Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2011), *25*, 25–40.

doi:10.1017/S0890060410000156

Using contexts to supervise a collaborative process

AVELINO J. GONZALEZ,[1] SETSUO TSURUTA,[2] YOSHITAKA SAKURAI,[2] JOHANN NGUYEN,[1] KOUHEI TAKADA,[2] AND KEN UCHIDA[2]

[1]Intelligent Systems Laboratory, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, USA
[2]School of Information Environment, Tokyo Denki University, Tokyo, Japan

(RECEIVED April 15, 2008; ACCEPTED December 16, 2009)

Abstract

This article describes an investigation into the feasibility of using contextual reasoning to monitor and supervise the collaborative work of several knowledge workers working together on a project. Managing large and complex projects is a difficult task that requires *situational awareness* by the project manager to be able to be proactive when possible and to react correctly in the presence of events. In complex projects, effective oversight of the project personnel and the progress of the project are essential in ensuring that project objectives are met. This is especially true of projects that require contributions from various experts, whose interaction may be limited to a Web-based collaborative tool. Such oversight is typically the job of a project manager who is tasked with avoiding cost overruns, shipment delays, and ensuring product effectiveness. We utilize *context-based reasoning* and *contextual graphs* as the tools of choice for implementing an agent that emulates the function of a competent project manager. We use rocket design and manufacture as the domain to evaluate our technique. We use a public domain rocket design software package developed by the National Aeronautics and Space Administration as a guide to the domain. The article describes the investigation, its results, and the related works in a collaborative design project.

Keywords: Computer Supported Cooperative Work; Context-Based Reasoning; Project Management

1. INTRODUCTION

Computer-supported cooperative work (CSCW) seeks to provide tools and techniques that effectively manage the process of combining the talents and skills of several sparsely located individuals in a closely coupled project. Cooperative work can be defined as those that cannot be done by one entity working on his/her/its own, but must instead rely on help from other entities to meet the collective objective. Military operations, team games, and large-scale design, and/or construction projects share this common trait.

In this research, we investigate the feasibility of building an intelligent CSCW tool for project management that can autonomously support a human project manager (PM) by performing the same functions done by a competent PM. This can be used to provide advice and decision support to the PM or to unilaterally perform the tasks normally done by the PM. Our work focuses primarily on projects where the collaboration is done via the Internet.

Reprint requests to: Avelino J. Gonzalez, Intelligent Systems Laboratory, School of Electrical Engineering and Computer Science, University of Central Florida, 4000 Central Florida Boulevard, HEC 346, Orlando, FL 32816-2450, USA. E-mail: gonzalez@ucf.edu

We specifically seek to develop a tool that can provide a PM with a full awareness of the situation faced in the project. *Situational awareness* (SA) reflects a person's ability to be fully cognizant of the situation and of its implications. The field of SA was pioneered by Endsley (1988) for training US Air Force pilots. SA is particularly important in conflict-based situations such as sports games or military operations, where a full understanding of the situation is necessary before effective action can be taken by an agent, human or artificial. Endsley (1988) defines SA as

> . . . the perception of the elements in the environment within a volume of time and space, their comprehension of their meaning and the projection of their status in the near future.

Endsley's definition implies three levels of SA:

1. Perception of the elements in the environment—where is my opponent? What kind of opponent am I facing?
2. Comprehension of the current situation—what is my opponent trying to do?
3. Projection of future status—what does that mean vis-à-vis my mission? Level 3 involves short-term planning and option evaluation when time allows.

Much research has been done in SA by the US military in efforts to improve the war-fighting capability of its forces. Early work on SA included Stiffler (1988), Dominguez (1994), Gaba et al. (1995), and Harwood et al. (1988), as well as Endsley's pioneering work. Gero (2004) discussed SA in the context of design, so that the design took into account its environment and an interaction therewith by the design agent (DA). Endsley also wrote an updated review of SA (Endsley, 2000). More recent works support our hypothesis that SA can be useful in nonmilitary applications (e.g., Nehme et al. 2008; Feng et al., 2009; Sapateiro & Antunes, 2009). The application of SA to team collaboration was also investigated extensively in the literature (e.g., Gutwin & Greenberg, 2004; Fan et al., 2005). Gutwin and Greenberg's work in particular supports the notion of using SA for tasks such as those seen by a PM.

A full discussion of SA is beyond the scope of this article. We refer the reader interested in learning more about SA to Clancey (1997). Nevertheless, we assert that based on the large body of research in SA, it would be highly beneficial for a tool that supports PMs to be based on an approach that enhances SA. We address this problem (and opportunity) through a combination of two contextual reasoning paradigm called *context-based reasoning* (CxBR) and *contextual graphs* (CxG). Before describing our approach in detail, a review of the relevant literature in automated project management support is appropriate, and this is found in Section 2. Section 3 describes our context-based approach to PM support. This section includes brief discussions on CxBR and CxGs, as well as the motivation for our use of these technologies. Section 4 describes the prototype, and Section 5 describes the test program undertaken to evaluate the hypothesis as well as the results. Section 6 includes the summary, conclusions, and future work.

2. CSCW IN PM SUPPORT

There is clearly a stated need to provide tools to assist PMs (Nienaber & Cloete, 2003; Smith et al, 2005). Fox and Spence (2005) provide an excellent review of the literature on the need for special project management tools for successful software project management. Brewer (2005) discusses the desired qualities of a PM for software projects, and discusses how these can be taught in a college curriculum in information science. Abernethy et al. (2007) likewise discuss the teaching of software project management, but they advocate doing so from an experiential standpoint.

Bardram (1996) developed a tool called *Project Manager* to implement CSCW by assisting the management of projects at a manufacturing company. This tool, however, was not designed to be an intelligent assistant to a PM, but rather to serve mostly as a communication medium between interested parties in a company. Medina-Mora et al. (1992) describe a tool called the Coordinator that controls workflow. The intent of this tool is similar to our work described here. However, the Coordinator does not appear to have the capability of being proactive when changes in the plan or budget take place. Furthermore, no details of the evaluation of the prototype are included in this paper.

More recent work on the subject includes the Virtual Design Team (Levitt & Nissen, 2003), which provides a simulation environment for design project teams to predict backlogs and other such organizational obstacles. However, the Virtual Design Team is not designed to assist a PM in real time on a collaborative project. Wu and Kotak (2003) describe their collaborative project management system based on multiagent architecture. Their work in many ways resembles our own, but they concentrate on project scheduling and rescheduling, rather than on our more general goal of supporting a situationally aware PM in potentially conflictive situations.

Lee et al. (2006) build an ontology to provide a project monitoring and control function to the capability maturity model integration process (a process improvement approach for organizations). However, their work focuses on the natural language capabilities required to monitor the progress of collaborators based upon their written report. Nienaber and Cloete (2003) discuss a multiagent-based architecture to support software development project management. Although their work points in the same direction as ours, they merely provide a general architecture that covers the functions that need to be addressed in this important application. Fox and Spence (2005) present an interesting study on the suitability of current project management tools to different cognitive strengths of individual PMs.

Lee and Pena-Mora (2005) propose and implement the use of system dynamics in a simulation of a construction project. Smith et al. (2005) address the problems of software project management from a risk analysis viewpoint, as does Bannerman (2007). They also include in their work the concept of learning from past examples of software development projects.

3. OUR CONTEXT-BASED APPROACH TO AUTOMATED PM SUPPORT

The literature reviewed above on project management describes tools for PM support that can be used for specific and rather limited purposes. No project approaches the problem with a fully autonomous tool that can track the progress of the project and is unilaterally able to identify problems and propose (and implement) solutions to them. Furthermore, the SA of a PM is never mentioned.

We assert that a PM can be best assisted by an intelligent tool that provides him/her with the SA to always be "on top" of the situation. Only by knowing this is he/she able to undertake action to resolve or, even preferably, prevent problems. Our work focuses on investigating the feasibility of a situationally aware agent (called *Project Manager Agent*, or *PM-Agent*) that assists a human PM in a project involving design and manufacturing. This agent duplicates the function of a PM and (indirectly) alerts the PM to any situation that arises that may have negative implications on project objectives.

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

The project objectives of interest are related to three universal constraints (Nienaber & Cloete, 2003): *scope*, *cost*, and *delivery time*. Furthermore, a PM-Agent recommends corrective action, and in our prototype, is able to unilaterally undertake such action. However, our PM-Agent does not design.

A PM-Agent monitors the progress of the collaborators and discovers conflicts at the earliest possible time. We use a context-driven paradigm to provide a situationally aware agent for PM assistance. Using context to guide CSCW is not a new concept. Borges et al. (2004) introduced the idea of contextual reasoning in CSCW. They discussed it in terms of providing a tool to facilitate the design of a system. We take their general idea further by applying it to an entire project and providing a tool to do so.

In particular, we employ CxBR and CxGs as the means for imparting intelligence and SA to the PM-Agent. By its context-driven nature, CxBR facilitates the introduction of SA in its agents by providing an agent with what can be expected in a situation. This ability to suggest what to expect under the current situation can facilitate SA without the need for a complex cognitive model of SA. CxBR has been successfully used to model tactical military agents that represent human forces in a battle simulation. CxBR was developed by the lead author (Gonzalez & Ahlers, 1998; Gonzalez et al., 2008). In the next section, we introduce contextual reasoning in general and CxBR and CxGs in particular. We explain our approach in the subsequent section.

