# Two Approaches to Implementing Metacognition

Emily Hand*, Darsana Josyula*†, Matthew Paisner*, Elizabeth McNany*, Donald Perlis*, and Michael T. Cox *
*Department of Computer Science
University of Maryland, College Park, College Park, Maryland 20742
†Department of Computer Science
Bowie State University, Bowie, MD 20715
Email: {emhand, darsana, mpaisner, beth, perlis, mcox}@cs.umd.edu

***Abstract*—Metacognition, the ability to monitor and regulate cognition, is important for an agent to adapt to novel situations and fix discrepancies in its knowledge base. In this paper, we discuss two different approaches to implementing metacognition in artificial systems: internally and externally. In the internal approach, metacognition is built into the agent, and thus, is combined with its cognitive reasoning abilities. The same Knowledge Base (KB) is fully shared between the metacognitive and cognitive processes of the artificial system. In the external metacognition approach, only portions of the agent's KB are shared between the agent and the external metacognition. We describe the implementation of external metacognition using our own Metacognitive Loop (MCL2) and internal metacognition using active logic in the context of a dialog agent called Alfred. We discuss how the two systems handle long pauses in dialog and compare the pros and cons of each. Our experiments show that for a system with time-related expectations, it is more efficient to use interleaved metacognition rather than an external metacognition module as the internal metacognition has access to the entire KB of the agent.**

***Keywords–Metacognition; Dialog Management.***

## I. INTRODUCTION

Humans are capable of reasoning about situations and developing expectations about themselves as well as the world around them. They are also able to handle anomalies. Humans can recognize that an expectation has been violated, decide on the best response, and then, implement that response to restore the desired state. Intelligent agents situated in the real world should also have these capabilities. An agent must possess some form of cognitive abilities in order to make decisions. In order to make an agent more intelligent, metacognitive abilities must be added to the system. Metacognition is the ability of an agent to explicitly monitor, evaluate, and improve upon its own internal processes. One approach to providing the agent with the capability to recognize and correct problems is to have the agent maintain some set of expectations about itself and its environment. These expectations would be a part of the metacognition used by the agent in order to properly reason about its situation. There has been some promising work in the fields of Artificial Intelligence and Cognitive Science with metacognition, including [3] and [16].

When faced with anomalies in dialog, humans are able to *note* the anomaly, *assess* the anomaly and determine how to handle it, and finally *guide* a response to resolve the anomaly. This is called the N-A-G cycle [6]. In our previous work, we have found that a N-A-G cycle works quite well in detecting and correcting anomalies in an intelligent agent.

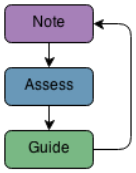


Figure 1. N-A-G Cycle.

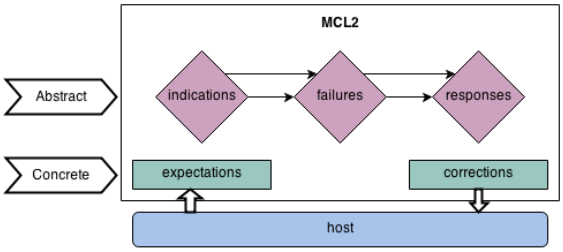


Figure 2. The MCL2 architecture.

We call the algorithm that implements the note-assess-guide-repeat cycle to deal with anomalies as the metacognitive loop (MCL ); this basic algorithm is shown in Figure 1. We have developed a domain-independent implementation of MCL, referred to here as MCL2, (Figure 2) which can be used as an external module [5][20]. MCL2 uses three ontologies organized as a Bayes net [5]: *indications* nodes representing different types of expectation violations; *failure* nodes indicating the probable type of problem that is being experienced, and *response* nodes associated with solutions that can be suggested to the host system. Periodically, a system is expected to send its current set of observations to MCL2 using a "monitor" call. If an expectation violation is noted, MCL2 responds with a suggestion of corrective action for the host to implement.

