Available online at www.sciencedirect.com
ScienceDirect
Int. J. Human-Computer Studies 67 (2009) 90–110
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

Designs for explaining intelligent agents

Steven R. Haynes[a,*], Mark A. Cohen[b], Frank E. Ritter[a]

[a]College of Information Sciences & Technology, Penn State University, 301J IST Building, University Park, PA 16802, USA
[b]Department of Business Administration, Computer Science, & Information Technology, Lock Haven University, Lock Haven, PA 17745, USA

Received 10 June 2006; received in revised form 6 August 2008; accepted 16 September 2008
Communicated by M. Atwood
Available online 27 September 2008

Abstract

Explanation is an important capability for usable intelligent systems, including intelligent agents and cognitive models embedded within simulations and other decision support systems. Explanation facilities help users understand how and why an intelligent system possesses a given structure and set of behaviors. Prior research has resulted in a number of approaches to provide explanation capabilities and identified some significant challenges. We describe designs that can be reused to create intelligent agents capable of explaining themselves. The designs include ways to provide *ontological*, *mechanistic*, and *operational* explanations. These designs inscribe lessons learned from prior research and provide guidance for incorporating explanation facilities into intelligent systems. The designs are derived from both prior research on explanation tool design and from the empirical study reported here on the questions users ask when working with an intelligent system. We demonstrate the use of these designs through examples implemented using the Herbal high-level cognitive modeling language. These designs can help build better agents—they support creating more usable and more affordable intelligent agents by encapsulating prior knowledge about how to generate explanations in concise representations that can be instantiated or adapted by agent developers.

Keywords: Explanation; Intelligent agents; Design rationale; Design guidelines

1. Introduction

In this article, we analyze and synthesize the findings from prior research, including our own ongoing studies, into a design for including explanation facilities in intelligent agents. We do this by describing and demonstrating a design for explanation facilities.

Since the Mycin experiments (Buchanan and Shortliffe, 1984), providing explanations of how and why an intelligent system works has been an enduring concern for developers, users, and researchers. The goal of the Mycin project was to produce an expert system to support physicians diagnosing blood disease. Very early in the project, researchers identified the ability to explain its problem-solving logic as a fundamental requirement of the system (Buchanan and Shortliffe, 1984).

Mycin's first explanation facilities were considered useful primarily as a resource to support developers debugging system reasoning. When researchers investigated how Mycin could be extended to assist with physician training, they realized that explanation facilities were central to exposing the system's reasoning steps, how a request for information translated into a system process, and the relationship between system goals and how these goals or outcomes were achieved (Clancey, 1983). They also realized early that providing explanations contributed to the acceptance of the system by its intended users, in this case, physicians and medical students (Dhaliwal and Benbasat, 1996).

Explanations are important. They have been used to support users in several ways. They can: (a) give justifications for a given design (Chandrasekaran and Swartout, 1991; Carenini and Moore, 1993; Gregor and Benbasat, 1999); (b) increase trust in system behavior and results (Ye and Johnson, 1995; Herlocker et al., 2000; McGuinness and da Silva, 2004); (c) aid comprehension and learning

*Corresponding author. Tel.: +1 814 865 7886; fax: +1 814 865 6426.
E-mail addresses: shaynes@ist.psu.edu (S.R. Haynes), mcohen@lhup.edu (M.A. Cohen), ritter@ist.psu.edu (F.E. Ritter).

1071-5819/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2008.09.008

(Conati and VanLehn, 2001; Lam and Barber, 2005); and (d) increase the likelihood that a system will be accepted into use (Ford et al., 1993; Papamichail and French, 2003).

Much of the research on explanation facilities has taken the form of normative design studies, where new intelligent system tools are developed and demonstrated in use. Research that empirically assesses the explanation requirements of users as they interact with (and try to understand) a running intelligent system is less well represented in the literature. This paper helps address this gap through a study of intelligent agent users and the questions they ask.

Design of the knowledge base for an intelligent system results in a knowledge resource that is potentially valuable beyond its original application context (Reich, 1995). Much of the rationale underlying a design, however, is made invisible through the process of abstraction—a fundamental characteristic of the software development process (Brooks, 1987). For an intelligent system to be comprehensible to its users (and for its design and knowledge base to be reusable by future developers) requires access to the full depth of analysis embedded in its rationale. Explanation facilities are one acknowledged approach to expose and make explicit this inscribed knowledge (Swartout et al., 1991).

This article focuses on one aspect of explanation, that of users interacting with an intelligent system, and derives a design for implementing explanation facilities to answer the kinds of questions that arise in the use context. We describe our approach in five sections. The rest of Section 1 provides a review of prior research on explanations and how they contribute to the usability of intelligent agents, and describes some of the research on developing embedded explanation facilities for users of intelligent systems. Section 2 presents results from an analysis of questions asked by users as they worked with an example agent. Section 3 describes the design derived from our own research, and then Section 4 demonstrates how this design has been implemented in the *Herbal* agent modeling and development environment. Section 5 summarizes and concludes the article, noting how with further implementation and testing our design could be the basis of design patterns for providing explanations in intelligent agents.

1.1. Intelligent agents

Intelligent agents are software programs designed to act autonomously and adaptively to achieve goals defined by their human developers or runtime users (the latter can be other intelligent agents). These systems make use of a knowledge base and algorithms to carry out their responsibilities. A knowledge base typically includes domain and problem-solving knowledge specific to the agent's focal tasks, control knowledge to bridge between this domain and problem-solving knowledge, and software components to realize the system's desired behaviors. The operational concept of an agent may be derived from a theory of collaborative intelligence, for example, CoJACK (Ritter and Norling, 2006) or CAST (Yen et al., 2001), a cognitive architecture such as Soar (Newell, 1990; Jones et al., 1999) or ACT-R (Anderson et al., 2004), or may be based on a more generic, instrumental approach to implementing intelligent systems, as for example, in the Jess expert system framework (Friedman-Hill, 2003).

Comprehension and confidence are important attributes of the relationship between intelligent systems and their users for two reasons. First, because agents are designed to operate semi-autonomously, users need to both understand and trust the problem-solving approach they employ. Second, because these systems are often used in consequential domains, such as military operations and medicine, users may be called upon to justify the actions that are carried out by agents operating on their behalf. One approach to supporting these goals of intelligibility and credibility is to have agents explain the rationale behind their design as represented in their structure and behaviors. Explanation facilities are central to development of agents that both "know what they're doing" (Brachman, 2002), and that can show and tell what they are doing. We next describe what we mean by explanation, and then provide a more in-depth discussion of the role of explanation facilities in intelligent system development and usability.

1.2. Explanation

A key challenge in design of effective explanation facilities is deciding the appropriate form and content for the explanations they convey. While a range of perspectives from a number of fields address this issue, little consensus exists as to what constitutes a correct or complete explanation. In part, a lack of empirical work has undermined the ability of both researchers and practitioners to form a consensus. In this section, we review the most enduring theories of explanation and relate them to the kinds of user questions that arise in the intelligent agent domain before reporting empirical work that can help understand explanation needs.

Formal accounts of explanation from the philosophy of science, such as the deductive-nomological (D-N) model, suggest that explanations should take the form of deductive statements predicated on the application of formal laws or well-established universal truths (Hempel, 1965). At the opposite extreme, the pragmatic theory holds that the correct form of an explanation is contingent on the motivation of the explanation requestor and the context of the request (van Fraassen, 1991). In between are probabilistic (Hempel and Oppenheim, 1988), inductive-statistical (Salmon, 1993), and functional (Cummins, 1975) theories, all providing arguments for what constitutes a generally ideal, or context-appropriate explanation. Functional explanations are particularly important in design as they explain by reference to the purpose or requirement that caused creation of the artifact.

This tension between the formal and the contingent is not just a philosophical rumination; accounts from empirical work in psychology point to the same disagreement between those who have attempted to establish models of explanation dialogue (Lehnert, 1978; Graesser et al., 1992) and those who interpret explanatory dialogue as a negotiation between the explanation requestor and provider (Antaki, 1994). This lack of agreement on the necessary and/or sufficient information content of an explanation requires us, as designers of explanation facilities for intelligent agents, to identify our own theory of what constitutes an effective explanation.

An obvious starting point is psychological and naturalistic conceptions of explanation, which are descriptive models of what takes place in explanatory dialogs between people. Several taxonomies of explanations and explanation-seeking questions have been developed. For example, Graesser et al. (1992) constructed a taxonomy of questions and answers derived from studies of naturalistic human-to-human explanation discourse. This framework is derived from work by Lehnert (1978) to support development of tools for natural language processing. The taxonomy divides question types into those requiring short and those requiring long answers. The taxonomy is shown in Table 1.

The explanation types identified by short-answer questions (1–5) are relatively simple, static, and declarative, but are important to define the *ontological status* of the entity or entities being explained; this status offers an explanation that answers questions about *which* entities exist (and are important in a given domain). Answers to these questions play a role in helping to define the scope of the explanation space, and act as the basis for assessments of the potential explanatory relevance of the information provided. Establishing which factors *might* be important to a given explanation request is a critical antecedent to ensuring that explanations conform to the tenets of the pragmatic theory, which argues for the centrality of relevance-to-context as essential in explanatory dialog. Similarly, long answers 6–13 from Table 1 provide *what* information to help the explanation requester make sense of a concept and understand it relative to their own background knowledge and the context that defines their requirement for an explanation. We refer to these types of answers generally as *ontological explanations*.

Table 1
Explanation seeking questions

Short answer
1. Verification: Is a fact true? Did an event occur?
2. Disjunctive: Is X or Y the case? Is X, Y, or Z the case?
3. Concept completion: Who? What? When? Where? What is the reference of a noun argument slot?
4. Feature specification: What qualitative attributes does entity X have?
5. Quantification: What is the value of a quantitative variable? How many?

Long answer
6. Definition: What does X mean?
7. Example: What is an example label or instance of the category?
8. Comparison: How is X similar to Y? How is X different from Y?
9. Interpretation: What concept or claim can be inferred from a static or active pattern of data?
10. Causal antecedent: What state or event causally led to an event or state?
11. Causal consequence: What are the consequences of an event or state?
12. Goal orientation: What are the motives or goals behind an agent's action?
13. Instrumental/procedural: What instrument or plan allows an agent to accomplish a goal?

Long answers 10 and 11 provide explanations that offer insight into *mechanical* functions of the entity or event, in particular on its causes and its consequences, and on how entities and behaviors are linked together. In the context of intelligent systems, these explanations describe how an agent works, on how events and pre-conditions cause agents to enact certain behaviors (which in turn may create or effect events and pre-conditions for the next behavior in sequence). We refer to these types of answers developed by this work as *mechanistic explanations*.

Long answer 12, goal orientation, defines an entity with respect to its intended *purpose*. These types of answers conform to the theory of functional explanation (as discussed earlier), and are particularly visible in the space of designed artifacts including intelligent agents and other systems. The purpose underlying a system or other artifact is an important part of its *design rationale*. Design rationale and its role in explanation of intelligent agents are discussed in some detail in the next section.

Finally, long answer 13 refers to instrumental or procedural explanations that relate goals to the mechanics designed to realize them. These explanations are critical to the use of complex tools, as they provide the user with a means to bridge their goals and the artifacts that can help achieve these goals. In the systems context, we refer to information designed to help make use of a system as *operational explanations*.

A version of the Graesser/Lehnert framework was used as the basis for a study of the kinds of questions people ask of different human–computer interfaces (Lang et al., 1992). The study used questions asked in two contexts: as students learned to use a computer system and the ARPANET network. Frequencies of different types of explanation-seeking questions ($N = 500$) across the two tasks are relatively similar, with instrumental/procedural questions being by far the most common type of question asked (77%). Other common types include verification (15%), concept completion (16%), goal orientations (3%), and enablement (3%). The causal antecedent and causal consequence categories each were represented in 2% of the questions. The prevalence of relatively simple questions was striking; suggesting that people rarely seek explanations drawing on the deeper *why* knowledge underlying the systems they use.

