1

From Intentions to Social Commitments: Adaptation in Multiagent Systems

FABIANO DALPIAZ, AMIT K. CHOPRA, JOHN MYLOPOULOS, AND PAOLO GIORGINI

ABSTRACT. Runtime adaptation of software systems is an area of software engineering that is gaining increasing attention from researchers. Adaptive software can successfully cope with failures and the volatility of the environment it operates in. Researchers in software requirements and architecture are especially interested in model-driven adaptation. The idea is that software would reflect upon its own model in order to reason about adaptation.

This chapter represents a summary of our recent work for supporting adaptation in multiagent systems. Our key contribution is to exploit the notion of *social commitment* in order to reason about adaptation in multiagent systems. While other approaches for adaptation assume cooperation or some form of logical centralization, our approach works for open multiagent systems wherein agents are autonomous and heterogeneously constructed. We conceive adaptation from the perspective of an intentional agent, who needs to interact with other agents in the multiagent system—by engaging in social commitments—to achieve its goals.

0 An Allegory[1].

Imagine! You are an enterprise architect with HyperTech Industries Unlimited (“Hype”, for short) and a client comes along wanting to re-engineer all business processes for her multinational publications organization, Home Periodicals Inc. (hereafter “Hope”) to eliminate redundant positions while improving performance. Your mission—should you decide to accept it—is to re-engineer these business processes in consultation with Hope staff and, most importantly, keep them happy and off your boss’ back.

Now, you happen to be a recent graduate from a Computer Science program and you are aching to use all these great ideas you learned back in college. So, you dig up your course notes from your home basement and proclaim at the next meeting with your boss and your client that you intend to go about this using the latest work from the best minds in the field. Ignoring the anxious looks on their faces—and the sneers behind your back among your fellow architects the following day—you begin your task going over all your material, reading up references and taking notes.

At the end of your search, you proudly present your boss with your findings. There seems to be unanimous agreement among experts, you note, that existing business process specification techniques—based on old and well-tried system modeling concepts such as finite state machines, interaction diagrams, and Petri nets—lead to inflexible, procedural descriptions of what’s to be done. Since human activity is intrinsically situated and emergent, such specifications lead to serious discrepancies between what is specified and what

[1]Adapted loosely from [Mylopoulos 1992]

actually happens. Such specifications also fail with respect to flexibility, as they are difficult to customize to user preferences, or adapt as the environment changes. While your boss waits patiently, you point out that there seems to be no consensus on how to deal with these fundamental problems. Some organizations fall back on informal specifications of their systems, whereas others adopt simpler diagrammatic techniques (flow charts and so on) that leave out important details. Sketches of processes (formal or informal) are fine sometimes, you remark, but they often lead to misunderstandings among Hope employees, and suboptimal performance in the execution of its business processes. Your boss (who, you just notice, bears remarkable resemblance to Dilbert's Pointy-Haired boss[2]) shakes his head knowingly. Following your carefully laid out script, you draw an analogy between what current business process specifications practices do and the proverbial drunk, who late one night is looking under a street lamp for his keys, lost elsewhere, because there he can at least see.

As your boss becomes restless, you get to the punchline. According to your recently completed multiagent systems course, the key concepts for talking about multiagent systems (which business processes surely are) are *agent* and *social commitment*. A social commitment consists of one agent x committing to another agent y to fulfill proposition q if p (often, p would be something that y would have to bring about). Such atomic contracts can be used to model complex interactions, such as those involving the execution of a multiparty process. To specify such processes, we need to abstract away from the concept of agent to that of a *role*. Then, system specifications consist of generic commitments among roles. These can be instantiated and enacted whenever concrete agents are bound to all the roles that are part of a specification. With this approach, you conclude while waiving didactically your longest finger, we avoid the pitfalls of current process specification techniques. Multiagent system specifications based on commitments are neither procedural nor do they talk about tasks, activities, and plans. Instead, they specify the social expectations (via social commitments) that would arise from the agents' interactions.

Your boss stops looking at the ceiling and is now staring directly into your eyes (and yes, he does look Pointy-Haired!). "Could it be that this guy is on to something?", he wonders. The tools offered for commitment-based specifications, you continue, are based on ideas from multiagent systems. The key concern is to change the nature of multiagent interaction specifications so that they are social models, rather than procedural ones.

Your boss is ecstatic. He has recently watched the movie *The Social Network* and is fleetingly favorable towards all things social. "Here is another wacky idea", he thinks, "but it actually sounds better than all the others we have been peddling to our customers. Let's try it!" He gives you his blessing and you give him a good book on multiagent systems ([Singh and Huhns 2005]) for background reading before embarking on the Hope project. For the rest of the week you are definitely on the good side of your boss' balance sheet as you rock-and-roll between Hype and Hope. And it's all thanks to multiagent systems. . .