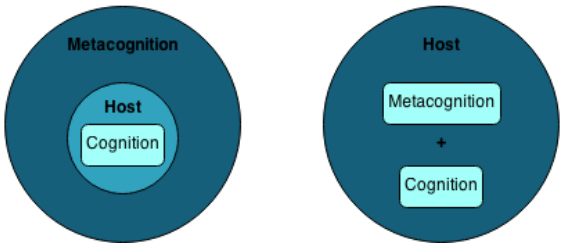


Figure 3. The host system connected to a metacognitive component with only cognitive internal capabilities (left) and the host with full metacognitive and cognitive internal capabilities (right).

 ISBN: 978-1-61208-340-7 

In this paper, we will discuss how to incorporate the N-A-G cycle into a cognitive system. Figure 3 shows the two implementations of metacognition. The agent could implement this process internally in conjunction with its regular cognition, allowing it to perform reasoning as well as reasoning about its reasoning. The agent could instead have only cognitive capabilities and be in communication with an external metacognitive module like MCL2. We investigate the two design approaches in detail, and discuss their performance when integrated with a dialog agent Alfred [12][13].

We discuss recent work in dialog agents, as well as metacognition, in Section II. Section III highlights the challenges associated with metacognition in dialog and our agent Alfred. Section IV discusses external metacognition, while Section V describes internal metacognition. Section VI compares the two implementations of metacognition.

## II. Related Work

In recent years, there has been a great deal of work in the use of metacognition as a tool for monitoring and regulating cognitive activities including natural language dialog. Monitoring and controlling cognitive activities has been a popular topic, with researchers focusing on explicitly setting time limits for deliberation [10], or limiting the size of current knowledge used for deliberation [18], or both [1][2][17]. Fox and Leake proposed to focus the cognitive activities of an agent by narrowing the knowledge base the agent uses to the subset relevant to the circumstances [9]. They showed that the reduced focus set enables the agent to spend the available deliberation time to produce the optimal action by adapting to the current circumstances. A hybrid approach, where the deliberative process simultaneously produces an immediate action and a strategy, has also been used [8]. Since the action is immediately available, the component of the agent responsible for taking action does not need to wait for the strategy in circumstances that do not permit doing so. This approach implies that the action component is intelligent enough to determine when to wait for a strategy.

McNany et al. [14] discusses metacognition as an integral part of natural language dialog in an artificial agent. The importance of metareasoning has been shown in linguistics. One such example is in [11], where Hymes expresses the idea of communicative competence. This idea was expanded by Canale [7], by detailing four different types of communicative competence. Before the aforementioned work in linguistics, Rieger [19] showed that metareasoning in a natural language dialog can improve deficiencies, and McRoy agreed with this in [15] further stating that in dialog, the ability to handle mistakes is essential. In more recent work on metareasoning in natural language dialog, Anderson and Lee [4] found that a large part of dialog involves metareasoning in order to properly handle misunderstandings, as well as references to previous utterances. Metareasoning and metacognition are a very large part of an intelligent dialog agent.

As metacognition and metareasoning have been identified as essential in an intelligent dialog agent, the next natural question is how to properly integrate these into the agent. Recently, Anderson, Oates, Chong, and Perlis [4] developed a metacognitive system, called MCL, to be used as an external module by an intelligent agent. Continuing with this idea, Schmill et al. [20] extended MCL to use metacognition to reason about and handle anomalies. In our work we extend MCL to handle anomalies in a dialog agent. We adapt MCL to reason about time-related expectations and anomalies and we compare a version of Alfred with internal metacognitive capabilities to Alfred connected with the adapted MCL.