1.3. Explanation facilities for intelligent agents

As described earlier, work on the Mycin expert system (Buchanan and Shortliffe, 1984) was among the first to identify and elaborate on the need for explanations in intelligent agents. Most explanation facility research since then has pointed back to the original finding that explanations are required to help developers trace and debug the behavior of knowledge-based systems. Investigations into the difficulties experienced when attempting to expose the core logic of these systems highlighted the implicit and embedded nature of the problem-solving strategies employed by intelligent agents (Clancey, 1983). This concern with multiple perspectives or levels of analysis echoes Dennett's (1987) observation on the three different "stances" that people can take toward an object: the *physical* stance concerned with the physical or structural nature of an object; the *design* stance concerned with how the object's components carry out some function; and the *intentional* stance, where the object is considered in terms of the human beliefs, goals, and intentions underlying its existence. To be effective in different contexts, explanations must provide information from each of these perspectives, as appropriate.

Building on these findings from the Mycin experiments, the Explainable Expert System (EES) project (Swartout et al., 1991) resulted in the important insight that much of the knowledge needed to explain an intelligent system was outside the boundaries of the knowledge base proper, residing instead in the knowledge employed by the system's creators when they translate domain requirements into a software-based solution. A key EES objective, therefore, was to provide explanations of an expert system by reference to the design rationale underlying the system's architecture. Design rationale consists of the entire design space explored by a development team including all of the design questions identified, the alternatives considered in response to these questions, and the criteria used to select a solution from these alternatives (MacLean et al., 1989). Design rationale asks developers to make explicit much of the knowledge underlying a system's control and knowledge representation strategy, and thereby relates a system's structure to domain concepts and problem-solving strategies (Clancey, 1983). Results from the EES project suggested that explanatory content could be derived from the design process and that these explanations were useful for developers and for system end users.

Research into making intelligent agents more transparent to both developers and end users has built on this earlier work. Related efforts include formal models of explanatory discourse (Cawsey, 1992; Moore and Paris, 1993), empirical work on the use and effects of providing explanation facilities to intelligent agent users (Berry et al., 1995; Ye and Johnson, 1995; Dhaliwal and Benbasat, 1996), and design studies proposing functional solutions to address explanation facility shortfalls (Chandrasekaran and Swartout, 1991; Wolverton, 1995). Much of this work is dispersed among different disciplines, including artificial intelligence and information systems (Keil and Wilson, 2000). Relatively few efforts have attempted to integrate findings from these related (but disconnected) research streams.

The vision of the *semantic web* (Berners-Lee et al., 2001) describes a distributed computing environment characterized by autonomous, composable web services being used

Table 2
Explanation framework for intelligent agents

Ontological explanation	
What—identity	Requests for basic ontological information such as the existence of an agent or agent component, and its identifier.
What—definition	Requests for defining attributes of an agent or component of the most basic type. Goes beyond simply identifying an agent or component and involves providing it with some meaning in context.
What—relation	Requests for information about the relationships between agents or their components. These are static, structural relations however, not the more dynamic relations that might be described in a how it works explanation as described below. This category also includes information requests related to where, for example, spatial information to help place an agent within a simulation or one of its components spatially within the structure of the agent.
What—event	Requests for information about a special kind of entity within an agent's ontology, events are distinguished from static entities and play an important role as a primitive in causal explanations. This category also includes information requests related to when, for example, requests for temporal information related to an event or an ordered causal chain of events and actions.
Mechanistic explanation	
How does it work?	Requests for information describing the mechanics of an agent's behavior, how different components and events interact to give rise to more complex actions.
Operational explanation	
How do I use it?	Requests for instructional content that describes the steps to be performed to enact some agent behavior including instantiation.
Design rationale (Why)	
Deductive-nomological	Requests for why explanations that refer to some law or law-like relation between entities and/or events.
Functional	Requests for why explanations that refer to the purpose or reasons why an agent or one of its components has been created.
Structural	Requests for why explanations that refer to existing structure as a system of constraints that cause an entity or event to take a particular form.
Pragmatic	Pragmatic explanations are those information requests that are entirely interest relative. They include two major attributes: explicit relevance relations between the information provided and that which is being explained, and information on contrast classes that explain with reference to how an entity or event could be otherwise.

semi-autonomously by intelligent agents at the behest of their human delegators. This vision suggests an expanded role for explanation facilities as people come to rely on agents' ability to both describe and justify the manner in which they carry out tasks, for example, in the business-to-business e-commerce purchasing domain. Some current research is focused on the need to ensure trust and understanding between people and agents working together in the semantic web by exposing the inference mechanisms agents use to select from among available services (McGuinness and da Silva, 2003; McGuinness and da Silva, 2004). Other work is attempting to define an ontology for agents that need to explain *to each other* to effectively coordinate their activities toward some shared goal (Su et al., 2003).

Explanation capabilities for agents and other intelligent systems clearly are related to research and practice on computer system user documentation and help facilities. This research has generally shown that a minimalist approach to documentation and help is the most effective approach to providing users with what they need to know when they need to know it (Carroll, 1990, 1998). However, the relationship between a user and an intelligent system is different from that of the day-to-day productivity tools for which minimalism is typically prescribed. Intelligent systems are designed to carry out tasks much as a human would, and to communicate how and why they carry them out in a particular way, again, much as a human would. This suggests requirements for explanation that go beyond instructions to include rationale and reasons for why these systems behave in certain ways. This level of tool knowledge is required to achieve the trust and credibility that are prerequisites for effective intelligent system use (Lerch et al., 1997).

Our review of cross-disciplinary theories of explanation resulted in construction of a framework or content theory for explanations of intelligent agents and other intelligent systems. Table 2 provides a summary of the different explanation types based on this review. In the next section we describe a study carried out to assess the appropriateness and applicability of our framework. We include a more complete description of the different explanation types and give examples of explanation-seeking questions raised by study participants to illustrate each type.

2. Analyzing explanation requests

To ground and to situate the explanation types developed in our review, we analyzed a set of explanation-seeking questions obtained from users of a commercial cognitive model, an intelligent agent designed to act as a virtual pilot in a tactical air-combat simulation. The analysis helped us to understand the different kinds of explanations and gain a preliminary measure of their relative frequency.

2.1. Method

The results reported here were derived from an analysis of walkthrough transcript data obtained from a usability study (Avraamides and Ritter, 2002) of the Tac-Air-Soar Situation Awareness Panel (SAP) (Taylor et al., 2002). Soar is a widely used and successful agent architecture, and Tac-Air Soar is one of the most complex cognitive models to have been developed using it (Jones et al., 1999). In this study, Tac-Air-Soar intelligent agents flew planes in a synthetic environment and executed a combat mission that took approximately 15 minutes. The SAP user interface, shown in Fig. 1, provides a graphic display of the agent's behavior, including a plan view display, some of the agent's external state variables, and a trace of the agent's goal structures, actions, and major milestones. It provides a useful example of an agent that users want to understand, and as such serves as a test bed for studying users' explanation requirements.

Study participants were chosen to represent the range of users for many intelligent systems, from domain experts and users to developers and experts who could comment on various aspects of the design. They were four former military aviators, a Marine Major (not an aviator), a military instructor pilot, an expert on social and group processes, two cognitive psychologists with interests in HCI (one with some amateur flying experience), a former software developer, one university instructor in AI, and one geographic information systems specialist. Four of the 12 study participants had previously used the SAP. So, there was a range of domain knowledge and a range of user types, and a range of knowledge about usability.

Study participants were provided with a short introduction to the SAP and were then given a scenario describing the executing model:

> The Defensive Counter Air mission involves defending an area against airborne threats. An Airborne Early

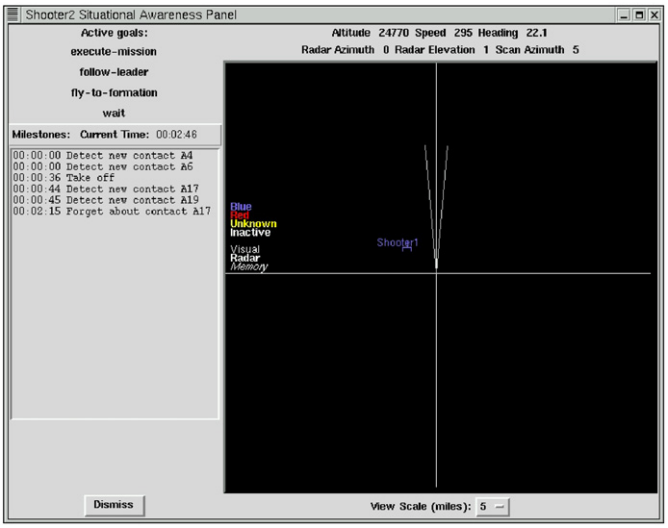

Fig. 1. The Situation Awareness Panel user interface.

Warning (AEW) aircraft is used for its long-range radar to watch for distant threats. When threats arise, the AEW dispatches an airborne 2-ship flying a Combat Air Patrol (CAP) to engage the bogeys.

Participants were requested to perform three sets of tasks with the SAP to ensure that each was exposed to all of the tool's displays and commands. These sessions took 60–90 minutes to complete. The first set of tasks was designed to familiarize participants with the controls. The second set required them to use the displays to observe four agents and answer questions afterwards. Finally, they were provided with time to explore the interface.

Participant interactions with the system were videotaped and transcribed. Results from the initial study were used to identify a set of suggested improvements in the SAP (Avraamides and Ritter, 2002). The secondary analysis reported here was used to identify both the participant's and the study moderator's questions about the agent and its interface. The study moderator's questions were included because the moderator was blind to the purpose of this secondary analysis at the time the data were collected. The questions were coded using the explanation taxonomy derived from theories of explanation discussed in Sections 1.2 and 1.3 and summarized in Table 2. We measured inter-rater reliability using a second coder on a sample of the transcripts (12%) to assess the relative fit of the framework's concepts to the data. Agreement between coders was 81%—greater than 70% agreement is usually considered acceptable for exploratory studies such as this (Lombard et al., 2002). In the sections following, we describe findings for the different explanation categories, and then describe how these results can be combined with results from prior research to create explanations for intelligent agents.

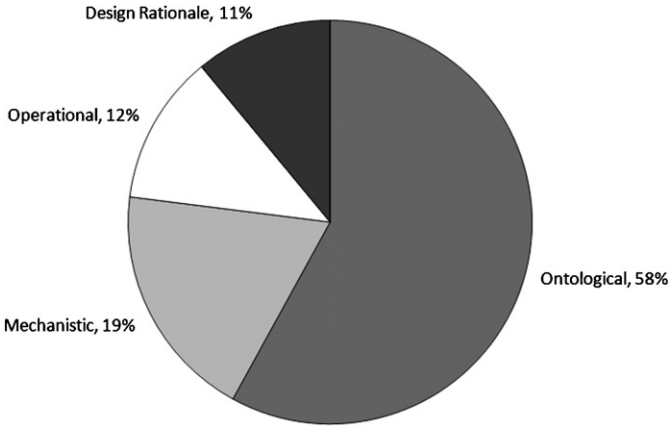

Fig. 2. Explanation seeking questions.