3.1. Contextual reasoning and CxBR

Contextual information is constantly used by humans in our everyday reasoning activities. Knowing the context of a conversation reduces the need to explicitly reveal information that is essential for a meaningful discourse. In the same way, contexts play a role in our everyday lives by incorporating much information implicitly, thereby avoiding extensive explicit communications. For example, knowing that one is driving a car in an interstate highway carries some well-known assumptions and expectations about the process. For instance, the speeds of the vehicles are comparatively high, and there will (normally) be no oncoming traffic. Therefore, a tire blowout has much more serious implications in an interstate highway than if it were to happen in a normal city street where speeds are much lower. It also has some functionality that must be brought to bear in order to manage the situation properly, such as, for example, how to pass another vehicle in an interstate highway.

Contextual approaches have been widely accepted for representing human behavior. Several investigations have implemented context-driven behaviors to represent human behavior. Turner (1998), Gonzalez and Ahlers (1998), Gonzalez et al. (2008), Kokinov et al. (2007), and Brézillon (2005) are some such works among many others. They all have in common that behaviors and expectations are implicit when a person or an agent is in a given context. Therefore, knowing one's context (i.e., being situationally aware) allows one to apply several procedures that permit the agent to successfully manage that context. When the external (or internal) situation changes, then the agent must know how the context has changed and shift to a new context that best addresses the emerging situation.

CxBR (Gonzalez & Ahlers, 1998; Gonzalez et al., 2008) is an open architecture designed expressly for efficient control of time-based missions in a real or simulated world. Its ability to efficiently and effectively control agent platforms in tactical situations has been proven in several application areas such as automobile driving (Henninger & Gonzalez, 1997; Brown, 1994; Norlander, 1998; Gonzalez et al., 2000; Fernlund, 2004), armor warfare (Barrett & Gonzalez, 2002), poker (Stensrud, 2005), submarine warfare (Gonzalez & Ahlers, 1998), maritime navigation (Gumus, 1999), and antisubmarine warfare (Proenza, 1997). CxBR is based on the following ideas:

1. A situation calls for a set of actions and procedures that properly addresses it.
2. As the mission plays out over time, a transition to another set of actions and procedures may be required to address new situations.

Things likely to happen under the current situation are limited and well known (SA). CxBR encapsulates knowledge about appropriate actions and/or procedures for specific situations, as well as compatible new situations, into hierarchically organized contexts. All the behavioral knowledge is stored in th*e Context Base* (i.e., the collection of all contexts). The top layer of contexts in the hierarchy contains the *Mission Context*. At the next layer are the *Major Contexts* and then several layers of *Minor Context*s (e.g., Sub-Contexts, Sub-Sub-Contexts).

Figure 1 shows an example of a context structure from a simple, unlabeled context base. Mission Contexts define the mission to be undertaken by the agent. Although it does not control the agent per se, the Mission Context defines the scope of the mission, its goals, the plan of execution, and the constraints imposed (time constraints, weather, resources, etc.).

The Major Context is the primary control element for the agent. It contains functions to control the actions of the agent, the rules that provide the SA, and a list of compatible Major Contexts that can succeed the currently active one. Identification of a new situation can now be simplified because only a limited subset of all situations is possible under the currently active context. Sub-Contexts are abstractions of functions performed by Major Contexts to reduce complexity. Sub-Contexts may be used by more than one Major Context; this encourages reusability. Sub-Contexts will know when to deactivate themselves, either upon completion of their actions, or prematurely when in reaction to events in the environment.

Figure 2 depicts how an active Major Context interacts with the agent to control its actions. The agent resides in the environment (real or simulated). It perceives the environment, and in turn, its actions can affect the environment. The environment reflects its information in the *Global Fact Base* (GFB), onto which all relevant information about the

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

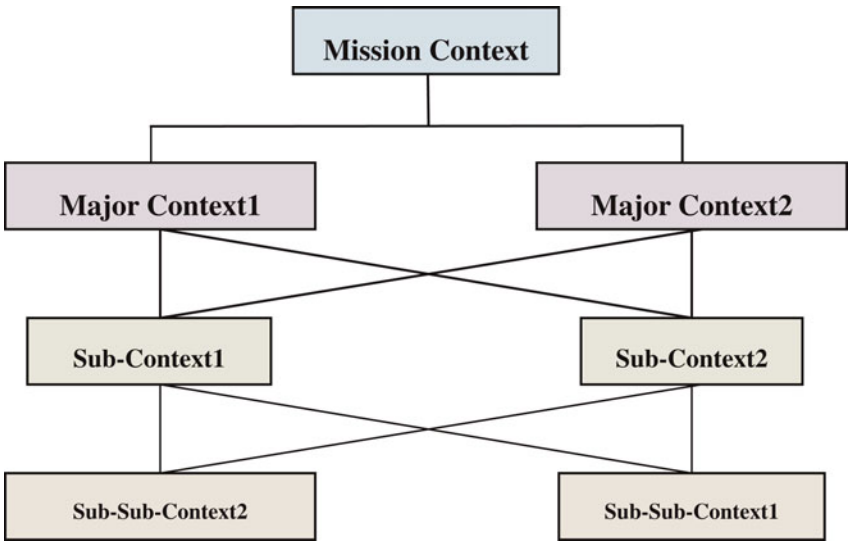

Fig. 1. The context-based reasoning organization of contexts. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

environment is posted in a continual basis for all agents to see. The agent (an automobile driver in Fig. 2) also has a *Local Fact Base* (LFB) that contains the same type of information about the world as the GFB except it is only visible to the agent with whom the LFB is associated. The LFB typically contains information about the internal state of the agent not known to the other agents.

The actions of a CxBR agent are determined by the *CxBR Control System*, which encompasses the *CxBR Framework* and the *Context Base*. The context base contains the application-specific knowledge (the contexts), whereas the framework is the mechanism that activates the contexts and executes the functions and rules found therein. It is somewhat analogous to the traditional expert systems concept of the inference engine and the knowledge base. This CxBR control system can even include a traditional rule-based inference engine when rules form part of the context base. The control system, however, must also be broader and more flexible to be able to work with many types of knowledge representations, including conventional functions and methods.

One and only one Major Context is always *active* for each agent, making it the sole control element for the agent. When the situation in the environment changes, a transition by the agent to another Major Context may be required to properly address the emerging situation. For example, a driver agent maneuvering its car in a freeway may come to an exit at which it must leave the freeway and thereafter drive on a rural two-lane road. The knowledge about how to manage the two types of roads is somewhat different, so the agent must recognize that the exit represents the end of one context and the beginning of another. Transitions between contexts can typically be triggered by events in the world; some are planned, but others are unplanned. Events internal to the agent (e.g., mechanical breakdown) can also trigger transitions. Expert performers

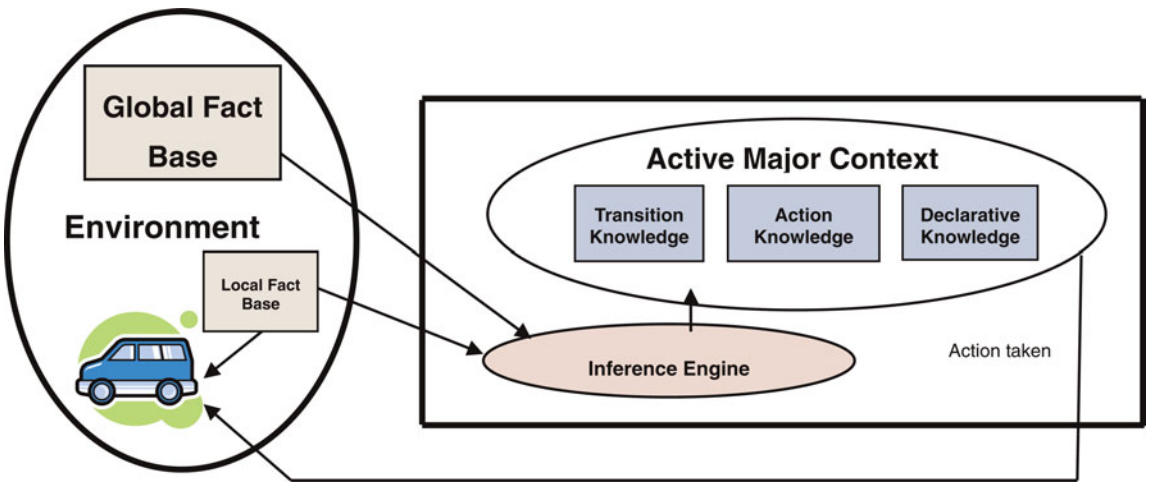

Fig. 2. The active Major Context controlling the agent. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

are typically able to recognize and identify the transition points quickly and effectively.

The CxBR framework incorporates the CxBR algorithm and places its execution within an iterative program loop of a user-defined frequency. It is responsible for activating and deactivating Major Contexts, executing the inference engine, and calling conventional functions within the Context code. Figure 3 shows the location of the CxBR Framework for a CxBR agent.

