Introspective Self-Explanation in Analytical Agents

Anita Raja[1] and Ashok Goel[2]

[1] Department of Software and Information Systems, The University of North Carolina at Charlotte, Charlotte, NC 28223, anraja@uncc.edu
[2] AI Laboratory, School of Interactive Computing, Georgia Institute of Technology, Atlanta, Georgia 30308, goel@cc.gatech.edu

Abstract. There is a critical and urgent need for automated analytical agents operating in complex domains to provide meta-level explanations of their reasoning and conclusions. In this paper, we identify the principles for designing analytical agents that can explain their reasoning and justify their conclusions at different levels of abstractions to potential human customers with varying goals. We also analyze the goals of users of an automated agent for investigative analysis along the dimensions of why, what, how and when, and develop a taxonomy of human goals that will leverage the explanations generated by the agent.

1 Introduction

Automated agents are increasingly used to assist human decision-making in various domains ranging from engineering design to medical diagnosis to economic prediction. In general, the principles of designing these automated agents are based on factors like the quality of decision-making, the efficiency of computation, and the value of information. However, in many complex domains, there is a critical and urgent need for explanations of the reasoning and conclusions of the automated agents. Equipping automated agents with the ability to generate meta-explanations of their behavior will enable them to be self-introspective of their end-to-end problem solving process. In the domain of investigative analysis, for example, an automated analytical agent must not only collect data from a variety of data sources and generate hypotheses about a set of data, but it also must explain its reasoning and conclusions to a range of potential human users. This requirement of generating explanations imposes a new set of requirements on the design of automated agents for investigative analysis.

In this paper, we describe an automated analytical agent and identify the principles of designing it such that can explain their reasoning and justify their conclusions to potential human customers. The desiderata for explanation generation [1] are that: (1) the explanation must be generated autonomously, without any human intervention; (2) the contents of the explanation in general may include justification for the conclusions, explanations of the decision-making process, and justification of the decision-making knowledge. The precise contents

of the explanation will depend on the human consumer and his/her goals. (3) The level of abstraction of the explanation must be tunable to different goals of various human consumers. Thus the focus of this work is on the capturing of and providing access to explanations at multiple levels of abstraction. This requires intelligent integration of many pieces of information as described in this paper.

The current state of automated agent design is such that these agents in general do not know what they are doing, or why or how they are doing it. Thus, the explanations and justifications are inherently limited and typically an afterthought in the design of these systems. We propose that the generation of meaningful explanations and justifications requires introspection by the agent over its own reasoning and knowledge. The design requirement of introspection in turn requires an explicit representation of the tasks the agent addresses, the methods it uses to address them, and the knowledge used and created by the methods.

Thus, the primary hypothesis of this paper is that self-explanations of the decision-making process are enabled by introspection over a task structure (called introspective task structure) containing meta-knowledge about the decision making. Introspection pertains to the ability to inspect and capture the invocation of tasks and methods in this task structure. We propose that introspection over the task structure, which is a recursive decomposition at multiple levels of abstraction, enables generation of explanations at multiple levels of abstraction. Further, since introspection is an autonomous process, it enables automated generation of self-explanations.

The above hypothesis originates from earlier work on knowledge systems. MYCIN [2] was an early production system for diagnosis in the domain of E. Coli bacteria. When an explanatory interface called GUDION [3] was added to MYCIN for training medical students, it was found that explanations of MYCIN's decision making in terms of chains of activated production rules were at too low a level of abstraction for human consumption. NEOMYCIN was a reimplementation of MYCIN in terms of the task structure of diagnostic decision-making, and it provided explanations in terms of tasks rather than rules [4]. Since NEOMYCIN there has been significant amount of new work on explanation in intelligent systems. For example, Tanner et al. [5] used knowledge of task structures and domain models to generate explanations of different kinds. Sormo et. al [6] and Cox [7] describe the role of explanation generation in model-based and case-based reasoning systems. Murdock and Goel [8] used knowledge of task structures and domain models to interleave explanations of decision making and justifications of solutions. Languages for specifying task structures of complex decision making include TAEMS [9] and TMKL [10].

The remainder of this paper is organized as follows: In Section 2, we describe a mixed-initiative analytical agent for investigative analysis capable of self-introspection and providing different types of explanations for its actions. In Section 3, we present our conclusions and future work.

2 An Automated Analysis Agent

Threat perception in intelligene analysis [11, 12] generally involves the tasks of recognizing and characterizing a problem based on some initial evidence about an event or activity; generating multiple explanatory hypotheses based on the evidence; collecting and assimilating additional data; evaluating the multiple explanatory hypotheses; and selecting the most plausible hypothesis. This analytical task is very complex because of the constantly evolving, and often unreliable and conflicting nature, of the data. The evolving nature of data implies a need for ongoing monitoring and continual generation and evaluation of hypotheses so that new evidence can be accounted for as it arrives and the most likely explanation can be produced at any given time. Further, the analytical task is hard not only because of the need to assimilate new information that fits known patterns but also because of the need to discover novel patterns. The need to discover new patterns implies construction of explanations that can not only account for the current data but also make predictions and generate expectations so that violation of an expectation can signal a novel situation. In this section we describe the Analytical Agent (AA) capable of mixed-initiative reasoning that will assist investigative analysts to choose from and reason about enormous databases of text, imagery, video and web cast.

