Self-Adaptivity via Introspection and Monitoring of Norms and Values

Henry Hexmoor and Gordon Beavers

hexmoor@uark.edu beavers@engr.uark.edu

Computer Science & Computer Engineering Department, Engineering
University of Arkansas
Fayetteville, AR 72701

1 Introduction

Awareness of norms and values among agents can be used by agents to adapt to changing conditions. One agent must recognize the values and norms being observed by other agents in order for the agent to respond appropriately to the actions of the other agents. Without this recognition an agent, for example, might be benevolent and have the well-being of other agents in mind, but work at cross purposes to those agents because it does not recognize the plan being followed by the other agents, or the roles that those agents have adopted or the norms that are associated with those roles.

To see the importance of values, roles, and norms in cooperative activity consider the following concrete example. Suppose that three agents have formed a team to fulfill the duties at the framastat installation station on a widget assembly line. They first agree to adopt the values "be a team player", "work efficiently", "be independent", and "don't interfere with the activities of other agents on your team" since these values are seen to be to their mutual benefit. Note that the value "be independent" and "be a team player" are in some conflict while the value "be a team player" seems to subsume the value "don't interfere". The agents then agree to assume the roles required to fulfill their duties.
Suppose that their duties include (i) two agents guide and push the widget to the framastat installation station, (ii) one agent retrieves the framastat and inspects it while another agent obtains the fasteners, (iii) one agent positions the framastat and holds it in place while the second and third agents place and secure the fasteners, (iv) the widget/framastat assembly is inspected, and finally, (v) the widget is guided to the next station on the assembly line. The roles seem to be 'pusher', 'navigator', 'framastat fetcher', 'fastener fetcher', 'framastat inspector', 'framastat positioner', 'framastat fastener', and 'widget/framastat inspector'. It is apparent that the adopted roles have dependency among them. Furthermore, it is possible that the agents will undertake the completion of their respective tasks in such a manner that there is redundancy among them. Redundancy is between tasks. For example, multiple agents might duplicate the inspection of the widget

on the assembly line. The agents attempt to remedy redundancy and dependence illustrates adherence to two of the values given above.

Each role will have norms associated with it. For example roles requiring agents to work in close quarters might include “don’t come into contact with other agents”, and “always perform actions in the same way so that your actions are easily interpreted”.

Assume that each team member accurately estimates its dependence on other team members, which is represented as a deviation from independence. The following table shows the three agents in their roles. Each agent due to its role is considered fully independent at level 0.0. Each cell in the table from row i to column j is agent i’s dependence on agent j due to their roles. For instance, agent 2 depends on agent 3 by 20%, which can be interpreted as agent 2 cannot finish 20% of its task without agent 3 completing its task. The three entries in the last row are each agent’s total dependence on others. For instance, agent 1 depends upon agent 2 by 10%. These values reflect the team’s dependence on the agent. In this example, agent 2 is the most dispensable whereas agent 1 is the most needed. In column 4, we show the sum of each agent’s dependences on other agents. For example, agent 1 depends on agents 2 and agent 3 by 10% and 30% respectively. Since agent 2 depends on agent 3 by 20%, agent 1 indirectly depends on agent 3 by another 20%, which adds up to 60%. In this example, agent 1 is the most dependent on other members of the team, whereas agent 3 is least dependent. In this example, agent 2’s dependence on agent 1 is high (0.7) but agent 1 does not depend on 2 as much (0.1).

	1	2	3	
1	-	0.1	0.3	0.6
2	0.7	-	0.2	1.3
3	0.1	0.0	-	0.2
	0.8	0.1	0.6	

Table 1 Dependence

In the following table each cell from row i to column j contains agent i’s overlap (redundancy) with agent j. For instance, agent 2 overlaps with agent 3 by 20%. Redundancy is symmetric. In the fourth column we add the overlap with other agents and see the agent’s total overlap. For example, agent 3 has 50% overlap with agent 1 and agent 2. Since much of the activity of agent 3 is redundant, the value “be efficient” dictates that the agents redefine their roles so as to minimize the redundancy.

	1	2	3	
1	-	0.1	0.3	0.4
2	0.1	-	0.2	0.3
3	0.3	0.2	-	0.5

Table 2 Redundancy

Values can conflict, as the values "be independent" and "be a team player" seem to. Resolving these conflicts is one of the subtlest problems in the use of values to guide behavior. Earlier we suggested that social norms at the right granularity govern action selection [Hexmoor, 2001c]. We continue with this assertion and build from that to architectures that exhibit how norms and value awareness enhances adaptivity. In the rest of this paper we will outline our architectures and follow that by an assessment of similar attempts in the literature. Throughout this paper, we feature teaming among agents as a special class of norms. We will then offer some concluding remarks.

2 Adaptive Architectures

In this section we describe the development of adaptive architectures, both for individual agents and groups of agents, as well as the investigation of abilities that aid in the maintenance of efficient interactions, and of methods that guarantee responsible behavior.

2.1 Adaptive Architectures

2.1.1 Inter-agent and Agent Community Architecture

In our framework, agents control themselves and exert influence on other agents. We model agent control as a networked hierarchy of feedback loops within a single agent with higher loops providing guidance for lower loops. For example, an agent who decides to be in a team with two other agents, instantiates a feedback loop for teaming. At a lower level, the agent instantiates other feedback loops dependent on the team loop[1]. If the team loop ceases to exist, the other loops cease as well. With two teammates, two feedback loops are instantiated for maintaining working relationships in the context of the team with the two other agents. Additionally, for each teamwork criterion, the agent instantiates a

[1] Subordinate feedback loops are in service of their superior feedback loops. This is a bit like inheritance hierarchy in object-oriented programming. However, the relationship here is control hierarchy and not data inheritance.

feedback loop to uphold that particular criterion. Instantiating templates from prior cases create the feedback loops dynamically. We assume the agent will have a library of cases and patterns of loops for different scenarios, e.g., for teaming.