The CxBR Framework algorithm is as follows:

1. **The transition criteria for planned transitions is sent to the Major Contexts.**
2. **The initial (Default) Major Context is activated.**
3. **For every program cycle, until a goal is reached or becomes impossible to reach (other end of mission criteria), do:**
   a. Determine whether mission goal(s) attained
      - If yes, go to step 4
   b. Determine whether other end of mission criteria reached
      - If yes, go to step 5
   c. Premises of all Transition Rules are evaluated to determine whether the situation has changed enough to warrant a transition to a new active Major Context
      - If change in situation exists, then
        - Select new Major Context to be activated
        - Execute transition to activate the appropriate new Major Context
        - Initialize new active Major Context
        - Return to step 3(a)
   d. Execute control function(s) to control agent actions
   e. Premises of all Action Rules are evaluated to determine whether the any Minor Context is to be activated
      - If yes, then Action Rules fired to activate Sub-Contexts
        - Initialize Minor Context
        - return to 3(d) inside Minor Context activated
   f. Return to 3(a)
4. **Mission successfully completes.**
5. **Stop.**

In summary, CxBR is a very intuitive, efficient, and effective representation technique for human behavior that facilitates SA. The original description of CxBR can be found in Gonzalez and Ahlers (1998). An updated description can be found in Gonzalez et al. (2008). We use CxBR in this project to control the actions of the PM-Agent.

3.2. CxGs

CxGs are contextual paradigms that are similar in some ways to CxBR, but they are quite different in other ways. CxBR and CxGs are synergistic with each other because the latter can serve to organize the knowledge within a Major or Sub-Context within the CxBR paradigm.

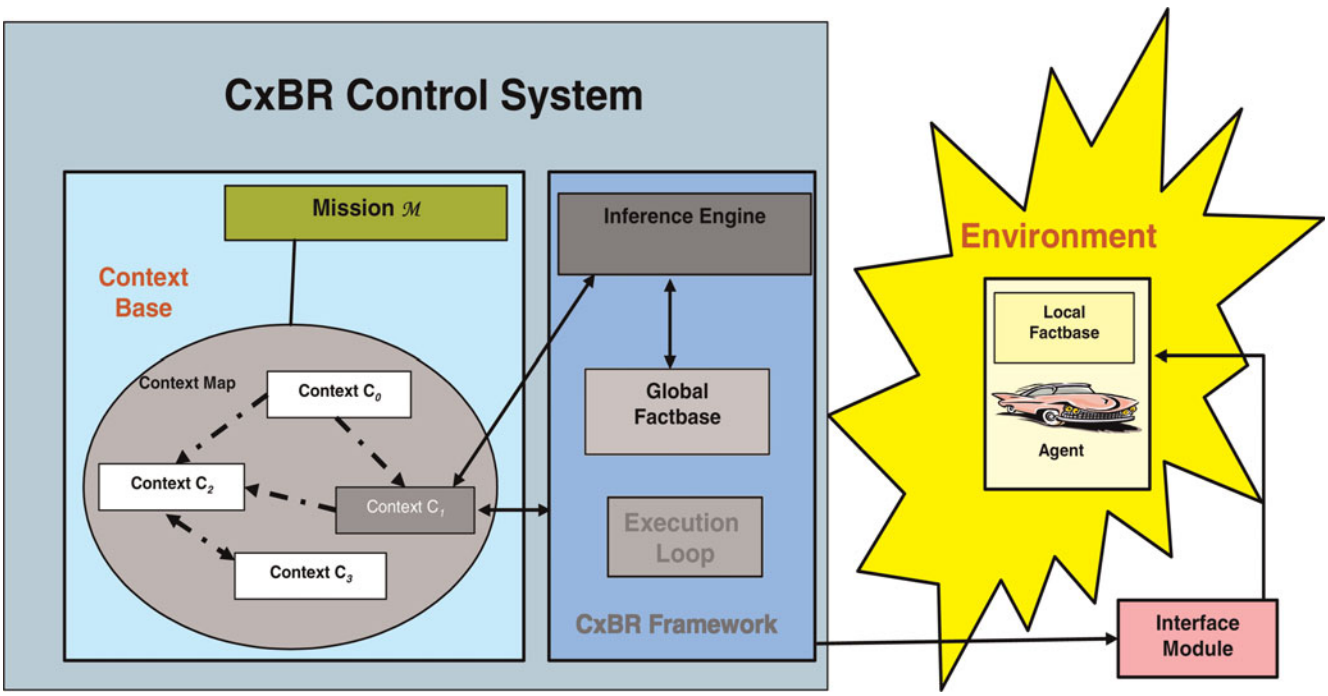

Fig. 3. A context-based reasoning control system controlling an agent in the environment. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

CxGs were conceived by Brézillon (2002) to better organize procedural knowledge in a contextual fashion. They are acyclic graphs that have one beginning point and one ending point. The share some common traits with decision trees, except that they incorporate actions and activities, and permit flexibility in how these actions are defined. Their ability to refine the context through contextual nodes allows the context to be defined in finer and finer granularity, all within the CxG itself. An example of a CxG is displayed in Figure 4.

CxGs have several basic components. *Contextual nodes* are the large dark circles in the graph of Figure 4. They generally pose a question, either to the human user or to the agent, and the answer permits further definition of the context of the problem. Contextual nodes can branch in two or more directions to take alternative courses of action, depending on the determined context. *Actions* are represented by the light-shaded squares. These are actions executed by either a human user or a software agent to accomplish something or to learn something. The results of an action can be taken in consideration in a subsequent contextual node. Several sequential actions can be grouped together into an *activity*. Activities are shown as the soft-edged squares in Figure 4. Last, recombination nodes are the small circles that serve to reunite several branches created in a prior contextual node. In effect, they serve to create subgraphs within a larger graph.

CxGs are entered on the left-hand side and exited on the right-hand side. The first node can be either a contextual node or an activity. The last element must be a recombination node. The path cut through the CxG by the answers provided to contextual nodes and subsequent actions taken is called the *proceduralized context*. It can serve as the record of the reasoning process for the purposes of explanation. The reader is referred to Brézillon (2002) for more details on CxGs. Although Brézillon has published more recent works on CxGs, his 2002 article provides an excellent description of the conceptual basis of CxG.

3.3. The project management process and its representation in CxBR and CxG

The same expectation-driven feature of contextual reasoning can be used to advantage in CSCW systems. Before discussing implementation details in the next section, we briefly describe the project management process in general terms to provide the basis for later assumptions.

A PM has a time-based mission: to bring the project to completion while meeting the goals stated *a priori* by her management and/or the customer. Typically, a PM is an experienced individual in the domain of the project, such as engineering design, manufacturing, construction, or nontechnical missions such as producing a documentary film or organizing a major event. However, it is unlikely that the PM will be sufficiently knowledgeable to override decisions made by the experts without seeking some outside expert advice. Most expert contributors to a project have only a limited view of the entire project. It is the PM who understands the overarching scope of the entire project and must make decisions based on the best outcome for the overall project.

There are assumed to be several people involved in a project. Otherwise, there would be little need for a CSCW approach. Some of these may be integral members of the project and thereby directly controlled by the PM (e.g., designers, schedulers, accountants). Others involved may indirectly contribute to the project but are not controlled by the PM (e.g., customers, upper management). Yet others may have only peripheral, long-term influence over the project (e.g., politicians making decisions that affect the price of oil, union negotiators). We only consider the first two types of people in our work.

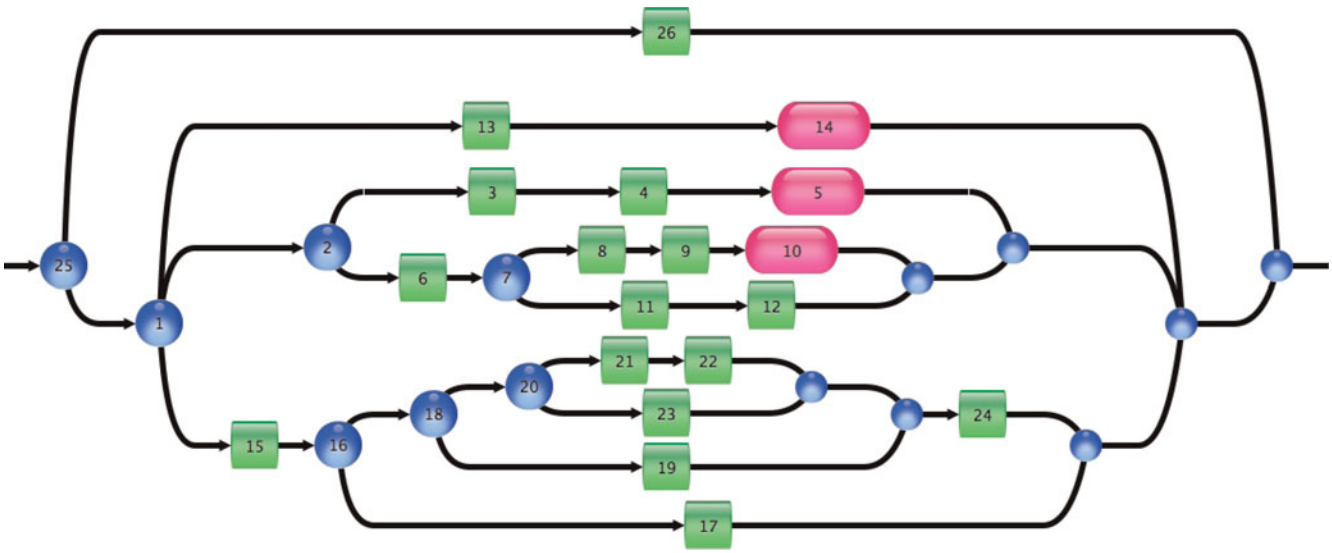

Fig. 4. A contextual graph. Reprinted from "Integrating two context-based formalisms for improved representation of human tactical behavior," by A.J. Gonzalez and P. Brezillon, 2008, *Knowledge Engineering Review 23(3)*. © 2008 Cambridge University Press. Reprinted with permission. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

