# Adaptation in Open Systems: Giving Interaction its Rightful Place

Fabiano Dalpiaz, Amit K. Chopra, Paolo Giorgini, and John Mylopoulos

Department of Information Engineering and Computer Science, University of Trento
dalpiaz,chopra,paolo.giorgini,jm@disi.unitn.it

**Abstract.** We address the challenge of adaptation in open systems. Open systems are characterized by interactions among autonomous and heterogeneous participants. In such systems, each participant is a locus of adaptation; nonetheless, a participant would typically have to interact with others in order to effect an adaptation. Existing approaches for software adaptation do not readily apply to such settings as they rely upon control-based abstractions.
We build upon recent work on modeling interaction via *social commitments*. Our contributions in this paper include (1) formalizing the notion of a participant's strategy for a goal not just in terms of goals and plans, but also in terms of the commitments required, and (2) a conceptual model and framework for adaptation built around this notion of strategy that allows using arbitrary strategy selection criteria—for example, trust. We illustrate our contributions with examples from the emergency services domain.

## 1 Introduction

One of principal challenges in software engineering is supporting runtime adaptation in software systems. In this paper, we address the challenge of adaptation in *open systems*. Open systems involve autonomous and heterogeneous participants who interact in order to achieve their own respective goals [8]. Autonomy implies that no participant has control over another; heterogeneity that the participants' internal constructions, not only in terms of code but also in terms of goals and policies, may be different. Additionally, a participant will likely keep its internal construction private to others. In this sense, they are completely independent of each other.

Many of the applications that we rely upon today are open—for example, banking, foreign exchange transactions, trip planning. Trip planning, for example, involves a customer, a travel agency, airlines, credit card companies, and so on—each an autonomous entity with its own private goals and policies. The participants interact with each other in order to fulfill their respective goals.

Clearly, supporting adaptation in open systems is as valuable as in any other kind of system. In the trip planning application, the travel agency may book an alternate flight for his customer in case the workers of the airline with which the customer is currently booked are likely to go on strike the day of the flight. Typically, however, the travel agent will not do that before interacting with the customer and getting his approval for the change. Neither is the customer's approval guaranteed—he could prefer to arrive a day early than two hours late. Thus, in adapting, the travel agent must interact with another autonomous entity—the customer. The point we emphasize here is twofold.

1. In open systems, each participant, being autonomous, is an independent locus of adaptation.
2. Nonetheless, in effecting an adaptation, a participant needs to interact with others to achieve goals it cannot achieve by itself.

This paper addresses the challenge of understanding what it means for a participant to adapt in open systems, and thus how one might design adaptive software that represents the participant. It emphasizes interaction among participants and the corresponding *social* abstractions. By contrast, recent trends in software adaptation emphasize centralized, control-based abstractions and largely ignore interaction. Our emphasis on interaction is not a matter of technique—interaction is what makes things work in open systems, and our approach reflects that reality.

From here on, we refer to a participant as an *agent*. An important kind of agent—that ties our work to software engineering—is a software system that pursues the interests of a particular stakeholder. Our contributions specifically are the following.

- A conceptual model for adaptation in open systems that emphasizes interaction.
- The formalization of the notion of an agent's strategy for a goal. The notion of strategy covers interaction with other agents, and forms the common semantic substrate for adaptation across agents, whoever they may be designed by.
- A framework for adaptation that allows plugging in arbitrary agent-specific criteria in order to *select* and *operationalize* alternate strategies.

In previous work we proposed *design-time* reasoning about the suitability of interaction protocols for a participant's goals [5]. Here, such results constitute the baseline of our framework for participant *runtime* adaptation in open systems.

The rest of the paper is organized as follows. Section 2 introduces a conceptual model for adaptive agents in open systems. Section 3 presents a set of motivating examples of agent adaptation drawn from a firefighting scenario. Section 4 formalizes the notion of a strategy. Section 5 explains the overall framework for adaptive agents. It presents an agent control loop and focuses on the selection and operationalization of a strategy. Section 6 contrasts our model to existing approaches, summarizes the key points, and outlines future work.

## 2 Modeling Adaptive Agents in Open Systems

Section 2.1 recaps a conceptual model for agents in open systems [4]. Section 2.2 then describes the concepts involved in agent adaptation.

### 2.1 Agents in Open Systems

Following Tropos [1], we model an agent as a goal-driven entity (Fig. 1). An agent has goals that reflect its own interests. An agent may have the capability to achieve certain goals; for others, he may have no such capability. A capability is an abstraction for specific plans that an agent may execute to achieve the goal. To support those goals, an agent may depend on other agents. Conceptually, the approach in [4] goes beyond

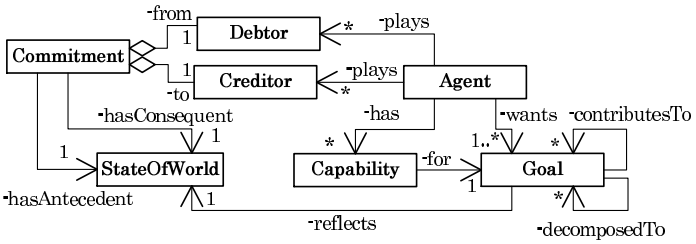



**Fig. 1.** Conceptual model for agents in open systems [4]

Tropos in that it makes these dependencies concrete and publicly verifiable by explicitly modeling interaction. In other words, the agent interacts with others (via messaging) to realize those dependencies.

