Collaboration behavior enhancement in co-development networks

Shadi, M.

Publication date
2017
Document Version
Other version
License
Other

Chapter 4

Normative Virtual Organizations Supervisory Assistant Tool - VOSAT

This chapter contains some material from the following paper:

- Shadi, M., Afsarmanesh, H. and Dastani, M. M. (2013). Agent behavior monitoring in virtual organizations. In *22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 9-14). IEEE.

- Shadi, M., Afsarmanesh, H. and Dastani, M. M. Virtual Organization Supervisory Tool (VOSAT). Submitted to *International Journal of Networking and Virtual Organisations*. Inderscience Publishers.

4.1 Introduction

To increase the success rate of VOs, it is needed to design and develop a supervisory framework aiming at monitoring the partners' behavior, and diagnosing the behavior-related risks. In an open multi-agent system that allows agents to enter and exit the system at runtime, an exogenous normative artifact can be defined to organize and control the behavior of agents [10]. This normative artifact consists of norms and sanctions in which activities of participating agents are monitored, violations of /obedience to norms are specified, and finally the consequences for the observed violations/obedience are realized. The VBE is an open border environment, and thus represents an open society of agents, in which an exogenous normative organization can be defined. In Chapter 3, a normative multi-agent framework is proposed for VOs, based on which our VO supervisory artifact, called VOSAT (VO Supervisory Assistant Tool), is designed and developed. VOSAT aims to monitor and control agents' interactions, through checking their compliance with some norms, and imposing related sanctions against the norms' violations. It includes five components, as shown in Figure 4.1. In

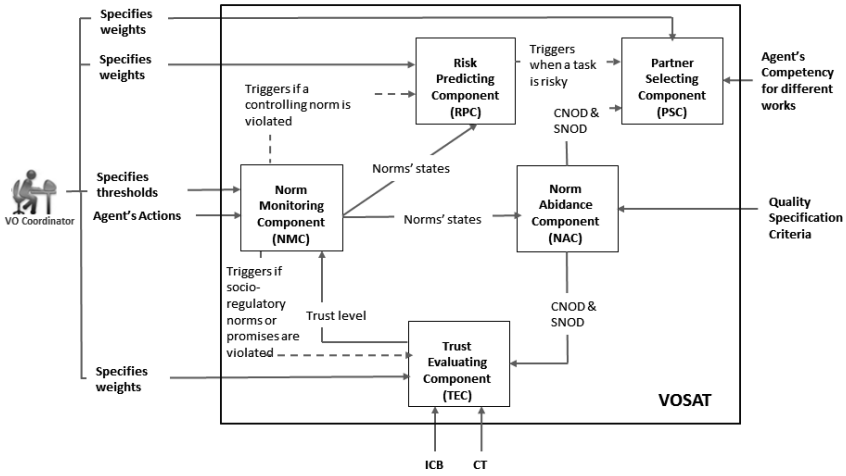

Figure 4.1: VO Supervisory Assisting Tool (VOSAT)

VOSAT, Norm Monitoring Component (NMC) is responsible for monitoring and controlling the agents' behavior against their defined norms, and imposing related sanctions against the norms' violations. To monitor the trust-related norm, it is necessary to continuously measure the trust level of an agent, which is measured by the Trust Evaluating Component (TEC), and set as input to NMC. Norm Abidance Component (NAC) is responsible for measuring the committing norms obedience degree (CNOD) and socio-regulatory norms obedience degree (SNOD) for each agent, using the information related to the norms specified by NMC. If an agent's trust-related norm is violated then the Risk Predicting Component (RPC) is triggered to find the risky tasks based on the PRIT information, as well as the information related to the risk factors. These information are finally used to assist the VO coordinator to potentially intervene through reassignment of risky tasks and therefore to remove the risk condition at the VO. Moreover, Partner Selecting Component (PSC) provides a mechanism to select the suitable partner for task reassignment or reward distribution among partners. TEC is addressed in Chapter 5, RPC and PSC are further addressed in Chapter 6, while details of NMC, and NAC are discussed in this chapter.

