Trustable Task Processing Systems

Alyssa Glass, Deborah L. McGuinness, Paulo Pinheiro da Silva, and Michael Wolverton

As personal assistant software matures and assumes more autonomous control of user activities, it becomes more critical that this software can tell the user why it is doing what it is doing, and instill trust in the user that its task knowledge reflects standard practice and is being appropriately applied. Our research focuses broadly on providing infrastructure that may be used to increase trust in intelligent agents. In this paper, we will report on a study we designed to identify factors that influence trust in intelligent adaptive agents. We will then introduce our work on explaining adaptive task processing agents as motivated by the results of the trust study. We will introduce our task execution explanation component and provide examples in the context of a particular adaptive agent named CALO. Key features include (1) an architecture designed for re-use among different task execution systems; (2) a set of introspective predicates and a software wrapper that extracts explanation-relevant information from a task execution system; (3) a version of the Inference Web explainer for generating formal justifications of task processing and converting them to user-friendly explanations; and (4) a unified framework for explaining results from task execution, learning, and deductive reasoning.

1 Introduction

Personalized software assistants have the potential to support humans in everyday tasks by providing assistance in cognitive processing. If these agents are expected to achieve their potential and perform activities in service of humans (and possibly other agents) then these agents need to be fully accountable. Before their users can be expected to rely on cognitive agents, the agents need to provide justifications for their decisions, including that those decisions are based on appropriate processes and on information that is accurate and current. Further, if the agents are to be used to perform tasks, they need to explain how and under what conditions they will execute a task, as well as how and why that procedure has been created or modified .

One challenge to explaining adaptive assistants is that they, by necessity, include task processing components that evaluate and execute tasks, as well as reasoning components that determine conclusions. A comprehensive explainer needs to explain task processing responses as well as more traditional reasoning systems, providing access to both inference and provenance information, which we refer to as knowledge provenance [1].

Work has been done in the theorem proving community, as well as in many specialized reasoning communities, to explain deductions. A limited amount of explanation work has been done in the task execution community. What has not been done is work explaining task execution in a way that is also appropriate for explaining *both* deductive reasoning and provenance. Our work provides a uniform approach to representing and explaining provenance and results from both communities, in addition to learned information.

We present our work in the setting of the DARPA Personalized Assistant that Learns (PAL) [2] program, as part of the Cognitive Assistant that Learns and Organizes (CALO) [3] project. The CALO system includes work from 22 different organizations. This presents a complex challenge where CALO users must understand and trust conclusions from multiple knowledge sources, both hand built and automatically generated, with multiple reasoning techniques including task processing, deduction, and learning. In this paper, we first describe a study of CALO users, in which we investigate issues of trust and usability, discussing several design guidelines implied by the results of this study. Using these guidelines, we then present our representation, infrastructure, and solution architecture for explaining BDI-based task processing and learning in adaptive agents. We describe how it has been implemented in our new Integrated Cognitive Explanation Environment (ICEE), and show how it has been used to provide explanations in CALO.

2 Trust Study

We conducted a structured, qualitative trust study to identify what factors influence user trust in adaptive agents, to understand the types of questions users would like supported by such a system, and to evaluate the general usability of adaptive agents.

Procedure Our study was conducted in two basic stages: the usage stage and the interview stage. For the usage stage, we piggy-backed on a broad study aimed at testing the learning capabilities within the CALO system. The broad study is part of a long term effort involving a large set of testers and researchers, extensive participant training for the use of various CALO components, and detailed analysis of complex data logs and learning algorithms through intensive system use by dedicated users over approximately two weeks. During the usage stage, participants typically used the system for a full eight hour work day, each day, for the entire duration of the test period.

During the interview stage, we interviewed each participant after the usage period. The interviews were structured to follow a fixed script for all participants. The script contained 40 questions—eleven questions using a typical 5-step Likert scale (from “hardly ever” to “extremely often”) and 29 open response questions. The script was organized around five main topics: failure, surprise, confusion, question-answering, and trust. Each interview was audio recorded. We used these recordings to make notes and to organize the responses into common themes.