A distinguishing feature of [4] is that the interactions are modeled in terms of *commitments*. A commitment $\mathsf{C}(\mathsf{debtor}, \mathsf{creditor}, \mathsf{antecedent}, \mathsf{consequent})$ is a promise made by the debtor to the creditor that if the antecedent is brought about, the consequent will be brought about. For example, $\mathsf{C}(\mathsf{customer}, \mathsf{travel\ agency}, \mathsf{tickets\ delivered}, \mathsf{paid})$ represents a commitment from the customer to the travel agency that if the tickets are delivered, then the payment will be made.

In open systems, an agent would have to interact with others and rely on commitments with them in order to support its goals. The agent can either play the debtor or the creditor in a commitment. In the above example, the customer's goal of having tickets delivered (tickets delivered) is supported by the commitment he makes (by sending the appropriate message, details in [4]) to the travel agency. Alternatively, the customer could request a commitment from the travel agency for tickets. If the travel agency responds by creating $\mathsf{C}(\mathsf{travel\ agency}, \mathsf{customer}, \mathsf{paid}, \mathsf{tickets\ delivered})$, then the customer's goal is supported (provided the customer can bring about the payment). Roughly, supporting a goal means identifying a strategy which will lead to the fulfilment of such goal [4] at run-time, provided that the strategy is enacted faultlessly. In principle, goals can be related to commitments and vice versa, because both goals and commitments (via the antecedent and consequent) talk about states of the world.

It is well known that commitments abstract over traditional descriptions of interactions (such as message choreographies) in terms of data and control flow [17]. More importantly, commitments support a standard of compliance suitable for open systems: an agent behaves in a compliant manner as long as it fulfills its commitments.

Commitments are notably different from traditional Tropos goal modeling:

- Commitments are a social abstraction better suited to open systems than dependencies. Commitments relate agents; they are created by explicit and observable messaging and hence are publicly verifiable (whether they hold or have been discharged) [14]. Dependencies relate agents, but are not tied to communication.
- Commitments decouple agents: to support a goal, in principle, an agent has only to enter into the appropriate commitment relationships with another agent. The agent need not care if the latter actually has the intention of achieving that goal. This is possible since commitments are publicly verifiable, and thus socially binding.

By contrast, reasoning with dependencies assumes a centralized perspective, where traditional AI planning techniques can be exploited [2].

### 2.2 A Model for Agent Adaptation

Fig. 2 shows the concepts involved in agent adaptation and the relations among them. The figure helps answer the following questions. *Why* should an agent adapt? *When* is adaptation required? *What* is the object of adaptation?

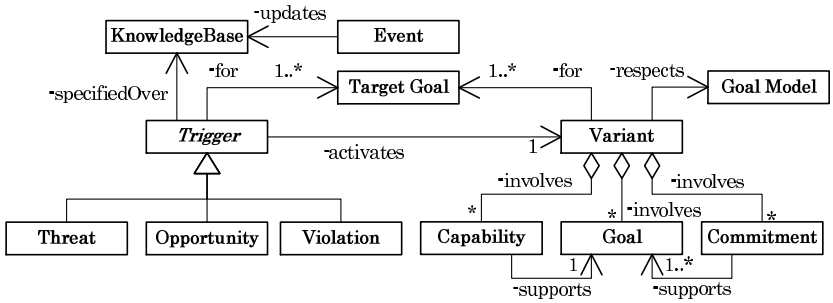



**Fig. 2.** Conceptual model for agent adaptation

An agent acts on the basis of its motivational component, the *target goals* it currently wants to achieve. The reason for adaptation—the *why*—is that the current strategy for achieving the current goals is inadequate or can be improved.

Every agent maintains its own state in a *knowledge base*, which is updated based upon the *events* the agent observes. An agent adapts *when* a certain adaptation *trigger* is activated. A trigger is a condition that is monitored for one or more target goals and is specified over the knowledge base. Three types of trigger are shown in Fig. 2; this list of types is not meant to be exhaustive but rather illustrative. A *threat* means that target goals are at risk; an *opportunity* means that a better strategy may be adopted; a *violation* means that the current strategy has failed. Both threat and opportunity relate to proactive adaptation policies—the agent adapts to prevent failures. By contrast, violation triggers are set off upon failures. Table 1 extends such triggers with more specific trigger types.

The object of adaptation, that is, *what* to adapt to, represents the new strategy, technically a *variant*. When the trigger goes off, the variant is activated. The variant represent the set of *goals* that need to be achieved in order to achieve the target goal. These goals are supported either by the agent's *capabilities* or *commitments*. Additionally, the variant is computed with respect to the agent's *goal model*—for example, this ensures that the set of goals to be achieved are sound with respect to goal decomposition (this notion is formalized later in Definition 3).

As explained above, the model in Fig. 1 motivates the *supports* relation. Fig. 2 exploits the *supports* relation in the notion of variant. A variant is essentially a collection of goals, commitments, and capabilities that are necessary to support the agent's target goals (Definition 4 formalizes this notion).