Table 3
Explanation types requested

	Count	Percentage
Ontological		
What—identity	41	17
Definition	79	32
Relation	15	6
Event	7	3
Mechanistic		
How does it work?	46	19
Operational		
How do I use it?	30	12
Design rationale		
Deductive-nomological	0	0
Functional explanation	15	6
Structural explanation	0	0
Pragmatic explanation	13	5
Total	246	100

2.2. Results of analysis

Analysis of the walkthrough transcripts yielded 246 identifiable explanation-seeking questions. Of these, 218, or 89% were classified as requiring *ontological*, *mechanistic*, or *operational* explanations, and 28, or 11%, were classified as *design rationale (why)* questions requiring deeper explanatory content, as shown in Fig. 2.

Table 3 provides a breakdown of the types of questions asked. Proportions of the different categories, the specific codes assigned to different question types, and examples of the questions as coded appear in the sections that follow.

2.2.1. Ontological explanations

Recall that ontological explanations are those that appeal for information related to *what* explanation-seeking questions might be asked. We classify what explanations as falling into one of four categories including identity, definition, relation, and event questions, as detailed below.

2.2.1.1. What—identity questions. The relatively common frequency (17%) of *what—identity* questions points to the importance of this category of simple explanation seeking questions. This type of question generally seeks to identify the agent, one or more of its components, and sometimes other agents operating within the simulation. For example:

Okay, who am I on here, and who are the agents?

…and what are these here?

These questions attempt to understand what the user is seeing in a simulation view. Answers to *what—identity* questions map the ontology of a given agent and its environment, providing information about different concepts, their attributes, and their relations.

In many cases, users attempt to attach identifiers to the entities they see in the simulation. Devices such as rule traces, state space displays (working memory and current

attributes values), and other instrumentation often list the existence and sometimes the behavior of agents in the simulation but it is not always clear to users how these map to graphical displays provided in the simulation. For example:

> They are of the agent that—it says up here. So one of the agents is called Sniper?

> Notice destroy contact A43, well what's A43?

The second of these questions highlights a potential development problem, the use of unhelpful abbreviations for entity identifiers in complex simulations. Somewhat more problematic are identity questions that seek to relate elements from the agent simulation to real-domain concepts. For example, answering the question:

> are we sure that we have these types of aircraft in our inventory?

requires more than just identification locally, but also requires relating the existence of the entity in the simulation to actual entity types in the domain, in this case, a combat air patrol simulation.

2.2.1.2. What—definition. Definitions were the most prevalent operational explanation-seeking questions at 32%. These questions are requests for information about a known entity or event, and appear generally to appeal for information to help characterize what is being queried, to help establish the meaning of a term or concept. They differ from identity questions by making use of more in-depth information than is provided by simply enumerating lists of simulation or agent component identifiers. For example:

> And inactive—what does that mean?

> so these are different units on the ground?

> What about these labels here?

> I don't have any idea what I'm looking at. I mean I can-as I look at this obviously we're looking at an aircraft-altitude, speed, and a heading of negative 27?

Because they are so common and so fundamental to more advanced understanding of an agent, definitions are among the most important information that can be provided to explanation-seeking users. Definitions are also problematic because the notion of a definition provided and externally available for every component of an agent places demands on agent software engineers that may not be currently realistic within the constraints of real-world development projects.

2.2.1.3. What—relation. Explanation requests seeking to understand the relation between elements in the agent's world were less common than other operational types at just 6%. These are requests for static, structural information rather than the mechanistic, causal information provided in the "how does it work" category described later on. For example:

> This is Shooter 1. Okay, and this is what Shooter 1 sees?

> how do these relate, actually?

Answers to what—relation questions play an important role in providing more advanced structural information than is provided by a simple listing of the elements in the model to be explained. They are also problematic. Depending on the architecture, users may have difficulty understanding how different elements relate to each other. Also, this problem often emerges only at model runtime, making this difficult to predict. Further, providing inter-agent relationship information is problematic because, especially in complex systems, these relationships are intended to emerge from the interactions between agents and their operating context.

Questions seeking information about *where* (spatial information) were included in the *what*—relation category, but were not very common. These questions seek information about the placement of entities (or events) relative to other entities in the simulation environment. For example:

> where is this racetrack pattern?

> Where is it [the agent] in this particular part?

The importance of *where* information may be increase in certain applications that are expressly designed for domains where spatial information is central. These may also be critical as agents and the environments in which they work become increasingly distributed and the ability to reflect on and reason about a system's topology becomes central to an agent's self-awareness.

2.2.1.4. What—event. Event type explanation requests are also not common, at 3%. Events in intelligent agents often signal state transitions, and are thus important to understanding process sequences. Most agent development environments are inherently event driven. Explicating the causal chain, why some particular set of agent behaviors occurred, involves understanding the event that triggered the behavioral sequence as well as the intervening events that caused salient changes to the current state. Exceptions play this role in a programming language such as Java, as do impasses in the Soar cognitive architecture. For example:

> Oops, what did I do?

> Now did I activate those?

In the two examples above, the user is seeking information about the consequences of some event in the simulation. Answers to such questions provide the raw material for understanding more complex questions, such

as "how do I use it?" and "how does it work?" Much of the burden for making this information available to explanation facilities is placed on the agent developer, who specifies which events are likely to support end-user explanations.

The *what—event* category also included time-related events. Despite their relatively low frequency, time-related questions may be important to a user's understanding *if* they seek information about where in a sequence a particular event occurred relative to others. For example:

> …how about the most recent milestone event and when that took place?

As with questions seeking *where* information, the extent of functionality required to meet this requirement may be especially dependent on the agent's domain, specific tasks, and the support an interface provides. Real-time domains, for example, are likely to demand that agents maintain some sense of temporal awareness, and have the capability to express how time impacted the decisions that they made while operating in the domain.

2.2.2. Mechanistic explanations

How does it work questions were relatively common (19%). These are the first of what can be called "white box" explanation requests, as they seek information about the internal mechanisms of an agent—the mechanisms and parameter data that work together to produce behavior. These explanations supervene on lower-level, simpler explanatory information, especially their interactions. Information provided in a *what—identity* explanation combined with *what—event* information provides some of the building blocks for mechanistic explanations, such as:

> So what happens if I click on this top goal?
>
> It doesn't make any sense because how would anything get on your panel if it wasn't on your radar?
>
> what's he thinking—does he know what the other one knows?

Meeting these requests involves accessing runtime actions of the model and how parameter data guide their operating sequences and branching behavior. Though not classified here as a why-type explanation, *How does it work?* information has the potential to reveal causal-mechanical relationships about how actions work together to produce outcomes.

2.2.3. Operational explanations

Operational or *how do I use it* explanation seeking questions was fairly common (12%). For designed systems, especially interactive systems, providing answers to these types of explanation seeking requests are important to ensure the operational capability of system objects and to contribute to usability. As noted earlier, instrumental/procedural questions such as this were by far the most common found by Lang et al. (1992) in their study of the kinds of questions asked at the human–computer interface. Some examples include:

> Can you change the scale of this display?
>
> I lose track of where — if you want to shift back to the map?
>
> can you turn it off if you don't want it?

However, not all intelligent agents are interactive. In fact, by definition most agent-based systems are designed to operate semi-autonomously, and therefore do not require information about how to use them at runtime. Still, most agents are parameterized, equipped with initial values before being introduced into the simulated world where they operate, and these initial parameter settings may need to be explained to their users. Humans who delegate tasks to intelligent agents concede some authority regarding how the task is accomplished; using an agent is therefore largely a function of this initial parameter setting.

2.2.4. Design rationale

Design rationale or *why* explanations are those that appeal to deep information about a model's structure or behavior. Why explanations cut across the different types of explanation-seeking questions discussed so far. For example, an ontological question might be augmented or followed by an appeal for the reasons or rationale for why a given entity or attribute exists as part of the agent. Similarly, an operational explanation can be elaborated with information about why a particular agent user interface feature is used in a certain way. Two types of why explanation-seeking questions were identified in the transcripts, functional and pragmatic.

2.2.4.1. Functional explanation. Questions related to the goal or purpose of an entity, process, or event were the most common of the *why* explanation types at 6%. Functional explanations play a key role in providing the *why* behind the *what* information imparted in operational explanations. They use design intent to introduce the rationale for a given entity, behavior, or event coming to exist within a particular system framework. Given how common requests for functional explanation were in the study, and given prior evidence of the centrality of functional explanations (Lang et al., 1992), these appear to be among the most important explanations to provide. Functional explanations exist at multiple levels. Sometimes the functional explanation requested may appeal to the purpose of the agent or even its environment, for example, a simulation. Explanation requests such as these:

> …okay, and what is the ultimate goal of this?
>
> why would a pilot want to watch this?
>
> What's it supposed to do?

show that knowledge of purpose and intent are fundamental to user understanding of an intelligent system.

2.2.4.2. Pragmatic explanation. Questions seeking the *pragmatic* dimension of explanation were common in the *why* portion of participant questions, at 5%. In particular, participants sought information about contrast classes, the reasons why one entity, behavior, or event exists in the model rather than some other. Some examples from the SAP transcripts show the different forms of these explanation requests:

why didn't he do that?

What if you had 5 times more people with you?

oh, so this isn't meant to represent a system interface?

Providing explanations based on contrast classes, or negative questions, presents some special difficulties for explanation facilities. They require, for example, that the explanation knowledge base include not only comprehensive information about what is in the agent and what behaviors it can perform, but also the rationale for why alternative structures and behaviors were not selected. Even in simpler cases (such as in the first of the examples above), answering pragmatic queries requires examining the branching behavior of an agent and the situation that caused a particular branch to be followed.

2.3. Discussion

The results of this study provide a description of the types of questions users ask and an initial distribution of explanation-seeking question types. The questions were taken from a wide range of user types, deliberately chosen to sample the range, not a random sample or a sample of convenience. In addition to the literature review, these results ground the design we present next. This study presents only one case for explanation-seeking with intelligent agents, so applying this design to other domains is important future work. Other task domains, and even other user-agent interface designs may lead to different results. These designs present a successful foundation from which other researchers and developers can learn, and from which they can adapt and extend to fit their specific requirements.

Fig. 3 illustrates the inherent dependencies between the different explanation types, and may account for why some types of explanation-seeking questions are more common than others. For example, the question types discussed here typically assume the participant has a basic level of understanding of the domain in which the agent is operating. As a result, many *mechanistic*, *operational*, and *design rationale* (why*)* questions are preceded by an inquiry about the agent's ontology, the identity and definition of it and its components. Such initial questioning serves to ground each user's knowledge in the ontology of the domain before moving on to deeper explanations about how an agent is designed to work and why.

As a result, explanation-seeking questions related to *ontological* information were the most common questions, and were often followed by operational or mechanistic questions. These ontological questions cover the attributes, behaviors, events, and relations within and between agents. Ontological explanations provide structure and state-of-the-world information that act as a scaffold for mechanistic, operational, and design rationale-based explanations. The information required to produce ontological explanations may be among the easiest to provide. Many modern development languages (e.g., Java, C#) include facilities for class and object *introspection* (also called reflection), which allow programmers to interrogate the identity, components, parents, and other attributes of a given runtime object. Many agent development and modeling languages, however, such as Soar and ACT-R, do not possess this functionality and this information must be provided as part of the explanation facility. The ontological explanation design in the next section describes an approach to providing this.