CALO Agent Overview The CALO system provides capabilities for a wide range of office-related tasks [4], including maintaining calendars and schedules of meetings, managing contact information, scanning and sorting email and other documents, performing Web searches, scraping information from the Web, helping to prepare new documents and presentations, managing tasks, purchasing new equipment, planning travel, and learning new procedures for previously unknown tasks.

Typical tasks the participants performed with the help of their agents included scheduling mutually convenient meetings with groups of people; planning detailed travel arrangements to attend conferences; and teaching their agents how to independently find and store contact information for colleagues using Web searches and scraping. In many cases, participants were provided with guidance about how best to use CALO to perform the tasks. In other cases, participants were free to use any portion of the CALO system that they felt would best enable them to accomplish specific goals.

The main objective for the CALO project is to explore the use of machine learning techniques as applied in robust, complex assistant agents capable of reasoning, execution, explanation, and self-reflection. Questions of usability, though important when building a deployed personal assistant, were not central research questions for the project.

3 Study Findings

After completing the interviews, we grouped similar comments from multiple users into topics, and discarded topics that only a few participants commented on. Several themes are discussed below, focusing on those which impact the use of explanations. While not all participants commented on all themes, each of the themes were significant to a majority of the participants. Further detail can be found in [5].

Theme 1: Granularity of Feedback. In the cases where the agent provided feedback to the participants, many of them commented that the feedback was at the wrong level of detail for their needs. Several of the agent components provided simple status messages indicating simply "Okay" or "Not Okay." Participants found this type of feedback frustrating, because they desired additional feedback to explore details about the cause of the problem, and possible solutions. Participants commented, "[The component] would say 'I'm confused' and there was no other feedback," and "I definitely wanted to know WHY!" Lacking more detailed feedback, they were unable to fix the problem, nor to avoid it in the future.

Equally frustrating to many participants were other system components that provided an overwhelming amount of feedback. The constant stream of status information from these components was so frequent and cumbersome that most users found them to be unhelpful even when there was a problem.

Theme 2: Context-Sensitive Questions. To investigate the value of different types of explanations to user needs, we asked our users to rate a list of question types according to how often they would have utilized questions of that type if an answer to it had been available during the test.

We used Silveira *et al.*'s "Taxonomy of Users' Frequent Doubts" [6], an enumeration of user information needs, as our candidate question types, and each was ranked by each user on a Likert scale from 1 ("would never want to ask") to 5 ("would want to ask extremely often"). We averaged the ratings to produce an overall score of usefulness for each question. These questions can generally be divided into two categories. *Context-independent questions* have answers that are not dependent on the context in which they are asked; these questions can generally be addressed by standard help systems or mouse-over pop-up text boxes. *Context-sensitive questions* require the system to consider what is currently happening ("task sensitivity") or high-level goals inferred from the user ("user-intent sensitivity").

Of the question types presented, five of them had average scores of 3.0 or higher. Of these five question types, three are context-independent and could be supported in a software system through the use of easily-accessed documentation. The other two top question types, the Interpretive questions (e.g., What is happening now? Why did it happen?) and Guidance questions (e.g., What should I do now?) are context-sensitive, and point to the need for more complex explanation capabilities to address these user needs.

In addition to rating the standard taxonomy of questions, we also asked our users to identify on their own the questions they most wanted to ask.[1] The most common questions our users requested were:

1. What are you doing right now?
2. Why did you do that?
3. When will you be finished?
4. What information sources did you use?

Of the 34 questions mentioned by our participants, 16 of them (47%) were variations on these four questions.

We also note two final observations about the explanation requirements identified by the participants. First, we note that the questions most often identified by the participants before being presented with the taxonomy of question types are entirely context-sensitive. We conclude that the majority of the confusion encountered by the participants cannot be solved with the use of simple help systems or documentation, but rather requires a deeper solution. Second, we were surprised by the reaction of our participants when presented with the question types from the taxonomy. Common reactions included comments like "I would love to ask that!", "That's a cool [question]... I'd use that if it existed!", and "I was asking that [to myself] all day!" The majority of our participants expressed that these were questions that they would definitely want to ask, but it would not generally occur to them to ask the questions because they do not expect systems to be able to provide useful answers.