The life of a PM therefore consists of composing a plan and then monitoring the events that affect the project plan and making decisions and taking action that lead to the project being ultimately successful. Events that affect the project positively or negatively, unfold on a daily basis. The PM must interpret those events vis-à-vis the project, and react to them appropriately. Throughout the project, the PM must have complete understanding of all events in the environment that may impact the project's success. This is the aforementioned SA. Examples could be completion of an internal milestone, a shift in project goals because of changes in customer requirements or upper management philosophy, and reduced availability of funds for completion of the project. There must be some common information medium upon which unfolding events that are relevant to the project are posted. In effect, this medium reflects the relevant environment. The PM must be able to continually and periodically monitor the contents of this medium to gauge the progress of the project. For our PM-Agent, this medium is the GFB. We follow no particular protocol in the communications between the environment and the CxBR control system through the GFB. We chose to use a simple, fact-based format where a fact is posted as a list of attributes that can be decoded to extract the information contained. This is a common practice in production systems.

The GFB does not evaluate the consistency of the facts posted therein. It is conceivable that facts that conflict with or contradict each other could be simultaneously present in the fact base. Therefore, there is no error detection for the inputs. Admittedly, this is also a function for a human PM and should at some point be done by the PM-Agent. We leave this for future research.

3.4. Motivation for implementing a context-based PM-Agent

The motivation for our prior work on CxBR was the need to more simply and efficiently model tactical behaviors in systems where a human would normally be in control of a platform. One characteristic common to these applications is that decisions must be made on a continual basis by the decision maker and that these decisions are found at different levels. For example, an automobile driver must always keep the car in the lane and at the appropriate speed (low level), decide when to commit to a gap in traffic to make a left turn (middle level), and make decisions on the route taken and plan another route if there are any delays in the original route (high level). Furthermore, CxBR applications are for agents that must deal with their environment over a long period of time (several minutes to months). The agent's mission requires constant action, interruptible at any time by circumstances, and constrained by what can be physically or legally done.

For these applications, CxBR links the knowledge applicable to a specific situation faced by the agent in an intuitive fashion, thereby pruning the search space for an agent that needs to quickly decide what to do next. Furthermore, it facilitates SA by always knowing the current situation and what can be expected therein. This expectation-driven process can facilitate knowing what to do because the current situation has been correctly identified.

The *Project Manager* application meets some, but not all of these characteristics. A PM works in an environment that does not require continual decision making. In most cases, only when a problem or a milestone arises does it require her attention. Furthermore, the deliberative process is not always subject to interruptions. Finally, the task of a PM is to resolve conflicts, something covered by other approaches such as constraint programming techniques.

Nevertheless, the many situations that a PM encounters can make a contextual approach beneficial. Further, the ability to recognize the situation and carry out a plan of action is greatly facilitated by CxBR. We enhance our CxBR approach here with CxGs to add the more finely defined contextual information and to further organize the procedural aspects of the PM's job. Note that CxBR and CxG are not constraint-based approaches in the formal sense of the latter. As such, these in no way replace the power of constraint programming to find a solution. However, the context-based organization of the solutions can help better define the ranges of some of the variables that are to be satisfied. This can simplify the constraint-based computations in many cases.

There are some similarities between CxBR and a blackboard architecture (BBA; Engelmore & Morgan, 1988). BBAs were originally conceived to promote collaborative problem solving by creating a blackboard where all work could be done. This is very similar in concept to our GFB. In its most basic terms, a BBA defines a control agent that mediates among several competing knowledge modules, where each vies to contribute something to the solution. The control agent passes a token to a module and allows it to work to solve the problem or part thereof. Although each module can be likened in some ways to a Major Context, one might come under the mistaken impression that CxBR and BBAs are the same. The original BBA concept allowed the selected module to complete its work and then control would shift back to the central manager to determine to which other module it would pass the token to allow it to execute. No interruptions were allowed, and there was no interaction with an uncertain environment. The decision on where to shift the focus of attention was based solely on the output of the active module as posted on the blackboard. In effect, the original BBA was designed as a problem solver that depended on several team members to solve the overall problem. In contrast, CxBR was designed for continual control of an agent undertaking a tactical mission. It differs from a BBA in that control is distributed among the Major Contexts, rather than resting with a central control agent. The former is preferable because the Major Contexts can have a better appreciation for the situation than would a central agent. Otherwise, the central agent would have to contain all of the knowledge found in all Major Contexts. This would result in extensions being rather difficult to achieve. Furthermore, CxBR defines the different levels of granularity in the contexts, whereas BBA leaves that up to the respective modules. This is even

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

better defined when CxBR is combined with CxGs as we do in this investigation. The defined interaction between the Major Contexts, both in a horizontal (with other Major Contexts) and in a vertical manner (with Minor Contexts in the context hierarchy) separates CxBR significantly from BBAs.

In summary, the efficiency of a context-based organization and representation as well as its advantage for facilitating continual SA provides significant advantages for representation of a PM's function. In the next section, we describe our prototype used to determine the feasibility of our approach to this new application.

4. A PM-AGENT FOR APPLICATION TO A ROCKET DESIGN PROCESS

As mentioned earlier, our work is applied to the design and manufacture of a small sounding rocket for lower atmosphere launches. The PM is charged with ensuring that a design of the rocket be done according to the customer specifications, on time, and within budget. The specifications, the time line, and the budget are reflected in the Mission Context. We specifically make use of a National Aeronautics and Space Administration (NASA) simulation system (NASA, 2003) that can simulate the launch of a specifically designed rocket, and determine whether the launch of a rocket design will be successful.

The design of our rocket in this application is actually more like a specification of values for several attributes. It requires that the following components be designed (specified):

- the engine and body material;
- the rocket structure, including the number of stages and the thickness of the fins; and
- the control system.

The design decisions are made by *DAs* who, with their experience and knowledge, can make the best decision for the stated requirements. DAs can be either human or software. In a CSCW environment, it is assumed that they will be human. However, in some cases they may be software agents. We assume the DAs to be human collaborators in our prototype.

The PM-Agent has access to all "postings" of design activity made by the various DAs onto the GFB (see Figs. 2 and 3). An LFB was not deemed applicable in an application where open knowledge is assumed. Our PM-Agent reviews and interprets the postings made by the various DA collaborators in the design and manufacturing project, and determines in which context the overall project finds itself. In cases where it needs expertise to make these decisions, the PM-Agent can rely on *Auxiliary Agents* (AAs). These can be human or software. We assume that they are software for this prototype.

4.1. Conceptual architecture

This section describes how we reflect the entire system in our prototype. Note that this is not a software architecture, but merely a conceptual description of the agents involved. The formal software architecture follows later.

Figure 5 illustrates the conceptual relations between the agents that participate in our prototype. All events related to the project are to be posted in the GFB. A PM-Agent need only monitor the GFB to learn what new events affect the project.

4.1.1. The collaborating agents

We define three DAs and five AAs. The DAs will contribute the design of the engine, the material of the rocket body, the design of the rocket structure, and the selection of an appropriate control system. Each of these DAs has to decide on a limited number of design parameters, each of which has relatively few possible choices. These agents are the Engine DA (DA-1), the Structural DA (DA-2), and the Control DA (DA-3).

The five AAs include the NASA RocketModeler simulator (AA-1), a Scheduling Agent (AA-2), and a Cost Agent (AA-3). AA-1 (the rocket simulator) determines the validity of the rocket design. It does this by incorporating the design into a simulated rocket that is virtually launched. If the simulated rocket attains the specified altitude with the appropriate payload, it is considered to be an acceptable design. AA-2 determines whether any delays are expected as a result of a design or of an external event. AA-3 is in charge of determining the financial impact of design decisions and/or of delays on the project cost. In addition to the main AAs defined above, AA-0 will compute the initial cost and schedule plans as well as maintain the official schedule and cost estimates. The PM-Agent can only make changes to the expected cost or the schedule through AA-0. Finally, AA-4 is a problem solver loosely based on constraint-based algorithms that when given all the constraints involved in making decisions, it selects the values for the various variables that meet the criteria defined as constraints. By "loosely based" we mean that it does not employ a formal CSP algorithms but instead employs a rule-based approach that, for a simple application such as this one, works acceptably well. If the system is overconstrained (no solution is possible), AA-4 will inform the PM-Agent of this. These AAs are called by a PM-Agent when it deems necessary. Although DAs are human for this experiment, the AAs are software agents in our prototype. However, only the PM-Agent is based on CxBR/CxG.