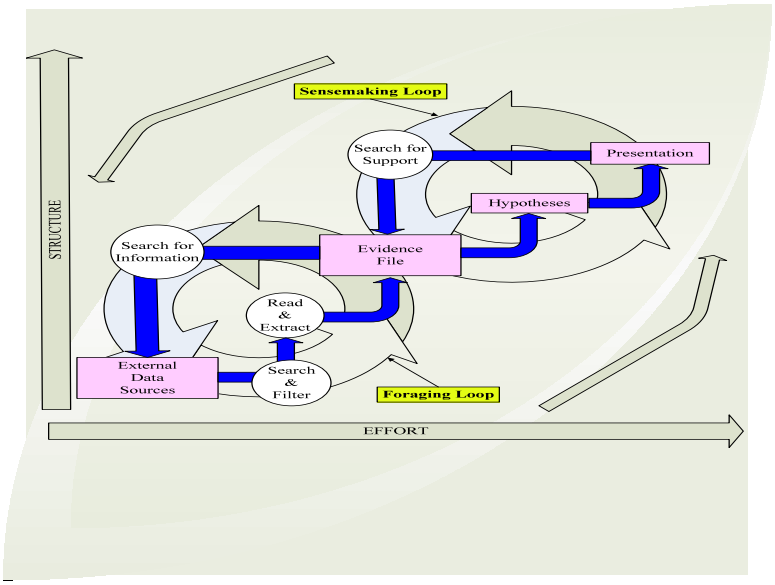

Fig. 1. Pirolli and Card's Sensemaking and Foraging Loops

Pirolli and Card [13] describe a model they developed by observing the cognitive task analysis performed by intelligence analysts as they did their jobs. They

have identified two main, overlapping loops in the analyst's problem solving approach, a foraging loop and a sensemaking loop. Figure 1 describes this process. The foraging loop involves finding the right data sources; searching and filtering the information; and extracting the information. The sensemaking loop involves iterative development of a conceptualization (hypothesis) from the schema that best fits the evidence and the presentation of the knowledge product that results from this conceptualization. We have designed AA for investigative analysis to contain both the Sensemaking and Foraging loops as suggested by Pirolli and Card. We describe STAB [14], the Sensemaking component in Section 2.1 and TIBOR, the Foraging component in Section 2.2.

As mentioned earlier, explanation generation in an analytical agent must be autonomous; should include justifications and explanations of the conclusions and decision making process and knowledge; and be tunable to the different goals of various human customers. In general, explanations can be of two types [15]: (1) abductive explanations, and (2) self-explanations. An abductive hypothesis provides a (best) explanation for a set of data. In investigative analysis, for example, an abductive hypothesis may explain how a set of seemingly unrelated events form a pattern of activity. An agent's self-explanation describes the agent's reasoning in reaching a conclusion. For example, in investigative analysis, an automated assistant may provide a description of its decision-making process.

A self-explanation in general may have three components [1]: (1) justification of conclusions; (2) explanation of the decision-making process; and (3) justification of the decision-making knowledge. In investigative analysis, for example, a conclusion about a specific pattern of activity may be justified by the evidence for and against it, the decision-making process may be explained in terms of the steps of the process, and the decision-making knowledge may be justified in terms of past cases of investigative analysis.

The decision-making process can be explained in many ways, each with its own benefits and drawbacks. Two commonly used methods are "audit trails" [16] and "design rationales" [17,18]. In financial decision-making, the decision-making may be explained by capturing "an audit trail" of the process. This method has the benefit of completeness but the disadvantage of a single and too low-level of an abstraction. In long-term, large-scale team design, decision-making may be explained in terms of issues, positions, and arguments underlying each decision. This has the benefit of tunable levels of abstraction but the disadvantage that it requires a human to capture the rationale for each decision.

Goal-Directed Explanations

Our first step towards designing introspective analytical agents is to analyze the goals of human customers of an automated agent for investigative analysis along the dimensions of why, what, how and when, and develop a taxonomy of human goals for reading explanations generated by the agent. This analysis will be based on interviews of investigative analysts. For each element in this taxonomy, we

Abstraction↓/Analyst→	A	OA	HASO	OASO	HADO
Conclusion(s)	X	X	X	X	X
Confidence Values of Conclusions	X	X	X	X	
Justifications for Conclusions	X	X	X	X	X
Alternate hypotheses/justifications	X	X	X		X
Analyst's Assumptions and Biases	X	X	X	X	X
High-Level Explanation	X	X	X	X	
Mid-Level Explanation	X	X		X	
Low-Level Explanation	X	X			
Supporting Raw data	X				
Justifications for Domain Knowledge	X				

Table 1. Decision Process Explanation Matrix A: Analyst; OA: Other Analysts; HASO: Higher Authority, Same Organization; OASO: Other Analysts, Same Organization; HADO: Higher Authority, Different Organizations

will identify the information that the explanation should contain, and develop techniques for generating, aggregating and abstracting the required information.

Table 1 presents our preliminary classification of human customers (the columns in the table) and contents of the explanation generated for the various customers (rows). The human customers may range from the analyst using our AA agent, to other agents in the same organization, to higher authorities in the same organization, to analysts in different organizations, to high authorities in different organizations. The contents of an explanation may range from the conclusions reached by the AA agent, to confidence values for each of the conclusions, to justifications, i.e., the evidence for and against, for each of the conclusions, to alternative hypotheses considered by AA, to AA's assumptions and biases (which reflect the assumptions and biases of the analyst who engineers its knowledge), to explanations of the decision-making process, to supporting raw data for the conclusions, to justifications for the domain knowledge used by decision-making process. Note that the decision-making process itself may be articulated at multiple levels of aggregation and abstraction such as high, medium or low.