## III. Metacognition to Handle Pauses in Dialog

In this paper, we discuss handling expectations and anomalies using metacognition in a dialog system. We believe this to be a good example for testing the two approaches to metacognition because we are dealing with time-related expectations and the work we present here is applicable to any system with time-related expectations, not simply dialog agents. Agents situated in a real-world environment must interact with humans and other agents. These interactions require that the agent has some concept of time, and some expectation for the amount of time certain interactions will require. We discuss these expectations in the context of a dialog agent, but they are generalizeable to any intelligent agent situated in the real world. To motivate our work, let us consider a few examples:

**Example 1:** Suppose we have two participants in a dialog $P_1$ and $P_2$. $P_1$ says something to $P_2$ expecting a response. If $P_2$ leaves the room without responding, a human would understand that something strange was occurring in the conversation. Perhaps $P_2$ was offended, or simply had to leave. In either case, a human $P_1$ would recognize the anomaly and not simply sit around waiting for a response. To simulate this behavior, an intelligent dialog agent should have some expectation for the length of conversational pauses associated with a particular user, so that it understands that an anomaly has occurred when $P_2$ fails to respond in a reasonable time period. Otherwise, it will not understand that there is a problem, and thus, will take no steps to correct it.

Listing 1. Example 1 Dialog in Alfred without Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) ...
...
(t=100) Alfred: "Please tell me what to do now."
(t=101) ...
...
(t=200) Alfred: "Please tell me what to do now."
```

Listing 2. Example 1 Dialog in Alfred with Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) ...
...
(t=100) Alfred: "Are you there?"
(t=101) ...
...
(t=200) Alfred: "Goodbye."
```

**Example 2:** We have the same dialog participants from Example 1. $P_1$ says something to $P_2$ and $P_2$ is thinking of something to say. While $P_2$ is thinking, there is a pause in the conversation. Now $P_1$ does have an expectation of the length of a typical pause, and it notices when the normal pause length has been exceeded. $P_1$ has an expectation that $P_2$ will respond within 100 seconds. $P_2$ is still thinking and 100

seconds pass. $P_1$ then says "Do you still want to talk?" Now, 101 seconds have passed since $P_1$ initially started waiting, and $P_2$ has still not responded. If $P_1$ does not take into account that it has recently spoken and that $P_2$ may need more time to respond, then since its expectation has been violated, $P_1$ would ask again "Do you still want to talk?" This exchange would continue in this way with $P_1$ continuously asking "Do you still want to talk?" until $P_2$ responds. An intelligent dialog agent would need to understand that there is a difference between an initial prompt and a prompt issued as a response to an expectation violation. $P_1$ should understand that when it prompts $P_2$ that $P_2$ will need more time to respond.

Listing 3. Example 2 Dialog in Alfred without Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) User: "Send Metroliner to Baltimore."
(t=2) Alfred: "Command sent to domain."
(t=3) Alfred: "Please enter another command."
...
(t=103) Alfred: "Please tell me what to do now."
(t=104) Alfred: "Please tell me what to do now."
(t=105) Alfred: "Please tell me what to do now."
...
```

Listing 4. Example 2 Dialog in Alfred with Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) User: "Send Metroliner to Baltimore."
(t=2) Alfred: "Command sent to domain."
(t=3) Alfred: "Please enter another command."
...
(t=103) Alfred: "Please tell me what to do now."
...
(t=150) User: "Send Northstar to Richmond."
...
```

**Example 3:** We have the same dialog participants from Example 1. Now, the dialog agent $P_1$ has an expectation for a typical pause associated with $P_2$ of 100 seconds. It also understands how to revise its expectations. If $P_1$ asks $P_2$ a few questions and for each of the questions, $P_2$ took longer than $P_1$ expected to respond, $P_1$ recognizes a pattern. $P_1$ realizes that $P_2$ responds more slowly in general than it had expected and so it revises its expectation, noting that $P_2$ has a longer typical pause length for conversations.