Theme 3: Being Ignored. Many participants complained about feeling ignored by the agent. After providing the system with personal preferences, as well as suggestions and feedback aimed at improving machine learning, many users were left with the impression that their effort was wasted and that the agent was ignoring them. Users complained that the agent was "not paying attention" during interactions. One user said, "You specify something, and [the system] comes up with something completely different, and you're like, it's ignoring what I want!"

Theme 4: Transparency. When asked what would help them to build trust in the system, the first thing most participants (71%) mentioned was transparency, and every participant

[1] In the study, users provided their free-form questions *before* being presented with the taxonomy.

(100%) mentioned transparency as a major factor affecting overall usability. Participants complained that the system was "too opaque" and "needs to be more comprehensible." Several users noted that the components of the system that they trusted the most were the ones that provided feedback about what they were doing. One user commented that "the ability to check up on the system, ask it questions, get transparency to verify what it is doing, is the number one thing that would make me want to use it." We note as well that transparency is particularly useful in building trust in a system for which a baseline of trust does not already exist through reputation-based methods (for example, through recommendations from other users).

Theme 5: Provenance. Access to knowledge provenance was also mentioned by many participants. Several users reported that explanations of knowledge provenance would enable them to trust results without the need for extensive further verification. One user commented that, "in general, I wanted information about the source," and another user said that "[the system] needs a better way to have a meta-conversation."

We also found, somewhat surprisingly, that providing access to knowledge provenance would increase trust not only in the answers provided by the system, but also in the reasoning of the entire system itself. Users tended not to blame incorrect answers on errors in the data, or even a lack of sufficient data, as was often the case with statistical machine learning components in the system. These "data-driven" errors, however, are often easy to fix, and when errors could be properly identified as being data-driven rather than logic-driven, the users' trust in the system as a whole was better maintained.

Theme 6: Autonomy and Verification. Most participants adopted a "trust but verify" approach. When asked how often they felt that they trusted the system, most participants responded that they trusted the system 25 to 60 percent of the time. Upon further investigation, it became clear that almost all participants actually meant that they *would* trust the system this often, but *only if* they were given mechanisms to verify the responses and override erroneous behavior when necessary. Typical participants said that they trusted the system when it "wasn't too autonomous," when the system performed "with supervision," and when they could "check up on" the system. Participants noted that "trust is an earned property" that the system would only earn when its behavior has been verified.

Guidelines for Adaptive Agents Our research and implementations address issues of user trust in complex adaptive agents. We observe that an explanation system that provides context-sensitive explanations of adaptive agents (for instance, as in [7]) is capable of addressing the themes described above. Our findings show that users have specific requirements in terms of transparency and verification that they expect from such systems before they are willing to trust the outputs and actions of the system. In addition, as these systems become more intelligent, they must increasingly support the ability to have a "dialogue" or "conversation" with users, to provide them with reassurances about the system's understanding of the user and the world.

Several theoretical frameworks for modeling these dialogues have been suggested; Walton [8] compares several of these models and suggests one such model that is particularly useful for modeling explanation dialogues of the type that would address the themes we identified here. In the HCI community, some previous work (for example, [9]) has taken a high-level view of making an intelligent system usable. The themes that we present here build on these broad guidelines, providing more grounded, concrete guidance for designing interfaces for adaptive agents.

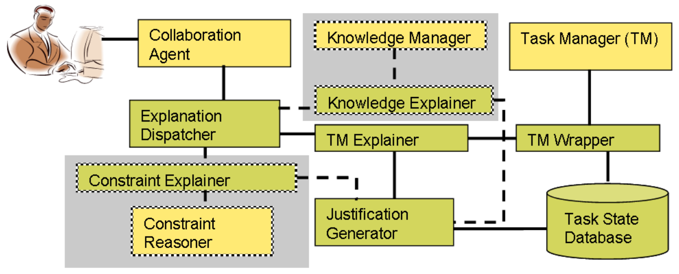


Figure 1: ICEE architecture. Shaded boxes show how additional explanation capabilities (beyond explaining task reasoning) are integrated into the overall framework; these additional components are not discussed in this paper.

4 The ICEE System

We used these guidelines to inform our explanation design and implementation. ICEE focuses on providing explanations of task processing and learning in hybrid adaptive agents and is integrated with the CALO system.