The PM-Agent examines the GFB on a regular basis. For the purposes of this application, we have set this sampling frequency to 1 day. However, for the purpose of practicality in demonstrations, it has been accelerated to once every few seconds to simulate several months in the space of a 10-min simulation. Upon retrieving the relevant information from the GFB, the PM-Agent will assess the situation. Its transition rules decide which new Major Context should become active, if any. If different from the currently active Major Context, this newly selected Major Context becomes activated and the current one deactivated. The Major Context contains the action appropriate for the context and executes this action through the framework. It will continue to do so until the situation is resolved or until the next sampling cycle if there are no problems to address.

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

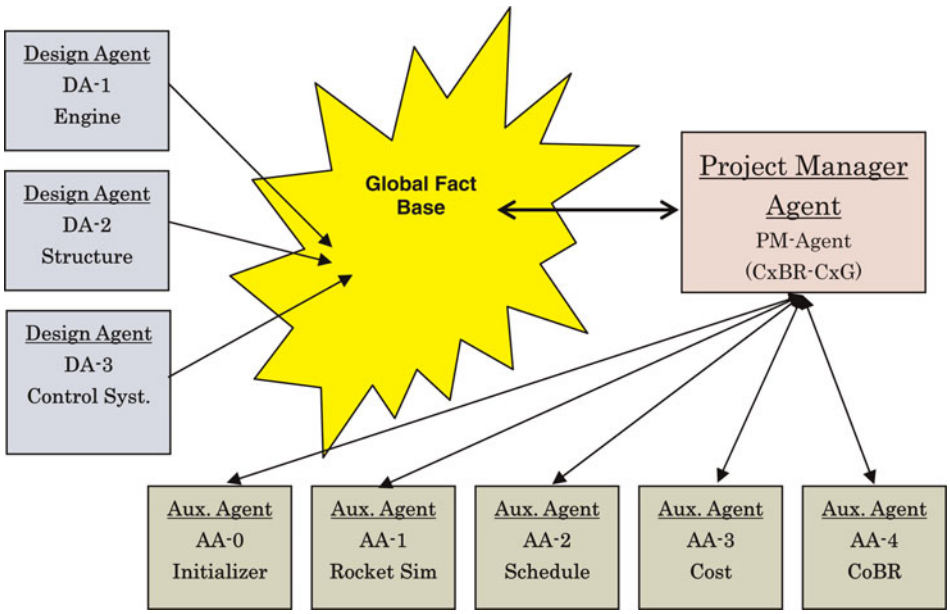

Fig. 5. A conceptual block diagram of participating agents in our prototype. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

4.1.2. Corrective action operators

The PM-Agent has at its disposal two corrective action operators that can take corrective measures when the situation is in conflict. In a robust, real-world application one would expect that there be many more such corrective action operators. Those used here are as follows:

1. *Corrective action operator 1:* Costs can be reduced by trimming staff. This can only be done twice during the lifetime of the project:
   - the first time for a reduction of up to 10% of the total projected cost and
   - the second time for a reduction of up to 5% of the total projected cost.
2. *Corrective action operator 2:* Component delivery time can be reduced by paying overtime to workers to accelerate delivery:
   - The relation is set to 50.0 kYen/day for reduced delivery time.
   - This can only be done for a maximum total reduction of 30 days in the life of the project being simulated.

When faced with a conflict to resolve, agent AA-4 first attempts to find a solution using either the above corrective action operators or through a design change. AA-4 may use more than one corrective action operator to achieve acceptable results. Should all attempts to solve the problem fail, then an impasse is declared by PM-Agent and the *Impasse* Major Context becomes activated (described later). This indicates that the situation is overconstrained, and turns the process over to a higher authority to authorize an extension to the project deadline, increase the maximum allowable cost (the budget), mandate a design change on the DAs that results in lower cost and/or faster delivery, or declare the project to have ended in failure. These all require human involvement.

4.1.3. Other assumptions

Here we describe some assumptions that we make to simplify the coding of these agents. We refer to these as our "rules of engagement."

- We assume no transportation time or cost. Delivery is to be at launch site.
- External events only relate to schedule and/or cost of rocket. They do not affect the design.
- Design changes can potentially affect both cost and delivery.
- A design change must be accepted unilaterally by the PM-Agent if it "works." That is, the rocket has been deemed flight worthy by AA-1 as per the specifications in the Mission Context.
- The PM-Agent must unilaterally reject designs that do NOT "work." That is, the rocket has been deemed NOT to be flight worthy with that particular design by AA-1.
- The PM-agent cannot reject a new design that works purely on the basis of negative schedule and/or cost implications. It must negotiate with the appropriate DA to achieve a design change.
- The rocket design can be changed to compensate for an unacceptably long or overly expensive project that results from external events unrelated to the design. This has to be accepted by the DA, unless mandated by a higher authority during an Impasse.

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

- The human PM being supported by our tool is not in the loop (i.e., does not interface directly with the PM-Agent tool). He/she merely monitors its actions and decides whether they are appropriate or not. The PM-Agent tool could conceivably be used in lieu of a human PM. However, we do not believe that the project management discipline is ready to accept that level of automation, and even if it were, considerable more research would be required to validate and verify its functionality before turning it loose on a real-world project.

4.2. The PM-Agent

The PM-Agent is always in control of the project (except during an impasse). The first thing that the PM-Agent needs to know is what are the technical specifications of the rocket, what is the maximum cost of the rocket, and what is the time frame for completing the work. The PM-Agent can find this information in the Mission Context. Although some of the information therein may change, the mission itself rarely, if ever, changes during the simulation. That is, the mission does not change, say, from *BuildSoundingRocket* to *BuildSchoolHouse* in the middle of a project.

4.2.1. The mission context for the PM-Agent

Our mission objective is the completion of a rocket that performs its required task in time and within budget. We have defined three different missions for our prototype. These are shown in Table 1. Mission 1 features a low budget, mission 2 features a short deadline, and mission 3 is technically challenging in that it needs to boost a heavy payload above 1000 m.

Of course, an important component of a Mission Context is the plan of action in terms of Major Contexts. For tactical operations where an agent must follow a plan, yet always be in a potentially reactive mode, this is indeed quite important. In such cases, the plan is a sequence of Major Contexts with carefully designed transition criteria. However, this is not applicable to our domain, as one can always hope that things go according to plan and that no extra effort is required on the part of the PM-Agent other than just regularly confirming that all is well with the project. Thus, the PM-Agent would always be in *Normal* Major Context and would be always be in a reactive mode. Nevertheless, it would not be a difficult future extension of our work to add a plan to the process of project management, detailing deliverables by the DAs and the Major Context in which the PM-Agent must be in order to act upon these deliverables.

Table 1. *Description of sample Mission Contexts*

Attribute	Mission 1	Mission 2	Mission 3
Maximum budget	10 MYen	18 MYen	20 MYen
Deadline	≤12 months	≤6 months	≤12 months
Altitude to be reached	≥100 m	≥400 m	≥1000 m
Payload weight	490 g	490 g	1000 g
Rocket weight	<2000 g	<3000 g	<3000 g

4.2.2. Major contexts

Each Major Context reflects a situation that the PM-Agent faces and must resolve. The Major Contexts contain functionality to do what the PM-Agent can do to return the project to normal status. The Major Contexts used in this prototype are

- Normal
- DesignChange
- ExternalEvent
- Impasse

These contexts introduce expectations as well as procedures on how to act in these situations. We describe each Major Context in the following section.

Normal Major Context. If all is normal, then the context is Normal. We define normalcy to be that the specifications have been met by the current design, the rocket is projected to be delivered on time and within budget, and no evidence to the contrary exists in the GFB. In other words, Normal can be defined to be the absence of any problems. In this Major Context, the PM-Agent continues to monitor the situation every computation cycle, or if that is too frequent, then every *n* computation cycles. Sampling the situation once per day is thought to be sufficient for this prototype. The Normal Major Context is also the default context as well as the initial context. The simulation always begins with Normal being active. Furthermore, when nothing else emerges as a problem, the PM-Agent returns to this context. When Normal acts as the initial context, the PM-Agent calls agent AA-0 as the active Sub-Context to initialize the schedule data structure and calculate the projected final cost of the project. A design change or an external event that reports a change in delivery time or increase in cost can cause the project to become not normal. Therefore, upon detecting either a design change or a new external event in the GFB, the PM-Agent will transition, respectively, to DesignChange or ExternalEvent to examine and determine the effect of the new situation.

DesignChange Major Context. When a design decision is posted into the GFB, the PM-Agent transitions into this Major Context, where it deals with a design change and its effect on the project. Within this Major Context, we define a CxG Sub-Context called CxG-1 that steps through a process to determine whether or not there are any ill effects from this design change and resolve the problem if there are. CxG-1 determines the effect of the design change and its effect on the cost and schedule. If it discovers any problems, it will seek to resolve them through AA-4.

