based on semantics specific to the construction domain. Entities are defined as the different classes of information in the simulation environment. The entities Independent and Dependent Variables have been discussed in Rojas and Mukherjee (2003b). The Event and Environment Variable, being classifications of variables defining the environment, are also entities. Every agent operation takes an information entity as an input and transforms it to another information entity.

Entities can be atomic in nature. Atomic entities can be combined to create superentities when the superentity is a logical parent of the atomic entities. For instance, the entity that describes Events is a superentity of the different types of atomic events, namely, Activity Dependent and Global. Similarly, Activity Specific Variables and Global Variables are atomic entities that can be combined to form the compound entity Environment Variables. The set of entities in the simulation environment consist of: *As-Planned Data*, *As-Built Data*, *Static Derived Data*, *Dynamic Derived Data*, *Activity Dependent Events*, *Global Events*, *Activity Specific Variables*, *Global Variables*, and *Independent Variables*.

Agents exhibit autonomous behavior. Autonomy gives an agent the ability to behave free of human intervention. Autonomous decisions cannot be taken by agents, which function by looking up matching facts from a set of built-in assumptions and knowledge, because that limits the agents' ability to deal with undefined situations. Autonomy of an agent calls for an "intentional stance" (Woodridge and Jennings 1995). To take an intentional stance is "to be the subject of beliefs, desires, etc." (Seel 1989), and intentional notions are abstraction rules that provide us with a convenient way of describing, explaining, and predicting behavior. For instance, some simple abstraction rules for the construction environment are "bad weather adversely affects productivity," "productivity affects activity duration," etc. These intentions are essentially attitudes that represent the agent and influence its behavior. The attitudes can be information attitudes or pro-attitudes. While information attitudes are related to the knowledge that an agent has about the world, pro-attitudes guide the agent's behavior. The agent has to have access to both these attitudes in behaving autonomously and rationally within the environment. Attitudes are inherited by agents from knowledge bases (KB), databases (DB), and feedback from user interaction (UL). These are collectively referred to as *bases* in this paper. The concepts of entities and bases developed here are akin to that of the Waffler architecture (Anderson and Evans 1996), which adopts and applies intentions under resource and time constraints. The knowledge bases provide the inference rules, while the database containing As-Planned data provides information about resource and precedence constraints and constitutes the "long-term memory" or conceptual knowledge possessed by the agent specific to the domain and the project under implementation. The feedback from user interaction influences the stances taken by the agent and hence provides information, which is stored in the "working memory." This is a set of variables in the environment that are stored and used in the reasoning as long as they are temporally valid.

Rational agent behavior calls for an agent to be able to reason about its perceptions. This requires a formal representation of the environment and compatible agent reasoning mechanisms. Logical reasoning about causal relationships between actions and events during the project implementation phase is based on axiomatic semantics. Reasoning about functional relationships between independent and dependent variables in the environment is based on a system of equations.

The reasoning mechanisms are represented as operators. The list of operators includes *infers*, *generates*, *computes*, and *unites*. Each agent has access to specific operators, which it uses to operate on entities.

Infers

Situational simulation representation uses first-order logic syntax and semantics based on a deduction model of belief (Konolige 1986). Konolige's model works on the observation that a knowledge-based system functions on the two components of (1) a database of symbolically represented "beliefs" (this could include rules, frames, semantic nets, or logical formulas); and (2) a logically incomplete inference mechanism. He defined this observation in terms of a deduction structure which can be expressed as $d=(\Delta,\rho)$ where Δ=base set of formulas in some logical language and ρ=set of inference rules representing the agent's reasoning mechanism. Deductive closure of the agent's base beliefs under its deduction rules is defined by the function *close*( ), which is given by

$$close((\Delta,\rho)) = \{\phi | \Delta | \text{--} \rho\phi\} \tag{1}$$

where $\phi|\Delta|\text{--}\rho\phi$ means that ϕ can be proved from Δ using only the rules in ρ.

Our logical agent comes into play between every consecutive discrete time points in the simulation—in other words, between any two consecutive "days" during the simulation of a project. We call this process "overnight inference." The agent infers after the time point T and before the time point $(T+1)$. Given the knowledge of the environment in terms of variables at the end of the T period and a set of assertions about the environment, the agent can identify the events that were triggered due to user interaction during T and predict the future outcomes of such present events. In addition, by identifying the existing conditions, the agent can suggest a list of actions to the event generator to appropriately create situations in the environment.

In its inference mode, the agent has complete access to information encoded in terms of sets of variables. Information in the simulation environment is coded as a finite set S of variables. Subsets of S convey information about the environment in different contexts. For instance, the subset $W(T)$ is a subset of S containing variables that define the environment across all activities at the time point T. Similarly, the subset $W'(i)$ is a subset of S containing variables that define the environment across all time points spanned by the interval defined over the context i. The variables are expressed in the following predicate form

$$c{:}V(s,t) \tag{2}$$

where c=context over which the variable encodes information. Typically, the context is itself a time interval specific to the activity to which the variable pertains. V=aspect of the environment the variable specifies; and s belongs to the finite set of discrete states that V can assume. t indicates the time interval over which the aspect holds the state s. The time interval t will always be a subinterval of c or the same interval as c. The predicate itself takes up a truth-value to indicate the state of a variable over an interval of time. Such predicates can be used for logical reasoning.