Table 1 also presents our preliminary estimates of a explanation's content for different human customers. Thus, the human analyst using the AA agent may want all the details, indicating by the marks in all the boxes in the corresponding column. In general, however, as the explanation is shared with higher decision-making authorities and outside the analyst's organization, the explanation's content would be less detailed and more abstract.

Introspective Self-Explanations

The primary hypothesis of this paper is that self-explanations of the decision-making process are enabled by introspection over the task structure which captures the decision making process, also called the introspective task structure.

So the next step towards building introspective agents is to construct these task structures. The Sensemaking and Foraging components of the AA agent each have an introspective task structure and we construct a third introspective task structure for meta-AA, a component that reasons about and generates explanations for AA's overall problem solving process. In the discussion that follows, we describe STAB, TIBOR and meta-AA; and their respective introspective task structures,

2.1 The Sensemaking Component

Studies of investigative analysis [11] suggest that human analysts typically view the task of threat perception as that of (1) generating abductive explanations for a set of data, and (2) using the explanations to make verifiable predictions. These studies also indicate that analysts use multiple kinds of knowledge including past cases, prototypes and patterns. Psychological studies of intelligence analysts [12] further indicate the three main errors made by human analysts in explanation generation. Firstly, due to limitations of human memory, investigative analysts often have difficulty keeping track of multiple explanations for a set of data over a long period of time. Secondly, analysts often quickly decide on a single explanation for the data set and stick to it as new data arrives. Thirdly, analysts often look for data that supports the explanation on which they are fixated, and not necessarily the data that may refute their explanation. The goal of the sensemaking component is to help address these three limitations.

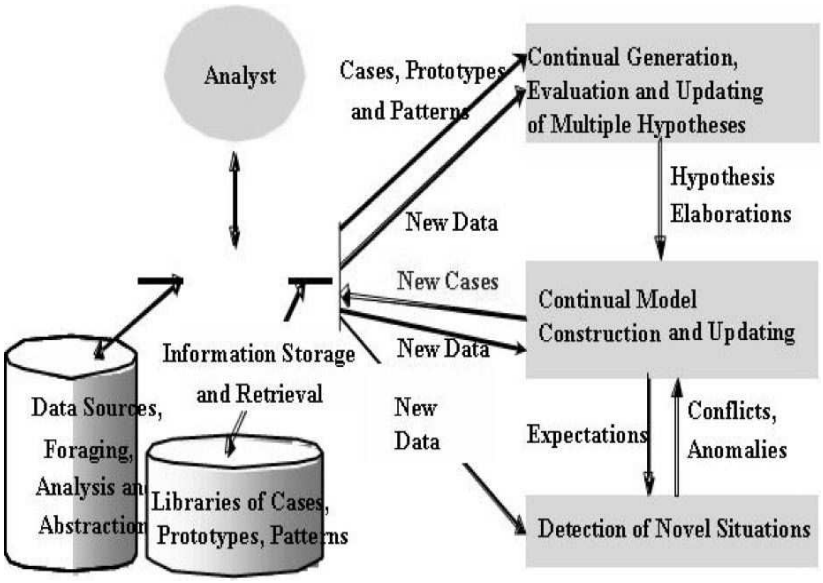

Fig. 2. Functional Architecture of Sensemaking Component

We view the task of threat perception as that of abducing a story whose plot explains the current data and makes verifiable predictions about both the future and the past. The story is composed out of multiple elementary hypotheses that explain different portions of the data. The elementary hypotheses originate from different kinds of knowledge of threats including cases, prototypes and patterns. Figure 2 illustrates a high-level functional architecture for the sensemaking component. The continual data foraging, analysis and abstraction loop (described in the next subsection) is indicated at the bottom-left of the figure. In the sensemaking loop, abstractions of the current data in the present situation are used to probe a library of past cases, prototypes and patterns of threats. The cases, prototypes and patterns that match the current data in the present situation are retrieved and invoked, thus generating multiple elementary hypotheses. The elementary hypotheses, which explain different portions of the data, are composed into multiple stories that explain much of the current data.

The sensemaking and the data foraging components work together to (i) generate, track and evaluate multiple elementary hypotheses; (ii) continually construct stories to accommodate the constantly evolving data; (iii) make and verify predictions about the future and the past; (iv) detect novel situations. The violation of a prediction (a conflict or an anomaly) signals a novel situation. Arguments for each story are generated by keeping track of the evidence in favor and against the story. Confidence values for stories are based on (a) coverage of the data; (b) evidence in favor of the story; (c) evidence against the story; and (d) parsimony.

The stories generated for a set of data, along with the argument structures and confidence values, are presented to the human analyst, who ultimately decides whether to accept a story, and, if so, which one. The accepted story is encapsulated as a new case and stored in the case memory. Thus, the case library is dynamic. The current version of the sensemaking component has been instantiated in a computer system called STAB (for STory ABduction).