Surprisingly, requests for *mechanistic* explanations were more common than those for operational explanations. This is surprising because one would expect that users would be more interested in how to use an agent rather than its internal operations, per the theory of minimalism. Though only cautious generalizations can be made from the results of this study, we suggest that this result points to an aspect of explanation for intelligent agents that differentiates the domain from that of more traditional productivity software systems. In particular, it

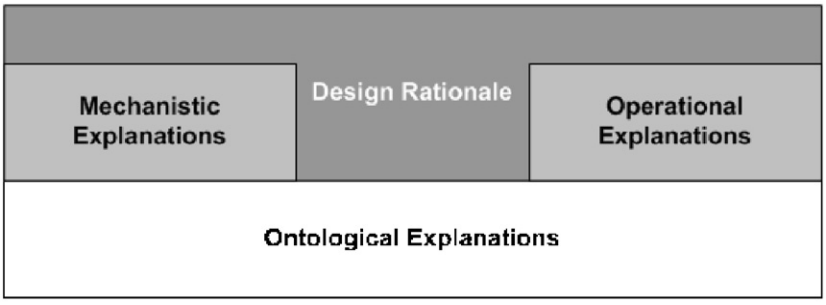

Fig. 3. Dependencies between operational explanation types.

may be the case that providing information about how an agent *works* may be more important than information related to how the agent is *used*. How an agent makes use of its operating parameters to carry out a task may be more important than the user interface through which those parameters are provided.

Operational explanation-seeking questions were the third most common type. Providing explanation in response to these questions is problematic for a number of reasons, not least that they require access to design documentation describing the intended use of the agent. This information is difficult to derive directly from the structure and behavior of the agent itself. The various interfaces of an agent's components may be intended for access from other components, from external agents, from user interface widgets, or from any combinations of these. Determination of which interfaces are meant to expose functionality to human users or delegators, and describing how they are meant to be used requires access to documentation not usually included in an agent.

Another issue that complicates efforts to answer operational questions is that they may involve access to exogenous knowledge, such as instrumental knowledge about a simulation in which an agent is running, to understand how the agent is controlled within the simulation. Also, answers to questions such as these are sometimes dependent on knowledge about the presentation layer software that provides the interface between a human user and the simulation. Because one of the design goals for intelligent agents is that they exhibit some measure of autonomy in how they work within simulated microworlds, it may be impossible to completely know this information at agent development time.

There are two types of *why* questions. *Functional* explanations are recognized as central to designed and engineered devices (Elster, 1983), and emerged from our analysis as the most frequently requested type of *why* explanation. A major challenge to providing this information is capturing design intent when agent and their components are created. Because almost any agent attribute or behavior may at some point be interrogated in its use context, and because this information is not derivable from the mere existence of a component or a mechanistic understanding of how different components work together, agent developers should capture this information at the time that an agent or one of its components is created. Only the designer and/or developer of a software component is equipped to document why they chose to create a particular component and how the component was designed to contribute to the system.

Perhaps most problematic of the findings is the apparent importance of pragmatic explanations to agent comprehensibility, which is the second type of why questions. *Pragmatic* explanations require that the explainer understand the context from which the question emerged. These pose special problems for designers of explanation facilities because of the difficulties inherent in agent access to details of contexts in which they are used. Our suggested approach, as described in the design below, is to provide access to as much of the design rationale as possible, and to carefully design access to this information so that its users can navigate to what they need in a given context.

The most striking result of this analysis is the relatively low frequency of explanation-seeking questions that appeal to design-rationale information about *why* the agent displays the structure and behavior that it does, such as the design constraints that guide how a given model was implemented. Results of the study suggest that those who interact with intelligent agents may be relatively satisfied (at least in the early stages of use) with explanations that appeal only to the operational aspects of the models they use. Relatively simple lists of the agents in a simulation, their attributes, and how they are parameterized to achieve their goals are often sufficient to satisfy many user concerns.

Despite the relatively low density of requests for design-rationale or *why* explanations, we believe these might still have an important role in intelligent system use and design re-use. These questions may not be frequent, but appear to be crucial when present. Some researchers have found that sophisticated users are more likely to use knowledge related to the underlying rationale of a system, in particular, the way in which the design of the system was related to tasks in the problem domain (Hackos, 1998), rather than surface-level information regarding the features of the system or the tasks it is designed to support. Other research indicates that users desire and require much more information about the genesis of the tools they use than is commonly attributed, especially in the case of more complex systems (Hackos, 1998; Mirel, 1998), and in more complex domains (Forsythe, 1995). However, these requirements may not manifest themselves on first exposure to a given agent or agent environment (as in our study) but over extended periods of use as a user's understanding of basic operations increases and their questions become more sophisticated. Further research is needed to understand the explanation requirements of agent users over time.

Some work on the essential nature of explanation has argued that what constitutes an effective explanation may be substantially dependent on the context in which the explanation request is made (van Fraassen, 1991). For example, consider one of the questions we used as the basis for part of our explanation facility design: *Where does dealing with this bogey [an enemy aircraft] fit into my goal hierarchy*? There are a number of plausible translations of this question including: *am I going to deal with this bogey? when am I going to deal with this bogey? how am I going to deal with this bogey*?, and so on. This pragmatic dimension of explanation content determination makes efforts to intelligently manage the explanation dialog difficult. One possible response to this concern is to consider explanation from a constructivist perspective, which effectively transforms the explanation delivery problem from one of dialog management to one of information

design. In other words, our designs assume that the most important goals of explanation delivery are to identify the content *potentially* most important to the explanation requestor, and then to make this available in a form that allows the user to actively navigate and construct the explanation that they need for the specific purpose of a given request.

Further research is required to better understand the different kinds of system and cognitive events that lead to a need for explanation. This study, while using a wide range of participants and a large, sophisticated agent, is still just a single study from a single domain; other system/user combinations are likely to have other distributions of question types. While we believe that the explanation types identified provide a solid theoretical and empirical grounding, we have not attempted to order their importance based on this study.

3. A design for providing commonly requested explanations

We describe our design using the structure of design patterns because this will help reuse the design, not because these designs can be considered patterns yet. We cast our results in the format of design patterns for three reasons. (a) Design patterns provide a recognized format for capturing the design lessons learned, as a summary of the results from our review and study. (b) Design patterns provide us with a formal and structured way to communicate our design for explaining intelligent agents. And (c) design patterns have made it easier for us to implement explanations across agent platforms.

In our description of each design, we use the four elements seen as forming the core of a design pattern by Gamma et al. (1995), including:

- *Name*—A short (two words) but descriptive and meaningful label.
- *Problem*—Generally the set of conditions suggesting when it is appropriate to apply the pattern.
- *Solution*—The body of the pattern describing an abstract description or template for addressing the pattern problem. The solution generally includes the components (e.g., classes or objects), component responsibilities, and component relations that work together to solve the problem.
- *Consequences*—A description of what occurs when we use the pattern, the trade-offs that are made, and the costs and benefits of using a particular pattern relative to other prospective solutions.

Other information is sometimes provided to augment the basic pattern structure, and to make it easier to assess a given pattern's relevance. This can include the forces or constraints that guide the approach to a problem, examples showing where and how the pattern was previously applied (or the solutions from the pattern were derived), and other related patterns (Bruegge and Dutoit, 2004).

Among the challenges to effective use and re-use of design patterns is identifying *which* design pattern is appropriate to a particular case, and *how* best to adapt and implement the pattern to solve the problem. One suggestion for identifying appropriate patterns is to focus on the pattern creator's intent as documented in the problem and solution sections of the pattern, and to review examples, if available, to see how well they map to the problem at hand. Design patterns typically include sample source code, diagrams, or other aids to understanding their function (as well as to show how the pattern can be adapted to the unique requirements of the design scenario). Much of a pattern's usability derives from how well it was written, its simplicity and clarity, and where it has been applied successfully. So, we include examples where we can see the same design implemented in other systems, and we illustrate these design patterns by including them in our design.

Three designs were derived from our review of explanation research and from our empirical work. The designs support providing the most commonly requested explanations found in the analysis of user's questions, and allow for the possibility of supporting users who require or desire deeper, *why* explanations derived from design rationale. These designs constitute an initial pattern language or framework for explaining intelligent agents. They support providing the 10 types of explanations, making the design clearer and easier to implement than specifying 10 different explanation facilities, where these 10 could be grouped into three more fundamental types.

The *Ontological* explanations design describes the structural properties of an agent or intelligent system—the elements, attributes, and relations that constitute a system's foundation. The *Mechanistic* explanations design addresses the more difficult problem of describing how the elements making up an agent's structure work together to carry out tasks, solve problems, and communicate with the environment. Finally, the *Operational* explanations design describes an agent's interfaces: how people and other agents delegate responsibilities to the agent, and how they should interpret and use the results produced by programmed agent behaviors. Each of these designs includes *design rationale* as a cross-cutting part of the explanatory information available to agent users at runtime, so we include this information as part of each explanation type.

3.1. The ontological explanation design

Ontological explanations provide the *what* behind a system's static structure. The explanation products of this design, shown in Table 4, may be especially useful for new or novice users as they try to grasp the conceptual architecture of an intelligent system.

Ontological explanations are provided through access to static knowledge that is either inherent in the agent's construction or added to the declaration of a system component at development time. This latter information

Table 4
The ontological explanation pattern

Pattern name	Ontological explanation
Problem	Users need to understand an agent's structural properties: its components, their attributes, and how they are related to each other.
Solution	Every object in an intelligent system is tagged with attributes to capture explanatory knowledge during development. The types of these attributes are derived from theories of explanation (see Section 1.2) and comprise the following: • Identifier, including its class • Components • Relations • Definition • Purpose • Design rationale Access to this information is provided by an explanation viewer or other instrument that accesses and displays the ontological knowledge captured during design and development.
Consequences	A major constraint when providing ontological explanations is access to the exogenous knowledge that maps the entity to the related concepts in the domain. Though some of this knowledge is captured in the course of everyday agent development activities, unless this pattern is used, much of the rationale that maps domain concepts to design decisions is lost after consideration. This knowledge is obtained and generated by knowledge engineers and agent developers in the course of agent design and programming, but capturing it during design adds additional overhead to the agent development process. It has been shown that asking software developers to do significant additional work beyond that directly related to their immediate goals, when they are not the beneficiaries of that work, is problematic (Clancey, 1983; Grudin, 1988). One challenge is to make apparent the benefits (i.e., having agents explain themselves) of access to this knowledge to the agent developers tasked with capturing the design knowledge. Making this knowledge useful through a design pattern may help both uses.

may include information related to the definition of the component including its purpose and other design rationale. Structural properties are provided by exposing the parent-child, class-attribute, and attribute-value relations that are built into the system, and by accessing the explanatory information tagged to components.

Fig. 4 shows the two types of explanations implementing the ontological explanation. The first, *search or browse component*, shown at the top of the sequence diagram in Fig. 4, describes a reflective or introspective process that extracts the agent's structure or the structure of one of its components. The second shows the user expanding the component's structure to show its *is-a* and *has-a* component relations. Ontological explanations may also contribute to the user's understanding of the mechanics of a given agent's behavior by showing why a particular behavior was expressed.

Examples of ontological explanations include the original Mycin explanation facility (Buchanan and Shortliffe, 1984), which provides access to its rule base via a simple stack trace mechanism, and the Explainable Expert System (EES) project (Swartout et al., 1991), which provides design rationale for its components as generated by the automatic expert system programmer. Modern tools for programming documentation, such as JavaDoc, implement a version of this design by capturing and presenting information about a class and its members. The SAP model used in the study provides such explanations for Soar models running in its environment.

3.2. The mechanistic explanation design

Mechanistic explanations, shown in the design in Table 5, describe how the components of an agent interact to perform a task; they show *how* the agent works. Instances of this design rely on being able to recognize agent state changes, the events and actions that cause these changes, and the post-conditions that are obtained as a result of a state change. To implement this design, each agent component must have the ability to provide a trace of how it obtained its current state, and the consequences of that state. Once that information is captured, then the explanation facility can trace back through these state transitions and expose them to the user. How to present them to the user in a useful fashion is environment dependent; a programmer can probably read a stack trace, but less sophisticated users will require a more stepwise and less computational explanation.