As variables cannot change value during the inference process, the agent's inference environment is static. It is also discrete, because there are a finite number of possible values that each variable can take. Finally, from the agent's point of view, the environment is nondeterministic, as it cannot predict all user interactions or event generator decisions in the immediate future.

It may be noted here that, for every event, the set of event conditions may be referred to as the postcondition set for the action triggering the event. The precondition set of the action and event are identical. The following assumptions of closure can also be made:

- Event closure: An occurrence of an event implies that an action occurred; and
- Attribute closure: This reflects a closure of the attributes and variables and expresses that any change in attributes of variables implies that an event has occurred.

All agent inference and reasoning is done on the basis of assertions about actions and events in a knowledge base of facts. The inference mechanism is sound and complete within the definitions of actions and events defined in the knowledge base. Hence, if an action is defined in the knowledge base, then it will be predicted every time its precondition set is fulfilled. Also, an event defined in the knowledge base will always be inferred if it has occurred. However, if there is a combination of variable attributes that the participant can change but which are not documented in the knowledge base, they will simply go unnoticed. Therefore, an efficient implementation of this agent lies in developing an accurate knowledge base of facts and creating appropriate closures on participant interactions.

Generate

The second function of the logical agent in the system is to generate events. The possible actions forecasted for a particular day provides the set from which the agent chooses some actions based on a stochastic model for the generation of events. The stochastic model is Bayesian in nature. Therefore, the probability of a particular event occurring is conditional to specific circumstances and to the frequency of its occurrence. The details of this model are beyond the scope of this paper.

Unite and Compute

The mathematical agent reasons about the sensitivities of the simulated environment to its component aspects. The component aspects of the environment are interrelated. Random events, which create constraint violations in the simulated environment, have the ability to deflect the progress of a project from its planned path. For instance, a delayed material delivery can postpone a particular activity and all dependent activities and, in turn, affect indirect and direct costs and the remaining duration of the project.

The set of indicators [*Total Direct Cost (TDC)*, *Total Indirect Cost (TIC)*, *Remaining Duration (RD)*, and *Production Rate (PR)*] traps the sensitivity of the system to the essential aspects of time and cost. By maintaining a direct quantitative comparison between the As-Planned and As-Built values of each of the preceding quantities during the execution phase of the project, an accurate picture can be created regarding the stability of the simulated system. Delays in the As-Planned schedule indicate that there have been constraint violations in the project implementation and are therefore a pointer to instability in the system. By taking corrective measures to satisfy constraints, the participant aims at maintaining stability in the system.

Rojas and Mukherjee (2003a) explain the mathematical model that defines the relationships between direct cost, indirect cost, remaining duration, and production rate during the construction process. The equations are time dependent. In the simulation, programs compute each of the equations. We can denote each program by a function. The most basic parameters taken by the functions are the independent variables (defined over continuous domains), which describe the availability and quantities of materials at any point in time during the simulation. Given the unit costs of materials, the total direct cost at the time point T is given by

$$TDC[T] = \sum_i C_{i,T} \tag{3}$$

$$TDC^P[T] = \sum_i C^P_{i,T} \tag{4}$$

where

$$\forall i \cdot T \subset i \tag{5}$$

$TDC^P[T]$ and $TDC[T]$=values of the total direct cost across all context intervals as calculated from the As-Planned database and the total direct cost across all context intervals as calculated from the independent variables generated from the simulation, respectively. $C_{i,T}$ indicates the total cost of material, equipment, and labor used in the context interval (activity) i during the time point t. The superscripted cost $C_{i,T}^{P}$ denotes the As-Planned cost. Based on the mathematical model, the relationships between *Total Cost, TIC, RD, and PR* (the dependent/derived variables in the mathematical model) can be functionally expressed.

We used the following functional forms to express the interrelated operations

$$Y[M] = x_{\_}^{*}(p_{*}, Y[M] \dots) \tag{6}$$

where $[M]$ belongs to $\{[i,T],[i],[T],o\}$; and x belongs to $\{f,F\}$. x denotes a program which takes an independent (p_{*}) and derived parameters and returns a value $Y[M]$. Y=derived variable; and $[M]$ denotes the aspect of Y that the value describes; that is, Y can be described across contexts at a particular time point $[T]$ or only specific to a time activity element $[i,T]$ or across the interval of the activity $[i]$. M can take up a value of o only when it denotes a value that has been derived from static planned data. $Y[M]$ can, in turn, be fed to some other function as a derived parameter. In keeping with the preceding expression, the following equations can be written

$$TC[T] = TIC[T] + TDC[T] \tag{7}$$

$$TIC[T] = f_I(TIC[T-I], RD[T]) \tag{8}$$

$$RD[T] = f_2(RD'[i,T]) \tag{9}$$

$$RD'[i,T] = f_3(PR[i,T]) \tag{10}$$

$$PR[i,T] = f_4(PR_0, \eta) \quad \text{where } prod(\eta, i) \tag{11}$$

$$PR_0 = TDC^P[i] = C_i^P \quad (constant) \tag{12}$$

$TDC^P[i]$=total planned direct cost for the context interval i. $RD'[i,T]$ gives the remaining duration for the activity represented by the context interval i. The function f_2 represents a program that returns the remaining duration of the project at time t, $RD(T)$, given the remaining durations of the concurrent activities. This set of equations can also be used to calculate the same parameters for the As-Planned implementation of the project.