At present, STAB contains knowledge only of patterns of threats. It analyzes the VAST data set generated by the Pacific National Labs (PNNL). This dataset pertains to normal and typical activities, as well as illegal and unethical activities, in a fictitious town in the United States. It contains over a thousand news stories, and a score of tables, maps and photographs. It is said to capture many of the ambiguities and subtleties encountered in real investigative data. The VAST dataset is input into STAB as a data stream containing predicates representing objects, relations, and events in the chronological order in which they are described in the news stories. Some of the apparently isolated events in this data stream form patterns of illegal/unethical activities. STAB outputs multiple abductive hypotheses that capture these illegal/unethical patterns of activity and explain the causal and intentional relationships among events comprising an activity.

The generic patterns of illegal/unethical activities in STAB are represented in a knowledge representation language called TMKL, which, for this task, has the expressive power of Hierarchical Task Networks. Each pattern is organized in

task-method-subtask hierarchy, where more than one method may be applicable to a task in the hierarchy.

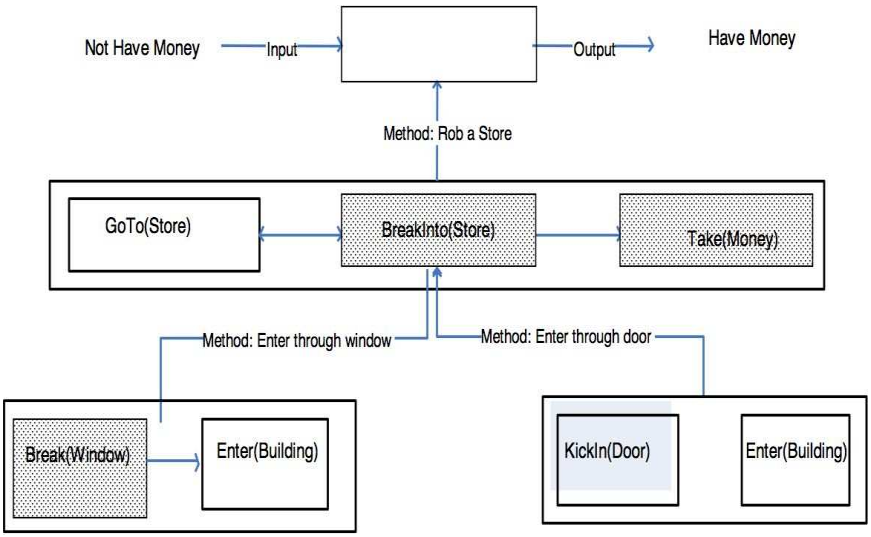

Fig. 3. Activity Pattern for "Rob a Store"

As events in the input data stream arrive incrementally, STAB matches the events with the task nodes in the patterns. This matching is done using feature vectors. Figure 3 and Figure 4 illustrate the activity pattern for *Rob a Store* and *Destroy Property*. Suppose the first input events are Break(Window) and Take(Money). The matching task nodes in the two patterns are shown in dotted boxes. Note that BreakInto(Store) provides the intentional context for the event Break(Window).

STAB invokes the patterns whose task nodes match the first input event Break(Window) and considers them as candidate hypotheses for explaining the data. In particular, it uses these explanatory hypotheses to generate expectations about additional data not yet seen by STAB. Thus, the hypothesis of *Rob a Store* generates the expectation of Take(Money) and *Destroy Property* generates the expectation of Take(Null). As additional data in the form of event Take(Money) arrives as input to STAB, the system matches the data with the expectations generated by the candidate hypotheses. This may lead to abandonment of some hypotheses. For example, in the current scenario, the event Take(Money) results in the refutation of the *Destroy Property* hypothesis . The new data may also lead to the acceptance of some candidate hypotheses with some confidence value. Again in the current scenario, the event Take(Money) results in the acceptance of the *Rob a Store* hypothesis with a confidence value that measures how many of the task nodes in a task-method subtree of the hypothesis were matched by the input data. The confidence value of a story plot depends in part on verification

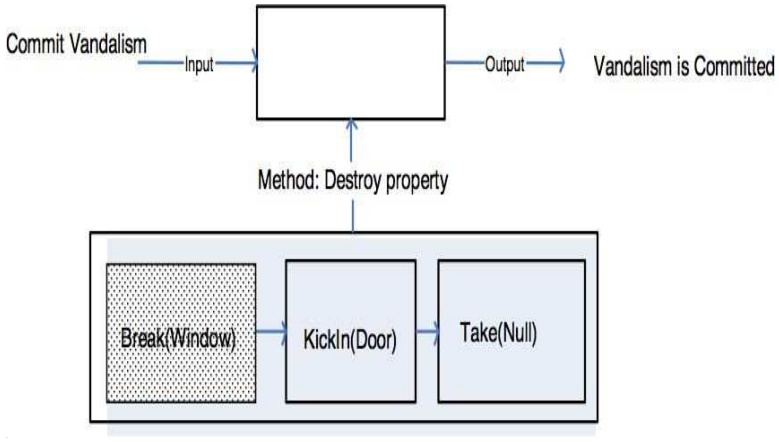

Fig. 4. Activity Pattern for "Destroy Property"

of expectations generated by it: the confidence value increases if the expectation is met by evidence, and decreased if is not. The *Rob a Store* plot generates an expectation of taking money and *Destroy Property* generates an expectation of not taking money. In the current scenario, the expectation generated by the *Rob a Store* is met and hence its confidence value is higher than that of *Destroy Property*. In this way, STAB addresses all three main limitations mentioned in the introduction of this section: limited memory, cognitive fixation, and confirmation bias. STAB currently contains patterns of all illegal/unethical activities in the VAST dataset.