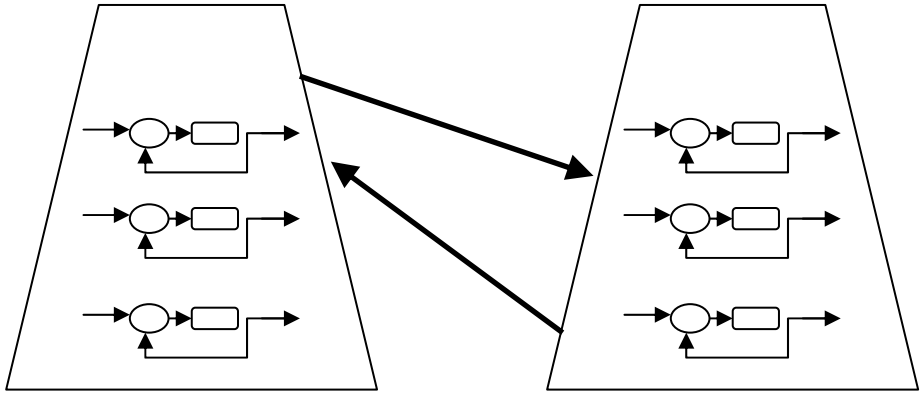

Figure 1 Two agents, each with a hierarchy of feedback loops, may interact at different levels of the agent. The arrows between agents show that in general agents will relate at different levels. The diagram is not suggesting specific levels.

The ability of an agent to self-monitor that results in the selection and instantiation of a feedback loop provides an important type of self-adaptivity. As designers of self-monitoring agents, we provide a library of feedback loops to be used by the agent for solving particular types of social problems centered on agent interaction typically involving a common set of norms or values. Self-monitoring is part of this self-adaptivity. Yet another type of self-adaptivity is achieved by monitoring other agents, which we will now describe.

Interacting agents provide guidance for one another to dynamically select a level, which generates an interaction. For example, an agent may suggest the idea of teaming to another agent (who is not currently a teammate). The agent may choose to formulate a message for this suggestion with concerns at a high level so it will choose the topmost teaming loop. Two agents already in a team choose a level at which to interact, which may be the top team loop level. The ability to monitor other agents in order to select an appropriate feedback loop level for interaction provides another important level of self-adaptivity. The idea of interaction at different levels is shown in Figure 1. An agent may initiate a level due to inputs it receives from another agent. For instance, if one agent points to an inefficiency of teaming in another agent, the second agent may instantiate a loop to monitor itself and perform adjustments to bring about the desired performance.

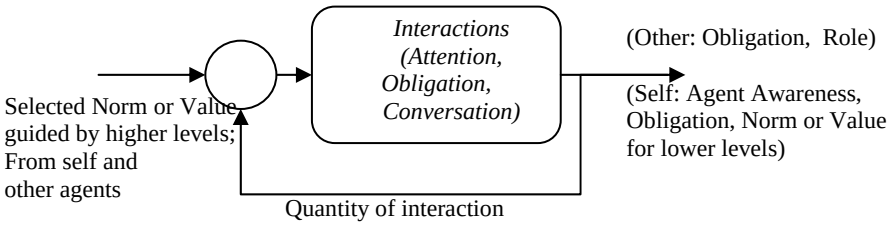

Figure 2 Social Feedback Loop at level i

Figure 2 shows one such feedback-loop at level i. The feedback loop receives a reference value as a norm or a value to uphold, which it has received from a higher level. At this level the agent monitors and quantifies how well the norm or value at this level is upheld. This is shown as the input to the comparator (depicted as a circle). The comparator determines discrepancies and sends them to the Interactions module. The interactions module adjusts the agent's mental attitudes and produces a revised decision on the agent's obligations and roles for other agents, as well as possibly awareness of other agents, obligations, norms or values for its self. The feedback loops discussed here are combinations of several functions we call revision functions, which we will discuss in the section on agent architecture. The revision functions being combined are attention revision, obligation revision, and conversation revision. Attention revision keeps track of objects of attention. Obligation revision maintains commitments to committed obligations. Conversation revision consists of several modules including modules that maintain deictic centers and negotiation protocols, and conversational states.

The agents considered here exhibit various degrees of sociability in the form of adherence to norms, roles, values, cooperation, motives, responsibilities, autonomies, and rights. Intentional agents have been modeled in multi-modal BDI logics, e.g. [Wooldridge, 2000], with operators for belief, desire, and intention. We propose the integration of social notions into BDI agent architectures to account for social decision-making. Although a large collection of notions is needed to explain the actions of complex social agents, this project will investigate a somewhat simplified model of social agents built on a small set of agent properties (norms and values) and intentional notions (beliefs, desires, intentions, and obligations). Since this model is a starting point for the further investigation of social agents, it is expected that the model will be improved and expanded as the result of further research.

Figure 3 shows some of the influences among social agents, groups and social notions that a model should take into account. The figure has been simplified to emphasize the most salient features of sociality. The reality is much more complex and many of the relationships shown are part of ongoing research projects that will clarify the nature of the relationships.

Values (or guiding principles) can have varying scope in terms of the set of agents to which they apply. Norms may guide the actions of individuals, groups, societies, or

even be global guides to behavior. Values may be taken to be constraints that are determined by roles, so that norms can be modeled with a filter on possible worlds. A group may set the values to which its members will adhere with each agent helping to determine what these values should be through negotiation. When the group adopts a joint intention, the members of the group will negotiate a division of responsibilities, these responsibilities, in turn, determine the roles assumed by each agent.