1 Introduction

In the age when the department of Computer Science at the University of Toronto was in its infancy, Hector Levesque had a trail-blazing career that took him from junior undergraduate to a faculty position within the span of a little more than a decade. Hector was the undergraduate who asked—by correspondence—Marvin Minsky when he had questions about Artificial Intelligence (AI). And it was him who put the instructor of the AI course to task

[2]http://www.dilbert.com/

for not using Lisp as the programming language taught in the course[3]. His MSc thesis was inspired by Lisp in defining a self-descriptive knowledge representation language[4]. And as a PhD student he taught his supervisor things (modal logics, and more), rather than the other way around, as he was working towards an epistemic account of knowledge bases. Among other ideas, that thesis used a functional account of knowledge bases [Levesque 1981] that was later adopted, first by KRYPTON [Brachman et al. 1983] and then by Description Logics. And all that was a preface to a sparkling research career.

Multiagent interactions among human, organizational and artificial agents account for much of the social activity of modern world, be they business processes, socio-technical systems or collaborative efforts. To better manage and support such multiagent interactions, computer scientists have used a variety of specification techniques—often based on Petri nets, finite state machines and the like. Unfortunately, such techniques are too rigid, prescriptive and inflexible to describe human activity in two orthogonal ways. Firstly, they don't account for agent autonomy. Secondly, they are static and difficult to dynamically adapt in the face of varying user preferences and environmental settings.

A multiparty interaction (business process or otherwise) is above all else a social activity. During such an interaction, participants care about whom they interact with; what contractual relationships arise from their interactions and if they have recourse in case others violate their contractual obligations; whether their goals are likely to be met by participating in the activity; whether they can interact flexibly and adapt in appropriate ways when necessary.

This paper adopts concepts and ideas from the literature on multiagent systems to address the problem of *agent adaptation*—an area Levesque has explored in the past [Kumar et al. 2000]. Levesque's work relies upon the notion of *internal commitment* to intentions. By contrast, our proposal builds upon ongoing work by Munindar Singh and his group on the specification of multiagent interactions in terms of *social commitments*. Social commitments, as we show later, are distinct from internal commitments.

Organization. Section 2 contrasts cognitive abstractions against social abstractions. It introduces a key social primitive, that of commitment among agents. Section 3 introduces a conceptual model of adaptation that takes into account social commitments. Section 4 introduces an architecture for adaptive agents. Section 5 summarizes the paper and concludes with directions for future research.

2 From Intentions to Social Commitments

Broadly, we understand an agent as an active autonomous entity that acts according to its own motivations. Typically, an agent would not be able to achieve all of its goals by interacting with inactive objects in the environment. Rather, it would need to interact with other agents and enter into social relationships with them to achieve its goals. The interactions give rise to a multiagent system.

Multiagent systems may be specified a priori. Such specifications would prescribe the legal interactions among the agents, not with reference to specific agents however, but with reference to *roles*. We call such multiagent system specifications *protocols*. An agent could then adopt a role in a protocol if doing so suited its goals. For example, if Barbara wanted to sell her cellphone, she may consider adopting the role *seller* in some auction protocol.

[3]The University of Toronto didn't offer any form of interactive computing at the time, so Lisp was out of the question.

[4]Elements of that thesis were published in [Levesque and Mylopoulos 1979].

Researchers in multiagent systems are keenly concerned with protocols and agents and the abstractions in terms of which they can be specified. Generally speaking, computer science has tended to address more the challenges of high-level agent specification rather than high-level protocol specification. This is likely due to traditions from AI, where agents are viewed as intelligent computational entities capable of reasoning in terms of cognitive (mental) abstractions such as beliefs, goals, intentions, and obligations [Bratman 1987].

Cohen and Levesque's work [1990] on cognitive abstractions has been especially influential over the decades. They formulated a rich theory of rational action based on the concepts of *intention* and *internal commitment* to intention. For an agent to succeed with its intentions, it must be internally committed to realizing them. Further, the agent should not be overcommitted or undercommitted to the intentions as that would produce undesirable results.