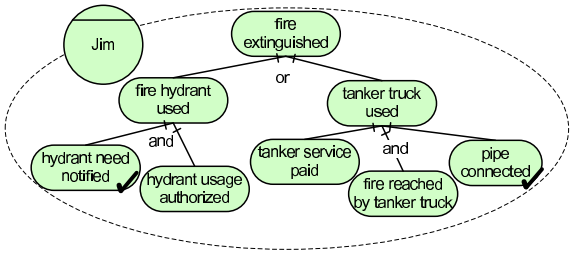
| Type | Description |
|---|---|
| Capability threat | The capability (plan) for an active goal is undermined |
| Commitment threat | The fulfillment of a commitment from another agent is at risk |
| Capability opportunity | An alternative capability has become more useful to exploit |
| Social opportunity | A new agent is discovered with whom it is better to interact |
| Commitment violation | A commitment from another agent is broken (timeout or cancellation) |
| Capability failure | A capability was executed but failed |
| Quality violation | An internal quality threshold is not met by the agent |

**Table 1.** Taxonomy of adaptation trigger types

## 3 Motivating Examples

We discuss examples of agent adaptation in a firefighting scenario. It serves as the running example throughout the rest of the paper.

Jim is a fire chief. His top goal is to extinguish fires. That goal may be achieved either by using a fire hydrant or a tanker truck. The top part of Table 2 shows Jim's goal model in Tropos. Ticked goals are those for which Jim has capabilities. The bottom part of Table 2 lists the commitments used in Figures 3–6.




| Label | Commitment |
|---|---|
| $C_1$ | C(Brigade 1, Jim, hydrant need notified, hydrant usage authorized) |
| $C_2$ | C(Jim, Brigade 1, tanker service paid, tanker truck used) |
| $C_3$ | C(Tanker 1, Jim, tanker service paid, fire reached by tanker truck) |
| $C_4$ | C(Tanker 2, Jim, tanker service paid, fire reached by tanker truck) |

**Table 2.** Running example: Jim is a fire chief (top); commitments in the scenario (bottom)

Each of the Figures 3–6 depicts Jim's active variant for the goal of having fires extinguished before (on the left of the dark, solid arrow) and after adaptation (on the right of the arrow). The chorded circles are agents. A variant depicts the active part of Jim's goal model, that is, those goals in the goal model of Jim that he has instantiated, and the capabilities and commitments required to *support* that goal. Commitments are represented by labeled directed arrows between agents—the debtor and creditor are indicated by the tail and the head of the arrow respectively.

We lack the space to explain the details of how Jim's fire extinguished goal is supported after adaptation in each example. The key point to take away from these exam-

ples is that Jim's set of active goals, the capabilities required, and his commitments to and from others change as a result of adaptation in response to some trigger.

**Tactic 1 (Alternative goals)** (Example 1) Choose a different set of goals in a goal model to satisfy target goals. The agent believes the current strategy will not succeed.

*Example 1.* (Fig. 3) Jim tries to achieve fire extinguished via a variant that relies upon using the fire hydrant. However, the fulfillment of $C_1$, which is necessary to support the goal, is threatened because Brigade 1 hasn't authorized hydrant usage yet. So Jim switches to another variant that supports fire extinguished via tanker truck used. To support tanker truck used, Jim makes $C_2$ to Brigade 1 and gets $C_3$ from Tanker 1.

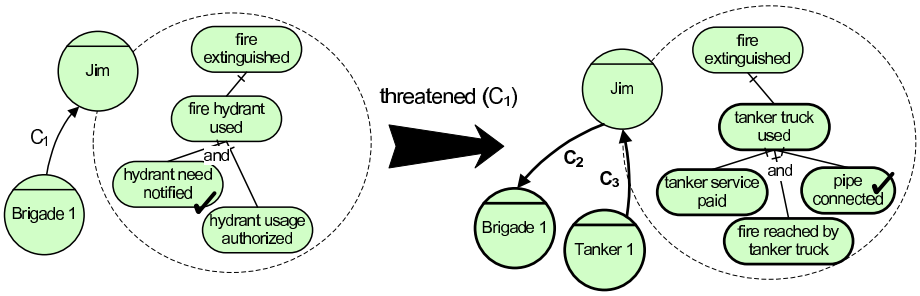



**Fig. 3.** Alternative goals: Jim switches from a variant involving fire hydrant usage to another involving tanker truck usage

**Tactic 2 (Goal redundancy)** (Example 2) Select a variant that includes redundant ways for goal satisfaction. Useful for critical goals that the agent wants to achieve at any cost.

*Example 2.* (Fig. 4). Jim's current strategy is to fight the fire via the hydrant. However, $C_1$ is threatened. So Jim adopts a strategy which involves *also* calling a water tanker truck. By contrast, Example 1 involves no redundancy.