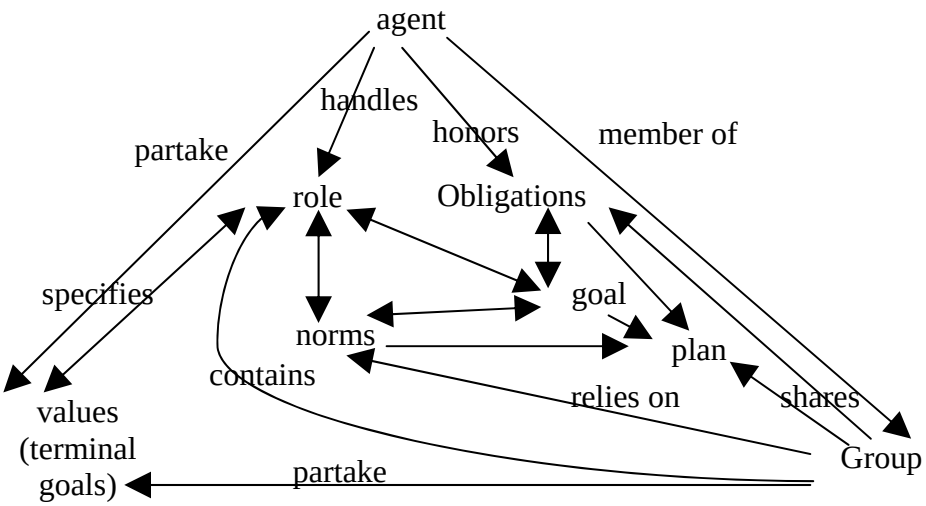

Figure 3 An agent as a member of a social group

An individual role will normally be fulfilled in a standard way, that is, each role will imply a set of norms to which the agent is expected to comply in addition to the values it will observe[2].

[2] Values are viewed as more general and abstract, e.g., "do no gratuitous harm" while norms are considered more specific and concrete, e.g., "when in area A and moving at a speed of 1 meter per second or faster make sure that there are no obstacles within a range of five meters". Norms are determined by roles and designate a range of behaviors that are consistent with the agent's having adopted a given role. When an agent accepts a role, the agent is expected to acquire the norms that are appropriate for the role. Some norms are characteristic behaviors that evolve over time in response to selective pressures while other norms are conventions selected through deliberation. Norms and values differ in specificity. "When fulfilling the role of an ATM, always offer a receipt for each transaction" is a norm that can be rephrased as an obligation "ATMs ought to offer receipts for transactions" or as a fact "ATMs offer receipts for transactions". The specificity and concreteness of this standard suggest that it is a norm. In contrast "always cooperate with team members" is more general and abstract, in part because what constitutes cooperation requires interpretation. So "always cooperate with team members" is a value. Obligations are implemented as modal operators, with distinct obligations having distinct operators. If agent A

A decision involving teamwork is an example of agent interaction. Teamwork requires adherence to a set of norms. One such set of norms concerns performance. The performance of a team is meaningful for an agent along many dimensions. First we list the dimensions (below) and then state exemplar norms from them. We have reported on this elsewhere [Beavers and Hexmoor, 2001a]. Agents in teamwork monitor the following dimensions:

(a) role dependence on other agents, by agents, as a group,

(b) redundancy/overlap with others, as a group,

(c) influence on others, influence on group, influence from others, influence from group, and

(d) efficiency of self judged by others, efficiency of others judged by self, group efficiency.

A list of exemplar goals that can be derived from these performance factors include:

(a) lower overall group role dependence,

(b) lower overall group role redundancy,

(c) increase overall group positive influence, and

(d) increase overall group efficiency.

If an agent notices an unexpected value along any dimension, it will either take corrective measures for itself or suggest corrective measures to its teammates. Expected values are established prior to team formation or are agreed upon through negotiation after the team is formed.

Both individual agents and groups of agents will make use of values, obligations, norms, beliefs, desires, and intentions, but only the architecture at the agent level explicitly incorporates these in our individual agent architecture named VONBDI. For example, social groups of agents do not have beliefs, but rather make use of the fact that the individuals comprising the group have beliefs. At the inter-agent level, the architecture is concerned with influence between agents.

2.1.3 VON-BDI Agent Architecture

In this subsection we present an architecture for an individual agent. This architecture relates the interaction feedback loops we saw earlier in greater detail. In addition to the three feedback loops attention, conversation, and obligation that were abbreviated in a single feedback loop (shown as one feedback loop in Figure 2), Figure 4 shows the following feedback loops that are internal to an agent: belief, intention, and role. These feedback loops are known as revision functions [Hexmoor and Beavers, 2002]. In all, six

---

and agent B are both on the same team as agent C, then “agent C ought to cooperate with agent A” and “agent C ought to cooperate with agent B” are distinct obligations. Distinct obligations give rise to distinct modal operators in order to accommodate conflict of obligation without having the logical system degenerate into triviality.

feedback loops are shown as rounded comparator circle attached to each[3]. Desire does not have a revision function. This is partly because inconsistent desires are allowed and the set of desires changes less often. Desires are reassessed when intentions and beliefs change. Revision functions monitor prevailing conditions and maintain an agent's commitments by implementing revisions only when necessary. For example, the attention revision function has an agent maintain its attention on particular agents until the condition that justifies the attention no longer obtains. Planning, like desire determination, is a function and not a revision function. This is because intention revision is responsible for guiding planning when it becomes necessary [Lacey, et al 2002]. In addition to the intentional notions of Belief, Desire, and Intention (BDI), Value, Obligation, and Norm (VON) are three notions that will prove useful in adding social properties to artificial agents. Taken together, they guide an agent's high-level behavior and help to provide a level of predictability, accountability, and responsibility. Values are understood as principles that govern the agent's behavior and which the agent will attempt to uphold as end-goals. Likewise, norms yield default behaviors that the agent is expected to observe whenever the agent finds itself in a situation to which the norm applies. We invoke a function that maps an agent **A**, a set of currently imposed values V, a set N of currently active norms, and a set of current Beliefs B to a set of obligations for the agent in that situation: f: **A** × V × N × B → {o1, … , on}.