Another widely influential strand of research, following research in the philosophy of language, concerns itself with the semantics of communication. Searle's account of speech act theory [1969] categorizes communications into types such as *assertives*, *commissives*, and so on, and formalizes each in terms of the mental states of the communicating agents. For example, if Barbara asserts to Alice "It is raining outside", it would be taken to mean that Barbara actually believes it is raining outside. If Barbara commits to all bidders to honor the winning bid for her cellphone, it would be taken to mean that she intends to honor the winning bid. In effect, speech act theory gives meaning to communication—an observable social phenomenon—in terms of the private mental states of agents. In a similar vein, Cohen and Levesque [1990] extrapolated internal commitment as being foundational for understanding communication among agents. We shall refer to this class of approaches as *mentalist*. The mentalist approach to communication influenced greatly the agent-oriented programming paradigm [Shoham 1993] and the attempts to standardize agent communication languages (KQML [Finin et al. 1994] and FIPA ACL [FIPA 2003] being the prominent ones).

Meanwhile, another strand of research focused on a convention-based view of communication, where agents interact with each other on the basis of public conventions [Winograd and Flores 1986]. Singh [1991] argues that *social commitments* are distinct and orthogonal to internal commitments, and they are intimately tied to public convention. By contrast, internal commitments, as their name suggests, are private to an agent. Let us return to our example to illustrate the difference. On the one hand, Barbara may be internally committed to honor the winning bid, but may not have communicated so to the bidders; in other words, even though Barbara is internally committed, she is not socially committed. On the other hand, Barbara may be socially committed to honor the winning bid (because she communicated that to the bidders) but not internally committed to do so (maybe because she has a minimum price in mind). Singh pointed the importance of keeping the two notions separate: while social commitments could be used to formalize convention and communication, internal commitments could be used to design specific agents.

Singh [1998] expanded on the argument against mentalism to show that speech act theory could not form the foundations of agent communication. He argued against the approach taken by KQML and FIPA ACL by showing that they led to unrealistic assumptions about the nature of multiagent systems. For example, assumptions about the sincerity of agents were unrealistic because one agent could not possibly know what another intended. In particular, such assumptions violated the autonomy of agents, and limited multiagent systems only to systems of cooperative agents. A recent joint manifesto [Chopra et al.

2011] by researchers in agent communication

- *affirms* the necessity of a social semantics for communication,
- *affirms* social commitments as a key part of any such social semantics, and
- *rejects* mentalist semantics and approaches based thereupon such as the FIPA ACL.

The distinction between social and internal commitments is crucially relevant to the design of multiagent interactions, where agents could be organizations, humans, or their software surrogates, with distinct, private, and often competitive motivations. These entities interact with each other on the basis of public conventions, not on the basis of internal commitments or any other mentalist notion. These public conventions are in fact protocols.

Cognitive abstractions have a place. However, this place is not in formalizing communication; it is in capturing agent intentions. Such intentions lead to communication and account for its purpose. For example, Barbara's desire to buy a cruise ticket may be the motivation for selling her cellphone, but her desires cannot explain the social effects of actually communicating the offer to others. Chopra et al. [2010] offer an account of the reasoning that connects goals with social commitments.

A social commitment is of the form $\mathsf{C}(\mathsf{debtor}, \mathsf{creditor}, \mathsf{antecedent}, \mathsf{consequent})$, where debtor and $\mathsf{creditor}$ are agents, and $\mathsf{antecedent}$ and $\mathsf{consequent}$ are propositions [Singh 1999]. From now on, we will simply use *commitment* to refer to social commitments. Other kinds of commitments, for example internal, will be appropriately qualified. $\mathsf{C}(x, y, r, u)$ means that x is committed to y that if r holds, then it will bring about u. If r holds, then $\mathsf{C}(x, y, r, u)$ is *detached*, and the commitment $\mathsf{C}(x, y, \top, u)$ holds ($\top$ being the constant for truth). If u holds, then the commitment is *discharged* and doesn't hold any longer. All commitments are *conditional*; an unconditional commitment is merely a special case where the antecedent equals $\top$. (For general rules of commitment reasoning, see [Singh 2008].)

For example, $\mathsf{C}(Barbara, Alice, paid, deliverPhone)$ means that Barbara commits to Alice that if payment is made, the phone will be delivered. When the payment is made, then the unconditional commitment $\mathsf{C}(Barbara, Alice, \top, deliverPhone)$ holds.

Commitments are rooted in communication. A commitment can be created or otherwise manipulated only by explicit communication among agents. The primitive messages for manipulating commitments are *Create*, *Cancel*, *Release*, *Delegate*, *Assign* [Singh 1999].

The notion of commitment protocols [Yolum and Singh 2002] has been highly influential in multiagent systems. The basic idea there is that application-specific messages can be assigned meanings (by a protocol designer or collectively by the application community) in terms of commitment-specific messages. Thus for example, in a particular community of fruit-sellers, a quote from a merchant to a customer may be formalized as creating the corresponding commitment:

Quote(m,c,item,price) counts as *Create(m,c,item,price)*

In another community, a quote may not mean any such commitment. The community provides the *context* of the commitment, which we omit here.