Listing 5. Example 3 Dialog in Alfred without Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) User: "Send Metroliner to Baltimore."
(t=2) Alfred: "Command sent to domain."
(t=3) Alfred: "Please enter another command."
...
(t=103) Alfred: "Please tell me what to do now."
(t=104) Alfred: "Please tell me what to do now."
...
(t=150) User: "Send Northstar to Richmond."
(t=151) Alfred: "Command sent to domain."
(t=152) Alfred: "Please enter another command."
...
(t=252) Alfred: "Please tell me what to do now."
(t=253) Alfred: "Please tell me what to do now."
...
(t=300) User: "Send Bullet to Washington."
(t=301) Alfred: "Command sent to domain."
(t=302) Alfred: "Please enter another command."
...
(t=402) Alfred: "Please tell me what to do now."
(t=403) Alfred: "Please tell me what to do now."
...
```

Listing 6. Example 3 Dialog in Alfred with Metacognition

```
(t=0) Alfred: "Welcome."
(t=1) User: "Send Metroliner to Baltimore."
(t=2) Alfred: "Command sent to domain."
(t=3) Alfred: "Please enter another command."
...
(t=103) Alfred: "Please tell me what to do now."
...
(t=150) User: "Send Northstar to Richmond."
(t=151) Alfred: "Command sent to domain."
(t=152) Alfred: "Please enter another command."
...
(t=252) Alfred: "Please tell me what to do now."
...
(t=300) User: "Send Bullet to Washington."
(t=301) Alfred: "Command sent to domain."
(t=302) Alfred: "Please enter another command."
...
(t=450) User: "Send Metroliner to Buffalo."
```

We implemented our two metacognition techniques using a particular dialog agent, Alfred. Alfred acts as an interface between a human user and a task-oriented domain. As input, it accepts English sentences which it then parses and takes as commands to be sent to a specific domain. We consider the problem of Alfred encountering conversational pauses of varying length when interacting with a human user. Alfred has expectations about the typical pause length associated with a particular user, and is able to detect when an expectation has been violated.

We have implemented a version of Alfred which uses metacognition to handle situations similar to the three examples above. We are presently focused on Alfred's expectations concerning interaction with the human user.

We will focus on the implementation of the two systems with Alfred, but much of the following discussion is applicable to any host system. In our implementation of internal metacognition, we combine the cognitive and metacognitive capabilities, creating an interleaved cognitive and metacognitive reasoning within Alfred. For our external implementation of metacognition, we used our MCL2, which Alfred communicates with through monitor calls.

## IV. External Metacognition using MCL2

First, we will discuss the architecture where the host is connected to an external metacognitive system. In this architecture, the host initially gives the metacognitive system its set of expectations. The metacognitive system then determines (via observations also provided by the host) when expectations have been violated and recommends how the host should respond to such violations.

One important design problem in this architecture involves knowledge sharing. Initially, the host sends its expectations to the metacognitive system, and the host must also send updates about the current state of the world in order for it to reason about possible violations. A key question arises: how often should the host system send information to the metacognitive system? As the host sends more information, there is more overhead incurred in sending information between the host and the metacognitive component.

We have a particular implemented metacognitive module called MCL2. MCL2 is capable of accepting expectations

 ISBN: 978-1-61208-340-7

and information about the current state of the world from a host, reasoning about expectations, and sending suggestions to the host when it notes an expectation violation. The host communicates with MCL2 through monitor calls in which it shares information about particular sensor readings. The host can perform monitor calls at any time; there is no specified frequency of monitor calls from the host.