While STAB generates explanatory hypotheses for the VAST dataset, it itself does not know what it is doing, or why or how it is doing it. We envision a Meta-STAB component that encodes the introspective task structure of STAB in the TMKL knowledge-representation language. Figure 5 illustrates this encoding for a small portion of STAB. The rectangles in the figure represent tasks; thus, the highest level task is Generate Explanations. The rectangles expanded into ovals represent methods used by STAB; thus the Pattern-Match Method addresses the task of Generate Hypotheses. This Pattern-Match Method decomposes the Generate Hypotheses task into two simple subtasks: Feature Vector Matching and Retrieval from Library. The transition machine for the Pattern-Match Method depicted in the oval represents the control of processing of the sub tasks. In general, the control of subtasks need not be linear; similarly, in general, multiple methods may be available for addressing a particular task. The Retrieve from Library Task is directly encoded in STAB; a primitive, or directly encoded, task may use both domain knowledge, K, and procedure, P. In this way, the TMKL model of STAB in Meta-STAB explicitly captures the entire introspective task structure of STAB including the relationship between specific tasks and the domain knowledge. TMKL is more expressive than the HTN (Hierarchical Task Networks) language because it provides constructs for capturing the

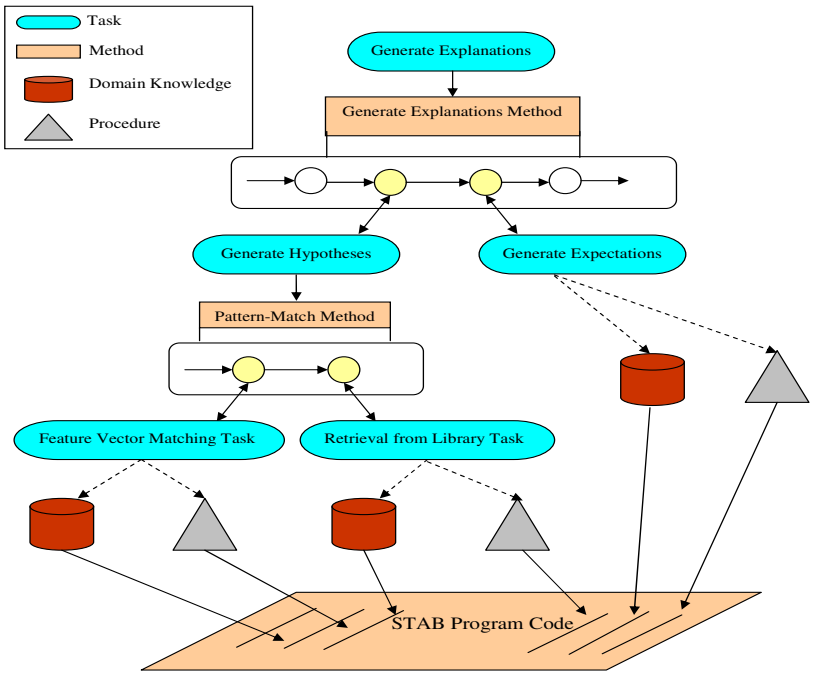

Fig. 5. Meta-STAB's Introspective Task Structure

relationships not only between tasks and methods, but also between primitive tasks and domain knowledge. When input data arrives, Meta-STAB executes its introspective task structure, dynamically selecting and invoking tasks and methods up to the level of primitive tasks. This selection and invocation of tasks and methods depends on the knowledge conditions generated by the preceding tasks. The primitive tasks in turn execute the corresponding code in STAB. Meta-STAB keeps track of the trace of processing in the vocabulary of tasks and methods. This scheme satisfies all three desiderata for explanation generation. STAB itself generates the explanatory hypotheses and the justifications for them. Meta-STAB generates explanations of the decision-making process in the task-method language. Since TMKL directly captures the relationship between tasks and knowledge, Meta-STAB also justifies its decisions by relating them to its domain knowledge. To the degree to which patterns are generalizations of prototypes and prototypes are generalizations of cases, Meta-STAB can also justify one of kind knowledge by pointing to the deeper kind. Further, since the decision-making process can be explained to different levels of depth in the introspective task structure of STAB, the level of abstraction of the explanations is tunable to different goals of various human consumers. Finally, the explanations and the justifications are generated autonomously.

2.2 The Foraging Component

The TIme BOunded Reasoning (TIBOR) component handles the foraging analysis. TIBOR leverages an AI blackboard system [19] and resource-bounded control mechanisms to support hypothesis tracking and validation in a highly uncertain environment like the analysis domain. Blackboard-based architectures have been previously used for complex information gathering and analysis tasks. Morrison and Cohen [20] use a Bayesian blackboard called AIID to serve as a prototype system for analysis and data fusion. BIG [21] is a resource-Bounded Information Gathering agent that combines several sophisticated AI components. TIBOR differs from these architectures in that it focuses on the end-to-end decision reasoning of analytical tasks and uses a sequential decision process approach to reason about the inherent uncertainty of the domain. TIBOR is designed to be a fully-functional mixed-initiative component and handles three types of decisions: gathering of large scale, high dimensional data from a variety of sources, which might be linked multimedia data as found on the web, broadcast video, time-dependent transactional data, telecommunication data, or other types of data; determining the type of processing to extract the data from these sources; and determining appropriate interactive visualization of these data.