4.1 Architecture Overview

The architecture of ICEE, shown in Figure 1, is designed to be flexible and allow explanations to be derived from justifications gathered seamlessly from a variety of task processing and knowledge systems. An *explanation dispatcher* gathers structured explanation requests from the user through a collaboration agent or user interface. The assistant's user interface provides specialized mechanisms for users to request explanations.

Based on the type of the explanation request, the explanation dispatcher determines which explainer will handle the request, and forwards it to the proper explainer component. For questions related to task processing, the *Task Manager (TM) explainer* handles the request. The TM explainer instructs the TM *wrapper* to gather task processing information about the requested tasks. The TM wrapper is closely coupled with a BDI execution system, or task manager. We have provided a TM wrapper for the task execution system used in CALO, which is based on the SRI Procedural Agent Realization Kit (SPARK); however, any similar task execution system could be similarly enhanced with explanation capabilities.

The TM wrapper stores the gathered task processing information in the *task state database*. This database is then used by the *justification generator* to create a justification for the tasks currently under execution, including any additional processing that is related to the current tasks. The justification can then be used by the TM strategies to create alternative explanations and select the one most salient to the user's questions. The explanation dispatcher returns the selected explanation to the collaboration agent for appropriate display to the user. Each of these architectural components is discussed below.

4.2 Task Oriented Processing

Complex cognitive agents must have a mechanism for representing and executing tasks. A belief-desire-intention (BDI) model [10] is a common framework for task reasoning components. BDI systems cover a range of execution capabilities, including hierarchical task encoding, control of procedural agent behavior, sequential and parallel execution of sub-procedures, conditional execution, branching, flexible preconditions, and meta-level reasoning to determine applicable procedures.

Task management in CALO is provided by SPARK [11], a BDI agent framework, which maintains procedures that define agent actions. The CALO Task knowledge base includes human-authored procedures along with procedures that were learned based on evolving information. ICEE gathers information on both static aspects of procedures within SPARK as well as dynamic information about its past and current execution state.

4.3 Introspective Predicates

ICEE is designed to provide cognitive agent users with the ability to ask detailed questions about task execution and to engage in a dialogue about past, current, and future task execution, as well as task procedure provenance information. To provide explanations of task processing system behavior, justifications are annotated with meta-data. In order to generate detailed explanations, the task execution system must be able to expose this meta-information. One of our contributions is a specification of a set of *introspective predicates* that were designed to provide access to meta-information required for explainable task processors. These introspective predicates fall into three categories:

1. *Basic Procedure Information*: relatively stable, static information that is not dependant on when a task is executed. Provenance information about how task definitions have been created or learned is a key aspect of these introspective predicates.
2. *Execution Information*: dynamic information that is generated as a task begins being executed, and remains valid in some form throughout the execution of that task. This information also includes history related to completed tasks.
3. *Projection Information*: information about future execution, as well as alternatives for decision points that have already passed.

A task execution system that provides access to these introspective predicates can be linked to ICEE and allow it to fully explain all the question types and strategies described above. Details on the introspective predicates can be found in [12].

4.4 Wrapper, Action Schema and Action Database

To collect explanation-relevant information from the task execution agent and store it in a format understandable by the explainer, we designed and built a wrapper for SPARK and an intermediate *action schema* in which to record task information. These elements were designed to achieve three criteria:

- *Salience*. The wrapper should obtain information about an agent's processing that is likely to address some possible user information needs.

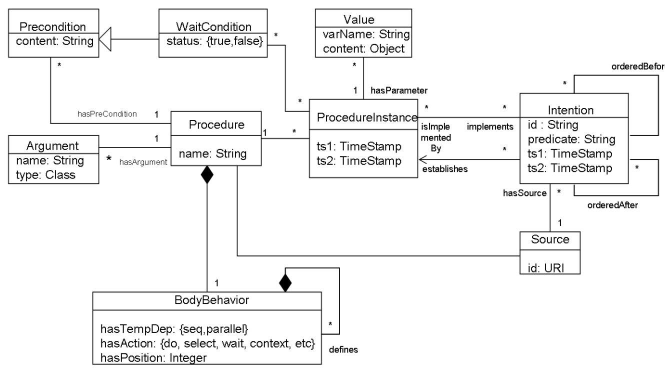


Figure 2: Part of the schema representing execution information.