In our implementation, Alfred provides MCL2 with an expectation for a typical pause length, *expected_pause(100)*; at each step, Alfred increments its current waiting time, *curr_wait_time(t)*, and sends this information to MCL2. When the current waiting time exceeds the expected pause length (*curr_wait_time(101)*), MCL2 notes that an expectation has been violated and provides the host with a suggestion as to how to respond. Currently, MCL2's suggestion is for the host to prompt the user, so Alfred then prompts the user with "Please tell me what to do now." Since Alfred is sending information to MCL2 at every time step, Alfred now sends the current wait time (*curr_wait_time(102)*) to MCL2. MCL2 only keeps track of the expectation and the current information being sent to it by the host. It receives *curr_wait_time(102)* from Alfred and treats it as a new expectation violation so MCL2 again sends a suggestion to prompt the user. This happens at every time step until the user enters a command to Alfred, at which time Alfred would reset its wait time (assert *curr_wait_time(0)*) and send that to MCL2. This is not the desired behavior of an intelligent agent. When a user is prompted, the system should give the user time to respond to the prompt before prompting again. Therefore, MCL2 needs to maintain another expectation for an appropriate response time associated with a prompt (*expected_response_time(100)*).

Another design problem which is even more complicated is that of deciding what exactly it means to implement a suggestion given to the host by the metacognitive system. In the case of MCL2 and Alfred, when MCL2 sends the suggestion to Alfred that it should prompt the user, MCL2 expects a response from Alfred indicating if the suggestion was a success or a failure. However, this does not make sense in the context of time-related expectations. If MCL2 tells Alfred to prompt the user, and Alfred does so, then MCL2's suggestion has been implemented, but both MCL2 and Alfred must wait in order to see if the suggestion was a failure or a success, that is, if the effect occurs. The suggestion is a failure if after the prompt, the user does not respond within the *expected_response_time* and it is a success if the user responds within that amount of time. However, is this truly a failure if the user simply takes longer to respond than expected? After all, if the user responds at step 102, the *expected_response_time* has been exceeded, but the prompting was successful.

In order to handle Examples 1-3 when connecting Alfred with MCL2, we used two expectations: *in_set(sensor_pause_id, num_pause_violations, 0, 1, 2, 3)*, and *discrete_range(sensor_pause_id, pause_length, 0, 100, add(1))*. The first expectation states that the value for *num_pause_violations* can only be 0, 1, 2, or 3. If it is any other value, there is a violation. The second expectation says that the value for *pause_length* must be between 0 and 100 in increments of 1. The value of *num_pause_violations* is incremented for each user input which required a prompt. If Alfred starts up and the user walks away, Alfred will prompt the user at step 101, and again at step 202, and so on. If we consider each one of these a violation; then, at step 404, Alfred would change its expectation. There is no need for Alfred to change its expectation if there is no user there. So, we do not consider repeated prompts for the same user input to be separate violations. If a user requires two prompts before a response, then that would be one violation and *num_pause_violations* would be incremented by one. On the fourth violation of the expectation for *pause_length*, the expectation for *num_pause_violations* is violated. When this expectation is violated, it means that Alfred has been repeatedly noticing the same expectation violation and the most probable source of the problem is a model error, so the expectation must be revised. This system of two expectations works well because it accounts for user error. If a user walks away, Alfred will not change its expectation, and if a user is distracted in some way and takes a long time to respond to one prompt, but from then on continues responding in a reasonable amount of time, Alfred will not change its expectation. This is generally a good idea as an agent should not change its expectation based off of one violation. More information is needed in order to determine if there is an internal or external error occurring.

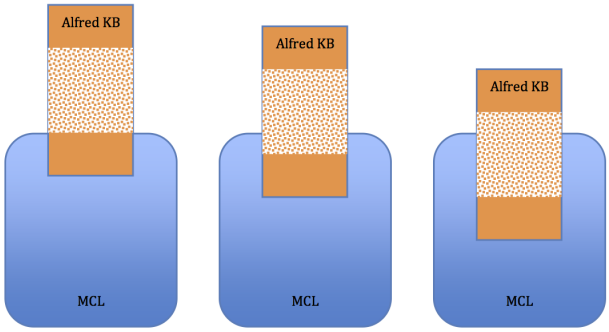


Figure 4. Amount of knowledge being shared between Alfred and MCL2 in Examples 1-3 from left to right.