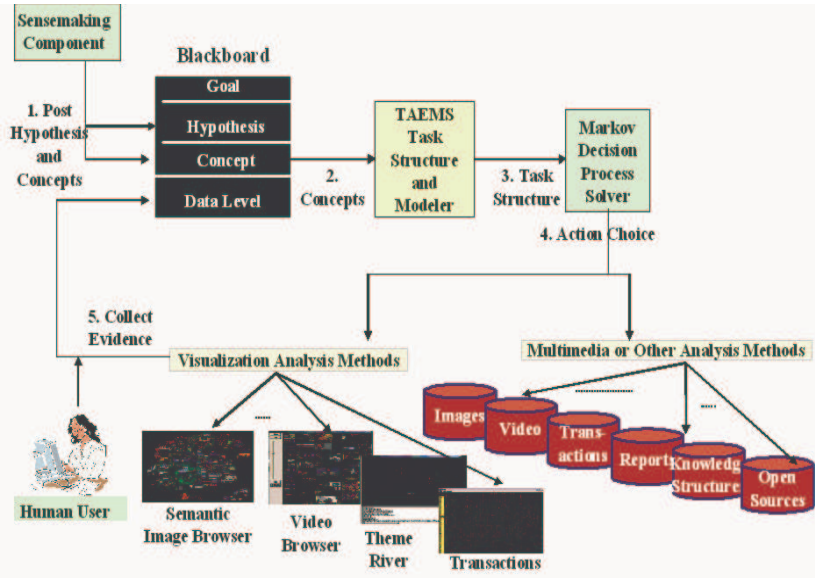

Fig. 6. Functional Architecture of Foraging Component

Figure 6 describes the architecture and control flow in TIBOR. The sensemaking component posts a set of hypotheses and concepts to the blackboard (Step 1 in Figure 6) and this triggers the TIBOR component. The concepts are

searchable entities that represent a hypothesis. To support reasoning about time and quality trade-offs, and thus a range of different resource/solution paths, TIBOR contains problem-solving models called Taems [9] task structures. Taems task structures are abstractions of the low-level execution model and these structures are generated by the Task Structure Modeler sub-component. The Task Structure Modeler takes the concepts as input (Step 2) and generates the corresponding Taems task structure (Step 3) that enumerates multiple different ways (choices for databases, visualization tools and user interactions) to obtain evidence to verify the set of concepts. These different choices are described statistically in the task structure in two dimensions, duration and quality via discrete probability distributions. We provide a detailed description of Taems in the discussion that follows. The Taems task structure associated with the concepts is then translated [22, 23] into a Markov Decision Process (MDP) [24] in the control sub-component. The MDP Solver computes the optimal policy for the MDP given the resource (deadline) constraints and uncertainty in the environment and prescribes the best action (Step 4).

User interactions play an important role in the foraging analysis making this a mixed-initiative system. The analyst must be given the ability to manually direct a search or override actions suggested by the MDP control mechanism, in order for this automated agent to be accepted by the analyst community. The contingency plans built into the MDP policy will allow the control system to adjust dynamically to such overrides. The automatic processing of the visualization tools along with the user interactions will determine the confidence in the concepts (Step 5). These evidential data as well their confidence values are then posted on the blackboard. The blackboard will then propagate the confidence values to verify the associated hypothesis. It is crucial for TIBOR to support both opportunistic and planned verification of hypotheses and concepts. It is probable that while gathering information to verify a concept supporting one hypothesis, the belief for a competing hypothesis increases. The control sub-component will model this possibility and then opportunistically redefine the problem solving process to gather evidence to verify the competing hypothesis. Liu et al. [25] provides a detailed discussion of role of the blackboard in hypothesis tracking and the interactions with the visualization components.

At present we have completed implementation of the MDP-based control sub-component. When TIBOR receives a hypothesis, the Task structure modeler component generates a Taems structure. Figure 7 describes a simple example Taems task structure for the hypothesis Rob_A_Store which has to be validated. The top-level task, Verify Rob_A_Store is decomposed into two searchable subtasks/concepts, Break_Window and Take_Money. In order for the Verify Rob_A_Store task to achieve quality, both Break_Window and Take_Money should get non-zero qualities which are denoted by the q_min quality attribution factor (qaf). Each concept X has two subtasks, the first subtask Visualize_X_Concept will determine which database and visualization tool to use; and the UI_X_Concept subtask will determine the quality obtained towards verifying the concept as a result of user interaction.