Figure 4 shows the salient relationships among VON-BDI concepts. These relationships are incorporated in an algorithm provided later. For instance the revision function for attention revision has obligations and intentions as inputs, and produces cues for the agent to be aware of certain other agents. Intentions are derived from the states of VON-BDI as determined by an intention revision function. In the model being developed all the elements of VON-BDI play a part in the determination of the revised set of desires, since it is reasonable to assume that current desires will influence future desires as will beliefs, intentions, values, obligations and norms. It is standard to require that beliefs, desires and intentions all be consistent sets. Unlike intentions, which agents normally attempt to keep consistent, agents do not require their desires to be consistent and thus our model differs from common BDI systems, by having distinct modal operators for distinct desires. Beyond that, social forces provide an influence on an individual agent's desires through obligations. Agents with plans might enter negotiation with other agents about obligations and roles. We discuss negotiation in a later section.

[3] Figure 4 contains an abbreviated rendering of Feedback loops with the comparator circle gathering inputs. The feedback arc in each loop is not shown for brevity.

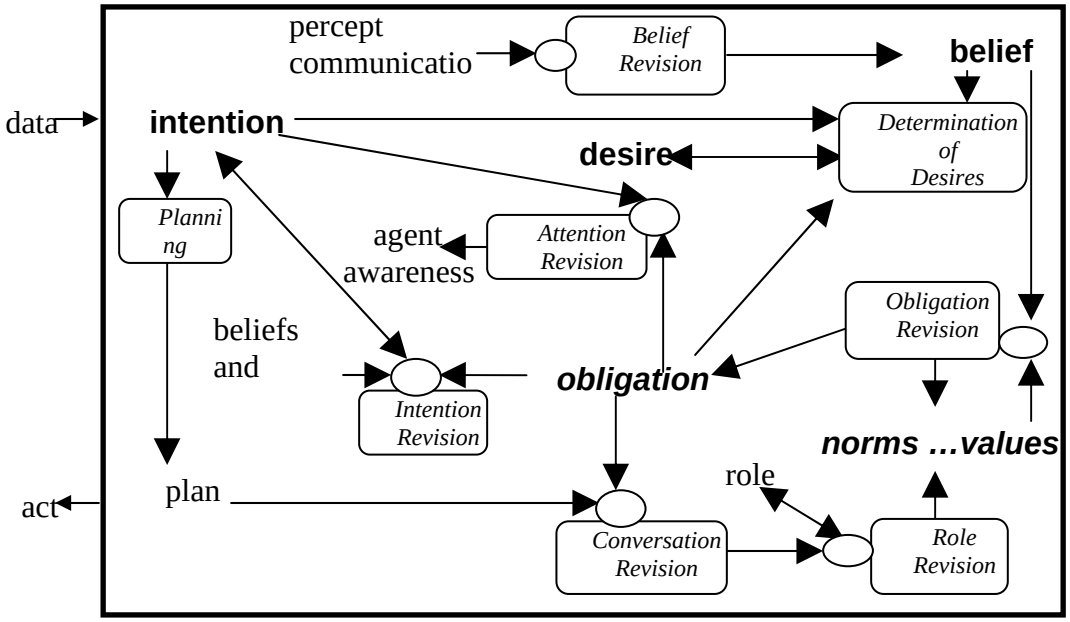

Figure 4 Intra-agent notions

2.1.2.2 The Algorithm

The following algorithm introduces a flag whose value is determined concurrently with, but outside the algorithm. α := sound(π, I, B) checks the current plan for consistency with current beliefs and intentions. The function that sets this flag is envisioned to be running continuously in the background. When a flag is set, that value is communicated to the process in the foreground, namely the algorithm below. All revision functions carry consistency checks with them. This algorithm extends current BDI approaches in two ways. First, we have introduced an obligation revision function (step 5.) that updates the agent's obligations against its norms and values and in light of new percepts. Obligation revision considers the effects of the current beliefs on values and norms. Our second extension is to make Wooldridge's *options* function and *intention revision* function (steps 9 and 10) account for the influences of obligations. We have added an attention revision function (step 7) and a role revision function (step 8). We have changed the "filter" function to a revision function over intentions.[4]

[4] Space does not permit us to explain the revision functions in detail.

2.1.2.3 Interagent Sociality

Agents that work together must reciprocate in order to reach equilibrium levels of sociality [Walsh and Wellman, 1999]. This means agents must adjust their own social attitudes in order to experience a sense of fair exchange. Game theory calls agent actions that are equilibrium inducing, *policies*. We borrow this notion from game theory to refer to an agent's mental attitude about social relationships.

```
1. B = B0;
2. I := I0;
3. while true do
4.        get next percept ρ;
5.        O := orf(O,ρ);              // obligation revision, O is obligation
6.        B := brf(B, ρ);             // belief revision, B is beliefs
7.        A := af(I, O);              // attention revision, A is agent awareness
8.        R := rrf(C, r);             // role revision
9.        D:=options(B, D, I, O);     // determination of desires, D is desires
10.       I := irf(B, D, I, O);       // intention revision, I is intentions
11.       π := plan(B, I);            // plan generation
12.       C := crf( π, O),            //conversation revision, C is conversational Context
13.       if α is true execute(π) else re-compute π
14. end while
```

Figure 5. Deliberation algorithm

2.2 Inter-agent Negotiation

Nick Jennings has identified three components of negotiation. First, *negotiation protocols* are the set of rules that govern the interaction. These protocols determine who may participate, what states may exist, and how transitions between states area to be accomplished. Second, *negotiation objects* are the issues over which agents negotiate. These may include price, timing, etc. Third, *reasoning models* are used for decision-making [Lomuscio, et al 2000].