This implementation of Alfred with MCL2 is fully functioning with the pause time example. However, Alfred is doing most of the reasoning. Alfred is keeping track of the current waiting time, as well as the number of pause violations, and sending all of this information to MCL2 at each time step. In this framework, MCL2 is doing very little to add to the overall system, and it is not doing anything Alfred, or any similar host, could not do itself. The design problem of knowledge sharing becomes very complex when connecting a host system with an external metacognitive module. External metacognition requires that the host decide what to share with the external component and when to share it. Figure 4 shows the knowledge sharing between Alfred and MCL2 in Examples 1 through 3 discussed earlier. The figure shows Alfred's KB overlapping with MCL as the portion which is being shared. The orange portion of Alfred's KB that is overlapping with MCL is the general knowledge the Alfred shares with MCL, which is not related to pause time. The orange dotted portion of Alfred's KB is Alfred's knowledge about pause time. In Example 1, there is no overlap between the orange part of Alfred's KB and MCL. In the second example, Alfred sends some of its KB concerning pause time to MCL2 and in the

 ISBN: 978-1-61208-340-7 

final example even more of the pause time knowledge is shared with MCL2.

As implemented, MCL2 requires the host to periodically send a subset of its observations using monitor calls. Thus, whether an expectation violation gets noted in a timely manner depends on whether the appropriate observations are being sent to MCL2 and the frequency at which the host issues monitor calls. If the host shares very little information with MCL2, such as in Alfred's pause time example, then flaws in the host's initial knowledge base cannot be found or corrected by MCL2. For these reasons, we decided to focus on designing and implementing the second framework (Figure 3), where the host has cognitive and metacognitive internal capabilities.

## V. Interleaved Metacognition using Active Logic

In this version, Alfred has full metacognitive and cognitive abilities, including expectations about itself and the world in which it is situated. Alfred has expectations that the user will respond to a prompt, that a user will respond with a certain type of answer, and that the user will respond within a specified amount of time. Alfred's expectations for interleaved metacognition are similar to those when connected with MCL2. It has an expectation for user pause length (or when it should receive user input), and an expectation for an appropriate number of violations before revising its expectation.

Alfred's expectations concerning the amount of time associated with a particular result are represented as time intervals, because Alfred expects a user response between two time steps. Initially, Alfred has the expectation of user input within 100 steps, represented in a predicate in its knowledge base as *in_set(pause_sensor_id, user_input, 0, 100, add(1))*. The predicate specifies the start and end points of the interval in which the user should respond. If the current time step exceeds the endpoint of the expected pause interval, Alfred notes that its expectation has been violated. When Alfred's expectation is violated, it prompts the user with "Please tell me what to do now." This is Alfred taking action to fix the expectation violation, but the violation will not be fixed until the user responds to Alfred's prompt. When Alfred prompts, it must wait for user input, so now the expectation is *in_set(pause_sensor_id, user_input, 100, 200, add(1))*, giving the user another 100 steps to respond to the prompt.

Alfred also has expectations about its expectation violations which look like *in_set(sensor_pause_id, num_pause_violations, 0, 1, 2, 3)*, exactly the same as with MCL2. Each time a violation of the expectation associated with *user_input* is violated, then the property *num_pause_violations* is increased. Once *num_pause_violations* reaches a value greater than 3, the expectation is violated, and Alfred knows that it needs to revise its expectation for the property *user_input* so that it can more effectively communicate with the human user.

Interleaved metacognition has lesser communication overhead than external, as all knowledge is shared between the cognitive and metacognitive components. With all information being shared, all expectations can be monitored properly and all expectation violations can be detected. Interleaved metacognition is implemented in time-tracking Active Logic [5], so the metacognition and cognition processes proceed in parallel, step-by-step. In Active Logic, a step is the fundamental measure of time passage. Since both the cognition and metacognition are being processed internally, the same concept of a step is shared by both, and therefore new observations are handled at the same time by both the metacognition and cognition processes.