In the preceding equations, the values that can be considered as "raw data" or the independent variables are the components of the cost variable, the resource requirement (As-Planned), and the actual resource usage/availability (As-Built) data. This information can be read off from the simulation environment in the case of actual resource usage/availability and from the As-Planned database in the case of planned resource allotment/requirement. For example, the value of $prod(\eta, i)$, the productivity of workers during the context i, is a discrete environment variable that is read from $W'(i)$. The raw data compute to C, which can be calculated across all contexts for a particular time point ($\Sigma_i C_{i,T}$) or across a particular context from the As-Planned database (C_i^P). The information from the As-Planned database is static and is used only as benchmark constants. PR_0 is a constant. Therefore, we can reduce the whole system to (where F denotes a program)

$$\sum_i C_{i,T} = F_I(W(T)) \tag{13}$$

$$\eta = F_2(W'(i)) \tag{14}$$

therefore

$$PR[i,T] = f_4(PR_0, \eta) = f_4(F_2(W'(i))) \quad \text{(absorbing the constants)} \tag{15}$$

$$TC[T] = F_I(W(T)) + TIC[T] \tag{16}$$

or

$$TC[T] = F_I(W(T)) + f_I(TIC[T-I], f_2 \circ f_3 \circ f_4) \tag{17}$$

or

$$TC[T] = F_I(W(T)) + f_I(TC(T-I) - F_I(W(T-1)), f_2 \circ f_3 \circ f_4(F_2(W'(i)))) \tag{18}$$

The symbol ($\circ$) denotes interconnections (i.e., $f \circ g$ implies that the value returned by g is passed as a parameter to f). Eq. (18) is a systemic representation of the simulation. It provides a single relationship, which reflects the progress in the simulation system at any time t dependent on the state at $T-1$. Each of the quantities when calculated for both As-Planned and As-Built information will provide comparative trends, which can be used to detect instabilities in the system. The values for the latter will be constants across all simulations of the same project, while the variations in the values of the As-Built will show the expertise of the participant.

The mathematical agent works within the vocabulary of the functions defined earlier. The functions denoted in the lower case f are collectively represented by the $\varnothing$ operator. They are programs that compute within the mathematical model. The functions denoted in the upper case F are collectively represented by the $\oplus$ operation. They are programs that unite the appropriate values to the appropriate variables in the mathematical model. The need for this arises because the same programs compute the As-Planned derived variables as well as the As-Built derived variables. While the data for the As-Planned calculations come from the As-Planned database, the As-Built independent variables come from the simulation environment.

General Purpose Multi-Agent Framework

The GPF is a protocol that can be used by developers to put together different simulations using the conceptual foundation of constraint satisfaction, planning, and the semantics of interval temporal logic to represent and reason about the CM domain. It is akin to an application programming interface as a programming language that can be used to program situational simulations for the CM processes. Hence, the GPF components will belong to one of the following fixed classes: agents, entities, operators, and bases. Members of these classes can combine according to a well-defined grammar to form operations, which are the basic building blocks of any situational simulation programmed using the framework. In this section we define each of the four mentioned classes and discuss the grammar that governs the framework.

Each agent has a finite set of operators associated with it. Operators are reasoning mechanisms attributed to each agent. Agents use operators to reason autonomously and make changes to the environment. Changing values of variables and/or variable collections, which are referred to as entities, makes changes to the environment. The nature of variables and their classification have been discussed in an earlier publication (Mukherjee and Rojas 2003). Variables can be classified into discrete and continuous