The challenge in mechanistic explanation is to show agent state changes as they occur over time and in response to events and other agent actions; to show how an event or other signal causes an agent component to react in a given way; and to show the consequences of that reaction. One way to approach this problem is to augment the explanation of a rule or production with a display of two key components: the conditions that cause the agent to perform some action, and the conditions that are obtained as a result of this action. A key challenge is to filter the components, actions, and events involved in a condition-action-condition trace so that only those elements relevant to the user's interest request are included in the explanation. This is a difficult problem for at least two reasons. First, a developer must identify those components of an agent potentially relevant to an explanation request. Second, the computational overhead of behavioral trace mechanisms may create performance problems. Further research is required to help ensure that the costs of explanation facilities are justified on both of these counts. The sequence diagram in Fig. 5 shows three ways the mechanistic explanations can be implemented in an agent user interface.

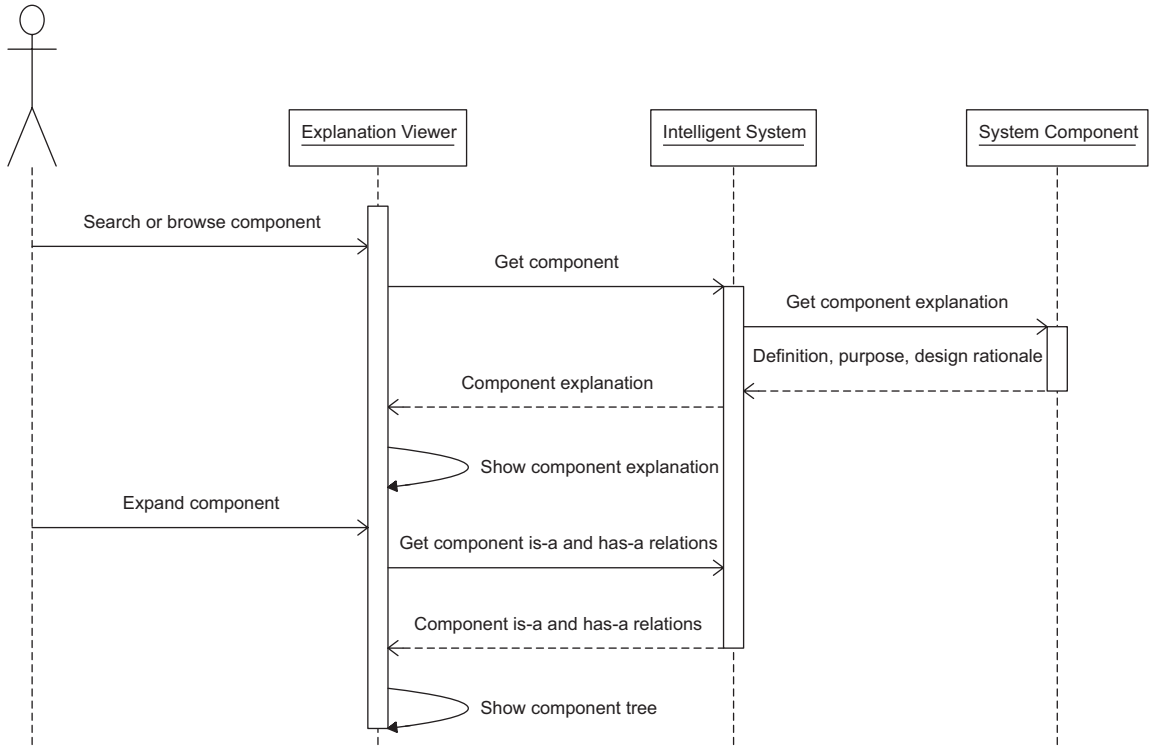

Fig. 4. UML sequence diagram for ontological explanation.

First, the explanation mechanism retrieves the *how-it-works* static explanation from the agent component. Second, the user can request further information about the condition-action pair that brackets the component (generally a rule or operator); this describes the state required for the rule or action to be called. Third, the user may request to view the status of the model component in the runtime trace.

Examples of mechanistic explanations in intelligent systems include the Debrief system (Johnson, 1994), which uses introspection to retrace and replay the results of a problem-solving trace, and Mycin, Vista (Taylor et al., 2002), and other rule-trace systems that provide a rule firing log. The advantage of the design described here is that elements that appear in a trace can also include information from their design rationale including their definition, purpose, and other reasons why they are designed to behave in a certain way, which helps situate agent behavior relative to a user's task.

3.3. The operational explanation design

Operational explanations, the design is given in Table 6, provide users and potentially other agents with information about how to access an intelligent system and its functionality. These types of explanations are most similar to end-user documentation; they describe how the behavior of the agent is initiated and controlled by an external user. In the case of an agent acting as the user of another agent, operational explanations resemble an application programming interface (API) specification, which describes the use signature of the desired functionality. The goals of operation explanations are generally short term and pragmatic; they relate less to long-term learning and deep understanding than they do to the immediate goals of the intelligent system user. Only those systems that expose an interface to the external environment might potentially require such information. Fig. 6 shows a sequence diagram for the operational explanation design.

Examples of operational explanation include the ubiquitous user manual, as well as those related to self-describing web services and the components of the semantic web (McGuinness and da Silva, 2004). The service-oriented architecture paradigm generally employs operational explanations to expose a standard public interface to other client applications including web-based ones.

3.4. Summary of the design

We presented three designs for providing explanations to users of intelligent agents. These designs are based on a review of the literature on providing explanations, on a study of a deliberately sampled diverse range of users, and with reference to existing systems that have attempted to provide explanations of intelligent agents to users. Each design is thus a potential solution to commonly seen problems, and draws upon lessons from several sources about how to provide single types of explanations. It is a relatively novel approach that provides the range of

Table 5
The mechanistic explanation pattern

Pattern name	Mechanistic explanation
Problem	Users need to know how agents work. A process within an agent or cognitive model begins with recognition of a new event, including creation of a new goal or new state. An event may originate internally, or may be obtained from its interaction with the environment. Any state transition that occurs in response to recognition of an event is potentially explanatory.
Solution	Providing mechanistic explanations involves explicating state transitions. Though any transition is potentially explanatory, only a subset are likely to be relevant to comprehension of the system. Production based intelligent agents follow a basic sequence as follows: 1. A goal is activated and recognized 2. Rules are proposed in response to the goal based on a set of pre-conditions 3. A rule is selected from the set based on a preference 4. The rule fires and takes actions such as activating a new goal or changing structures (a) if new goal (subgoal), repeat step 1 (b) else, repeat step 2 For each reasoning trace component: *goal*, *rule* or *operator*, *condition*, or *action*, the pattern specifies the following explanatory elements: Identity (ontological) Definition (ontological) Purpose (ontological) How it works (ontological) In addition to providing trace information for each of the components, the explanation facility should possess the ability to provide information about contrast classes: The program branches (rules or larger structures) that were not selected as part of the reasoning process, and *why*. For example, mechanistic explanation involves showing how a particular goal and set of conditions caused particular rules to fire in the reasoning component. These rules may in turn set new conditions and/or cause the system to take actions that may involve its output component. The scheduler provides rule and temporal ordering information, which helps expose the causal sequence underlying system behavior. Each system component is marked up with additional information, such as its definition and purpose, which helps map behavior back to the domain principles that they represent.
Consequences	Mechanistic explanations are computationally expensive because they involve keeping track of the sequence of state changes that realize system behaviors.

explanations in a single design. What can this design look like in practice? We take that up next.

4. Explanation design implemented: the Herbal agent development environment

We have used these three designs to create explanation facilities in the Herbal agent integrated development environment (IDE). Herbal (Cohen et al., 2005) is a suite of tools for developing intelligent agents using both the Java Expert System Shell (Jess) (Friedman-Hill, 2003) and the Soar cognitive architecture (Laird et al., 1987). Herbal is based on a high-level, ontological description of the Problem Space Computational Model (PSCM) (Newell, 1990). The Herbal IDE includes tools for creating agents and capturing their design rationale to support explanation, and a graphical explanation viewer to support interacting with the agent in runtime. The explanation designs introduced above (ontological, mechanistic, and operational) emerged from the development of Herbal with reference to both prior research on explanation in intelligent systems, and direct empirical research including the analysis presented in Section 2.

The Herbal IDE is provided as a customized version of the Eclipse Integrated Development Environment (eclipse.org). Developers create a model by graphically instantiating an ontology representing an Herbal agent framework. This ontology is based on the PSCM, and is realized as an XML schema. A custom compiler translates the XML schema into either Soar or Jess source code, which is annotated with the explanatory information described in the design.

Information related to the ontological, mechanistic, and operational explanation designs are captured, both explicitly and implicitly, as the modeler creates an agent using the Herbal Eclipse plug-in. In addition, the modeler can include design rationale justifying the decisions made with respect to the ontological, mechanistic, and operational design of the agent's behavior. This information is consolidated within a custom view in Herbal that allows the modeler to browse an agent and answer the questions supported by the design.

The Herbal suite of tools has been successfully used to teach agent development and cognitive modeling in undergraduate and graduate courses (Cohen et al., 2005) and as a conference tutorial (Ritter et al., 2005). One early study demonstrated a three-fold increase in programmer efficiency when using Herbal to create Soar agents (Morgan et al., 2005). A later study shows a 3× speed up for psychology undergraduates compared to computer science graduates (Cohen, 2008). Herbal is distributed at no cost and is currently available for download (acs.ist.psu.edu/Herbal). In the following sections, we describe two example Herbal agents that demonstrate the explanation designs in action.

4.1. Sense and respond agents in Herbal

The first example agent operates in a simulation environment created to evaluate the design of a *Sense and Respond* architecture for the United States Marine Corps. The purpose of the architecture is to reduce the time, cost, and effort involved in the maintenance of light armored vehicles (LAVs), and to increase the readiness and reliability of the LAV fleet. Due to the size and complexity