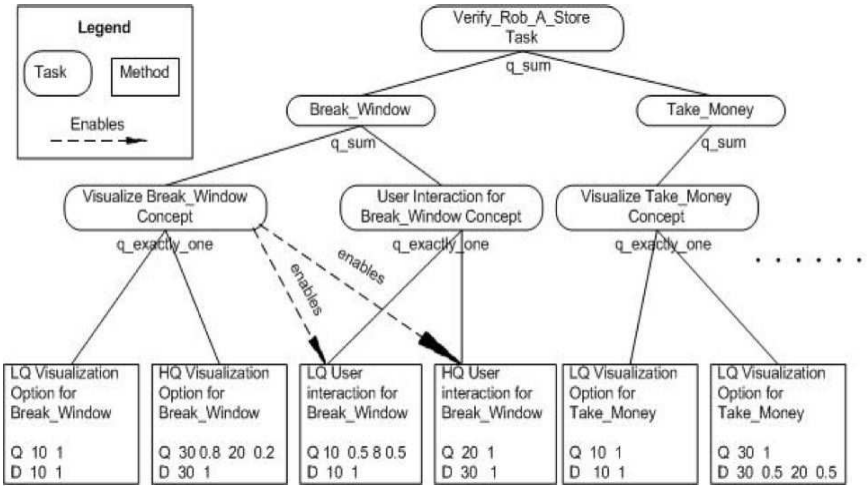

Fig. 7. Taems Task Structure for Verify Rob_A_Store Task

All primitive actions in TAEMS, called methods, are statistically characterized in two dimensions: quality and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. Thus, different applications have different notions of what corresponds to quality. A quality distribution Q 30 0.8 20 0.2 implies that the action will obtain a quality of 30, 80% of the time and a quality of 20, 20% of the time. LQ_Vis_Break_Window and HQ_Vis_Break_Window are the two alternative ways of visualizing the Break_Window concept, the former will open up the images quickly in low quality mode using a low pixel rate; the latter will take a longer time to load the images but will have higher precision of the images. The *enables* non-local relationship means the target method or task cannot accrue quality until the enabling task or method has non-zero quality. In other words, Visualize_Break_Window Concept has to accrue non-zero quality for user interaction to be successful. In other words, the images in the database have to be successfully rendered by the visualization tool for the user to have useful interactions with the data. As described earlier, the Taems task structure is translated into a MDP and the MDP policy prescribes the action choices.

Like Meta-STAB, we envision a Meta-TIBOR component that encodes the introspective task structure of TIBOR in the TAEMS language. This Meta-TIBOR introspective task structure will capture the various alternative ways for information foraging and visualization along with their quality and duration tradeoffs. Similarly we capture the end-to-end reasoning process of the AA agent using a Meta-AA component and describe its introspective task structure in TAEMS. Figure 8 and Figure 9 describe the introspective task structures of the Meta-TIBOR and Meta-AA components respectively. The introspective task structures described in TAEMS will allow the agent to autonomously justify the trade-offs that determine the action choices.

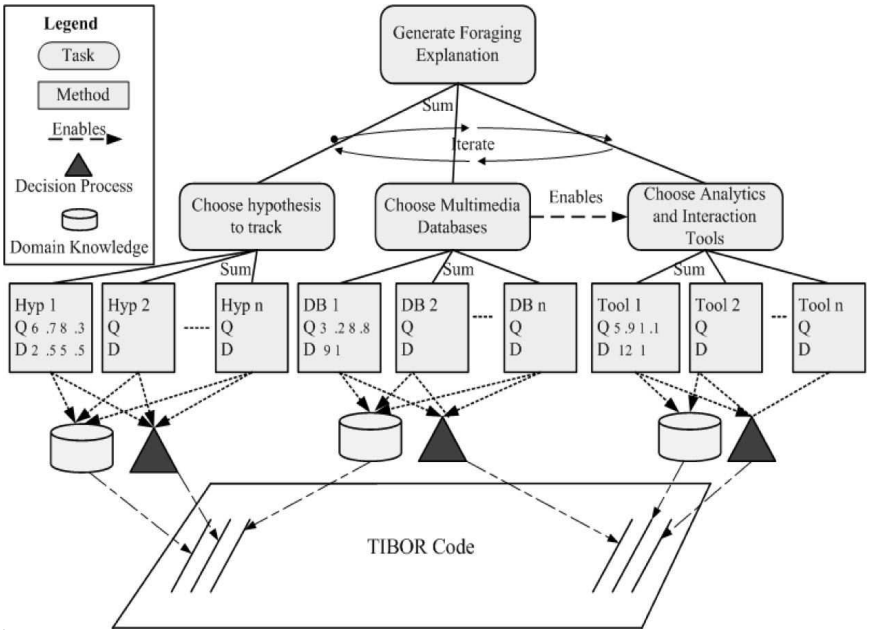

Fig. 8. Meta-TIBOR's Introspective Task Structure

Note that the explanations are both causal and intentional. The explanations are causal because the execution of a task in the introspective task structure sets up the knowledge conditions for the selection, invocation and execution of the succeeding task. Thus, the execution of various tasks is linked by the knowledge states they take as inputs and give as outputs. The explanations are intentional because the execution of a task (except the dummy task at the root of the task structure) takes place in the context of some higher-level task. Thus, this scheme can help answer not only the question of what the agent is doing at any given state of processing (the task), but also how (the method), and why (the higher-level task).

3 Conclusions and Future Work

In this paper we argue that agents operating in complex domains like investigative analysis should be capable of self-introspection over the task structure of their decision making process. This type of self-introspection enables agents to generate meaningful explanations and justifications of their action choices. We describe the components of an analytical agent and discuss how to include introspection and self-explanation as first-class design goals. We plan to continue implementation of the agent and study the effectiveness of the explanation matrix and meta-reasoning task structures at multiple levels of abstraction.