- *Reusability*. The wrapper should obtain information that is also useful in other cognitive agent activities that require reasoning about action - for example, state estimation and procedure learning.
- *Generality*. The schema should represent action information in as general a way as possible, covering the action reasoning of blackboard systems, production systems, and other agent architectures.

The wrapper collects a snapshot of SPARK's current state as well as the decisions that led to that state. It uses SPARK's expanded introspective predicates to extract the portions of its intention structure relevant to its current intentions, recursively querying for the supporting elements of intentions and procedures. Example queries include: What are the current intentions? What is the procedure instance that led to intention X? What preconditions were met before executing procedure P?

After collecting the snapshot, the wrapper stores it in a SPARK-independent task execution action database. A portion of the representation schema we use is shown in Figure 2. The schema reflects that most task execution systems share the same common structure. While the terminology in our schema is consistent with SPARK's, the concepts are general and consistent with other cognitive architectures. For example, "procedures" in our schema are equivalent to "knowledge sources" in BB* and other blackboard architectures, "procedure instances" are equivalent to "Knowledge Source Activation Records (KSARs)", etc. [13]. The database records the relationships between entities relevant to the agent's current state, for example, which intentions were established by which procedure instances, which procedure a given instance instantiates, and which of a procedure's termination conditions were satisfied and which were not.

We achieve multiple design goals by creating a system-specific wrapper and a generic action schema. When new task execution systems (other than SPARK) need to be explained, the generic action schema is reused and only a new wrapper is needed. No changes are required to the justification representation or the explanation strategies. Also, the action schema can be reused by other cognitive agent components for purposes beyond explanation, such as state capture, archiving, or snapshotting.

4.5 Generating Formal Justifications

A cognitive agent's actions should be supported by justifications that are used to derive and present understandable explanations to end-users. These justifications need to reflect both how the actions support user goals, and how the actions chosen by the agent were guided by the state of the world. More specifically, our approach to task justification breaks down the justification of a question about a particular task T into three complementary strategies, described here using terminology from SPARK:

- *Relevance*: Demonstrate that fulfilling T will further one of the agent's high-level goals, which the user already knows about and accepts
- *Applicability*: Demonstrate that the conditions necessary to start T were met when T started (possibly including the conditions that led T to be preferred over alternatives)
- *Termination*: Demonstrate whether one or more of the conditions necessary to terminate T has not been met.

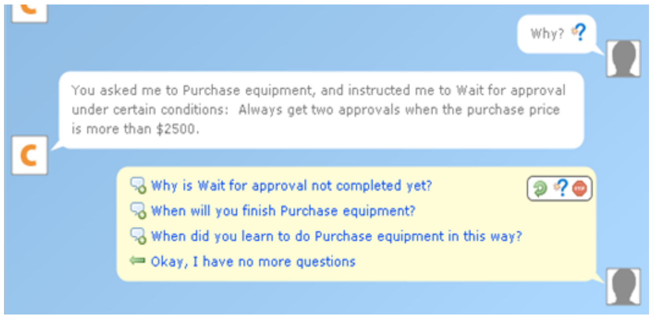
This three-strategy approach contrasts with previous approaches, most of which dealt with explaining inference [14, 15]. Previous approaches generally have not dealt with termination issues, and have not generally distinguished between relevance and applicability conditions. These are critical aspects of task processing and thus are important new issues for explanation.

Justifications can be seen and represented as proofs of how information was manipulated to come to a particular conclusion. We chose to leverage the Inference Web infrastructure [16] for providing explanations. Inference Web was designed to provide a set of components for representing, generating, manipulating, summarizing, searching, and presenting explanations for answers from question answering agents. At Inference Web's core is an Interlingua for representing provenance, justification, and trust encodings called the Proof Markup Language (PML) [17]. PML provides core representational constructs for provenance, information manipulation steps, and trust. Inference Web also provides PML tools for interactive browsing, summarization, validating, and searching [18]. In this work, we expanded the inference web infrastructure and underlying components to provide support for explaining task execution systems and learning.