## VI. Comparing Interleaved and MCL2

Consider a more general situation in which we either have interleaved metacognition or MCL2, where we are no longer just considering the pause time example. If in the case of MCL2, say we have a host with three sensors and it is sharing 60% of its initial belief set with the external module. If there is one anomaly detected, the knowledge being shared by the host is the same as prior to the anomaly. If there are several anomalies, the percentage of knowledge being shared is the same. This is counter-intuitive and not necessarily the behavior that we want from the connection between the host and an external metacognitive module. If an anomaly occurs, the host should share more information with MCL2 in order for MCL2 to better correct the problem. Another problem is even if there are anomalies not detected by MCL2 - because they are a part of the 40% not being shared - the amount of knowledge sharing remains the same. Therefore, these anomalies can never be corrected, and this, in principle, is against the entire point of the metacognitive component. In order for the metacognitive process to be effective, the host must be able to share any part of its KB with it, so that all anomalies can be detected and corrected appropriately. Interleaved metacognition provides just that very easily. The metacognitive external module, MCL2, however, is limited by its reliance on the host to share the required parts of its KB in a timely manner.

Figure 5 shows different scenarios of knowledge sharing for external metacognition. The shaded region is the portion of the knowledge base that is being shared with the external module for each scenario. The 10th scenario shares 100% of the KB, making it equivalent to interleaved metacognition. Only the violations that occur in the shaded area of the KB will be noticed when using an external module. Any violations that occur in the KB above the shaded area will go unnoticed with external metacognition.

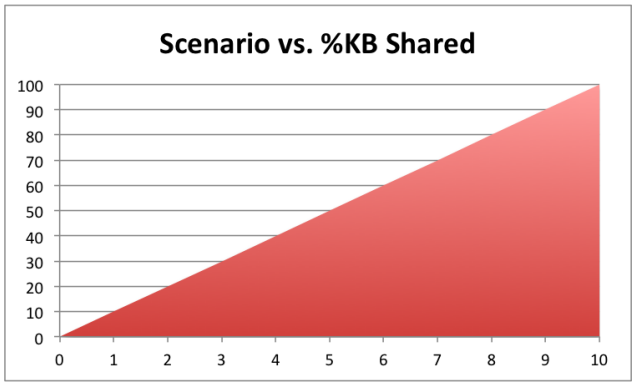


Figure 5. Different knowledge sharing scenarios vs the amount of knowledge being shared.

We note that the effectiveness of MCL2 is also dependent on the frequency of "monitor" calls. Going back to the pause

 ISBN: 978-1-61208-340-7 

time example, if Alfred only sent updates to MCL2 every 100 timesteps, MCL2 would only have available that information about the user's response. If Alfred had the expectation of a response within 100 timesteps, and the user consistently responded late, MCL2 would be unable to tell the difference between a 1-timestep delays and 99-timestep delays. Interleaved MCL does not have a dependency on the host to specify when to process information, and so can note and act upon any anomalies without waiting for a host to signal. We are currently working on an improved version of MCL2 that does not have these constraints.

One significant benefit to using external MCL with MCL2 is that the external metacognitive component does not need to be rewritten for each host system. MCL2 can be connected to any host system. Internal metacognition requires that the metacognitive portion be rewritten for each host system, which is quite a bit of work. The entire N-A-G cycle must be rewritten inside of the host system; but, with MCL2, it can just be interfaced with the host system with very few changes being made to the host system.

## VII. Conclusion and Future Work

In our research, we have found that an interleaved implementation of metacognition is much more useful than an external connection with MCL2. An external connection requires that a large amount of information be shared between the host and the metacognitive module in order for external metacognition to work properly with our current implementation of MCL2. However, if the host must communicate to the external metacognitive component a large portion of its knowledge base, then there seems to be little to no advantage to using an external metacognitive component like MCL2. A host-initiated data sharing model like MCL 2, causes significant overhead for timely sharing of information