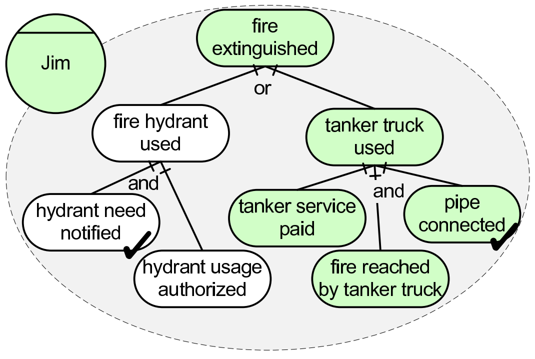
3 Conceptualizing Agent Adaptation

Adaptation in agents has largely been confined to two kinds of settings. One considers only a single agent; in other words, it does not consider interaction. The other considers multiple agents but in the context of a cooperative setting. Levesque, in joint work with others [Kumar et al. 2000], formulated adaptation in *team* settings. Essentially, a team of brokers has to ensure the fault tolerant operation of a system of agents (via services such as locating services, routing, and so on). The key feature of their approach is the establishment of *team commitments* and individual internal commitments among the brokers. For example, the brokers adopt the team commitment that if a client loses connection with a broker, the brokers will attempt to reestablish connection with the client. Further, if a registered client is disconnected, each broker would have the internal commitment to make that believed among the team, and so on.

Our approach to adaptation instead relies on *social* commitments. As such, it applies to cooperative as well as competitive systems. We characterize an adaptive agent in terms of both intentional concepts—the *goals* it wants to achieve—as well as social concepts—the social commitments it makes to or takes from other agents. We aim to ensure that a specific agent achieves its goals in a dynamic environment where failures and under-performance are common events. Although we do not talk explicitly about teams, extensions of our approach should be able to support teamwork as a special case. In general, our approach for agent adaptation should work in any open multiagent system, that is, a system where the agents are autonomous and independently designed.

We conceptualize adaption in terms of changing strategies in order to achieve a goal. Hence, the adaptive agent in our approach is goal-driven. Essentially, the agent monitors his goals, and depending on environmental conditions and interactions with other agents, adopts new strategies to achieve the goal if necessary.

We illustrate our approach on a scenario concerning firefighting, where Jim, a fire chief, has to decide how to efficiently respond to a fire. Jim's goal model is shown in Figure 1 using a subset of the concepts of Tropos [Bresciani et al. 2004]. Jim's top-level goal is "fire extinguished". This goal is OR-decomposed to indicate that there are alternative ways to achieve it. Jim can either use a fire hydrant or rely on a water tanker truck. Both goals are AND-decomposed. In order to use a fire hydrant, Jim has to notify this need and get authorized. To fight the fire with a tanker truck, the tanker service should be paid, the fire should be reached by a truck, and a water pipe should be connected to the truck. Jim's capabilities—goals he can achieve without interacting with other agents—are "hydrant need notified" and "pipe connected".



Commitments in the variant: C_2, C_3

Figure 1. Jim's goal model, emphasizing his current variant to extinguish the fire

Table 1 lists some possible commitments that may arise in the firefighting scenario. C_1 is made by fire brigade Brigade1 to Jim. Brigade1 commits that, if hydrant need is notified, then hydrant usage will be authorized. Jim commits (C_2) to the fire brigade that, if the tanker service is paid, then the water tanker truck will be used. Commitments C_3 and C_4 are made by different water tankers to Jim. Both commitments tell that, if the tanker service is paid, fire will be reached by the tanker truck.

C_1	C(Brigade1, Jim, hydrant need notified, hydrant usage authorized)
C_2	C(Jim, Brigade1, tanker service paid, tanker truck used)
C_3	C(Tanker1, Jim, tanker service paid, fire reached by tanker truck)
C_4	C(Tanker2, Jim, tanker service paid, fire reached by tanker truck)

Table 1. Some example commitments in the fire response scenario

The notion of *variant* is fundamental in our framework as it defines a common semantic substrate that characterizes a goal-oriented agent operating in a multiagent setting. The intuition is that, given a certain goal to achieve, a variant is a strategy that could lead to the achievement of the goal—if successfully carried out at runtime.