PML documents contain encodings of behavior justifications using PML *node sets*. An OWL [19] specification of all PML terms is available, which separates out provenance[2], justifications[3], and trust[4]. PML node sets are the main components of OWL documents describing justifications for application answers published on the Web. Each node set represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with a node set. A task execution justification is always a justification of why an agent is executing a given task T. The final conclusion of the justification is a FOL sentence saying that T is currently being executed. There are three antecedents for this final conclusion, corresponding to the three strategies discussed above. Each antecedent is supported by a justification fragment based on additional introspective predicates.

It is important to note that all the task processing justifications share a common structure that is rich enough to encode the provenance information needed to answer the explanation requests identified in the user study. By inspecting the execution state, explanation components can gather enough provenance information to support a wide range of explanations.

[2]http://inference-web.org/2006/06/pml-provenance.owl
[3]http://inference-web.org/2006/06/pml-justification.owl
[4]http://inference-web.org/2006/06/pml-trust.owl


Why?

You asked me to Purchase equipment, and instructed me to Wait for approval under certain conditions: Always get two approvals when the purchase price is more than $2500.

Why is Wait for approval not completed yet?
When will you finish Purchase equipment?
When did you learn to do Purchase equipment in this way?
Okay, I have no more questions

Figure 3: An example explanation, with follow-up questions.

4.6 Producing Explanations

Different users may need different types of explanations. In order to personalize explanations, ICEE uses *explanation strategies*. An explanation strategy provides a method for retrieving provenance and inference information from justifications, selecting the information relevant to the request, and presenting the information to the user. The feedback from our user study motivated our choice of supported explanation and follow-up strategies.

User modeling and strategy selection are handled by the explanation dispatcher. Currently, user modeling is restricted to user preferences. Additional approaches based on user interaction and machine learning techniques are under investigation. The explanation strategies are closely tied to the explanation request types discussed above. Example strategies include revealing task hierarchy, exposing preconditions or termination conditions, revealing meta-information about task dependencies, or explaining provenance information related to task preconditions, task learning, or other task knowledge.

Each individual user may also desire different explanations in different situations. ICEE provides context-dependent follow-up questions for the user. Follow-up questions might include requests for additional detail, clarifying questions about the explanation that has been provided, or questions essentially requesting that an alternate strategy be used to answer the original question. The user's familiarity with the particular task, current level of confusion, cognitive load, or other factors may influence which follow-up question(s) they choose in different contexts. Figure 3 shows an example user interface linked to ICEE. The user has requested an explanation of the motivation for a subtask of an executing task, and an explanation is provided along with three suggested follow-up questions.

5 Related Work and Discussion

Though there is increasing interest in building adaptive systems, and much has been written about the components that comprise these systems, little work has been done to evaluate their user acceptance. Cortellessa & Cesta [20] discuss this lack of research on what they call the "quality of interaction," and provide results of an initial study focused on user trust and the use of explanation in mixed-initiative planning systems. In this particular domain, they found that the use of explanations was highly correlated with the number of failures experienced by users.

Bunt *et al.* [21] provides for, but does not evaluate, a mechanism for simple explanations aimed at maintaining transparency. Our study extends that work to understand how such an explanation mechanism can influence user trust in a broader range of adaptive systems. Additionally, user studies focused solely on understanding machine learning [22, 23] have looked at how explanations can increase acceptance and usability of these learning algorithms in isolation, by testing user understanding of a variety of learning algorithms when explanations are available.

There has been an abundance of work in explaining expert systems and, to a lesser extent, explaining automated reasoning systems. Most of these have focused on some notion of explaining the deductive trace of declarative rules. Previous work on Inference Web and related work explaining hybrid reasoners added to the field by focusing on settings that are web-based or distributed. Our current work further expands the coverage to support explanations of task executions in the presence of learned knowledge, declarative rule processing, and provenance.

The literature on automated explanation research that explicitly addresses explaining actions is sparse. [24] presents a module to explain a Soar agent's actions by reconstructing the context in which action decisions were made, and then tweaking that context (hypothetically) to discover the elements that were critical to the decision. While Johnson's approach does have some similarities with the way we handle gating conditions, it does not deal with relevance and termination strategies that are important to our agent explanation module. Earlier, Schulman & Hayes-Roth [25] developed a BB1 module that explains actions using the architecture's control plan, but it does not address explaining when the control plan does not exist, as is the case in CALO and most other intelligent architectures. Work on plan description [26, 27, 28] has focused on summarizing an agent's aggregate behavior, rather than justifying individual task choices. Our work thus fills an important gap in explaining agent actions, providing fine-grained explanations of a wide range of agent activities, taking into account more aspects of BDI agent architectures, while using an approach that is compatible with explaining hybrid reasoning components, such as standard FOL reasoners. One promising area of future work would be to allow the user to switch between coarse-grained and fine-grained explanations, combining our work with the previous approach.