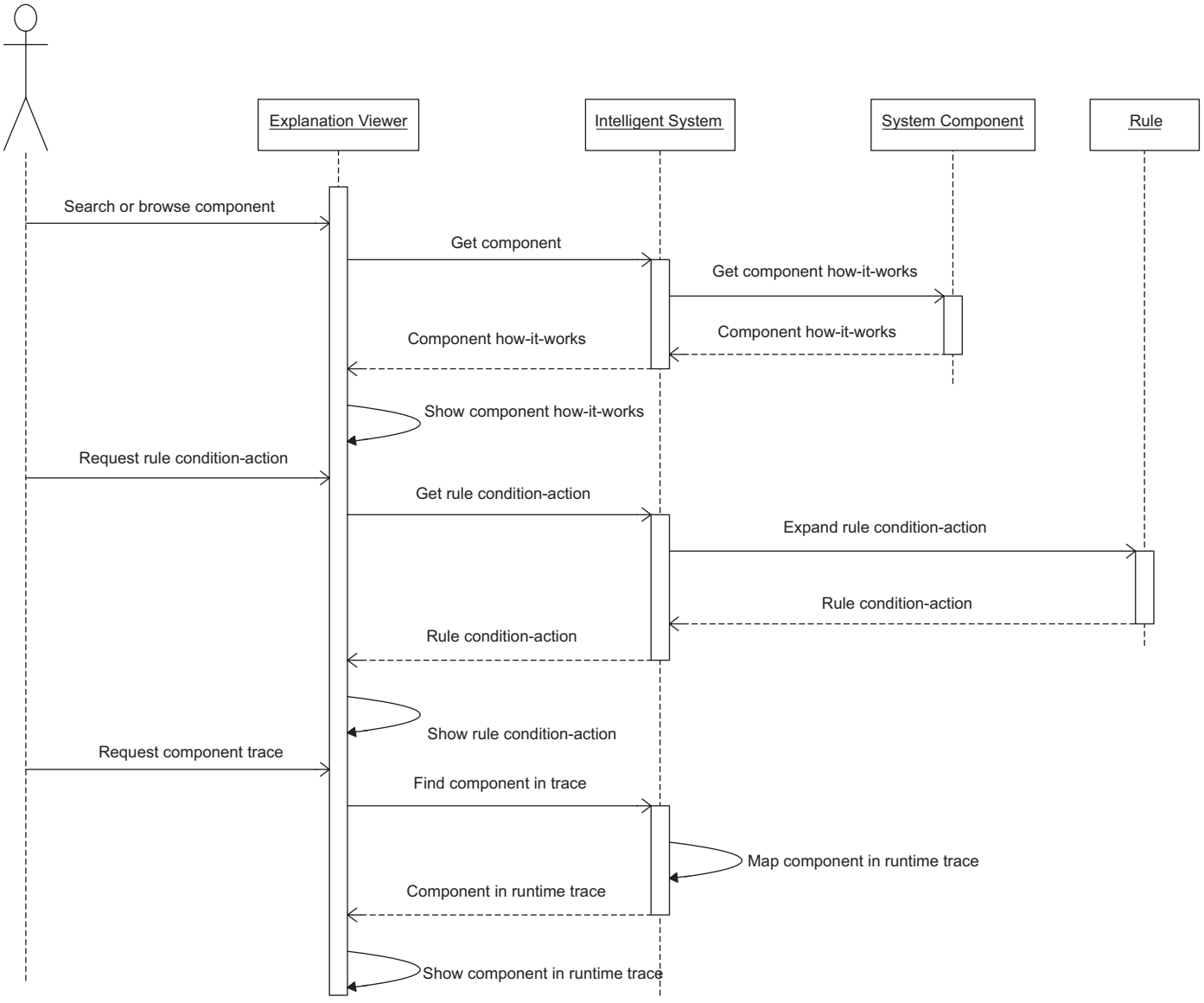

Fig. 5. UML sequence diagram for mechanistic explanation.

of the system, it is important to evaluate the performance of proposed LAV designs early in their development cycle. The simulation we have developed uses scenarios to simulate the relative utility of different design decisions. Herbal-based agents participate as virtual users in the execution of a scenario.

An agent playing the part of a maintenance officer has been chosen to illustrate Herbal's implementation of the explanation designs. Specifically, the maintenance officer agent schedules LAV repairs as shown in the use case in Fig. 7.

The use case shown in Fig. 7 is divided into three distinct activities. First, the agent must obtain a work order that contains the appropriate parts and documentation needed to perform the repair. The work order is dependent on the type of operational problem reported by the LAV autonomic logistics system. Next, the agent needs to determine the required maintainer competencies, again based on the current repair. Finally, the agent must schedule the repair by assigning an appropriate maintainer that is available when the work package arrives.

4.1.1. Ontological explanations in Herbal

Using Herbal, a maintenance officer agent was created to implement the behavior illustrated in Fig. 7. The Herbal explanation viewer provides explanations about the agent's structure and behavior. For example, explanations that describe the static properties of the agent and its components—ontological explanations—can be obtained in the Herbal explanation view by clicking on an agent component and viewing both the model structure and the "What is this?" section of the display (Fig. 8).

The ontological explanations, shown by the Herbal viewer of the model structure, match the structure represented in the UML diagram. This is especially important because these explanations are generated from

the agent code, and are automatically updated as the code is further developed. As a result, unlike UML diagrams and other external design documentation not explicitly linked to the program source code, the content of these explanations is automatically synchronized as the agent code changes.

Table 6
The operational explanation pattern

Pattern name	Operational explanation
Problem	Users need to know how to use an agent. A comprehensive explanation facility needs to include explanations of *how to use* a system. These kinds of explanations are more like instructions in that they specify how an agent's external interface, either user interface controls or an API, is employed to achieve desired results.
Solution	The first step is to identify the system behavior required by the human or other agent interacting with the system. This information may be of two types: the information required of interacting agents at runtime, and the information required by humans (or, potentially, other agents) who delegate tasks to the agent. In both cases we use the idea of a public interface and agent introspection as the basis for this capability. The approach involves tagging those components of the system that access its input/output capabilities. System developers are encouraged to provide information about how the system interprets inputs, and how the system's output is to be interpreted by users.
Consequences	An issue with this pattern is that different intelligent system development environments treat input/output and human–computer interaction in different ways. In Soar, for example, I/O is easily identified because it is done using specific and uniform data structures. In more open development tools, such as Jess, input and output are configurable by the developer, varies more between agents, and is less easy to identify. Operational explanations should preserve the encapsulation built into a well-designed API or user interface, hiding complexity unnecessary to realizing pragmatic goals.

4.1.2. Mechanistic explanations in Herbal

The Herbal Viewer provides mechanistic explanations by drawing upon the model's structure and on information entered explicitly and implicitly during model creation. The information displayed in the mechanistic view focuses on the how a particular element works and how it interacts with other model elements.

For example, in Fig. 9 the model structure illustrates that the ObtainWorkPackage problem space interacts with the ScheduleLAVRepair problem space and achieves its goal using the GenerateWorkOrder operator. In addition, the conditions that cause the ObtainWorkPackage problem space to become active, along with the initial actions performed by this problem space, are shown in the "How does it work?" section of the view. Finally, design rationale detailing why this problem space works the way it does is also shown. Again, it is important to note that all of this information is gathered, both explicitly and implicitly, during model creation.

Notice how both the ontological explanations contained in the model structure and the mechanistic explanations can be combined with design rationale to generate more complete and detailed explanations. This demonstrates how the explanation types can be leveraged and combined to provide more powerful explanations.

4.1.3. Operational explanations in Herbal

The Herbal viewer provides operational explanations by displaying information about how to use the various agent components. For example, the Herbal explanation viewer shows information about how a particular action works, making it easier for the action to be understood and reused. Fig. 10 shows the view displaying operational information about the determineParts action. Specifically, the "How do I use it?" section of the view

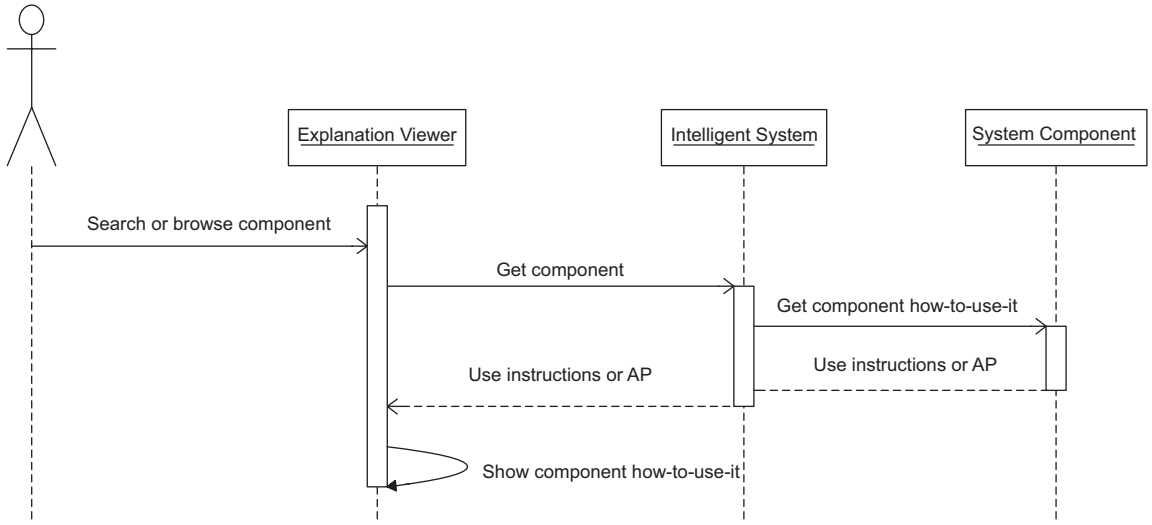

Fig. 6. UML sequence diagram for operational explanation.

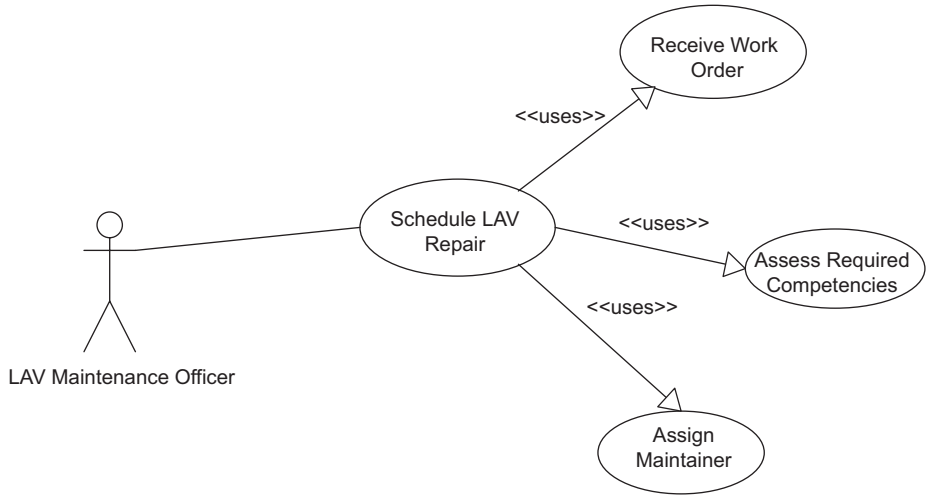

Fig. 7. Schedule LAV repair use case for the maintenance officer agent.

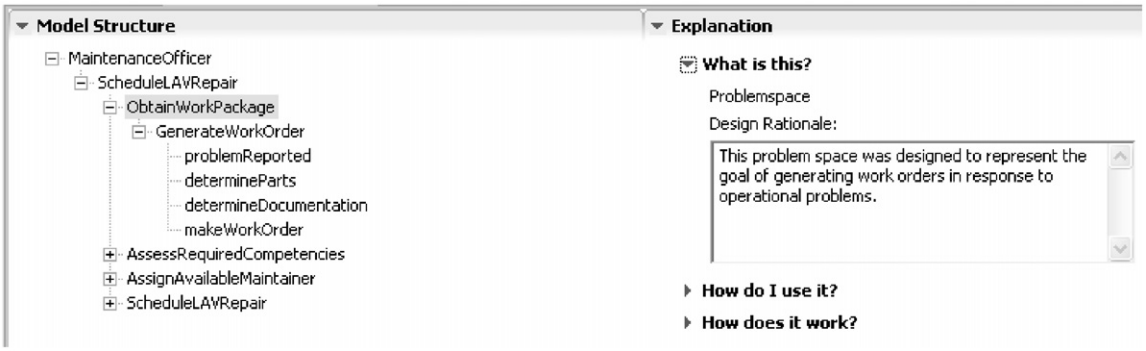

Fig. 8. Viewing the ontological explanation for the ObtainWorkPackage problem space.