ExternalEvent Major Context. This Major Context is activated by the PM-Agent when there is a new external event posted in the GFB. By definition, external events only affect the schedule and cost of the project, and not the design. When activated, ExternalEvent Major Context executes CxG-2 as the active Sub-Context. This contextual graph first calls AA-2 to determine whether there is any negative effect on the

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

schedule. If not, then it calls AA-3 to see whether there is any negative effect on the cost. If neither of these returns an indication of negative effects, then the Normal Major Context is reactivated. In contrast, if either one returns an indication of negative impact, then AA-4, the problem-solver agent, is called to arrive at an acceptable solution. If system is not overconstrained and an acceptable solution can be found by exercising the corrective action operators, then this is done and the Normal Major Context is reactivated with these new values. However, if the situation is found to be overconstrained, then the Impasse Major Context is activated.

Impasse Major Context. This Major Context is activated by the PM-Agent when the situation is overconstrained. In this Major Context, an extension of the mission time line or increase in the authorized budget is required to proceed. Otherwise, the project must be cancelled. The Impasse Major Context can also become activated if a design conflict is irresolvable within the PM-Agent's range of authority. Little action is executed in Impasse, other than a call to upper management for their intervention. Sub-Context CxG-3 reflects this logic.

In summary, the knowledge of an experienced PM can be represented in a context-based architecture that has an overall, coarse-grained CxBR hierarchy to organize the high-level knowledge, and CxGs to organize the finer, procedural knowledge involved. This is seen in Figure 6. Figure 7 provides the transition relationship among the Major Contexts. The directional edges indicate the allowable transitions between a pair of individual Major Contexts.

Figure 8 shows a high level description of the External-Event Major Context. Figure 9 depicts the high level description of Minor Context CxG-2. Figure 10 demonstrates the CxG for the CxG-2 Sub-Context.

5. PROTOTYPE TESTING AND EVALUATION

This section describes a test plan designed to evaluate the performance of our approach to modeling the project management process using contexts. The first section describes the tests as designed and their justification. Section 5.2 includes the results for each phase of the test plan, and Section 5.3 discusses conclusions about the success of the tests as well as their implications.

5.1. Test plan

In effect, we wish to test the effectiveness of our context-based implementation. We are unable to compare it to the published results of others because of the sketchy description of test results found in the most relevant publications as well as the different domain used here. Our tests, however, inject several realistic situations in a project that would have to be resolved by a human PM. Our focus is to see whether, given the knowledge and functionality found in the contexts provided to the PM-Agent, the latter can address the situation properly. Although this does not represent a real-world case study application to an actual project, it does subject the prototype to realistic, albeit not complex, situations normally faced by a PM. A full description of the tests can be found in Gonzalez (2007), where the expected solution of each test is described in greater detail.

5.1.1. Test set 1

Test set 1 represents relatively simple tests that seek to verify the basic functionality of the prototype. They are based on mission 1 and a relatively simple design of a one-stage rocket.

- Test 1-1 sets a basic benchmark by confirming the ability of the prototype to retrieve the design information

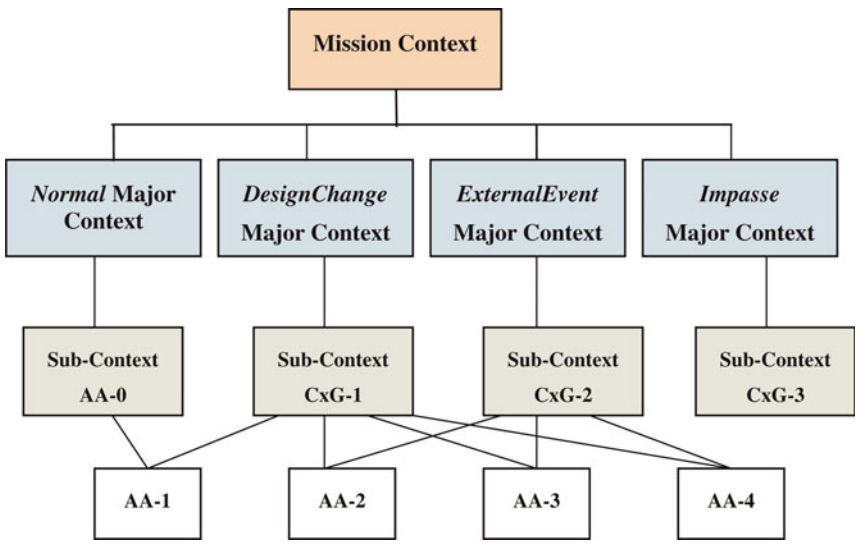

Fig. 6. The specific knowledge hierarchy for PM-Agent for vertical movement. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

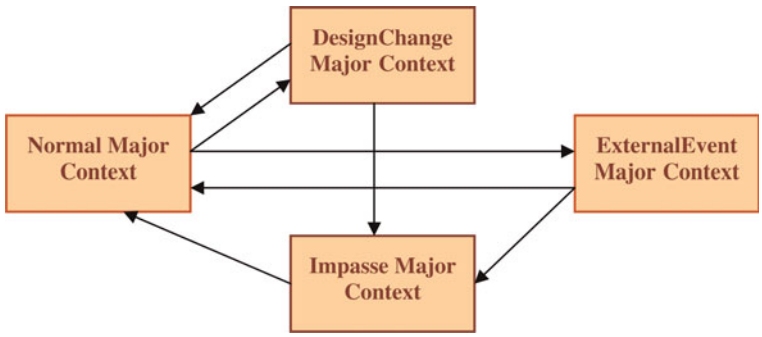

Fig. 7. A Major Context transition map of horizontal movement. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

and determine the impacts on the schedule and cost. There are no external events.

- Test 1-2 uses the same initial conditions as test 1-1, but it introduces an external event that causes the cost to rise, within the allowed budgetary limitation.
- Test 1-3 is the same as test 1-1, except that the external event makes the new cost exceed the budget. It evaluates the ability of AA-4 to make use of the corrective action operators to reduce the cost.
- Test 1-4 is designed to evaluate the prototype's reaction to schedule changes. The introduced changes do not cause the schedule to exceed the project's final deadline.
- Test 1-5 is designed to evaluate the prototype's reaction to schedule changes that cause it to exceed the final deadline.
- Test 1-6 is designed to evaluate the prototype's reaction to schedule changes and cost increases in the same test. However, the situation is not overconstrained.
- Test 1-7 is designed to evaluate the prototype's reaction to more complex schedule changes and cost increases in the same test. The situation is not overconstrained, but it does require multiple applications of corrective operators.
- Test 1-8 is designed to evaluate the prototype's reaction to schedule changes and cost increases in the same test. However, the situation is now overconstrained, leading to an impasse.

5.1.2. Test set 2

Test set 2 challenges the functionality of the prototype further than test set 1 by including design changes along the way that cause overconstrained situations, possibly leading to impasses. This test suite uses mission 2.

- Test 2-1 is designed to verify that the design change evaluation mechanism works as designed. A design change in the form of an engine change comes in on day 100, but this design change introduces no conflict.
- Test 2-2 is designed to verify that the design change evaluation mechanism works as designed. An engine design change occurs on day 5, but the new design is inadequate for the performance specification requirements. It should be rejected outright by the PM-Agent.
- Test 2-3 is designed to verify that the design change evaluation and recovery mechanism works as designed. This test introduces a design change that while technically acceptable, causes problems with schedule and cost. However, the situation is not overconstrained and a resolution can be found.
- Test 2-4 is designed to verify that the design change evaluation and recovery mechanism works as designed. This test introduces a design change that although technically acceptable, causes schedule and cost problems that overconstrain the problem, leading to an impasse.

Major Context Name: ExternalEvent

Action Functions

- Retrieves data on new event from GFB
- Activates Sub-Context CxG-2 through appropriate action rule

Transition Rules

- If schedule is not negatively impacted and cost limit is not exceeded, then transition to *Normal*
- If system overconstrained and no solution can be found, then activate the *Impasse* Major Context
- If system is underconstrained, then the acceptable values are assigned for cost and schedule and activate the *Normal* Major Context

Action Rules

- Activates Sub-Context CxG-2 to determine effect on schedule

Sub-Contexts

- CxG-2

Fig. 8. Description of Major Context ExternalEvent.

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

Minor Context Name: CxG-2

Action Functions

- Determines the next Major Context transition by activating the CxG

Action Rules

- Activates AA-2 to attempt to determine schedule impact
- Activates AA-3 to attempt to determine cost impact
- Activates AA-4 if necessary

Sub-Contexts

- AA-2
- AA-3
- AA-4

Fig. 9. The CxG-2 Minor Context.

- Test 2-5 introduces a design change to the body dimensions that makes the rocket unable to lift the payload to the required altitude.

5.1.3. Test set 3

Test set 3 involves tests with events that go beyond what the prototype was designed to handle. This provided an insight into the limits of the prototype as well as avenues for further research. This includes inadequate designs as well as late designs. Mission 3 is used for these tests.

- Test 3-1 introduces a different engine design as the initial design offered by the DAs. It includes a heavier payload and three-stage rocket. The initial design is not workable from a cost and schedule standpoint. The heavier payload of mission 3 requires multiple stage rockets, which complicate the decision making but provide several alternatives.
- Test 3-2 pushes the limits of the prototype by introducing late designs, causing uncertainty. The prototype was not been designed to handle lateness of designs. In this test, the engine selection lags behind schedule. The selection (design) is not submitted by the DA until day 2.