Driven by the needs of explaining cognitive assistants, we have focused on explanation infrastructures that can work with task execution systems as well as with deductive reasoners and learning components. We have described current progress on designing and implementing an extensible and flexible architecture that is capable of explaining the breadth required by cognitive assistants, with requirements gathered through a study of trust in one sample adaptive agent. Our architecture and implementation demonstrates that an explanation mechanism initially designed for explaining deductive reasoning can also be successfully applied to explaining task-oriented reasoning. Additionally, work devoted simply to explaining task execution has not traditionally focused on explaining a broader setting including deduction and provenance. We developed a framework for extracting and representing the state and history of a task system's reasoning, a framework that is already proving to be useful for building trust and explanation services in other agent activities.

ICEE includes initial implementations of all of the described components, and is seeded with a limited number of strategies, as prioritized by our trust study. Current work includes expanding these components to provide advanced functionality; additional capabilities for handling the results of procedure learning (from instruction and demonstration) that extend and/or update the procedures that the task execution system uses; as well as a strong focus on explaining conflicts, explaining failures, and further explaining knowledge provenance, including statistical machine learning algorithms.

6 Conclusions

We have studied issues governing the trust and usability of complex adaptive agents. Without trust in the actions and results produced by these agents, they will not be used and widely adopted as assistants and partners. By interviewing users of these agents, we have identified several themes that describe the willingness of users to adopt and trust these agents.

These recommendations are demonstrated in ICEE, our explanation infrastructure. While we focused here on explaining the task processing component, our solution provides a uniform way of encoding task execution, deductive reasoning, and learning components as justifications. The approach is integrated with the IW infrastructure for supporting knowledge provenance so that these explanations may be augmented with source information, improving end-user understanding. The resulting work provides not only an explanation infrastructure for a particular adaptive agent (CALO), but also provides a platform for explaining other hybrid adaptive agents that must explain task learning, deductive and non-deductive inference, and provenance.[5]

References

[1] P. Pinheiro da Silva, D. McGuinness, and R. McCool. Knowledge provenance infrastructure. *IEEE Data Engineering Bulletin*, 26(4):26–32, 2003.

[2] http://www.darpa.mil/ipto/programs/pal/.

[3] http://www.ai.sri.com/project/CALO.

[4] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2), 2007.

[5] A. Glass, D. McGuinness, and M. Wolverton. Toward establishing trust in adaptive agents. In *International Conference on Intelligent User Interfaces*, 2008.

[6] M. Silveira, C. de Souza, and S. Barbosa. Semiotic engineering contributions for designing online help systems. In *SIGDOC'01*, 2001.

[7] D. McGuinness, A. Glass, M. Wolverton, and P. Pinheiro da Silva. A categorization of explanation questions for task processing systems. In *AAAI Workshop on Explanation-Aware Computing*, 2007.

[8] D. Walton. Dialogical models of explanation. In *AAAI Workshop on Explanation-Aware Computing*, 2007.

[9] K. Höök. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12(4), 2000.

[10] A. Rao and M. Georgeff. BDI agents: From theory to practice. In *International Conference on Multiagent Systems*, 1995.

[5]This article combines and updates material from papers in the Proceedings of the 20th International FLAIRS Conference [29] and the Proceedings of IUI'08 [5].

[11] D. Morley and K. Myers. The SPARK agent framework. In *AAMAS-04*, 2004.

[12] A. Glass and D. McGuinness. Introspective predicates for explaining task execution in CALO. Technical Report KSL-06-04, Knowledge Systems, AI Lab., Stanford Univ, 2006.

[13] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.

[14] A. Scott, W. Clancey, R. Davis, and E. Shortliffe. Methods for generating explanations. In *Rule-Based Expert Systems*. Addison-Wesley, 1984.