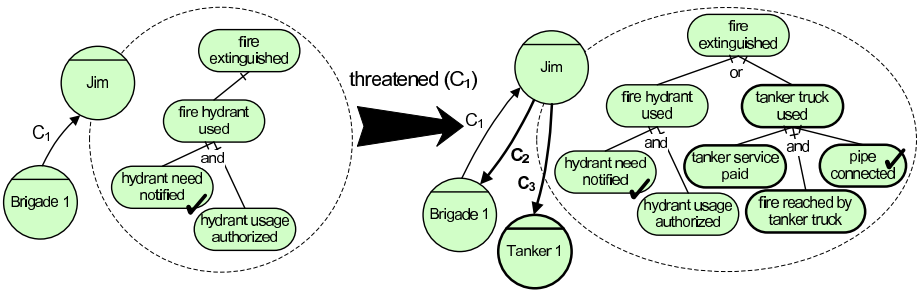



**Fig. 4.** Goal redundancy: Jim adopts a redundant variant, which involves also calling a tanker truck

**Tactic 3 (Commitment redundancy)** (Example 3) More commitments for a goal are taken. Useful if the agent does not trust some agent it interacts with. Also, it applies when a commitment from someone else is at risk due to the surrounding environment, and a different commitment is more likely to succeed.

*Example 3.* (Fig. 5) Jim doesn't trust Tanker 1 much for $C_3$. Therefore, he decides to get a similar commitment $C_4$ from Tanker 2.

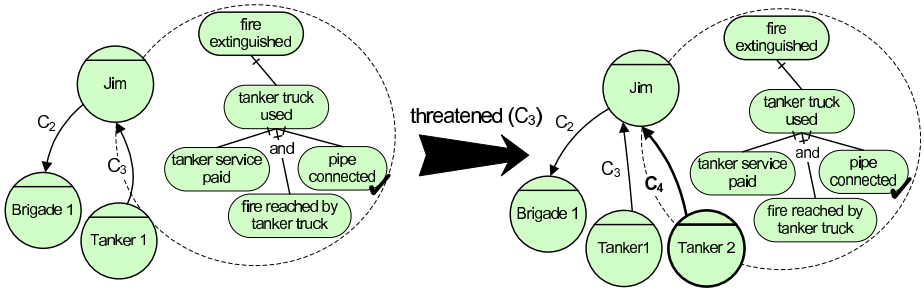



**Fig. 5.** Commitment redundancy: Jim gets $C_4$ from Tanker 2

**Tactic 4 (Switch debtor)** (Example 4) Get a commitment for the same state of the world but from a different debtor agent. Useful if the creditor believes the current debtor will not respect its commitment or a more trustworthy debtor comes into play. The original debtor is released from his commitment.

*Example 4.* (Fig. 6). Jim takes $C_3$ from Tanker 1, but fears that Tanker 1 will violate the commitment. Thus, Jim releases Tanker 1 from $C_3$ and instead gets $C_4$ from Tanker 2.

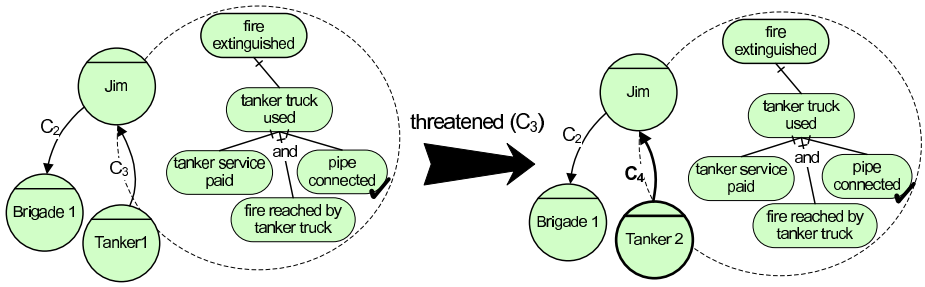



**Fig. 6.** Switch debtor: Jim releases Tanker 1 from $C_3$ and takes $C_4$ from Tanker 2

**Tactic 5 (Division of labor)** (Example 5) Rely on different agents for different goals instead of relying on a single agent. Distribution of work spreads the risk of complete failure.

*Example 5.* Suppose Jim wants to use both fire hydrant and a water tanker truck. Also, suppose Brigade 1 acts as a water tanker provider. Then Jim could use the tanker service from Brigade 1. However, Jim applies division of labor to minimize risk of failure: he takes $C_1$ from Brigade 1 and $C_3$ from Tanker 1.

**Tactic 6 (Commitment delegation)** (Example 6) An agent delegates a commitment in which he is debtor to another agent, perhaps because he can't fulfill it.

*Example 6.* Jim does not have resources to fight a fire. So he delegates his commitment to extinguish a fire to another fire chief Ron of a neighboring town.

**Tactic 7 (Commitment chaining)** (Example 7) Agent $x$'s commitment $\mathsf{C}(x, y, g_0, g_1)$ is supported if he can get $\mathsf{C}(z, x, g_2, g_1)$ from some $z$ and if $x$ supports $g_2$.

*Example 7.* Jim wants to achieve goal tanker truck used. It makes $\mathsf{C}_2$ to Brigade 1 so that tanker service paid is achieved. In such a way, it can get $\mathsf{C}_3$ from Tanker 1.

## 4 Formalization

We now formalize the notion of a variant. This formalization is not specific to any individual agent—it forms the common semantic substrate upon which we later build a comprehensive adaptation framework. We explain the formalization by referring to the examples introduced earlier.

Let $g, g', g'', g_1, g_2, \ldots$ be atomic propositions (atoms); $p, q, r, \ldots$ be generic propositions; $x, y, z, \ldots$ be variables for agents. Let $a_{id}$ be the agent under consideration. A *commitment* is specified as a 4-ary relation $\mathsf{C}(x, y, p, q)$. It represents a promise from a debtor agent $x$ to the creditor agent $y$ for the consequent $q$ if the antecedent $p$ holds. Let $\mathcal{P}$ be a set of commitments.