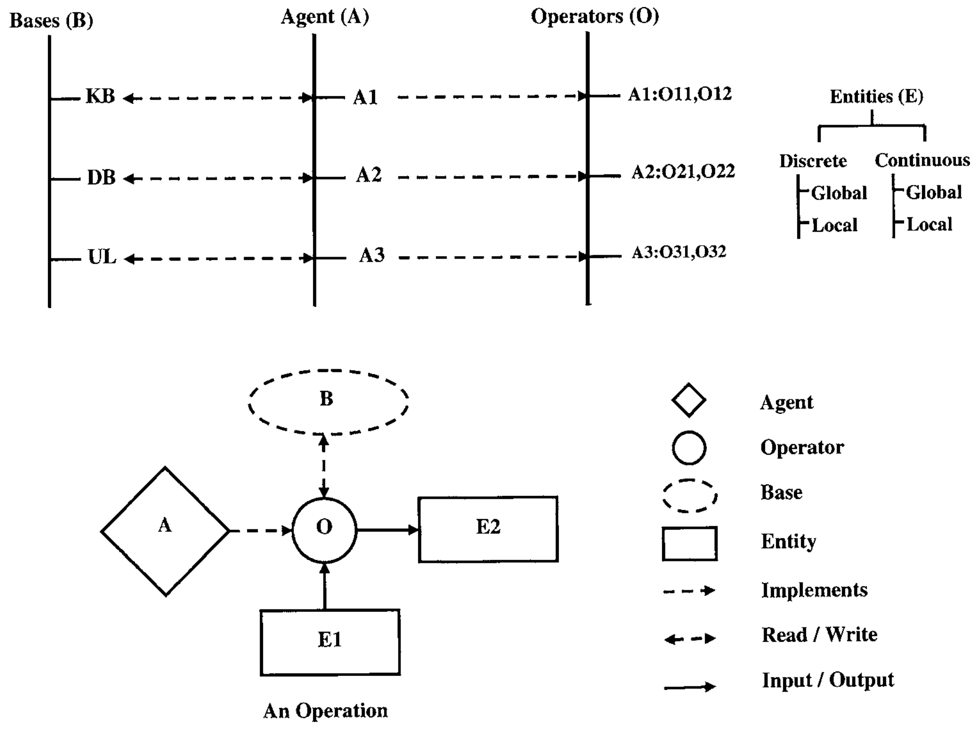


Fig. 1. Agent-operator-entity-base framework

variables, depending on the nature of the values they take up. Each variable can also be classified as activity specific (defines an aspect of a specific activity) or global (defines an aspect of the environment applying to all activities). Combinations of variables can also be classified into the following sets of disjoint entities:
- As-Planned Data, As-Built Data; and
- Activity Dependent Events, Global Events.

Agents function by implementing operators to change the values of entities.

Pro- and information attitudes (Woodridge and Jennings 1995) (inferred and factual information) are inherited by agents from knowledge bases, databases, and feedback from user interaction. This allows the agent to reason autonomously. Knowledge bases contain event definitions, and databases contain As-Planned cost and schedule information about the project being simulated. The framework consists of utility functions, which are not operators but can allow any of the agents to access the bases or to do routine repetitive tasks such as calculating remaining durations of activities or updating the floats on the schedule.

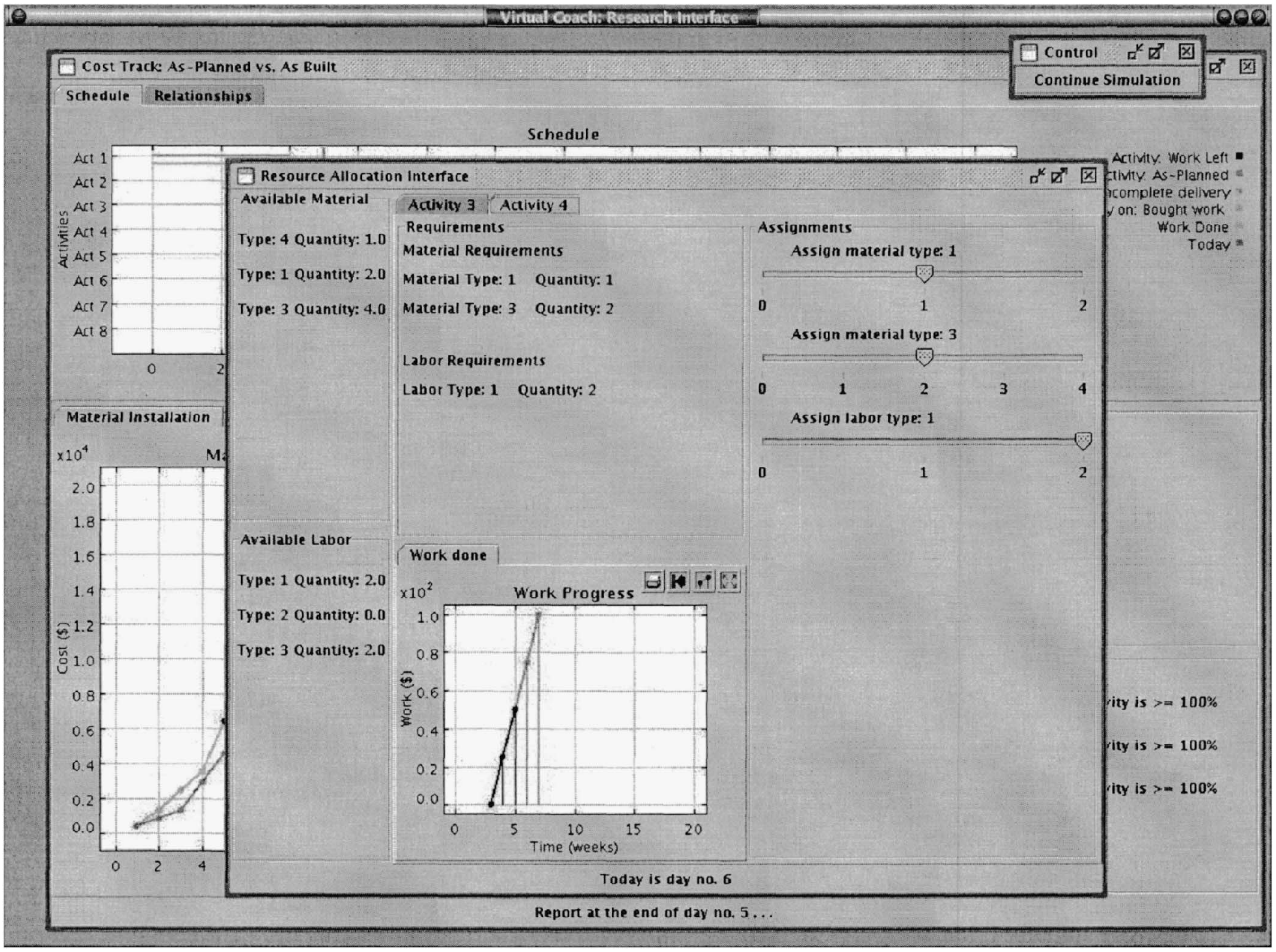
The basic unit of the GPF is an operation. In an operation, an agent inputs an information entity and outputs it to another information entity using a specific operator. Situational simulations built using the GPF can be expressed as a combination of operations, in series and/or parallel. This sets the grammar for creating simulations using the agent-operator-base-entity components of the framework, as illustrated in Fig. 1. In the parlance of the Java programming language, the GPF can be expressed as:

```
public interface Operation1{

  void 011 (Environment E);

  void 012 (Environment E);

  ...}

public class Agent1 implements Operation1{...}

public abstract class Variable{

  ...

  //Status of a variable: global or local

  public boolean global_local;

  ...}

public class DiscreteV extends Variable{...}

public class ContinV extends Variable {...}

public class Environment{

  ...

  //List of Discrete variables

  public DiscreteV discrete_list;

  //List of Continuous variables

  public ContinV contin_list;

  ...}

public class Simulation ( ) implements
Runnable

{
  public static Agent1 A1;

  public static Agent2 A2;

  public static Environment E;

  …

  run ( ){

    …

    //A typical operation

    A1.011 (E);
    …}
  public static void main (String Args [ ]){
    run ( );

  }

  public static void utility1 ( ) {…}

}
```