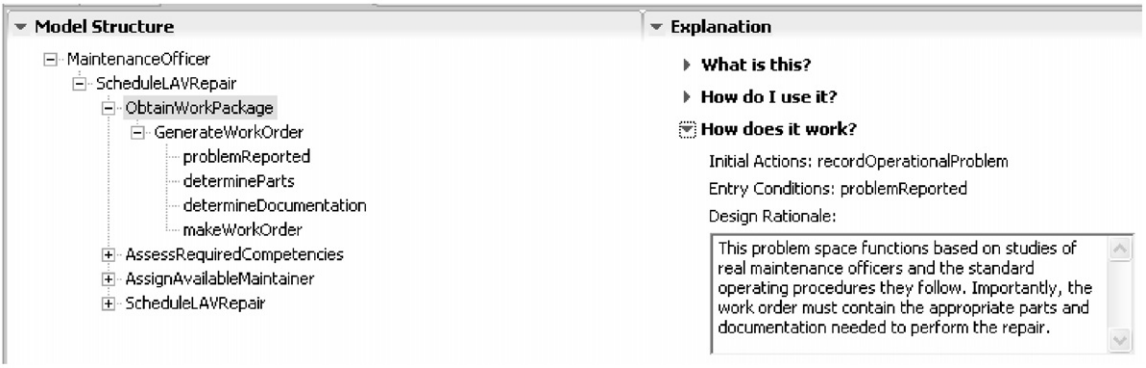

Fig. 9. Viewing the mechanistic explanation for the ObtainWorkPackage problem space.

provides information about what inputs the action requires and why.

4.2. A business to business e-commerce example

A second, more sophisticated agent was created to represent the buying behavior of an inventory manager in a business-to-business (B2B) e-commerce application. This agent is being developed to further explore the usefulness of Herbal agents and the explanation design, and to create an example agent of real-world complexity. The B2B agent's behavioral model is derived in part from theories and empirical studies of industrial buyer behavior (Bellizzi and Walter, 1980; Anderson et al., 1987). The agent embodies not only purely rational models of buyer behavior but also some of the contextual influences that

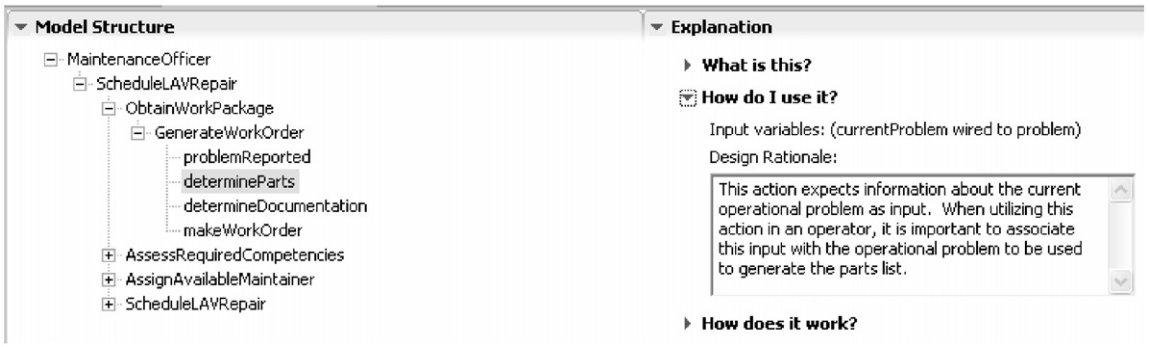

Fig. 10. Viewing the operational explanation for the determineParts action.

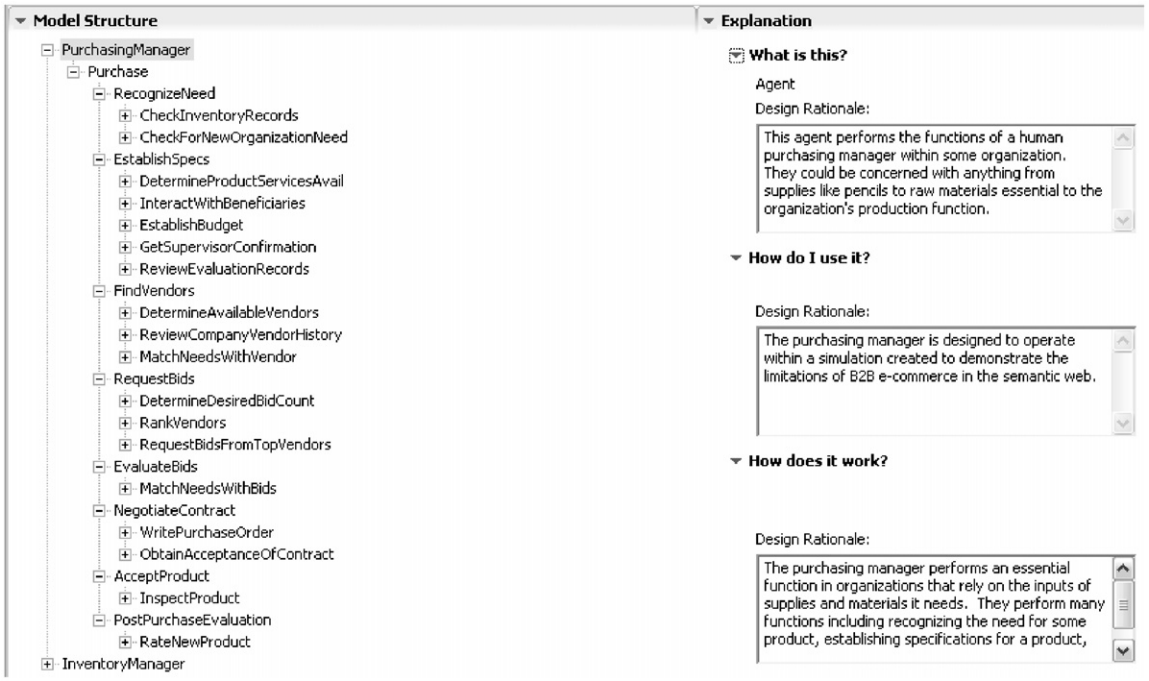

Fig. 11. Herbal explanation viewer illustrating the B2B agent.

appear to affect how buyers decide on a purchasing strategy. Fig. 11 shows the Herbal explanation view for the PurchaseManager agent.

From the view shown in Fig. 11, one can quickly obtain ontological information such as the fact that the inventory manager is an agent, or that *purchase* is the primary goal of the inventory manager. In addition, mechanistic information is equally apparent. For example, the model structure shows that the purchasing goal is accomplished by interacting with eight sub goals: recognizing the need for a purchase; establishing the specifications for the purchase; finding vendors; requesting bids from the vendors; evaluating these bids and selecting a vendor; negotiating a contract with the selected vendor; accepting the product shipment; and evaluating the purchase. Finally, an operational explanation is made apparent within the "How do I use it?" section, which shows that the inventory manager is designed to be used within a larger B2B simulation. By drilling down into the model components, more detailed explanations can be acquired in the viewer, including explanations about low-level agent components, such as the relevant conditions, actions, and data types.

The B2B agent example can be used to show how the *mechanistic* explanation design has been implemented in Herbal. As part of the vendor ranking process, the B2B agent includes a measure of vendor cooperation to determine the relative grades of suppliers who carry products meeting the specifications of the purchase requirement. The cooperation value is derived from purchase histories and is a function of the speed with which prior transactions were completed, modulated by any complaints logged by human purchasing agents in the course of the transaction. Using the explanation trace shown in Fig. 12, Herbal users can see how this data was used to assign a rank to the vendor PaperAndMore.

Fig. 12 shows the components of the "PurchasingManager" agent. The display includes the components of the identity of the components, including the problem space,

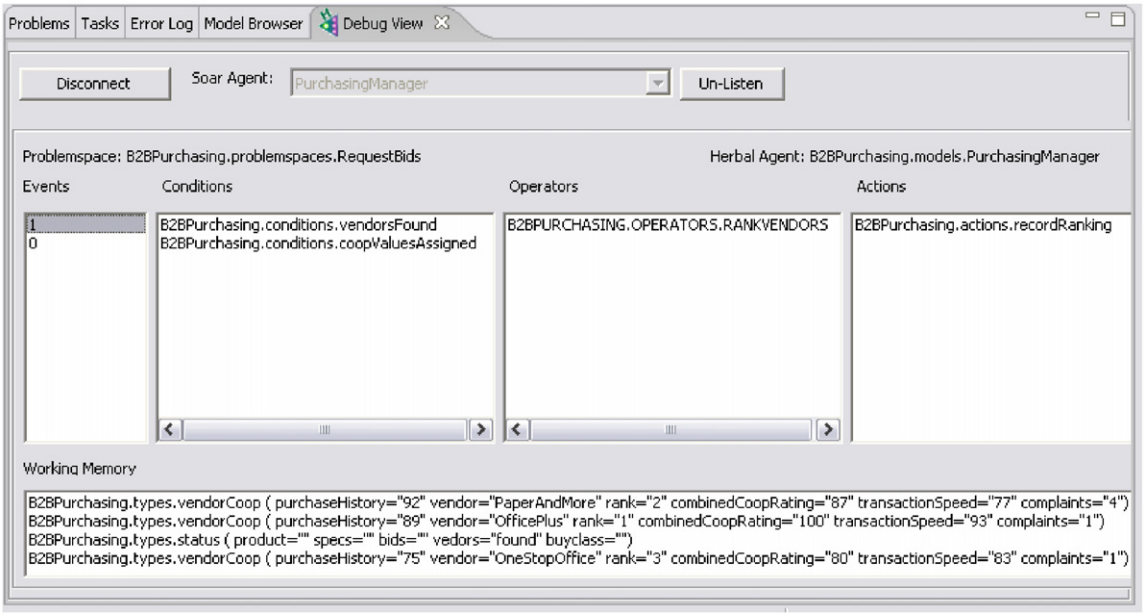

Fig. 12. Herbal mechanistic explanation trace for the B2B agent.

operators, actions, and conditions used, as well as the working memory elements in the agent. Some Soar knowledge is required to fully understand the mechanistic explanation, but the event displayed shows the agent's working memory just after vendors have been ranked and the trace showing how it was created. The vendors were ranked because the conditions "vendorsFound" and "coopValuesAssigned" were true. Using the vendors' overall cooperation values the "rankVendors" operator was applied causing the "recordRanking" action to execute. This action resulted in the ranking shown in the Working Memory subwindow in the figure: (1) OfficePlus, (2) PaperAndMore, (3) OneStopOffice). Further sub-explanations as noted in Table 5 are also available from this display.

4.3. Discussion

The examples presented here show instantiations of our explanation designs and provide explanation capabilities in agent architectures. Our use of Herbal to create agents in both Jess and Soar, and in two applications, military logistics and business-to-business e-commerce, suggest that the design is applicable to agent architectures that use domain knowledge and that have requirements to expose the what, how, why, and design rationale underlying their behaviors. The examples show explanations being provided at various levels (e.g., agent structure, behavior, and environment) and with different content corresponding to the ontological, mechanistic, and operational designs. We are currently implementing improved interactive graphical explanation displays in Herbal and we are exploring support for agent development in other architectures including ACT-R, CAST (Yen et al., 2001), and CoJACK (Ritter and Norling, 2006).

5. Conclusion

We have described a comprehensive design for including explanation capabilities in intelligent agents and shown an example implementation. The design supports providing answers to the range of questions users asked in our study and is consistent with and aggregates design results from the systems reviewed and the explanation literature. This design captures, reifies, and communicates the accumulated knowledge of intelligent system developers and researchers who have previously addressed the problem of explaining the structure and behavior of agents. This design has been useful in developing our integrated development environment for agent creation, and this suggests it may be useful to others when designing agents that require explanation capabilities.

We believe these designs can help build better agents. The individual designs are relatively simple, but as a set they realize a significant proportion of prior research in a single, unified design for intelligent system explanation. Because the designs are derived from prior research, previous research in a range of domains, and direct empirical research into the needs of users, they should be applicable where intelligent systems are used, including agent systems and cognitive models.

Creating effective explanation facilities for intelligent agents has been a goal of researchers and practitioners in AI since very early in the field's history. For computer programs to achieve the status of "intelligent" requires that these agents be capable of explaining themselves to the people and other agents with whom they interact. As

intelligent agents become more pervasive in our day-to-day computing environment, and as their role becomes more consequential with respect to human purposes, they will be increasingly called upon to communicate in a way that engenders trust and allows people to learn from the knowledge embedded in how they are designed to work. Intelligent agents that can explain themselves expose the knowledge inscribed into their structure and behaviors and make them more effective resources for their users and learners, and for agent developers who want to re-use and build on prior work.