5.2. Test results

This section summarizes the results obtained from running the tests described above. The reader interested in the details of the tests and results is referred to Gonzalez (2007). Table 2 summarizes the results as the sequence of transitions and actions among major contexts. This sequence is displayed in Table 2 in a somewhat cryptic, shorthand version of the results. Repeating the entries indicates that more than one external event or design change took place at different times. The expected and actual results indicate the respective major context transition and action sequence. As one can see from Table 2, all tests were successful except the last two (test set 3). This is discussed below.

Test set 3 presented the most difficult of all tests, for which not all situations were anticipated. It should be noted that for test 3.1, there was no context for objectionable initial design. Therefore, the transition improperly went to Normal nonetheless. This is not correct, even though the sequence of subsequent context transitions was as expected, given the limitation. There should be another Major Context that starts out the process by evaluating the initial conditions. The design was changed on day 2 to reflect a less expensive, three-stage design that did meet the cost criteria. This design change caused a DesignChange context to be activated, but it did not require corrective action operators to solve.

For test 3.2, where there is no design at the outset, the system should not have known what to do, as there was no context for incomplete design. Therefore, it improperly went to normal. All subsequent actions were done correctly.

5.3. Conclusions of testing

The testing proved to be successful for test sets 1 and 2. In these experiments, the PM-Agent context transitions and

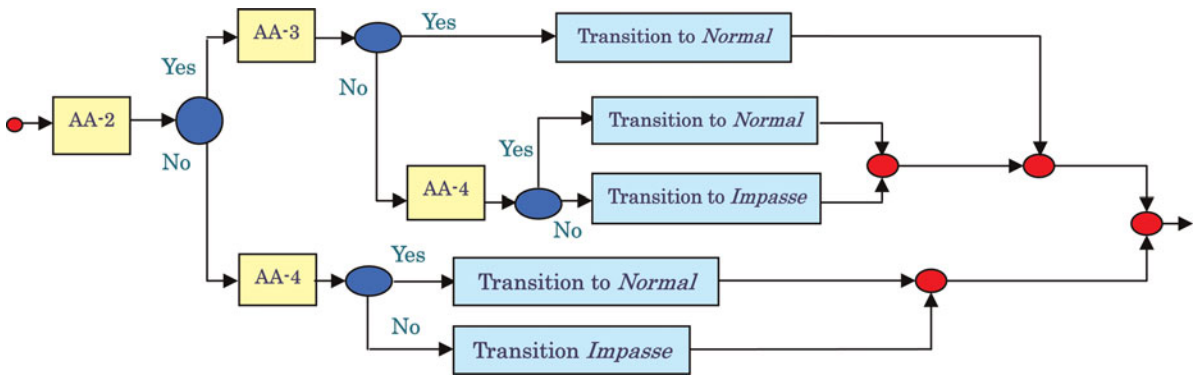

Fig. 10. A contextual graph for Sub-Context CxG-2. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

Table 2. *Summary of results for the tests*

Test No.	Results		Success?	Explanation or Comment
	Expected	Actual		
1.1	Normal	Normal	Yes	Successful
1.2	N-EE-N	N-EE-N	Yes	Successful
1.3	N-EE-ca1-N	N-EE-ca1-N	Yes	Successful
1.4	N-EE-N	N-EE-N	Yes	Successful
1.5	N-EE-ca2-N	N-EE-ca2-N	Yes	Successful
1.6	N-EE-ca2-N-EE-ca1-N	N-EE-ca2-N-EE-ca1-N	Yes	Successful
1.7	N-EE-ca2-N-EE-ca1-N-EE-ca2-ca1-N	N-EE-ca2-N-EE-ca1-N-EE-ca2-ca1-N	Yes	Successful
1.8	N-EE-ca2-N-EE-ca1-N-EE-ca2-ca1-I	N-EE-ca2-N-EE-ca1-N-EE-ca2-ca1-I	Yes	Successful
2.1	N-DC-a-N	N-DC-N	Yes	Successful
2.2	N-DC-r-DC-a-N	N-DC-r-DC-a-N	Yes	Successful
2.3	N-DC-a-ca1-ca1-N	N-DC-a-ca1–ca1-N	Yes	Successful
2.4	N-DC-a-ca1-ca1-I	N-DC-a-ca1-ca1-I	Yes	Successful
2.5	N-DC-a-ca1-ca1-N	N-DC-a-ca1-ca1-N	Yes	Successful
3.1	r-DC-a-N	N-DC-a-N	Partial yes	Initial context wrong
3.2	?-DC-a-ca1-N	N-DC-a-ca1-N	Partial yes	Awaiting late design not *normal*

Note: N, the Normal major context; EE, ExternalEvent; MC, Major Context; DC, DesignChange; I, Impasse; ca, corrective action operators were used to resolve a constraint conflict; ca1, cost operator; ca2, schedule operator (absence of ca entry indicates that no conflict arose); r, rejection of a design change by the project manager (PM); a, the design was accepted by the PM.

use of operators were as expected. In addition, the results achieved were the correct results. For test set 3, however, the tests were only partially correct.

We assert that additional contextualization within the bounds of this tool would address the problems uncovered with test set 3. New major contexts need to be added to evaluate the initial designs and/or wait for the design from the DA before declaring that all was normal. However, the question remains as to whether all situations need be defined *a priori* for the PM-Agent to work. Would some learning mechanism be capable of identifying new situations and constructing new Major Contexts? Very possibly, but we leave this for future research.

6. SUMMARY, CONCLUSIONS, AND FUTURE RESEARCH

The research succeeded in creating a PM agent that was situationally aware. It was able to see the external events and design changes and apply whatever corrective action operators existed to bring the project design, cost, and schedule within original objectives, if possible. If the problem was overconstrained, then it declared an impasse and called for external intervention on the part of the customer or upper management.

Knowledge engineering is always an important consideration when building intelligent systems. This application did not require significant knowledge acquisition because of its relative simplicity. The first author's (admittedly limited) experience in project management from an earlier position in his career proved sufficient for this application. For more complex and realistic applications, we are considering using a tool that can serve to collect the requisite knowledge through an automated interview of an expert. We have developed and evaluated a tool called CITKA (Context-Based Interactive Tactical Knowledge Acquisition), which is capable of significantly facilitating knowledge acquisition for CxBR applications (see Gonzalez et al., 2006). This tool permits the definition by an expert of several attributes (parameters) of the contexts to be used in the application. We leave this for future research.

As with all research, there are some simplifying assumptions and acknowledged limitations with our work. We discuss this in the next few paragraphs to give the reader a complete picture.

In our application to project management, we assume that the contexts are well defined as well as predefined. If the PM-Agent is faced with a new kind of situation not previously incorporated as a context, it will not know how to handle it. This application is not designed to be able to learn on the job, so to speak. There are technologies that can be brought to bear to accomplish learning with some degree of probable success. In fact, CxGs have been used in applications where they can be adjusted through an explicit dialog with a human expert. However, this is beyond the scope of this work and we leave it for possible future research.

The GFB makes no effort to ensure the validity or consistency of the facts posted therein. This can lead to incorrect and/or conflicting information upon which the PM-Agent will act, regardless of its veracity. This is akin to using sensor readings in a diagnostic application without verifying their accuracy. Sometimes their accuracy can be confirmed, but in other cases it cannot be. We purposely assumed correct and consistent input data as this is not deemed critical to our primary objective of evaluating the feasibility of a context-based approach to project management. We leave the verification of posted facts for future research.

We do not foresee that the agents developed for one application would be directly usable in another completely different application (e.g., software development vs. building con-

https://doi.org/10.1017/S0890060410000156 Published online by Cambridge University Press

struction). However, applications in the same domain (i.e., building construction) by the same organization for a new project may present significant opportunities for reuse, as the agents reflect the basic tenets of project management in a domain, as well as the specific procedures and policies of the organization. Nevertheless, we do expect some modification to be necessary, even in such cases. Furthermore, we do expect that the CxBR/CxG architecture, if not the specifically developed PM-Agent per se, will remain applicable across a wide range of project management domains.

The application used here was not as rigorous as we had hoped. The small size and weight of the rocket handled by the NASA simulator made it necessary to fictionalize (although within realistic bounds) several parameters of the project, such as cost and delivery. Furthermore, the system overburdened the AA-4 agent, asking it to solve problems that were occasionally beyond what it was designed to do. AA-4 would have to be unreasonably complex and computationally intensive when developed for a full-fledged real-world application. It would have been preferable to define new additional contexts that contextualized the problem further and allow AA-4 to be less complex. Nevertheless, a reconsideration of AA-4 will be necessary in future versions of this approach that will be examined in future research.

ACKNOWLEDGMENTS

The research was partially funded by the Japanese Society for the Promotion of Science, Tokyo Denki University (Chiba campus), and the University of Central Florida.

REFERENCES

Abernethy, K., Piegari, G., & Reichgelt, H. (2007). Teaching project management: an experiential approach. *Journal of Computing Sciences in Colleges 22(3)*, 198–205.