An implementation of such a framework would have definitions of multiple *Agents*, each implementing a particular *Operation* interface. The current pilot implementation of the framework, called the Virtual Coach, has three agents: the LA, the MA, and the visualization agent (VA). It also has the following events defined: bad weather, poor quality work, labor strike, no material delivery, and cost hike. Each of these events represents a resource constraint violation. The implementation also includes utility functions which read from the database and the knowledge base, run the scheduler, and calculate the remaining duration. In the next section we discuss the Virtual Coach implementation in greater detail.

Interfaces

There are three interfaces to the developed framework. One is the programmer's interface, the second is the developer's interface, and the third is the user interface to the developed situational simulation. We discuss the third interface in the next section. The first two interfaces allow the framework to be extensible. The implementation of such a framework will require the developer to input the following to simulate a specific project of their choice:



Available Material
Type: 4 Quantity: 1.0
Type: 1 Quantity: 2.0
Type: 3 Quantity: 4.0

Requirements
Material Requirements
Material Type: 1 Quantity: 1
Material Type: 3 Quantity: 2

Labor Requirements
Labor Type: 1 Quantity: 2

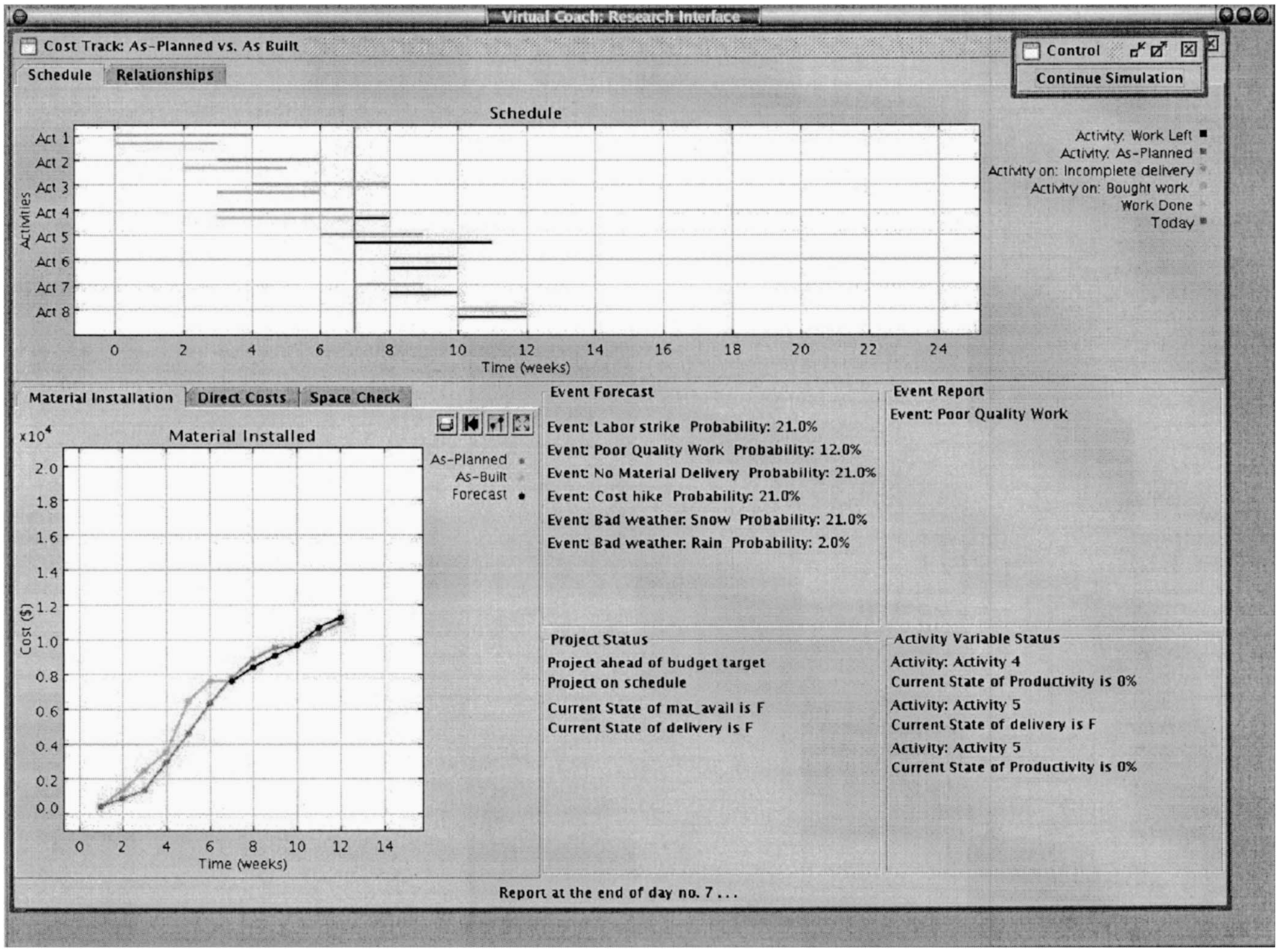
Assignments
Assign material type: 1
Assign material type: 3
Assign labor type: 1

Available Labor
Type: 1 Quantity: 2.0
Type: 2 Quantity: 0.0
Type: 3 Quantity: 2.0

Today is day no. 6

Report at the end of day no. 5 . . .

Fig. 2. Resource allocation interface



Event: Labor strike Probability: 21.0%
Event: Poor Quality Work Probability: 12.0%
Event: No Material Delivery Probability: 21.0%
Event: Cost hike Probability: 21.0%
Event: Bad weather: Snow Probability: 21.0%
Event: Bad weather: Rain Probability: 2.0%

Event: Poor Quality Work

Project ahead of budget target
Project on schedule
Current State of mat_avail is F
Current State of delivery is F

Activity: Activity 4
Current State of Productivity is 0%
Activity: Activity 5
Current State of delivery is F
Activity: Activity 5
Current State of Productivity is 0%

Report at the end of day no. 7 . . .

Fig. 3. End-of-day report

- As-Planned cost and schedule information;
- Definitions of variables characterizing the simulation (they can add to the defaults);
- Definitions of anticipated events using pre- and postconditions for the associated constraint violations; and
- Realistic probabilities of defined events based on historical data to enable the simulation to generate reasonable scenarios.