As these designs are adopted, adapted, and reused further, they may evolve to the status of design patterns. This may be expected because each design is based on a literature review, queries from users about a particular system, and previous system designs. Design patterns are useful because they help capture and make accessible the wisdom that accumulates from experimenting with and experiencing what works and what does not in a given design problem-solving context. Applying these design patterns to help provide explanations will make available to agent developers, users, and stakeholders the knowledge acquired in the analysis and engineering of these complex systems.

Society's intellectual capital is increasingly vested in the high-technology designs we create. To realize the full value of this capital requires conscious attention to building more comprehensible intelligent agents.

Acknowledgements

This work was supported by the Office of Naval Research under Contract N00014-02-1-0021 and N00014-06-1-0164. Soar Technology provided the SAP, and comments on earlier reports on it. Isaac Council contributed to early discussions of this work. Joshua Gross provided many useful comments.

References

Anderson, E., Chu, W., Weitz, B., 1987. Industrial purchasing: an empirical exploration of the buyclass framework. Journal of Marketing 51, 71–86.

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y., 2004. An integrated theory of the mind. Psychological Review 111 (4), 1036–1060.

Antaki, C., 1994. Explaining and Arguing: The Social Organization of Accounts. Sage Publications, London, Thousand Oaks, CA.

Avraamides, M.N., Ritter, F.E., 2002. Using multidisciplinary expert evaluations to test and improve model interfaces. In: Proceedings of the 11th Computer Generated Forces Conference, University of Central Florida, Orlando, FL, pp. 553–562.

Bellizzi, J.A., Walter, C.K., 1980. Purchasing agents' influence in the buying process. Industrial Marketing Management 9, 137–141.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The semantic web. Scientific American 284 (5), 34–43.

Berry, D.C., Gillie, T., Banbury, S., 1995. What do patients want to know: an empirical approach to explanation generation and validation. Expert Systems with Applications 8 (4), 419–428.

Brachman, R.J., 2002. Systems that know what they're doing. IEEE Intelligent Systems 17 (6), 67–71.

Brooks, F.P., 1987. No silver bullet: essence and accidents of software engineering. IEEE Computer 20 (4), 10–19.

Bruegge, B., Dutoit, A.H., 2004. Object-Oriented Software Engineering: Using UML, Patterns, and Java, second ed. Prentice-Hall, Upper Saddle River, NJ.

Buchanan, B.G., Shortliffe, E.H., 1984. Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, Reading, MA.

Carenini, G., Moore, J., 1993. Generating explanations in context. In: Paper Presented at the Intelligent User Interfaces, pp. 175–182.

Carroll, J.M., 1990. The Nurnberg Funnel. MIT Press, Cambridge, MA.

Carroll, J.M., 1998. Reconstructing minimalism. In: Carroll, J.M. (Ed.), Minimalism beyond the Nurnberg Funnel. MIT Press, Cambridge, MA, pp. 1–17.

Cawsey, A., 1992. Explanation and Interaction: The Computer Generation of Explanatory Dialogues. MIT Press, Cambridge, MA.

Chandrasekaran, B., Swartout, W., 1991. Explanations in knowledge systems: the role of explicit representation of design knowledge. IEEE Expert 6 (3), 47–49.

Clancey, W.J., 1983. The epistemology of a rule-based expert system—a framework for explanation. Artificial Intelligence 20, 215–251.

Cohen, M.A., 2008. Herbal: improved modeling through using a high-level behavior representation language to support reuse and maintenance. Unpublished Ph.D. Thesis, College of Information Sciences and Technology, The Pennsylvania State University.

Cohen, M.A., Ritter, F.E., Haynes, S.R., 2005. Herbal: a high-level language and development environment for developing cognitive models in Soar. In: Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation, University of Central Florida, Orlando, FL.

Conati, C., VanLehn, K., 2001. Providing adaptive support to the understanding of instructional material. In: Paper Presented at the International Conference on Intelligent User Interfaces, Santa Fe, NM, pp. 41–47.

Cummins, R., 1975. Functional Analysis. The Journal of Philosophy 72, 741–765.

Dennett, D.C., 1987. The Intentional Stance. MIT Press, Cambridge, MA.

Dhaliwal, J.S., Benbasat, I., 1996. The use and effects of knowledge-based system explanations: theoretical foundations and a framework for empirical evaluation. Information Systems Research 7 (3), 342–362.

Elster, J., 1983. Explaining Technical Change: A Case Study in the Philosophy of Science. Cambridge University Press, Cambridge.

Ford, K.M., Cañas, A.J., Coffey, J., 1993. Participatory explanation. In: Proceedings of Sixth Annual Florida AI Research Symposium, Ft. Lauderdale, FL, pp. 85–90.

Forsythe, D.E., 1995. Using ethnography in the design of an explanation system. Expert Systems with Applications 8 (4), 403–417.

Friedman-Hill, E., 2003. JESS in Action. Manning Publications, Greenwich, CT.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA.

Graesser, A.C., Person, N., Huber, J., 1992. Mechanisms that generate questions. In: Lauer, T.W., Peacock, E., Graesser, A.C. (Eds.), Questions and Information Systems. Lawrence Erlbaum, Hillsdale, NJ, pp. 167–187.

Gregor, S., Benbasat, I., 1999. Explanations from intelligent systems: theoretical foundations and implications for practice. MIS Quarterly 23 (4), 497–530.

Grudin, J., 1988. Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. In: Proceedings of CSCW '88, pp. 85–93.

Hackos, J.T., 1998. Choosing a minimalist approach for expert users. In: Carroll, J.M. (Ed.), Minimalism beyond the Nurnberg Funnel. MIT Press, Cambridge, MA, pp. 149–177.

Hempel, C.G., 1965. Aspects of Scientific Explanation, Aspects of Scientific Explanation and Other Essays. Free Press, New York.

Hempel, C.G., Oppenheim, P., 1988. Studies in the logic of explanation. In: Pitt, J. (Ed.), Theories of Explanation. Oxford University Press, Oxford, UK.

Herlocker, J.L., Konstan, J.A., Riedl, J., 2000. Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, PA, pp. 241–250.

Johnson, W.L., 1994. Agents that learn to explain themselves. In: Proceedings of the 12th National Conference on Artificial Intelligence. American Association for Artificial Intelligence, Menlo Park, CA, pp. 1257–1263.

Jones, R.M., Laird, J.E., Nielsen, P.E., Coulter, K.J., Kenny, P., Koss, F.V., 1999. Automated intelligent pilots for combat flight simulation. AI Magazine 20 (1), 27–41.

Keil, F.C., Wilson, R.A., 2000. Explanation and Cognition. MIT Press, Cambridge, MA.

Laird, J., Newell, A., Rosenbloom, P., 1987. Soar: an architecture for general intelligence. Artificial Intelligence 33, 1–64.

Lam, D.N., Barber, K.S., 2005. Comprehending agent software. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 586–593.

Lang, K.L., Graesser, A.C., Dumais, S.T., Kilman, D., 1992. Question asking in human–computer interfaces. In: Lauer, T.W., Peacock, E., Graesser, A.C. (Eds.), Question Asking in Human–Computer Interfaces. Lawrence Erlbaum, Hillsdale, NJ, pp. 131–165.

Lehnert, W.G., 1978. The Process of Question Answering: A Computer Simulation of Cognition. Erlbaum Associates, Hillsdale, NJ.

Lerch, F.J., Prietula, M.J., Kulik, C.T., 1997. The Turing effect: the nature of trust in expert system advice. In: Feltovich, P.J., Ford, K.M., Hoffman, R.R. (Eds.), Expertise in Context: Human and Machine. MIT Press, Cambridge, MA, pp. 417–448.

Lombard, M., Snyder-Dutch, J., Bracken, C.C., 2002. Content analysis in mass communication: assessment and reporting of intercoder reliability. Human Communication Research 28 (4), 587–604.

MacLean, A., Young, R.M., Moran, T.P., 1989. Design rationale: the argument behind the artifact? In: Proceedings of the ACM Conference on Human Factors in Computing Systems, pp. 247–252.

McGuinness, D.L., da Silva, P.P., 2003. Infrastructure for web explanations. In: Proceedings of Second International Semantic Web Conference (ISWC2003), Sanibel Islands, FL, pp. 113–129.

McGuinness, D.L., da Silva, P.P., 2004. Explaining answers from the semantic web: the inference web approach. Journal of Web Semantics 1 (4), 397–413.

Mirel, B., 1998. Minimalism for complex tasks. In: Carroll, J.M. (Ed.), Minimalism beyond the Nurnberg Funnel. MIT Press, Cambridge, MA, pp. 179–218.

Moore, J., Paris, C., 1993. Planning texts for advisory dialogues: capturing intentional and rhetorical information. In: Proceedings of the 27th Annual Meeting on Association for Computational Linguistics, Vancouver, British Columbia, Canada, pp. 203–211.

Morgan, G.P., Cohen, M.A., Haynes, S.R., Ritter, F.E., 2005. Increasing efficiency of the development of user models. In: Paper Presented at the Proceedings of the IEEE System Information and Engineering Design Symposium.

Newell, A., 1990. Unified Theories of Cognition. Harvard University Press, Cambridge, MA.

Papamichail, K.N., French, S., 2003. Explaining and justifying the advice of a decision support system: a natural language generation approach. Expert Systems with Applications 24, 35–48.

Reich, Y., 1995. Measuring the value of knowledge. International Journal of Human–Computer Studies 42 (1), 3–30.

Ritter, F.E., Norling, E., 2006. Including human variability in a cognitive architecture to improve team simulation. In: Sun, R. (Ed.), Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge University Press, Cambridge, UK, pp. 417–427.

Ritter, F.E., Morgan, G.P., Stevenson, W.E., Cohen, M.A., 2005. A tutorial on Herbal: a high-level language and development environment based on Protégé for developing cognitive models in soar. In: Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation. University of Central Florida, Orlando, FL, pp. xxix–xxxi.

Salmon, W.C., 1993. Scientific explanation and the causal structure of the world. In: Ruben, D.-H. (Ed.), Explanation. Oxford University Press, Oxford, UK, pp. 78–112.

Su, X., Matskin, M., Rao, J., 2003. Implementing explanation ontology for agent system. In: Proceedings of the IEEE/WIC International Conference on Web Intelligence, p. 330.

Swartout, W., Paris, C., Moore, J., 1991. Explanations in knowledge systems: design for explainable expert systems. IEEE Expert 6 (3), 58–64.

Taylor, G., Jones, R.M., Goldstein, M., Frederiksen, R., 2002. VISTA: a generic toolkit for visualizing agent behavior. In: Proceedings of the 11th Computer Generated Forces Conference, 02-CGF-044. University of Central Florida, Orlando, FL, pp. 29–40.

van Fraassen, B., 1991. The pragmatics of explanation. In: Boyd, R., Gasper, P., Trout, J.D. (Eds.), The Philosophy of Science. MIT Press, Cambridge, MA, pp. 317–327.

Wolverton, M., 1995. Presenting significant information in expert system explanation. In: Proceedings of the Seventh Portuguese Conference on Artificial Intelligence.

Ye, L.R., Johnson, P.E., 1995. The impact of explanation facilities on user acceptance of expert systems advice. MIS Quarterly 19 (2), 157–172.

Yen, J., Yin, J., Loerger, T.R., Miller, M.S., Xu, D., Volz, R.A., 2001. CAST: collaborative agents for simulating teamwork. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), pp. 1135–1142.