[15] M. Wick and W. Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54(1–2):33–70, 1992.

[16] D. McGuinness and P. Pinheiro da Silva. Explaining answers from the semantic web: The inference web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.

[17] P. Pinheiro da Silva, D. McGuinness, and R. Fikes. A proof markup language for semantic web services. *Information Systems*, 31(4–5):381–395, 2006.

[18] D. McGuinness, L. Ding, A. Glass, C. Chang, H. Zeng, and V. Furtado. Explanation interfaces for the semantic web: Issues and models. In *International Semantic Web User Interaction Workshop*, 2006.

[19] D. McGuinness and F. van Harmelen. Owl web ontology language overview. Technical report, World Wide Web Consortium (W3C), February 2004. Recommendation.

[20] G. Cortellessa and A. Cesta. Evaluating mixed-initiative systems: An experimental approach. In *ICAPS-06*, 2006.

[21] A. Bunt, C. Conati, and J. McGrenere. Supporting interface customization using a mixed-initiative approach. In *International Conference on Intelligent User Interfaces*, 2007.

[22] M. Pazzani. Representation of electronic mail filtering profiles: A user study. In *International Conference on Intelligent User Interfaces*, 2000.

[23] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker. Toward harnessing user feedback for machine learning. In *International Conference on Intelligent User Interfaces*, 2007.

[24] W. Johnson. Agents that explain their own actions. In *Conference on Computer Generated Forces and Behavioral Representation*, 1994.

[25] R. Schulman and B. Hayes-Roth. Plan-based construction of strategic explanations. Technical Report KSL-88-23, Knowledge Systems Lab., Stanford Univ, 1988.

[26] C. Mellish and R. Evans. Natural language generation from plans. *Computational Linguistics*, 15(4), 1989.

[27] R. Young. Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence*, 115(2), 1999.

[28] K. Myers. Metatheoretic plan summarization and comparison. In *International Conference on Automated Planning and Scheduling*, 2006.

[29] D. McGuinness, A. Glass, M. Wolverton, and P. Pinheiro da Silva. Explaining task processing in cognitive assistants that learn. In *20th International FLAIRS Conference*, 2007.

Contact

Alyssa Glass
Stanford University
Email: glass@cs.stanford.edu

Deborah L. McGuinness
Rensselaer Polytechnic Institute
Email: dlm@cs.rpi.edu

Paulo Pinheiro da Silva
University of Texas at El Paso
Email: paulo@utep.edu

Michael Wolverton
SRI International
Email: mjw@ai.sri.com

Alyssa Glass is completing her Ph.D. in Computer Science at Stanford University, and is a computer scientist in the Artificial Intelligence Center at SRI International. Her research involves explaining task execution and machine learning in adaptive agents, to make learning-based systems more trustworthy for users. She was previously a computer scientist at Xerox PARC. She received her Bachelor's degree in computer science and economics from Harvard University.

Deborah L. McGuinness is the Tetherless World Senior Constellation Chair and Professor of Computer Science and Cognitive Science at Rensselaer Polytechnic Institute. Her research focuses on the semantic web, ontologies, explanation, trust, and semantic eScience. Until recently, Deborah led the Knowledge Systems Lab at Stanford University. She received her B.S. from Duke Univ., M.S. from the Univ. of California at Berkeley, and her Ph.D. from Rutgers Univ.

Paulo Pinheiro da Silva is an Assistant Professor of Computer Science and leader of the Trust Laboratory at the Univ. of Texas at El Paso, and a member of the Cyber-ShARE Center of Excellence for Cyber-Infrastructure Applications. Paulo received his B.S. and M.Sc. from the Federal Univ. of Minas Gerais, Brazil, and his Ph.D. in computer science from Manchester Univ. He is a former Postdoctoral Fellow of the Knowledge Systems Laboratory at Stanford Univ.

Michael Wolverton is a Senior Computer Scientist in SRI's Artificial Intelligence Laboratory. There he has led research efforts in a wide range of areas, including link analysis, explanation, information management, planning, case-based reasoning, and analogy. He received his Ph.D. in Computer Science from Stanford University, and his bachelor's degree in Computer Science and Mathematical Sciences from Rice University.