All this information is currently fed into a PostgreSQL database. The information can be fed into the database via Web forms and/or directly imported from MS Project using the XML format.

The programmers have access to the source code and are free to add more operations to each of the existing agents and/or to add more agents to the framework with dedicated operators. Thus, the developer can either use the functionalities provided by the current implementation of the multi-agent framework to simulate projects of their choice or can add more functionality to extend the current framework.

Virtual Coach Implementation

The Virtual Coach is a particular implementation of the discussed general purpose multi-agent framework. It is a situational simulation that is run by three agents: the LA, the MA, and the VA.

The MA operators are Unite and Compute, while the LA operators are Inference and Event Generation. Entities are defined as the different classes of information in the simulation environment. Every agent operation takes an information entity as an input and transforms it to another information entity (Fig. 1). Atomic entities can be combined to create superentities when the superentity is a logical parent of the atomic entities.

Systemic reasoning in the Virtual Coach is based on a mathematical model defined by Rojas and Mukherjee (2003a). It deals with reasoning about how events affect the net equilibrium of the system. If the project is executed As-Planned, then the system equilibrium is not affected. However, every time there is an event that results in a crisis, the equilibrium is disturbed. This allows the simulation to constantly give the participant feedback regarding progress as compared to the As-Planned implementation. These graphs can be seen in the lower left corner of the As-Planned versus As-Built screen.

The logical agent can create events and also infer events, which follow as a result of user interactions with the simulated environment. It can create events in the situational simulation by violating developer-defined constraints. It can also predict future constraint violations based on its ability to infer from facts in the knowledge base. A default knowledge base can be used, or developers can create their own knowledge bases. A detailed discussion of how the agent functions can be found in Mukherjee and Rojas (2003). In the current pilot implementation of the Virtual Coach, events can be generated as a result of the following constraint violations:

- No work can be done unless necessary material and labor are available;
- Outdoor activities cannot be productive during snowy weather;
- Overworking a labor crew reduces productivity and increases the chances of rework;
- Labor hired on an emergency basis costs more and is less productive; and
- Schedule constraints.