We provide the basic intuition behind the notion of variant. More technical details can be found in [Dalpiaz et al. 2010]. A variant refers to one or more goals—typically, top-level goals—the agent aims to attain. A variant consists of a set of goals $\mathcal{G}$ the agent wants to achieve, a set of commitments $\mathcal{P}$ the agent intends to make or take, and a set of capabilities $\mathcal{C}$ the agent plans to exploit. Roughly, an agent's variant $\langle \mathcal{G}, \mathcal{P}, \mathcal{C} \rangle$ supports a goal g if:

- the agent is capable of goal g, i.e. $g \in \mathcal{C}$;
- the agent takes a commitment (in $\mathcal{P}$) from another agent; the commitments' antecedent is supported by the variant and the consequent entails g;
- the agent makes a commitment (in $\mathcal{P}$) to another agent; the commitment's antecedent entails g;
- a goal supported by the variant contributes positively to g;
- g is AND- (OR-) decomposed and all (at least one of) the subgoals is supported.

EXAMPLE 1. Figure 1 shows the active variant for Jim's goal fire extinguished (the goals in the current variant are grayed, and the commitments in the variant are shown below the goal model). The leaf-level goals in the variant are all supported: pipe connected via Jim's capability for that goal, tanker service paid by commitment C_2 made to Brigade1, fire reached by tanker truck via commitment C_3 Jim takes from Tanker1. Thus, the AND-decomposed goal tanker truck used is supported. In turn, it supports the OR-decomposed goal fire extinguished.

4 Adaptive Agent Architecture

Based on our conceptualization of agent adaptation, we propose here an architecture for adaptive agents. Our architecture operates in accordance with the Monitor-Diagnose-Reconcile-Compensate (MDRC) control loop:

(M) **Monitor** collects data about the state of the environment and the agents participating in the system from a variety of sources;

(D) **Diagnose** interprets the data with respect to goal models to determine if all is well. If not, the problem-at-hand is diagnosed;

(R) **Reconcile** searches for a different variant that best deals with the problem-at-hand;

(C) **Compensate** defines and executes a plan that enacts the new variant.

We exemplify how the MDRC cycle works in the fire response scenario where agent Jim is playing role fire chief.

EXAMPLE 2. Jim monitors fire severity and evolution, wind conditions, traffic in nearby areas, as well as retrieving messages from other agents, e.g., firefighters and water tanker truck providers. Jim's diagnosis would involve verifying whether the collected information threatens his goals. For example, if a firefighter notifies he cannot come, Jim's goal to respond to the emergency might be in danger. Jim can reconcile this situation by identifying a new strategy, e.g., relying on water tanker helicopters. Ultimately, Jim may compensate the nonavailability of firefighters by calling upon the helicopters.

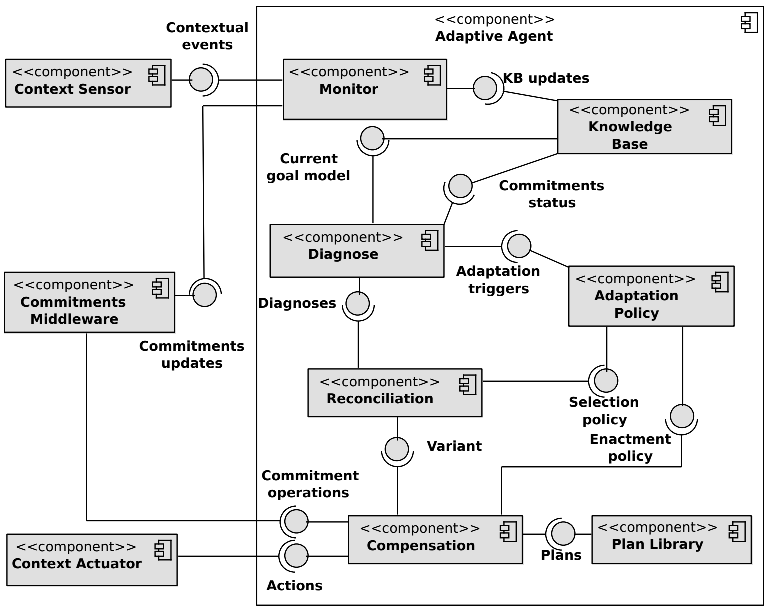


Figure 2. Logical view of the architecture for adaptive agents

The logical view of our architecture is shown in Figure 2. The architecture includes a number of external components the agent interacts with. *Context sensors* are computational entities providing raw data about the environment the agent runs in (e.g. to retrieve wind conditions, temperature, and traffic status). *Context actuators* represent effectors in the environment that can receive commands that modify the environment itself, e.g., sirens, loudspeakers, and door openers.

An essential external component of the architecture is a *Commitments Middleware*, the communication infrastructure that enables agents to interact in terms of commitments. The component allows for asynchronous message exchange between agents through an API that triggers a call-back method for every received message. Within this method, the agent defines how the message is handled according to its internal policy. Specifically, such middleware has to support the standard commitment manipulation operations [Singh 1998]. Below we list some messages the agents would typically exchange.