Commitments can be compared via a strength relation [6]. If an agent commits for something, it will also commit for something less. Also, it will commit if he gets more than expected in return. Such an intuition is captured via the transitive closure of $\mathcal{P}$.

**Definition 1.** Given a set of commitments $\mathcal{P}$, $\mathcal{P}^*$ is its transitive closure with respect to the commitments strength relation [6].

Let $\mathcal{P} = \{\mathsf{C}(\text{fireman, brigade, team assigned} \vee \text{ambulance sent, fire fought} \wedge \text{casualties rescued})\}$. Then, for instance, $\mathsf{C}(\text{fireman, brigade, team assigned} \vee \text{ambulance sent, fire fought}) \in \mathcal{P}^*$, $\mathsf{C}(\text{fireman, brigade, team assigned} \vee \text{ambulance sent, casualties rescued}) \in \mathcal{P}^*$, $\mathsf{C}(\text{fireman, brigade, team assigned, fire fought}) \in \mathcal{P}^*$.

**Definition 2.** A *goal model* $\mathcal{M}_{id}$ specifies an agent $a_{id}$ as:

1. a set of AND/OR trees whose nodes are labeled with atoms;
2. a binary relation on atoms $p\text{-}contrib$;
3. a binary relation on atoms $n\text{-}contrib$.

An AND/OR tree encodes the agent's knowledge about how to achieve the root node. The nodes are the agent's goals. $p\text{-}contrib(g, g')$ represents positive contribution: the achievement of $g$ also achieves $g'$. $n\text{-}contrib(g, g')$ is negative contribution: the achievement of $g$ denies the achievement of $g'$. The top part of Table 2 is a goal model for agent Jim. $\mathcal{M}_{\text{Jim}}$ has one AND/OR tree rooted by goal fire extinguished. $\mathcal{M}_{\text{Jim}}$ contains no contributions.

We introduce the predicate *scoped* to capture a well-formedness intuition: a goal cannot be instantiated unless its parent is, and if a goal's parent is and-decomposed, all the siblings of such a goal must also be instantiated.

**Definition 3.** A set of goals $\mathcal{G}$ is *scoped* with respect to goal model $\mathcal{M}_{id}$, that is, $scoped(\mathcal{G}, \mathcal{M}_{id})$ if and only if, for all $g_0 \in \mathcal{G}$, either

1. $g_0$ is a root goal in $\mathcal{M}_{id}$, or
2. exists a simple path $\langle g_0, g_1, \ldots, g_n \rangle$ in $\mathcal{M}_{id}$ such that $g_n$ is a root goal in $\mathcal{M}_{id}$ and $\forall i, 0 \leq i \leq n$ :
   (a) $g_i \in \mathcal{G}$, and
   (b) if $anddec(g_{i+1})$ $(i \neq n)$, then $\forall g$ such that $parent(g_{i+1}, g)$, $g \in \mathcal{G}$

*Example 8.* $\mathcal{G}_1 = \{$fire extinguished, fire hydrant used$\}$ is scoped with respect to $\mathcal{M}_{\text{Jim}}$. Indeed, fire extinguished is a root goal, whereas fire hydrant used is part of path $\langle$fire hydrant used, fire extinguished$\rangle$.

A variant is an abstract agent strategy to achieve some goal. It consists of a set of goals $\mathcal{G}$ the agent intends to achieve via a set of commitments $\mathcal{P}$ and a set of capabilities $\mathcal{C}$. A variant is defined with respect to a goal model $\mathcal{M}_{id}$. The notion of variant is more general than supports [5]: it considers scope and commitments between arbitrary agents.

**Definition 4.** A triple $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor$ is a variant for a goal $g$ with respect to goal model $\mathcal{M}_{id}$, that is, $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g$ if and only if

1. $scoped(\mathcal{G}, \mathcal{M}_{id})$ and $g \in \mathcal{G}$, and
2. $g$ is supported: $\nexists g' \in \mathcal{G} : n\text{-}contrib(g', g) \in \mathcal{M}_{id}$, and either
   (a) $g \in \mathcal{C}$, or
   (b) $\mathsf{C}(x, a_{id}, g', g) \in \mathcal{P}^* : \lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$, or
   (c) $\mathsf{C}(x, y, g, g') \in \mathcal{P}^*$, or
   (d) $ordec(g)$, and either
      i. $\exists g' : parent(g, g')$ and $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$, or
      ii. $\mathsf{C}(x, a_{id}, g', q) \in \mathcal{P}^* : q \vdash \bigvee_{parent(g, g_i)} g_i$ and $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$, or
      iii. $\mathsf{C}(x, y, p, g') \in \mathcal{P}^* : p \vdash \bigvee_{parent(g, g_i)} g_i$;
   (e) $anddec(g)$ and $\forall g' : parent(g, g')$ and $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$, or
   (f) $p\text{-}contrib(g', g) \in \mathcal{M}_{id} : \lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$.

The goals $\mathcal{G}$ that $a_{id}$ intends to achieve should be scoped with respect to the goal model $\mathcal{M}_{id}$ (clause 1). Goal $g$ must be supported: there should be no negative contributions to $g$ from any goal in $\mathcal{G}$ and one clause among 2a-2f should hold (clause 2).