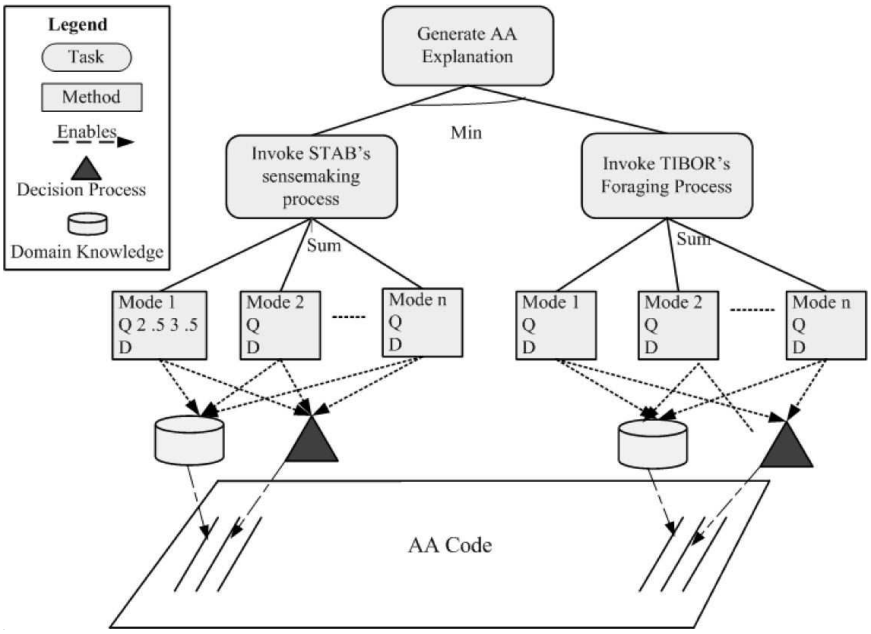

Fig. 9. Meta-AA's Introspective Task Structure

4 Acknowledgments

This work was sponsored by the National Visualization and Analytics Center (NVAC) under the auspices of the Southeastern Regional Visualization and Analytics Center. NVAC is a U.S. Department of Homeland Security Program led by Pacific Northwest National Laboratory.

References

1. Chandrasekaran, B., Swartout, W.: Explanations in Knowledge Systems: The Role of Explicit Representation of Design Knowledge. IEEE Expert **6**(3) (June 1991) 47–49
2. Buchanan, B., Shortliffe, E.: Rule-based expert systems : the MYCIN experiments of the Stanford Heuristic Programming Project. Addison-Wesley, Reading, MA (1984)
3. Clancey, W.J., Letsinger, R.: NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. In: Proceedings of the Seventh International Joint Conference on Artificial Intelligence. (August 1981) 829–836
4. Clancey, W.J.: The Epistemology of a Rule-Based Expert System - A Framework for Explanation. Artificial Intelligence Journal **20**(3) (1983) 215–251
5. Tanner, M.C., Keuneke, A.M., Chandrasekaran, B.: Explanation using task structure and domain functional models. (1993) 586–613

6. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in Case-Based Reasoning – Perspectives and Goals. Artificial Intelligence Review **24**(2) (October 2005) 109–143
7. Cox, M.T.: Metacognition in computation: a selected research review. Artificial Intelligence **169**(2) (2005) 104–141
8. Goel, A.K., Murdock, J.W.: Meta-cases: Explaining case-based reasoning. In: EWCBR. (1996) 150–163
9. Decker., K.: Taems: A framework for analysis and design of coordination mechanisms. In: In G. O'Hare and N. Jennings, editors, Foundations of Distributed Artificial Intelligence. Wiley Inter-Science. (1995)
10. Murdock, J., Goel, A.: Meta-case-Based Reasoning: Using Functional Models to Adapt Case-Based Agents. In: Proceedings of the International Conference on Case-Based Reasoning. (August, 2001) 407–421
11. Krizan, L.: Intelligence Essentials for Everyone. Daine Publications Company (1999)
12. Heuer, R.: Psychology of Intelligence Analysis. Center for the Study of Intelligence (1999)
13. Pirolli, P., Card, S.: The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In: Proceedings of 2005 International Conference on Intelligence Analysis. (May 2005) 2–4
14. Adams, S., Goel, A.: Making Sense of VAST Data. In: To appear in Proc. IEEE Conference on Intelligence and Security Informatics. (May 2007)
15. Josephson, J.R., Josephson, S.G.: Abductive Inference: Computation, Philosophy, Technology. Cambridge Univerity Press, Cambridge, MA (1994)
16. Bishop, M.: A Standard Audit Log Format. In: Proceedings of the 1995 National Information Systems Security Conference. (October 1995) 136–145
17. Josephson, J.R., Josephson, S.G.: Abductive Inference: Computation, Philosophy, Technology. Cambridge Univerity Press, Cambridge, MA (1994)
18. Buckingham Shum, S. In: Design Argumentation as Design Rationale. New York: Marcel Dekker, Inc. (1996)
19. Engelmore, R., A. Morgan, e.: Blackboard Systems. Addison-Wesley (1988)
20. Morrison, C.T., Cohen, P.R.: The hats simulator and colab: An integrated information fusion challenge problem and collaborative analysis environment. In: ISI. (2006) 105–116
21. Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., Zhang, S.: BIG: An Agent for Resource-Bounded Information Gathering and Decision Making. Artificial Intelligence Journal, Special Issue on Internet Information Agents **118**(1-2) (May 2000) 197–244
22. Raja, A., Lesser, V., Wagner, T.: Toward Robust Agent Control in Open Environments. In: Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, Spain, ACM Press (July, 2000) 84–91
23. Raja, A., Lesser, V.: A Framework for Meta-level Control in Multi-Agent Systems. To appear in Autonomous Agents and Multi-Agent Systems (January 2007)
24. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont, MA (1996)
25. Liu, D., Vaidyanath, J., Raja, A.: Tibor: A resource-bounded information foraging agent for visual analytics. UNCC Charlotte Visualization Center Technical Report CVC-UNCC-07-04, University of North Carolina at Charlotte (2007)