- *Create*(C_3): Tanker1 sends a message to Jim whereby C_3 is created.
- *Declare*(tanker service paid): Jim sends a message to Tanker1. As a result C_3 is detached and $\mathsf{C}(\mathsf{Tanker1}, \mathsf{Jim}, \top, \text{fire reached by tanker truck})$ holds.
- *Release*(C_3): Jim sends a message to Tanker1 telling that the truck is not needed anymore, thus releasing the provider from its commitment;
- *Cancel*(C_3): Tanker1 sends a message to Jim telling that the truck will not be able to reach the fire, as the truck was involved in a pile-up;
- *Delegate*(C_3): Tanker1 sends a message to another truck provider of a nearby town, thereby delegating its commitment to reach the fire.

In the next subsections, we detail the core components of the architecture that deliver the MDRC adaptive control loop.

4.1 Monitor and Diagnose

The purpose of the *Monitor* component is to detect relevant changes in the physical and social context. To collect events, this component relies on two sources: context sensors provide changes in the environment, whereas the commitments middleware furnishes the messages sent by other agents—in terms of commitments operations.

EXAMPLE 3. Jim's monitor receives *Cancel*(C_3) from Tanker1. Such message informs Jim that Tanker1 is canceling its commitment C_3, presumably due to an unforeseen traffic jam. Further, Jim's monitor has received data from a context sensor notifying it that there has been a car accident in the city center.

The collected data is then provided to the *Knowledge Base* component, which represents the agent's current knowledge. This corresponds to correlating collected data against the agent's goal model and the current commitments. The knowledge base provides these information through interfaces *Current goal model* and *Commitments status*.

The knowledge base does not analyze whether the agent's goals are at risk. Such activity is conducted by the *Diagnose* component. In particular, its role is to determine whether the current *variant* is adequate to achieve the agent's goals, if enacted correctly at runtime. While evaluating the adequacy of the current variant, the diagnose component takes into consideration the *adaptation triggers* specified in the agent's adaptation policy.

Adaptation triggers specify under what circumstances the agent should search for an alternative variant to achieve its goals. Various types of adaptation triggers exist [Dalpiaz et al. 2010] including the following:

- *Failure*. An agent may fail in using its capability (because the corresponding low-level procedure failed) or another agent may violate a commitment. For example, Jim's plan to deliver capability "pipe connected" may fail if the pipe socket is incompatible with the plug. The violation of a commitment is shown in Example 3, wherein Tanker1 cancels C_3 (by sending a message), which was part of Jim's adopted variant to extinguish fire.

- *Threat*. Threats are identified by specific agents on the basis of their own policies. A possible way to detect threats is to exploit risk analysis techniques. For example, if Tanker1 sends a message saying it could be late (it has not violated its commitment yet), Jim may interpret that as a threat to C_3. Whereas commitment failure is publicly observable, the evaluation of threats may be a matter of agent policy, i.e., one agent may consider a commitment threatened when another does not.
- *Opportunity*. Here monitored data suggests an opportunity to improve current performance. This kind of trigger constitutes an example of the proactivity of agents, as adaptation is stimulated by new opportunities rather than as reaction to failures or underperformance. For instance, if a new water tanker truck provider Tanker3 comes into play and commits to reach the fire in less time—its headquarters are close to the fire—Jim may check if this opportunity is exploitable. Deciding whether or not to adopt a new variant is up to the reconciliation component.

4.2 Reconcile and Compensate

Once the problem at-hand is identified and the root cause diagnosed, the architecture begins the reconciliation phase. The main activity here is to identify a new variant that is more likely to succeed than the current one. Specifically, the *Reconciliation* component takes as input the diagnoses as well as an agent variant selection policy, and returns the variant to be adopted.

After identifying possible variants, the agent has to select among them. Several criteria can be exploited and combined, such as (i) minimize cost expressed as money, resources, or time; (ii) preserve stability by minimizing change from the status quo; (iii) maximize quality, e.g. by considering soft-goals such as performance, security, and risk; (iv) choose preferred goals and commitments; (v) apply redundancy to ensure achievement of critical goals.