In order for agents to collectively arrive at levels that maximize teamwork, the negotiation objects include suggestions to one another and the group for adoption of norms and values, as well as offers of services, that, if adopted, can be beneficial to the group.

Although the field of agent negotiation is rich and very active, very little has been put forth to account for cross-talk during negotiation. For example, on the floor of the New York Stock Exchange, many unmediated, distributed negotiations occur where traders don't wait for complete exchanges and talk past one another. Here we suggest a variation of the well-known contract net protocol to account for unmediated distributed

negotiation. Instead of bidding being collected centrally, each agent individually puts forth its available services and suggestions for group improvement.

For simplicity, initially we make the following assumptions.

- All agents share and understand the negotiation protocols.
- Communication languages used are perfect and no loss of important information occurs.

We will relax these assumptions in the course of our research. Here we will sketch a very simple negotiation scheme that is based on the contract net [Smith 1980]. In contract net, a task is announced, agents bid on the task, and the lowest bidder is awarded a contract. In our variation, agents announce the perceived need for improvement in the form of suggestions. An example is the providing a service required to achieve a group goal. Agents capable of providing the service "bid" to provide the service, with the low bid from a qualified provider being accepted. We will develop a negotiation scheme that allows for asynchronous negotiation.

2.3 Design of Agent Responsibility and Work Toward Guarantees

Responsibility can be seen to span a continuum between moral, legal and practical extremes. A moral responsibility is a commitment to uphold a set of values. The values governing the responsibility might be in the context of an institution. Institutions may promote legal as well as moral responsibilities. In multi-agent systems, a social contract theory is commonly used to generate moral responsibilities. A practical responsibility is based on expectations among agents. Practical responsibility is inspired by task responsibility [Baier, 1980]. In the extreme case, it is merely between two individuals who make some kind of agreement about actions or mental constructs such as shared beliefs. Many forms of responsibility lie between the two and might involve an organization or a community.

Practical responsibility arises from interactions between agents. For example, if agent A delegates a task to agent B and B accepts it, there is a delegation relation between A and B represented as A deleg→ B. Reciprocally, there is a responsibility relation between B and A represented as B resp→ A. This kind of responsibility can be conceived of as an obligation for agent B and a dependence relation toward agent A. I.e., B now has an obligation to carry out the delegated task and A depends on B to carry it out. In a team setting, teammates can be expected to develop practical responsibility relationship among themselves. They may also form responsibilities to uphold the ideal of a team, that is to uphold the norms and values of a team. These responsibilities are practical but also have the flavor of legal responsibility.

Naturally, conflict is possible between different types of responsibilities, and humans' ability to resolve conflicts lies partly in their ability to prioritize categories of responsibility. This ordering reflects the agent's ontological categorization of responsibility types. By having responsibilities in different categories, not only can an agent impose priorities among responsibilities, but the agent can also indicate different

methods of dealing with different responsibilities. People who value personal commitment will favor their practical responsibilities whereas people, who value their moral stances, will favor moral obligations. As designers of agent systems we will design mechanisms for encoding the desired ontological level to match the agent's responsibility level. This design-time responsibility encoding is a method that can be used to assure predictable agent behavior. If we design obligation categories (i.e., responsibilities within ontological levels for the agent), an agent might be directed to adopt specific obligations about certain tasks to perform on behalf of a chosen agent or the human user in case the agent interacts with a human. This will affect the agent's autonomy and control with respect to the agent (or the user). For example if the project is safety-critical, overall project goals are given a higher ontological status in the agent. Issues of safety inclusion have been explored in other work such as [Jenkins, et al, 1997] where designer concerns were automatically elicited and logged for ease of bookkeeping. Here, we are including responsibility at the architectural level of the agent to make each agent accountable for its behavior instead of having accountability be a system wide property.

Our legal system holds the owner of an agent responsible for the actions of that agent; therefore, agents capable of considering their responsibilities could offer some protection to the owner of the agent. Likewise in a command and control situation, a commander is responsible for the actions of the agents under his/her/its control and therefore would have greater confidence in responsible agents capable of considering their obligations. The model suggested here allows agents to consider their individual responsibilities and thereby the model makes it possible for agents to account for their actions. Having responsible agents will provide a safeguard to the owner of the agent as well as help agents arbitrate practical actions and to recognize legal violations by other agents.

Values and norms guide the behavior of the agent through the generation of particular obligations. Responsible agents are true to their principles, obey the norms of behavior in specific situations, and take their obligations seriously. Varieties of responsibility include *responsibility to* (concerning an agent's obligation to perform an action), *responsibility for* (concerning an agent's obligation to see that a state of affairs obtains), *character responsibility* (concerning the agent's obligation to behave in accordance with its values), and *norms* (concerning the agent's obligation to behave according to a pre-established standard), and *interpersonal responsibility* (concerning responsibly derived from obligations of one agent toward another). Whereas responsibilities tend to restrict the agent's choices, rights leave certain choices open and thus can be used to explore the limits of actions permitted to the agent. The relationships between rights and responsibilities regulate the agent's commitments.