Bannerman, P.L. (2007). Software project risk in the public sector. *Proc. 2007 Australian Software Engineering Conf.*

Bardram, J.E. (1996). Organisational prototyping: adopting CSCW applications in organisations. *Scandinavian Journal of Information Systems 8(1)*, 69–88.

Barrett, G.C., & Gonzalez, A.J. (2002). Modeling collaborative behaviors in context-based reasoning. *Proc. 2002 Swedish American Workshop on Modeling and Simulation*, Orlando, FL, October 29–30, 2002

Borges, M.R.S., Brézillon, P., Pino, J.A., & Pomerol, J.-Ch. (2004). Bringing context to CSCW. *Proc. 8th Int. Conf. Computer Supported Cooperative Work in Design*, pp. 161–166.

Brewer, J.L. (2005). Project managers, can we make them or just make them better? *Proc. SIGITE'05*, pp. 167–173.

Brézillon, P. (2002). Modeling and using context: past, present and future. Accessed at http://www.lip6.fr/reports/lip6.2002.010.pdf

Brézillon, P. (2005). Task-realization models in contextual graphs. *Proc. Context 2005 Conf.*, pp. 55–68.

Brown, J. (1994). *Application and evaluation of the context-based reasoning paradigm.* Master's Thesis. University of Central Florida, Department of Electrical and Computer Engineering.

Clancey, W.J. (1997). *Situated Cognition—On Human Knowledge and Computer Representations*. New York: Cambridge University Press.

Dominguez, C. (1994). Can SA be defined? In *Situation Awareness: Papers and Annotated Bibliography* (Vidulich, M., Dominguez, C., Vogel, E. & McMillan, G., Eds.), pp. 5–15, Report AL/CF-TR-1994-0085. Columbus, OH: Wright–Patterson Air Force Base.

Endsley, M. (1988). Design and evaluation of situation awareness enhancement. *Proc. Human Factors Society 32nd Annual Meeting*, pp. 97–101.

Endsley, M.R. (2000). Theoretical underpinnings of situational awareness: a critical review. In *Situation Awareness and Measurements* (Endsley, M.R., & Garland, D.J., Eds.). New York: Erlbaum.

Engelmore, R.S., & Morgan, A. (Eds.). (1988). *Blackboard Systems*. Reading, MA: Addison–Wesley.

Fan, X., Sun, S., & Yen, J. (2005). On shared situation awareness for supporting human decision-making teams. *Proc. 2005 AAAI Spring Symp. AI Technologies and Homeland Security.*

Feng, Y., Teng, T., & Tan, A. (2009). Modeling situational awareness for context-aware decision support. *Expert Systems With Applications 36*, 455–463.

Fernlund, H. (2004). *Evolving models from observed human performance*. Doctoral Dissertation. University of Central Florida.

Fox, T.L., & Spence, J.W. (2005). The effect of decision style on the use of a project management tool: an empirical laboratory study. *DATA BASE for Advances in Information Systems 36(2)*, 28–42.

Gaba, D.M., Howard, S.K., & Small, S.D. (1995). Situation awareness in anesthesiology. *Human Factors 37(1)*, 20–32.

Gero, J.S. (2004). Constructive memory for situated design agents. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18(2)*, 163–198.

Gonzalez, A.J. (2007). *Using Contexts to Control a Collaborative Process.* Final Report. Tokyo Denki University. Accessed at http://isl.ucf.edu

Gonzalez, A.J., & Ahlers, R. (1998). Context-based representation of intelligent behavior in training simulations. *Transactions of the Society for Computer Simulation 15(4)*, 153–166.

Gonzalez, A.J., & Brézillon, P. (2008). Integrating two context-based formalisms for improved representation of human tactical behavior. *Knowledge Engineering Review 23(3)*, 295–315.

Gonzalez, A.J., Castro, J., & Gerber, W.E. (2006). Automating the acquisition of tactical knowledge for military missions. *Journal of Defense Modeling and Simulation 3(1)*, 145–160.

Gonzalez, A.J., Stensrud, B.S., & Barrett, G. (2008). Formalizing context-based reasoning—a modeling paradigm for representing tactical human behavior. *International Journal of Intelligent Systems 23(7)*, 822–847.

Gonzalez, F.G., Grejs, P., & Gonzalez, A. J. (2000). Autonomous automobile behavior through context-based reasoning. *Proc. Int. FLAIRS Conf.*

Gumus, I. (1999). *A threat prioritization algorithm for multiple intelligent entities in a simulated environment*. Master's Thesis. University of Central Florida.

Gutwin, C., & Greenberg, S. (2004). The importance of awareness for team cognition in distributed collaboration. In *Team Cognition: Understanding the Factors That Drive Process and Performance* (Salas, E., & Fiore, S.M., Eds.), pp. 177–201. Washington DC: APA Press.

Harwood, K., Barnett, B., & Wickens, C. (1988). Situational awareness; a conceptual and methodological framework. *Proc. 11th Symp. Psychology in the DoD* (McIntire, F.E., Ed.). Colorado Springs, CO: US Air Force Academy.

Henninger, A.E., & Gonzalez, A.J. (1997). Automated acquisition tool for tactical knowledge. *Proc. 10th Annual Florida Artificial Intelligence Research Symp*., pp. 307–311.

Kokinov, B., Petkov, G. & Petrova, N. (2007). Context-sensitivity of human memory: episode connectivity and its influence on memory reconstruction. *Proc. Context 2007 Conf.*, pp. 317–329.

Lee, C.-S., Wang, M.-H., Chen, J.-J., & Hsu, C.-Y. (2006). Ontology-based intelligent decision support agent for CMMI project monitoring and control. *Proc. North American Fuzzy Information Processing Society (NAFIPS)*, pp. 627–632.

Lee, S., & Peña-Mora, F. (2005). System dynamics approach for error and change management in concurrent design and construction. *Proc. 2005 Winter Simulation Conf.*

Levitt, R.E., & Nissen, M.E. (2003). The Virtual Design Team (VDT): a multi-agent analysis framework for designing project organizations. *Proc. 2003 KIMAS Conf.*, pp. 115–120.

Medina-Mora, R., Winograd, T., Flores, R., & Flores, F. (1992). The action workflow approach to workflow management technology. *Proc. CSCW 1992.*

National Aeronautics and Space Administration. (2003). *RocketModeler, version 1.2.* Accessed at http://www.nasa.gov/

Nehme, C.E., Crandall, J., & Cummings, M.L. (2008). Using discrete event simulation to model situational awareness of unmanned vehicle operators. *VMASC Capstone Conf.*

Nienaber, R., & Cloete, E. (2003). A software agent framework for the support of software project management. *Proc. SAICSIT 2003*, pp. 16–23.

Norlander, L. (1998). *A framework for efficient implementation of context-based reasoning in intelligent simulations*. Master's Thesis. University of Central Florida.

Proenza, R. (1997). *A framework for multiple agents and memory recall within the Context-Based Reasoning Paradigm*. Master's Thesis. University of Central Florida.
Sapateiro, C., & Antunes, P. (2009). An emergency response model towards situational awareness improvement. *Proc. 6th Int. ISCRAM Conf.*
Smith, J.L., Bohner, S.A., & McCrickard, D.S. (2005). Project management for the 21st century: supporting collaborative design through risk analysis. *Proc. 43rd ACM Southeast Conf.*
Stiffler, D. (1988). Graduate level situation awareness. *USAF Fighter Weapons Review.*
Stensrud, B.S. (2005). *FAMTILE: an algorithm for learning high-level tactical behavior from observation.* Doctoral Dissertation. University of Central Florida.
Turner, R.M. (1998). Context-mediated behavior for intelligent agents. *International Journal of Human–Computer Studies 48(3)*, 307–330.
Wu, S., & Kotak, D. (2003). Agent-based collaborative project management system for distributed manufacturing. *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, Vol. 2, pp. 1223–1228.

Avelino J. Gonzalez is a Professor of computer engineering in the School of Electrical Engineering and Computer Science at the University of Central Florida (UCF). He received his PhD in electrical engineering from the University of Pittsburgh in 1979. His area of expertise is artificial intelligence and contextual reasoning. Prior to joining the university in 1986, he was a Senior Engineer at the Westinghouse Electric Corporation, where he helped develop the GenAID intelligent diagnostic system.

Setsuo Tsuruta is a Professor in the School of Information Environment at the Chiba campus of Tokyo Denki University (TDU). He joined TDU after a long and illustrious career with Hitachi Ltd. His last assignment with Hitachi was with the Systems Development Laboratory in Yokohama. Dr. Tsuruta's research interests are in distributed artificial intelligence.

Yoshitaka Sakurai is an Assistant Professor in the School of Information Environment at TDU, Chiba campus. Dr. Sakurai's research interests are in machine learning.

Johann V. Nguyen is a doctoral student in computer engineering at UCF, where he expects to graduate in 2010. He has Bachelor's and Master's degrees in computer engineering from UCF. His area of research is composable agents for tactical simulations.

Kouhei Takada is a doctoral student in the School of Information Environment, TDU, Chiba campus.

Ken Uchida is a graduate of the School of Information Environment at TDU with a Bachelor's degree.