Element	**AC**	**CC**
hydrant need notified	6	4
pipe connected	17	8
C_1	15	7
C_2	8	0
C_3	13	20
C_4	31	16

Table 2. Activation (AC) and compensation cost (CC) for capabilities and commitments

Our architecture exploits a variant selection algorithm that takes into account cost and stability [Dalpiaz 2011]. As shown in Table 2, we associate cost values to capabilities and commitments. Activation cost refers to the effort required to exploit a capability and to make/take a commitment. Compensation cost represents the effort required to nullify/mitigate the effects of a capability and to cancel/release a commitment. Our selection algorithm considers the adaptation policy of the agent, which includes several factors:

- *Variant selection criteria* specify how to determine variant costs that determine selection. Two algorithms are supported: (i) *overall-cost* considers the overall variant cost, and selects the variant having minimal total cost; (ii) *delta* computes delta costs between a candidate and the current variant.

- *Variant exclusion factors* provide reasons why some variants should be excluded. An agent might want a different solution to avoid threatened capabilities and commitments. Also, failed capabilities and violated commitments might be excluded.
- *Compensation cost* represents how expensive it is for an agent to revert/mitigate the effects of a capability that is currently exploited and that will not be in the next variant. Compensation cost is considered when determining variant cost, if such option is in the policy.
- *Opportunity threshold* determines when an opportunity should be adopted. If the current variant is not at risk, the agent needs a clear incentive to switch to another variant. If the variant selection strategy is chosen, this threshold is how less the new variant would cost, in percentage. If the delta strategy is chosen, opportunities are taken only if their delta is lower than a fixed value.

We illustrate how our cost-based variant selection algorithm works at runtime to support agent Jim playing role fire chief (as in Figure 1) in the scenario depicted in Table 1 with respect to the costs in Table 2.

EXAMPLE 4. Suppose C_3 made by Tanker1 is threatened. This happens because Tanker1 notifies it will be late due to a traffic jam. C_3 is part of Jim's current variant for goal fire extinguished and the variant is threatened. In response, an adaptation process is triggered wherein variant selection is based on the delta criteria, and threatened/violated commitments should be avoided.

Jim can currently choose between three variants to extinguish the fire:

- V_1: Exploit his capability for hydrant need notified and get C_1 from Brigade1 to support goal hydrant usage authorized. Bringing about the antecedent of C_1 will make Brigade1 unconditionally committed to authorize hydrant usage.
- V_2: Make C_2 to Brigade1 to support tanker service paid, chain the commitment to C_3 so that Tanker1 is unconditionally committed to fire reached by tanker truck, and use his capability for pipe connected. Notice that this is the current variant, which is already partially enacted.
- V_3: The same strategy as the previous one, but relies on C_4 instead of C_3. This corresponds to making Tanker2 unconditionally committed to reach the fire.

Variant V_1 involves C_1 and Jim's capability for hydrant need notification. The variant supports hydrant need notified, hydrant usage authorized, fire hydrant used, and fire extinguished. To compute the variant cost, activation and compensation costs should be considered. The activation cost of hydrant need notified is 6; that of C_1 is 15. Compensation cost should be considered for capability pipe connected (8), C_2 (0), and C_3 (20). The variant overall cost is 49.

Variant V_2 (C_3, C_2, pipe connected) supports states of affairs tanker service paid, fire reached by tanker truck, pipe connected, tanker truck used, and fire extinguished. Unfortunately, this variant violates the agent's adaptation policy. Indeed C_3 in the variant is threatened.

Variant V_3 (C_4, C_2, pipe connected) supports the same states of affairs as V_2. Cost computation for capabilities is also the same and adds no cost. Cost computation for commitments adds the activation cost of C_4 (31). The only compensation cost to consider is that for commitment C_3 (20). Indeed, C_3 is in the current variant but not in the analyzed variant. The variant overall cost is 51.

Jim will therefore choose—according to his adaptation strategy—V_1. The next step is deciding on how to enact this variant, i.e. defining the course of actions to switch from the current variant to V_1. This is the role of the *Compensation* component, which uses the selected variant, the agent's enactment policy, and the agent's plans—taken from the plan library—to determine a course of action. Two types of atomic operations are possible: (i) actions that use actuators in the context and (ii) operations on commitments performed by sending messages through the commitments middleware.

To enact V_1, the architecture needs to carry out several steps. Some are intended to reverse the effects of the current variant, others to enact the new plan.

1. Cancel C_2 made to Brigade1, and release Tanker1 from commitment C_3. These operations are executed by sending messages through the commitments middleware;

2. Compensate capability pipe connected by using context actuators, such as an engine to retract the water pipe that was already ready to be plugged;

3. Carry out some actions to adopt the new variant: use capability for hydrant need notified (via an actuator such as a phone) to detach C_1 and make Brigade1 unconditionally committed to hydrant usage authorized by sending a message through the commitments middleware.