So far we have been discussing how agents may reason about their responsibilities and how we might guide that at design-time. It many environments it is desirable to offer guarantees that constrain agent behavior. These can be thought of as prohibitions or the opposite of rights. Guarantees are far stronger than responsibilities. Guarantees are

intended to be offered to end users of the system, and the agents will not reason about them or have the ability to change them.

Figure 6 illustrates the relationship between some social notions. The agent internally reasons about its values and norms, which leads to its adoption of obligations. That type of consideration influences the agent's social relationships in various ways. We choose to focus on one of the obligations, namely responsibility. Obligations affect the agent's dependence as well as autonomy. We have argued that autonomy depends upon ability and social permissions (which are influenced by obligations) [Beavers and Hexmoor 2001b, Hexmoor2002, Hexmoor 2001a, Hexmoor 2001b].

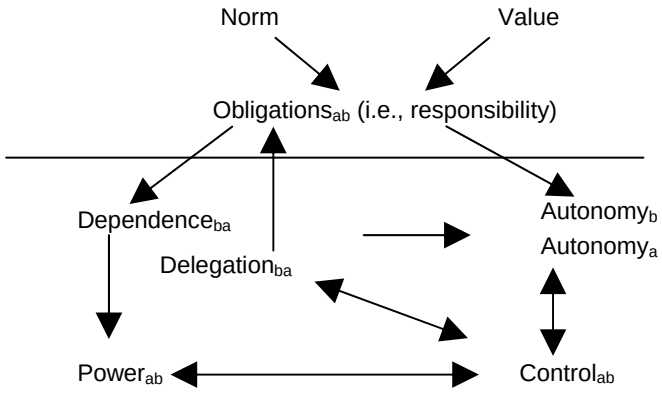

Figure 6 Exploring relationships for guarantees

We envision three possible approaches that can be used for building predictable behavior. Each approach focuses on manipulating a different social attitude in Figure 6. The first method we consider is to parameterize control. Consider a sphere of social control between two agents in which one agent sets goals and monitors the other agent. Control can be designed to be at various levels, e.g., master-slave, supervisory, or recommender levels[5]. Under the right circumstances, the tighter we set the control the more we can rely on the subordinate agent's behavior. The controlling agent is responsible for the behavior of the other agent. A second approach is to manipulate power levels. Everything else being equal, if two agents have a differential power relationship, they can affect one another's behavior. Command and control authority relationships are one example of established power relationships. By setting agent **A** in charge of agent **B** (i.e, explicitly establishing power relationships), **A** indirectly controls the actions of **B** (i.e., a

[5] We assume the right circumstances exist for this condition. We are not arguing for any one condition but only that these conditions must exist. This might require agents to be benevolent and to willingly agree to adopt control relationships.

control relationship is implicitly established). The control is indirect since we assume it is induced by direct manipulation of power levels.

Value and norm adjustment is a third method we are proposing. Although this is the least direct method of controlling behavior, it can be used to design an agent who will uphold certain general principles, e.g., Azimov's three laws of robotics constrain robot behavior.

4 Related work

In this section we will review related funded work that address adaptivity among agents with emphasis on teamwork. We will start by review of a specific DARPA program followed by review of a few NSF funded efforts. The most related DARPA program is ITO's Autonomous Negotiating Teams (ANTs) (BAA #00-24) with Dr. Robert Laddaga, and then Dr. Janos Szitpanovits as managers. The objective of ANTs was to encourage the development of scalable, distributed resource management systems capable of autonomously and efficiently negotiating the assignment of resources to tasks in real-time with real-time goal assurance. We will limit our reviews of ANTS activities to two activities conducted at Kestrel and Vanderbilt.[6]

Kestrel Institute has contributed to the ANTs project by providing a basic mathematical framework suitable for the high-level description of self-adaptive software the purpose of which is to observe the performance of the ANTs and attempt to adapt and improve the algorithms used by the ANTs based on those observations. Formalized comments are to be introduced as objects in programs [Pavlovic, 2001]. An adaptive interpreter, in conjunction with an automated specification engine, theorem provers, and code generators, are to test results and behavior for compliance with the formalized comments. The results are to be used to generate improved code. It would be possible to update the formalized comments during execution, perhaps on the basis of specification compliance and revised code. Both the specification and code need to be able to adapt to a dynamic environment that results in unpredictable changes. A program thus caries its own specification and thereby allows for automatic monitoring of correctness, reliability, safety and liveness properties of the program. As an example, consider a program that improves its performance by monitoring the characteristics of the input data and adjusting the algorithm to better fit the data. A program might also adjust its code based on the performance of other programs and components of the system. Application programming interfaces, APIs, would need to contain sufficient information about the structure and behavior of the given program to enable other programs to adjust their behavior with respect to the given program so as to allow optimization of the system.

Kestrel's project is ambitious and visionary and will require considerable inventiveness in software engineering to be able to synthesize code on the fly while

[6] Many other DARPA programs such as SDR, CoABS, MICA, UCAV are relevant but space limits us from including a review of these programs.

maintaining a specification of its functionality and its certificate of correctness. Kestrel provides the basic mathematical description of an abstract category of specification carrying modules that is intended to be the theoretical foundation for innovative self-adaptive software.