by the host to MCL. Instead, the host could easily perform metacognition interleaved with the cognition and not waste time and space with external metacognition. An external metacognitive module could be useful if the host only needed to share a small amount of information with it, but in this case, flaws in the hosts initial knowledge base (unshared) could not be exposed with that limited metacognition, as that information is not being shared with the external module. Another option would be to have a KB that is completely shared between the metacognitive component and the host without the host initiating the sharing. An agent with internal metacognitive capabilities allows for 100% knowledge sharing and therefore the ability of the agent to detect all expectation violations.

## Acknowledgment

This material is based upon work supported by ONR Grant # N00014-12-1-0430.

## References

[1] N. Alechina, B. Logan, H. N. Nguyen, and A. Rakib, "Logic for coalitions with bounded resources," Journal of Logic and Computation, 2009, vol. 21, no. 6, pp. 907-937.

[2] G. Alexander, A. Raja, and D. Musliner, "Controlling Deliberation in a Markov Decision Process-Based Agent," in Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, 2008, pp. 461-468.

[3] G. Alexander, A. Raja, E. Durfee, and D. Musliner, "Design Paradigms for Meta-Control in Multiagent Systems," Proceedings of AAMAS 2007 Workshop on Metareasoning in Agent-based Systems, 2007, pp. 92-103.

[4] M. Anderson and B. Lee, "Metalanguage for Dialog Management," in 16th Annual Winter Conference on Discourse, Text and Cognition, 2005.

[5] M. Anderson, T. Oates, W. Chong, and D. Perlis, "The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance," Journal of Experimental and Theoretical Artificial Intelligence, vol. 18, no. 3, 2006, pp. 387-411.

[6] M. Anderson and D. Perlis, "Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness," Journal of Logic and Computation, vol. 15, no. 1, 2005, pp. 21-40.

[7] M. Canale, "From Communicative Competence to Communicative Language Pedagogy," in Language and Communication, J. Richards and R. Schmidt, Ed., New York: Longman, 1983, pp. 2-27.

[8] F. Dylla, A. Ferrein, E. Ferrein, and G. Lakemeyer, "Acting and Deliberating using Golog in Robotic Soccer - A Hybrid Architecture," in Proceedings of the 3rd International Cognitive Robotics Workshop (CogRob 2002), Edmonton, Alberta, Canada, 2002.

[9] S. Fox and D. Leake, "Using Interospective Reasoning to Refine Indexing," in Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Quebec, Canada, 1995, pp. 391-397.

[10] E. A. Hansen and S. Zilberstein, "Monitoring and control of anytime algorithms:A dynamic programming approach," Artificial Intelligence, 2001, vol. 126, no. 1-2, pp. 139-157.

[11] D. Hymes, "On Communicative Competence," in Sociolinguistics: Selected Readings, J. B. Pride and J. Holmes, Ed., Harmondsworth: Penguin Books, 1972, pp. 269-293.

[12] D. Josyula, "A Unified Theory of Acting and Agency for a Universal Interfacing Agent," Ph.D. dissertation, University of Maryland, College Park, 2005.

[13] D. P. Josyula, S. Fults, M. L. Anderson, S. Wilson, and D. Perlis, Application of MCL in a Dialog Agent, in Papers from the Third Language and Technology Conference, 2007.

[14] E. McNany, D. Josyula, M. T. Cox, M. Paisner, and D. Perlis, "Metacognitive Guidance in a Dialog Agent," The Fifth International Conference on Advanced Cognitive Technologies and Applications, IARIA, 2013, pp. 137-140.

[15] S. McRoy, "Abductive Interpretation and Reinterpretation of Natural Language Utterances," Ph.D. dissertation, University of Toronto, 1993.

[16] A. Nuxoll and J. Laird, "Enhancing intelligent agents with episodic memory," Cognitive Systems Research, 2012, pp. 17-18, 3448.

[17] A. Raja and V. Lesser, "A Framework for Meta-level Control in Multi-Agent Systems," Autonomous Agents and Multi-Agent Systems, 2007, vol. 15, no. 2, pp. 147-196.

[18] U. Ramamurthy and S. Franklin, "Memory Systems for Cognitive Agents," in Proceedings of the Symposium on Human Memory for Artificial Agents, AISB'11 Convention, York, United Kingdom, 2011, pp. 35-40.

[19] C. Rieger, "Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances," Ph.D. dissertation, Stanford University, 1974.

[20] M. Schmill, M. T. Cox, and A. Raja, "The Metacognitive Loop and Reasoning about Anomalies," in Metareasoning: Thinking about Thinking, Cambridge, MA: MIT Press, 2011, pp. 183-198.

 ISBN: 978-1-61208-340-7 