2a. capabilities support goals;
2b. $a_{id}$ gets a commitment for $g$ from some other agent $x$ if $a_{id}$ supports the antecedent;
2c. some agent $y$ brings about $g$ in order to get a commitment for $g'$ from some other agent $x$ (possibly $a_{id}$ itself);
2d. an or-decomposed goal $g$ is supported if either: there is some subgoal $g'$ such that $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$ (2(d)i), $g$ is supported via commitment to (2(d)iii) or from (2(d)ii) other agents. These two clauses cover the case of an agent who commits for a proposition that logically implies the disjunction of all the goal children. For instance, a commitment for $g_1 \vee g_2$ supports a goal $g$ or-decomposed to $g_1 \vee g_2 \vee g_3$;
2e. an and-decomposed goal is supported if $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor$ is a variant for every children;
2f. positive contribution from $g'$ supports $g$ if $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g'$.

Definition 5 generalizes the notion of variant to sets of goals. A variant for a goal set should be a variant for each goal in the set.

**Definition 5.** A triple $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor$ is a variant for a goal set $\mathcal{G}'$ with respect to goal model $\mathcal{M}_{id}$, that is $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} \mathcal{G}$, if and only if, for all $g$ in $\mathcal{G}'$, $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{id}} g$.

*Example 9.* $\mathcal{G} = \{$fire extinguished, tanker truck used, tanker service paid, fire reached by tanker truck, pipe connected$\}$, $\mathcal{P} = \{\mathsf{C}_y = \mathsf{C}($Jim, $y$, tanker service paid, fire extinguished$), \mathsf{C}_z = \mathsf{C}(z$, Jim, tanker service paid, fire reached by tanker truck$)\}$, $\mathcal{C} = \{$pipe connected$\}$. $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{\mathsf{Jim}}}$fire extinguished. Shown in the right side of Fig. 3.

**Step 1.** From Definition 3, $\mathcal{G}$ is scoped with respect to $\mathcal{M}_{\mathsf{Jim}}$. Indeed, fire extinguished is a root goal, tanker truck used is and-decomposed, all its subgoals are in $\mathcal{G}$, and there is a path from all goals in $\mathcal{G}$ to fire extinguished.
**Step 2.** From Definition 4, we should check if $g$ is supported. Clause 2(d)i applies to fire extinguished if $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor \models_{\mathcal{M}_{\mathsf{Jim}}}$ tanker truck used.
**Step 3.** tanker truck used is and-decomposed into tanker service paid, fire reached by tanker truck, pipe connected; 2e tells to verify every subgoal.
**Step 4.** pipe connected is in $\mathcal{C}$, therefore 2a applies.
**Step 5.** tanker service paid can be supported if Jim commits for $\mathsf{C}_y$ to some agent (2c).
**Step 6.** fire reached by tanker truck is supported if some agent commits for $\mathsf{C}_z$ to Jim (2b), given that the antecedent tanker service paid is supported.

## 5 A Framework for Adaptive Agents

The conceptual model we introduced in the previous sections is the foundation to define a framework for the development of adaptive agents. First, we sketch a generic control loop for an adaptive agent in Algorithm 1. Then, we investigate agent adaptation policies for variant selection (Section 5.1) and variant operationalization (Section 5.2).

**Algorithm 1** An adaptive agent. AGENT is a generic agent control loop. TRIGGERED is an event handler for adaptation triggers

```
AGENT()
1  while OBSERVE(ε)
2  do UPDATE(σ, ε)
3     ...

TRIGGERED(goal [] G, goalModel M, state σ)
1  variant [] V ← GENVARIANTS(G, M)
2  variant V ← SELVARIANT(V, σ)
3  OPERATIONALIZE(V, σ)
```

Algorithm 1 shows the skeleton of an adaptive agent. The procedure AGENT sketches the part of a generic agent control loop related to adaptation. An agent observes an event $\varepsilon$ from the environment, then it updates the current state $\sigma$ of its knowledge base.

The procedure TRIGGERED is an event handler for adaptation triggers, which is executed whenever some event sets off an adaptation trigger. The input parameters are a set of target goals $G$, the agent's goal model $\mathcal{M}$, and the state $\sigma$.

First, the agent generates all variants $\mathcal{V}$ for the goals $G$ with respect to the goal model $\mathcal{M}$. Function GENVARIANTS is standard for any agent and is computed according to Definitions 4 and 5. Second, the agent selects one of the variants in $\mathcal{V}$. Such choice depends on the agent's internal policies. Section 5.1 details variant selection. A variant $V$ is an abstract strategy. It is a triple $\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor$ composed of goals $\mathcal{G}$, commitments $\mathcal{P}$, and capabilities $\mathcal{C}$. Neither commitments nor capabilities are grounded to concrete entities. The agent should therefore operationalize the variant: commitments must be bound to actual agents, capabilities to real plans. Section 5.2 describes operationalization.