In contrast, the Institute for Software-Integrated Systems at Vanderbilt University has approached self-adaptive software very differently than Kestrel. They demonstrate how techniques invented by classical control theorists can be utilized in the design and implementation of self-adaptive software [Karsai, et al 2001]. The Vanderbilt approach utilizes a hierarchical architecture, with optimization, fault management, and system reconfiguration being handled by a higher-level component. The adaptive control approach introduces a second higher-level of control (supervisory level) as the adaptation mechanism and assumes that the adaptation mechanism should be explicit and distinct from the main processing (ground level). The claim is made that introducing the supervisory level allows the system to be robust and flexible. Flexibility is achieved by having the designer build into the ground level, optional configurations that give the ground level the capability to react appropriately to changes in the environment. The supervisory level monitors the ground level configuration, its output, and current environmental conditions and determines whether or not the ground level should be reconfigured. Robustness is achieved by including explicit fault accommodation in the supervisory level. One of the problems yet to be solved is how to efficiently represent and manage the very large configuration space that is the result of allowing reconfiguration. The suggested solution is to represent only the components and the current configuration, and to generate representations of other configurations only as needed. Another complication is that the components of the ground level must be designed to be independent of one another so that they may be removed from, or inserted into the configuration of the ground level without disrupting the performance of the other components.

The Kestrel and Vanderbilt approaches to self-adaptive software stand in stark contrast to one another. The Kestrel approach has specifications built into the code at a single level and the specifications are modifiable so that the system can adapt in ways unforeseen by its designers. The Krestrel approach is truly adaptive and visionary, but little more than ideas at this stage and therefore will take considerable time to fully implement. The Vanderbilt approach is hierarchical and the code can be reconfigured but not modified, and thus is not truly adaptive. The designers must foresee every eventuality and code the proper response in advance. On the other hand the Vanderbilt approach can be implemented today.

Milind Tambe's recent work include a monitor for distributed teams that minimizes the amount of required communication through having the monitor infer the progress of team plans by observing the actions of member agents. Tambe's Overseer is a good example of agent assistants proposed in [Beavers and Hexmoor, 2001c]. Inferring agent states from team actions suffers from uncertainty and from growth of computational complexity that is exponential in the number of agents being monitored. By modeling the

group as a whole, rather than modeling the individual agents constituting the group, Overseer is able to effectively predict team responses during normal and failed plan execution in linear time. The gain is accomplished through restricting attention to coherent monitoring hypotheses. The trade off is that the team is not accurately modeled in some failure modes because an inaccurate, but efficient, temporal model is used. Knowledge bases containing the plan hierarchy, and containing the group hierarchy enable *socially attentive* monitoring. Tambe's approach blends reasoning about teamwork and reasoning about uncertainty.

Next we turn to work that is conducted jointly CMU and Microsoft. [Espinosa et al 2000, Cadiz, et al, 2001] considers experiments with an awareness tool to aid distributed, asynchronous groups in cooperative decision making. An awareness tool keeps track of information such as who has seen what information, who has examined which options, and what lines of reasoning have been used by whom. Groups using the tool tend to converge to a solution more quickly, however, group without the tool tend to come closer to the correct solution. Two types of awareness were considered: awareness of the group's collective long-term memory, and awareness of teammates, which consists of resource awareness, task-socio awareness and chronological awareness. Chronological awareness involves keeping track of who has done what when, which contributes to efficiency by avoiding duplication of effort, reduces the need for communication, and helps a team develop a shared mental model.

Chrysanthos Dellarocas' work [Dellarocas 2000] advocates the use of Contractual Agent Societies to implement fluid, open organizations capable of dynamically reconfiguring themselves from heterogeneous components. Such societies require the negotiation and communication of the shared social interaction context, which should include social control mechanisms to ensure stability. To test some his ideas, Dellarochas has created an electronic marketplace with various agent types including investor agents, stock intelligence agents, matchmaker agents, notary agents and reputation agents. This society allows investors to adapt their behavior based on the history of an agent's prior transactions. The behavior of the investor agents can be adaptive, although the structure of the society and the roles of the various service agents (intelligence, matchmaker, notary, and reputation agents) remains fixed. Dellarocas plans to make his agents able to negotiate details of transactions in future work.

Although Barbara Grosz calls her work "intention reconciliation" it is really only making a choice. Reconciliation would imply some kind of revision of intentions rather than merely making a choice between two options. Grosz and her students at Harvard have conducted simulations in which individual agents revise their intentions based not only on expected future income, but also on social rewards modeled as "brownie points". In their simulation, agents working together to achieve a group goal are assigned tasks leading to the goal. The tasks vary in their importance relative to the goal, with the more important tasks being assigned to those agents who have proved to be more reliable in the past. Some agents are presented an opportunity to accept higher paying tasks that require abandonment of the task leading to the group goal. If the agent accepts the higher paying

task to the detriment of the group, the agent will suffer a degradation of reputation with a resulting lower level of responsibility within the group and consequent lower future income. The agent must balance immediate gain against probable future loss. In addition, the agent loses social status in the form of brownie points. The value of the brownie points is interpreted as a level of social consciousness. Optimal group utility, measured as the sum of the incomes of the individual agents, was obtained with an intermediate level of social consciousness, that is, when social factors had some effect on an agent's commitment to the group goal, but did not prevent the agent from taking advantage of high individual gain even though that gain would be detrimental to the group.

5 Conclusion

Awareness of norms and values provide contexts for self-adaptation. We have presented architectures for communities as well as agents to make use of norms and values in guiding adoption of intentional social relationships such as role and obligations. We considered issues of responsibility and predictability of behavior. Principled design of agents with explicit responsibilities allows us to expect interactions that are to some degree predictable and benign. We plan to apply our architectures in certain problem domains to explore the extent to which it is scaleable.

Acknowledgements

This work is supported by AFOSR grant F49620-00-1-0302.