In the Virtual Coach, information visualization and user interactivity are handled by the visualization agent. The function of the VA is to make sure that the information being displayed to the user is consistent with the information in the simulation. The VA is also responsible for encoding participant reactions and passing them onto the other system agents in a format that can be easily processed.

The Virtual Coach pilot implementation currently runs a situational simulation for a twelve-activity hypothetical project with realistic constraint violations and event information. Figs. 2 and 3 provide screen shots of a preliminary deployment of the system. Fig. 2 is the resource allocation screen, which informs the participant of the total available resources in the environment and the total resource requirements specific to each ongoing activity in the simulation. Each activity panel also has a graph showing the As-Planned rate of work completion versus the As-Built one. The participant is allowed to assign more or less than the planned requirements, depending on availability, to accelerate or decelerate the project. In the absence of the necessary resources, the participant is also allowed to hire more labor and purchase more material at a premium price. This allows the participant to accelerate the project, at a higher cost, and is often an option to keep the project on schedule. While the direct costs go up, the participant does gain in terms of indirect costs by saving time.

Finally, Fig. 3 illustrates the report about progress at the end of a week. Participants can view the current state of the schedule as compared to the As-Planned schedule. They can also keep track of direct costs, indirect costs, and space requirements by following the graphics at the lower left-hand corner of the viewer. The lower right-hand corner of the viewer allows the participant to monitor the values of the discrete and continuous environment variables and to keep track of the possibilities of events that may occur in the near future. They can also keep track of recent events that have just occurred. This is important in allowing them to make future resource allocations. The final goal of the participant is to steer the project through generated scenarios and complete it within budget and time constraints.

Conclusions

The pilot implementation of the Virtual Coach situational simulation was administered to a sample of 19 senior-level construction management students, as part of a Project Management course at the University of Washington. Students were required to take a test before and after they ran the simulation. The pretest and posttest required students to rank (on a scale of 1–10), in their opinion, the importance of a list of factors in developing a plan for a 12-week period of a construction scenario. They were provided with a list of constraints governing the scenario. The constraints included schedule considerations, budget limitations, and the possibilities of events such as bad weather, material delivery delays, and labor shortage.

Four of the priority ratings assigned by the students, before and after using the simulation, were summed and compared using a paired-sample t-test. The ratings selected for analysis were those that related to the schedule and resource constraints and the need to anticipate delay on a project (giving priority to critical activities in case of delay, attention to space restrictions on site, anticipating future material delivery delays, accelerating activities to create a buffer for anticipated delay). The difference between the ratings was significantly different:

- Pretest: Mean=21.26, standard deviation=4.92;
- Posttest: Mean=25.31, standard deviation=4.70; and
- T-statistic: $t(18)=3.32$, p-value <0.01.

Based on qualitative feedback (postsimulation survey) from the students (16 of the 19 students thought that the Virtual Coach was a useful educational tool), the statistical significance of the post- and pretest results, and the high differential values of the confidence interval, we can conclude that an intervention using situational simulations could be useful in construction education. It is also encouraging that the students improved on issues related to the specific constraints that were programmed into the simulation. This reflects the constraint satisfaction philosophy that rules the underlying framework. It also indicates that curriculum developers could program simulations with constraints that are important for students to attend to. This study is preliminary, and the results are only indicative. They do, however, indicate the importance of exploring the use of situational simulations in construction management education.

References

AbouRizk, S. (1993). "Stochastic simulations of construction bidding and project management." *Microcomput. Civ. Eng.*, 8, 343–353.

Allen, J. F., and Ferguson, G. (1994). "Actions and events in interval temporal logic." *J. Logic Comput.*, 4(5), 531–579.

Allen, T. (1987). *War games*, McGraw-Hill, New York.

Allison, D., Wills, B., Hodges, L. F., and Wineman, J. (1997). "Gorillas in the bits." *Proc., VRAIS Ann. Conf.*, Institute of Electrical and Electronics Engineers, New York.

Anderson, J., and Evans, M. (1996). "Constraint directed improvization." *Proc., 11th Biennial Conf.*, Canadian Society for Computational Studies of Intelligence, Missassauga, Canada.

Brooks, R. A. (1991). "Intelligence without representation." *Artif. Intell.*, 47, 139–159.

Calder, R. B., Smith, J. E., Courtemanche, A. J., Mar, J. M. F., and Ceranowicz, A. Z. (1993). "ModSAF behavior simulation and control." *Proc., 3rd Conf. on Computer Generated Forces and Behavioral Representation*, Univ. of Central Florida Institute for Simulation and Training, Orlando, Fla., 347–356.

Choo, H. J., Tommelein, I. D., Ballard, G., and Zabelle, T. R. (1999). "WorkPlan: Constraint-based database for work package scheduling." *J. Constr. Eng. Manage.*, 125(3), 151–160.

Cremer, J., Kearney, J., Papelis, Y., and Romano, R. (1994). "The software architecture for scenario control in the Iowa Driving Simulator." *Proc., 4th Conf. on Computer Generated Forces and Behavioral Representation*, Univ. of Central Florida Institute for Simulation and Training, Orlando, Fla., 373–381.