### 5.1 Variant selection

Variant selection is the choice of one variant among all the generated ones. Function SELVARIANT takes as input a set of variants and a state and returns one of these variants.

$$\text{SELVARIANT} : 2^V \times S \rightarrow V$$
$$\text{SELVARIANT}(\{V_1, \ldots, V_n\}, \sigma) = V_i : 1 \leq i \leq n$$

Table 3 shows some common criteria for variant selection. Due to its autonomy, each agent is free to decide its own criterion.

| Name | Description |
|---|---|
| Cost | Minimize the overall cost, expressed as money, needed resources, time |
| Stability | Minimize the distance between the current strategy and the new one |
| Softgoals | Maximize the satisfaction of quality goals (performance, security, risk, ...) |
| Preference | Choose preferred goals and commitments |
| Goal Redundancy | Choose a redundant variant to achieve critical goals |

**Table 3.** Generic criteria for variant selection

*Example 10.* Table 4 specifies the function SELVARIANT for Fig. 4 as an Event-Condition-Action rule. We used such formalism to keep our explanation simple. In practice variant selection policies will be expressed via appropriate policy definition languages. Such a function is based upon the goal redundancy tactic. The triggering event is that commitment $\mathsf{C}_1$ is threatened. It applies if the target goal is fire extinguished, commitments $\mathsf{C}_2$, $\mathsf{C}_3$, $\mathsf{C}_4$ are not in place, Jim adopted goal fire hydrant used and not tanker truck used. The action specifies the transition to the new variant. Jim adopts goal tanker truck used and its children, uses his capability for pipe connected, gets commitment $\mathsf{C}_z$, and commits for $\mathsf{C}_y$. $\mathsf{C}_y$ and $\mathsf{C}_z$, from Example 9, are unbound.

| Event | Condition | Action |
|---|---|---|
| threatened($\mathsf{C}_1$) | target(fire extinguished),<br>¬made($\mathsf{C}_2$),<br>¬taken($\mathsf{C}_3$), ¬taken($\mathsf{C}_4$),<br>adopted(fire hydrant used),<br>¬adopted(tanker truck used) | adopt(tanker truck used),<br>adopt(tanker service paid),<br>adopt(fire reached by tanker truck),<br>adopt(pipe connected),<br>useCapability(pipe connected),<br>get($\mathsf{C}_z$), make($\mathsf{C}_y$) |

**Table 4.** Event-Condition-Action rule for variant selection with goal redundancy (Fig. 4)

### 5.2 Variant operationalization

The OPERATIONALIZE function takes as input the selected variant and an agent's state and returns a set of states.

$$\text{OPERATIONALIZE} : V \times S \rightarrow 2^S$$
$$\text{OPERATIONALIZE}(\lfloor \mathcal{G}, \mathcal{P}, \mathcal{C} \rfloor, \sigma) = \text{BINDTOPLAN}(\mathcal{C}); \text{BINDAGENT}(\mathcal{P})$$

Operationalization means identifying a concrete strategy to achieve goals and commitments in a variant. Commitments are bound to real agents (BINDAGENT), whereas capabilities are bound to executable plans (BINDPLAN). Let's explain why such a function returns a set of states instead of a single one. Suppose Jim's selected variant includes finding some agent that will commit for $\mathsf{C}_z$. Jim may send a request message for $\mathsf{C}_z$ to all known tanker providers—Tanker 1 and Tanker 2—and get a commitment from the first one that accepts. If Tanker 1 answers first, the function returns a state $\sigma_1$ where $\mathsf{C}_3$ holds; if Tanker 2 answers first, the returned state will be $\sigma_2$ such that $\mathsf{C}_4$ holds.

With respect to Table 4, operationalization would be invoked inside useCapability(pipe connected) to bind an appropriate plan to the capability, and inside get($\mathsf{C}_z$) and make($\mathsf{C}_y$) to bind $z$ and $y$ to the appropriate agents.

Table 5 shows some generic criteria an agent can exploit and combine to operationalize a variant.

| Name | Description |
|---|---|
| Comm Redundancy | More commitments for the same goal from different agents |
| Division of Labour | Involve many agents, each agent commits for a small amount of work |
| Delegation | Delegate some commitment where the agent is debtor to someone else |
| Trust | The agent gets commitments only from other agents it trusts |
| Reputation | Rely on reputation in community to select agents to interact with |

**Table 5.** Generic criteria for operationalization

*Example 11.* Let's operationalize the variant in Example 10. Jim wants to delegate firefighting with tanker truck to the agent he trusts more. He knows two fire chiefs, Ron and Frank. The one he trusts more is Ron.

**Step 1.** Bind capabilities to plans. Jim binds its capability for pipe connected to a specific plan where he connects a water pipe to the rear connector of a water tanker truck.

**Step 2.** Bind commitments to agents. Jim delegates $\mathsf{C}_2$ to Ron, but he doesn't get any response. Thus, he delegates such commitment to Frank, who accepts delegation. Frank creates a commitment to Brigade 1 and notifies Jim. Fig. 7 illustrates binding to agents.

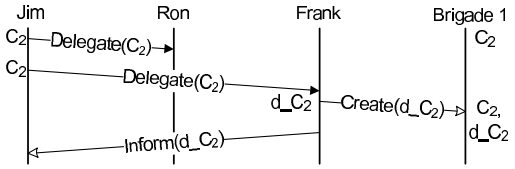



**Fig. 7.** Bind commitments to agents: Jim delegates $\mathsf{C}_2$ on the basis of trust

## 6 Discussion and Conclusion

Zhang and Cheng [18] introduce a formal model for the behavior of self-adaptive software. They separate adaptation models from non-adaptation models. Adaptation models guide the transition from a source program to a target one. Salehie *et al.* [13] propose a model for adaptation changes based on activity theory. They define a hierarchy for adaptation changes and match such concepts to a hierarchy for objectives. Both approaches are inadequate for open systems: they presume an omniscient view on the system which violates heterogeneity, and full control on system components which violates autonomy.