References

1. K. Baier, Responsibility and Action, In *Action and Responsibility* Michael Bradie and Myles Brand (eds.) (Bowling Green, OH: Bowling Green University Press, 1980), pp. 100-116.
2. G. Beavers and H. Hexmoor, 2001a. Teams of Agents, In Proceedings of the IEEE Systems, Man, and Cybernetics Conference.
3. G. Beavers, and H. Hexmoor, 2001b. Adding Values, Obligations, and Norms to BDI Frameworks, Submitted to *Cognitive Science Quarterly*.
4. G. Beavers and H. Hexmoor, 2001c. Team Assistants. Submitted to *AAMAS-02*.
5. JJ Cadiz, G. D. Venolia, G. Jancke, and A. Gupta Sideshow, 2001. Providing Peripheral Awareness of Important Information, Microsoft Research Technical Report MSR-TR-2001-83.
6. C. Dellarocas, Contractual Agent Societies: Negotiated shared context and social control in open multi-agent systems. In workshop on Norms and Institutions in Multi-Agent Systems, *4th International Conference on Multi-Agent Systems* (Agents-2000), Barcelona, Spain, June 2000.
7. A. Espinosa, J. Cadiz, G. Lautenbacher, L. Rico-Gutierrez, R. Kraut, and W. Scherlis, 2000. Coming to the Wrong Decision Quickly: Why Awareness Tools Must be Matched with Appropriate Tasks, In Proceedings of the 2000 *ACM Conference on Human Factors in Computing Systems*.
8. H. Hexmoor, (In Press, 2002). In Search of Simple and Responsible Agents, In the Proceedings of *NASA GSFC/JPL Workshop on Radical Agents*, MD.
9. H. Hexmoor, (In Press, 2001a). From Inter-Agents to Groups, In *International Symposium in Artificial Intelligence, ISAI-01*, India.
10. H. Hexmoor, 2001b. A Cognitive Model of Situated Autonomy, In Advances in Artificial Intelligence, Springer LNAI2112 -pages 325-334, Kowalczk, Wai Loke, Reed, and William (eds).
11. H. Hexmoor, (In Press, 2001c). Adaptivity in Agent-based Systems via Interplay between Action Selection and Norm Selection, In 2nd *International Workshop in Self-adaptive software*, (IWSAS-01), Robert Laddaga, Howard Shrobe, and Paul Robertson (ed), Hungary.
12. H. Hexmoor, 2000. Towards Empirical Evaluation of Tradeoffs between Agent Qualities, In *PRIMA-2000*, (C. Zhang and V. Woo, eds), LNAI Volume 1881, Australia.
13. H. Hexmoor, 2000. Conversational Policy: A case study in air traffic control, In Proceedings of the *International Conference in Artificial Intelligence*, (IC-AI'2000), H. R. Arabnia, (ed), Las Vegas, CSREA Press.
14. H. Hexmoor and G. Beavers, 2001a. Towards Teams of Agents, In Proceedings of the *International Conference in Artificial Intelligence*, (IC-AI'2001), H. R. Arabnia, (ed), Las Vegas, CSREA Press.
15. H. Hexmoor, and G. Beavers, 2001b. Measuring Team Effectiveness. Submitted to *Applied Informatics* (AI 2002).

16. H. Hexmoor and J. Vaughn (In Press, 2002). Computational Adjustable Autonomy for NASA Personal Satellite Assistants, In *ACM SAC-02*, Madrid.
17. H. Hexmoor and X. Zhang, 2001, Norms, Roles, and Simulated RoboCup, In 2nd *workshop on norms and institutions in multiagent systems*, (Agents 2001), Montreal, CA, ACM press.
18. H. Hexmoor, and H. Duchscherer, 2001, Efficiency as Motivation for Teaming, In Proceedings of *FLAIRS 2001*, AAAI press.
19. H. Hexmoor and H. Duchscherer, 2000. Shared Autonomy and Teaming: A preliminary report. In Proceedings of *Workshop on Performance Metrics for Intelligent Systems*, NIST, Washington, DC.
20. D. Jenkins, D. Livingstone, D. Maclean, A. Reglinski, A., 1997. Supporting Safety-Related Projects with a Designer's Assistant, in Proceedings of the *1st International Conference on Autonomous Agents*, Marina Del Rey, CA.
21. G. Karsai, A. Ledeczi, J. Sztipanovits, 2001. An Approach to Self-Adaptive Software based on Supervisory Control, In 2nd *International Workshop in Self-adaptive software*, (IWSAS-01), Robert Laddaga, Howard Shrobe, and Paul Robertson (ed), Hungary.
22. N. Lacey, G. Beavers, and H. Hexmoor, 2001. Planning at the Intention Level, Submitted to *AAMAS-02*.
23. R. Lomuscio, M. Wooldridge and N. R. Jennings, 2000. A classification scheme for negotiation in electronic commerce in *Agent-Mediated Electronic Commerce: A European Perspective* (eds. F. Dignum and C. Sierra), Springer Verlag, 19-33.
24. D. Pavlovic, 2001, Towards semantics of self-adaptive software, *Proceedings of the Workshop on Self-Adaptive Software*, P. Robertson (Ed.), Springer Lecture Notes in Computer Science 1936, Berlin, 2001, pp. 50-64. *Kestrel Institute Technical Report* KES.U.99.4, 1999.
25. R. Smith, 1980. The contract net protocol: High-level communication and control in distributed problem solver, *IEEE Transactions on Computing*, Vol. C-29 1980, pp. 1104-1113.
26. W. Walsh and M. Wellman, 1999. Efficiency and Equilibrium in Task Allocation Economies with Hierarchical Dependencies, In The International Joint Conferences on Artificial Intelligence Workshop on Agent-Mediated Electronic Commerce, August 1999.
27. M. Wooldridge, 2000. Reasoning about Rational Agents, The MIT Press.