5 Discussion

In this chapter we have proposed an approach to agent adaptation. Our proposal applies to different settings than Kumar et al.'s adaptive agent architecture [2000]. Whereas their work applies only to multiagent settings where agents are collaborative, ours supports also settings where agents are designed by different stakeholders and might be competitive. The core message of our chapter is that, in multiagent systems where agents are weakly controllable, *social abstractions*—here, commitments—are the mechanisms that make the system work. In fact, relying on the intentions of other agents is risky, due to their autonomy and heterogeneity.

A distinguishing objective of our work is to exploit high-level models to represent both the agent purposes and the social relationships with other agents (social commitments). The combined usage of these abstractions makes our proposal very flexible. By focusing on the *purpose* of the agent and the *meaning* of interaction, adaptation guarantees that the agent meets its strategic interests (its purpose) and acts in compliance with its commitments to other agents. Central to our framework is the notion of variability, the existence of multiple strategies (variants) to achieve an agent's goals. We have shown how variants are constructed and presented a cost-based framework that enables an agent to select the best variant to adopt.

Future work comprises different research lines: (i) early detection of failures, i.e. the capability of agents to anticipate failures by adapting; (ii) considering quality-of-service attributes to choose between alternatives; (iii) efficient variant generation algorithms, e.g. heuristics that generate good-enough variants; (iv) adaptation policies that guide the choices made by an agent throughout its adaptation control loop.

References

Brachman, R. J., R. E. Fikes, and H. J. Levesque [1983, October]. Krypton: A functional approach to knowledge representation. *Computer 16*, 67–73.

Bratman, M. E. [1987]. *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Bresciani, P., A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos [2004]. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems 8*(3), 203–236.

Chopra, A. K., A. Artikis, J. Bentahar, M. Colombetti, F. Dignum, N. Fornara, A. J. I. Jones, M. P. Singh, and P. Yolum [2011]. Research directions in agent commmunication. *ACM Transactions on Intelligent Systems*. To appear.

Chopra, A. K., F. Dalpiaz, P. Giorgini, and J. Mylopoulos [2010]. Reasoning about agents and protocols via goals and commitments. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 457–464.

Cohen, P. R. and H. J. Levesque [1990]. Intention is choice with commitment. *Artificial Intelligence 42*, 213–261.

Dalpiaz, F. [2011]. *Exploiting Contextual and Social Variability for Software Adaptation*. Ph.D. thesis, Information and Communication Technology, University of Trento.

Dalpiaz, F., A. K. Chopra, P. Giorgini, and J. Mylopoulos [2010]. Adaptation in open systems: Giving interaction its rightful place. In *Proceedings of the 29th International Conference on Conceptual Modeling*, Volume 6412 of *LNCS*, pp. 31–45. Springer.

Finin, T., R. Fritzson, D. McKay, and R. McEntire [1994]. KQML as an agent communication language. In *Proceedings of the International Conference on Information and Knowledge Management*, pp. 456–463. ACM Press.

FIPA [2003]. FIPA interaction protocol specifications. FIPA: The Foundation for Intelligent Physical Agents, http://www.fipa.org/repository/ips.html.

Kumar, S., P. R. Cohen, and H. J. Levesque [2000]. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pp. 159–166.

Levesque, H. J. [1981]. The interaction with incomplete knowledge bases: A formal treatment. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pp. 240–245. AAAI Press.

Levesque, H. J. and J. Mylopoulos [1979]. A procedural semantics for semantic networks. In N. V. Findler (Ed.), *Associative Networks: The Representation and Use of Knowledge by Computers*, pp. 93–120. Academic Press.

Mylopoulos, J. [1992]. Conceptual modelling and Telos. In P. Loucopoulos and R. Zicari (Eds.), *Conceptual Modelling, Databases and CASE: An Integrated View of Information System Development*. McGraw Hill.

Searle, J. R. [1969]. *Speech Acts*. Cambridge, UK: Cambridge University Press.

Shoham, Y. [1993]. Agent-oriented programming. *Artificial Intelligence 60*(1), 51–92.

Singh, M. P. [1991]. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pp. 104–106.

Singh, M. P. [1998, December). Agent communication languages: Rethinking the principles. *IEEE Computer 31*(12), 40–47.

Singh, M. P. [1999]. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law 7*, 97–113.

Singh, M. P. [2008]. Semantical considerations on dialectical and practical commitments. In *Proceedings of the 23rd Conference on Artificial Intelligence*, pp. 176–181.

Singh, M. P. and M. N. Huhns [2005]. *Service-Oriented Computing: Semantics, Processes, Agents*. Chichester, UK: John Wiley & Sons.

Winograd, T. and F. Flores [1986]. *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, New Jersey: Ablex Publishing.

Yolum, P. and M. P. Singh [2002]. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems*, pp. 527–534. ACM Press.