Component-based approaches to adaptation [9, 10] assume an external controller that adds, replaces, and rewires system components as necessary. The controller affects adaptations by reflecting upon the architectural model of the system. A centralized controller-based approach is unrealistic in open systems.

Our approach includes several elements of adapting at the architectural level, albeit without any central controller, when one considers that commitments are nothing but the interconnections among agents [15]. Our motivating examples may be seen as patterns constituting an *adaptation style* [9] for the commitment-based architectural style.

The meaning that we ascribe to (an agent) being *autonomous* is different from being *autonomic* [11]. A system is *autonomic* to the extent it can operate without supervision from its operator—the operator retains ultimate control. Broadly speaking, the self-* approaches refer to this notion of being autonomic.

Goal-oriented approaches for adaptation differ from the architectural ones in their emphasis on modeling the rationale for adaptation in a more detailed manner. However, the modus operandi remains similar. Either system code is instrumented to support adaptation [16] or a system is augmented with a controller that runs a monitor, diagnose, adapt loop [7] . They are inadequate for open systems, since they violate heterogeneity.

Approaches for adaptive agents characterize adaptation in terms of mentalistic notions such as goals, beliefs, desires, and intentions. Commitments, on the other hand, represent a social notion—they cannot be deduced from mentalistic notions, only from publicly observable communication [14]. Morandini *et al.* [12] give an operational account of goals so as to support adaptation in agents. However, lacking of commitments, their approach is not applicable for open systems. Unity [3] is a multiagent system created for autonomic computing. In Unity, *autonomic elements* (agents) collaborate to fulfill the system mission. Open systems, however, encompass competitive settings as well, and the agents may have no common goal.

In our framework, adaptation is conceived from the perspective of one autonomous agent which makes no assumptions about the internals of other agents (preserving, therefore, heterogeneity). An agent relies on interaction with others to achieve its own goals. Both the agent's goals and its architectural connections—specified in terms of commitments—are explicit and formally related to one another.

This paper provides the underpinnings of agent adaptation in open systems. Our contribution lies in incorporating interaction as a first-class entity in the notion of a variant for a goal. We built a framework around this notion of a variant that allows plugging in agent-specific variant selection and operationalization policies. Future work involves detailing these policies, building a middleware that understands the notion of a variant, and building implementations of adaptive agents on top of this middleware.

**Acknowledgements.** This work has been partially funded by the EU Commission, through projects SecureChange, COMPAS, NESSOS and ANIKETOS.

## References

1. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
2. Volha Bryl, Paolo Giorgini, and John Mylopoulos. Designing socio-technical systems: From stakeholder goals to social networks. *Requirements Engineering*, 14(1):47–70, 2009.
3. David M. Chess, Alla Segal, Ian Whalley, and Steve R. White. Unity: experiences with a prototype autonomic computing system. In *Proceedings of ICAC*, pages 140–147, 2004.
4. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. In *Proceedings of CAiSE*, volume 6051 of *LNCS*, pages from–to. Springer, 2010.
5. Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Reasoning about agents and protocols via goals and commitments. In *Proceedings of AAMAS*, pages 457–464, 2010.
6. Amit K. Chopra and Munindar P. Singh. Multiagent commitment alignment. In *Proceedings of AAMAS*, pages 937–944, 2009.
7. Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. An architecture for requirements-driven self-reconfiguration. In *Proceedings of CAiSE*, volume 5565 of *LNCS*, pages 246–260. Springer, 2009.
8. Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, December 2005.
9. David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, 37(10):46–54, October 2004.
10. William Heaven, Daniel Sykes, Jeff Magee, and Jeff Kramer. A case study in goal-driven architectural adaptation. In *Proceedings of SEAMS*, volume 5525 of *LNCS*, pages 109–127. Springer, 2009.
11. Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, January 2003.
12. Mirko Morandini, Loris Penserini, and Anna Perini. Operational semantics of goal models in adaptive agents. In *Proceedings of AAMAS*, pages 129–136, 2009.
13. Mazeiar Salehie, Sen Li, Reza Asadollahi, and Ladan Tahvildari. Change support in adaptive software: A case study for fine-grained adaptation. In *Proceedings of EaSE*, pages 35–44, 2009.
14. Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.
15. Munindar P. Singh, Amit K. Chopra, and Nirmit Desai. Commitment-based service-oriented architecture. *IEEE Computer*, 42(11):72–79, 2009.
16. Yiqiao Wang and John Mylopoulos. Self-repair through reconfiguration: A requirements engineering approach. In *Proceedings of ASE*, pages 257–268, 2009.
17. Pınar Yolum and Munindar P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of AAMAS*, pages 527–534, 2002.
18. Ji Zhang and Betty H. C. Cheng. Model-based development of dynamically adaptive software. In *Proceedings of ICSE*, pages 371–380, 2006.