Etzioni, O. (1993). "Intelligence without robotics. A reply to Brooks." *AI Mag.*, 14(4), 7–13.

Goldhammer, H., and Speier, H. (1959). "Some observations on political gaming." *World Politics*, 12(1), 71–83.

Hajjar, D., and AbouRizk, S. (1999). "Simphony: An environment for building special purpose construction simulation tools." *Proc., Winter Simulation Conf.*, P. A. Farrington, H. B. Newbhard, D. T. Sturrock, and G. W. Evans, eds., Univ. of Trier, Trier, Germany, 998–1006.

Halpin, D., and Woodhead, R. (1970). *A computerized construction management game*, Dept. of Civil Engineering, Univ. of Illinois, Urbana, Ill.

Hammond, J., Choo, H. J., Austin, S., Tommelein, I. D., and Ballard, G. (2000). "Integrated design planning, scheduling, and control with Desplan." *Proc., 8th Annual Conf.*, International Group for Lean Construction, Brighton, U.K.

Hanks, S., Pollack, M. E., and Cohen, P. R. (1993). "Benchmarks, test beds, controlled experimentation, and the design of agent architectures." *AI Mag.*, 14(4), 17–42.

Jaafari, A., Manivong, K., and Chaaya, M. (2001). "VIRCON: Interactive system for teaching construction management." *J. Constr. Eng. Manage.*, 127(1), 66–75.

Konolige, K. (1986). *A deduction model of belief*, Pitman, London.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). "Soar: An architecture for general intelligence." *Artif. Intell.*, 33(1), 1–64.

Lesser, V., et al. (2002). "Evolution of the GPGP/TAEMS domain-independent coordination framework." *CMPSCI 02-03*, Computer Science Dept., Univ. of Massachusetts, Amherst, Mass.

Martinez, J., and Ioannou, P. (1999). "General-purpose systems for effective construction simulation." *J. Constr. Eng. Manage.*, 125(4), 265–276.

McCabe, B., Ching, K. S., and Savio, R. (2000). "STRATEGY: A construction simulation environment." *Proc., Construction Congress VI*, K. D. Walsh, ed., ASCE, Reston, Va., 115–120.

Mukherjee, A., and Rojas, E. (2003). "Reasoning about actions and events in situational simulations." *Proc., Winter Simulation Conf.*, ACM, New York.

Ndekugri, I., and Lansley, P. (1992). "Role of simulation in construction management." *Build. Res. Inf.*, 20(2), 109–115.

Oppenheimer, P., and Weghorst, S. (1999). "Immersive surgical robotic interfaces." *Proc., Medicine Meets Virtual Reality*, Aligned Management Associates, Inc., San Diego, 242–248.

Ritchie, G. (1985). "Atlantis: The basis for management simulation development." *Simul./Games Learn.*, 15(1), 28–43.

Rojas, E., and Mukherjee, A. (2003a). "Modeling the construction management process to support situational simulation." *J. Comput. Civ. Eng.*, 17(4), 273–280.

Rojas, E., and Mukherjee, A. (2003b). "Visualizing situational simulation information." *Proc., Construction Congress* (CD-Rom), ASCE, Reston, Va.

Russell, S. J., and Norvig, P. (2002). *Artificial intelligence: A modern approach*, 2nd Ed., Prentice-Hall, Upper Saddle River, N.J.

Sawhney, A., Mund, A., and Koczenasz, J. (2001). "Internet-based interactive construction management learning system." *J. Constr. Educ.*, 6(3), 124–138.

Seel, N. (1989). "Agent theories and architecture." Ph.D. thesis, Surrey Univ., Guildford, U.K.

Soibelman, L., and Pena-Mora, F. (2000). "Distributed multi-reasoning mechanism to support conceptual structural design." *J. Struct. Eng.*, 126(6), 733–742.

Sucur, M., and Grobler, F. (1996). "Construction planning through multi-agent constraint satisfaction." *Proc., 3rd Congress of Computing in Civil Engineering*, ASCE, New York, 240–246.

Talukdar, S., Baerentzen, L., Gove, A., and de Souza, P. (1996). *Cooperation schemes for autonomous agents*, Engineering Design Research Center, Carnegie Mellon Univ., Pittsburgh.

Tambe, M., et al. (1995). "Intelligent agent for interactive simulation environments." *AI Mag.*, 16(1), 15–39.

Windschitl, M., and Winn, W. D. (2000). "A virtual environment designed to help students understand science." *Proc., Int. Conf. of the Learning Sciences*, B. Fishman and S. O'Connor-Divelbiss, eds., Lawrence Erlbaum Associates, Inc., Mahwah, N.J., 290–296.

Winn, W. D. (2002). "Learning in artificial environments: Embodiment, embeddedness, and dynamic adaptation" *Tech., Inst., Cognit. Learn.*, 1, 87–114.

Woodridge, M., and Jennings, N. R. (1995). "Intelligent agents: Theory and practice." *Knowl. Eng. Rev.*, 10(2), 115–152.