To develop NMC aiming at monitoring the VO partners' behavior against the defined norms in Chapter 3, it is needed to formalize the norm related concepts, such as promises, joint-promises, obligations and prohibitions. The proposed formalization for joint-promises and promises are to the best of our knowledge novel. In fact, all defined norms are formalized by a set of norm manipulating rules which are triggered by external events and partners' actions. Then, some new

propositions are derived from the fired norm manipulating rules. It means that it is possible to automatically specify the states of promises, and joint-promises, violation or obedience of socio-regulatory norms and controlling norms, based on which, different reactions can be planned.

This chapter also addresses how to measure CNOD and SNOD, which are internal measures to be used as criteria in partners' trust evaluation (addressed in Chapter 5), partner selection for task reassignment, and indirect reward distribution (both addressed in Chapter 6).

The rest of this chapter is structured as follows. The details of norm monitoring mechanism are addressed in Section 2. Section 3 discusses how we can measure the norm obedience degrees (CNOD and SNOD) for each VO partner. Section 4 addresses some conclusive remarks.

4.2 Norm Monitoring

In order to operationalize norms, a norm enforcement mechanism is required to be developed and provided. This mechanism is responsible for detecting the norms' violation and imposing sanctions in case of violations. AMELI platform [43], the normative framework of Cardoso [26], and the norm enforcement mechanism of Fornara et al. [45] are some developments in this area. The implementations addressed in [43] and [45] are all aimed at "procedural" norms, determining which actions should or should not be performed by agents, while the "declarative" norms (e.g. in [26]) specify a desired state that should be fulfilled within the environment in which the agents interact. In [33], the norms expressed as counts-as rules are declarative. We also define our norms as declarative counts-as rules.

The Norm Monitoring Component (NMC) does the monitoring of the agents' behavior through checking the state of their norms. The inputs of NMC component includes some predefined data related to the norms, such as the thresholds or agents' trust levels (see Figure 4.1). If agents' promises or socio-regulatory norms are violated then the Trust Evaluating Component is triggered to calculate the agent's trust level, while if one of the controlling norms is violated then Risk Predicting Component is triggered to identify the risky tasks. To implement the NMC, it is needed to first formalize norm-related concepts, and specify the configuration of NMC, as provided in the following sections.

4.2.1 Concepts Formalization

Promise Formalization. A promise with a specific state is formalized as a proposition, e.g. $Pr^C(x, y, p, d, q, d^{'})$ is used to represent the specific state of the promise $< x, y, p, d, q, d^{'} >$, as defined in Section 3.4.1 of Chapter 3. Label C in this expression refers to the conditional state (see Table 3.1). The agents' actions, and the external events manipulate the states of promises, and this manipulation

should respect a specific set of rules (see Table 4.1). In other words, these rules can be used to express how agents' actions and external events influence the state of promises. The rules have the general form of $\rho, \phi, \alpha \Rightarrow \rho^{'}$ where ρ, ϕ, α represent respectively a promise with a specific state, such as $Pr^C(x, y, p, d, q, d^{'})$, an environment-related fact describing the environment, such as deadline d, an action, such as $Fulfill(x, p)$, and $\rho^{'}$ represents the promise with a new state.

In these rules, shown in Table 4.1, the notations T and nop are used for True and for no-operation action, respectively. The agents' actions are as follows:

- $Agree(x, y, p, d, q, d^{'})$: x agrees with y to make the proposition p true, before the deadline d, if the proposition q is true before deadline $d^{'}$.
- $Withdraw(x, y, p)$: x tells y that he withdraws his promise to make proposition p true.
- $Release(x, y, p)$: x tells y that it is no longer needed to keep his promise to make proposition p true.
- $Fulfill(x, p)$: x makes proposition p true.

In these rules, $Fail(p)$ is considered as an external event showing that proposition p can no longer be made true, because of an external failure.

Rule Name	**Promise Manipulating Rules**
Making conditional promise	$T, T, Agree(x, y, p, d, q, d^{'}) \Rightarrow Pr^C(x, y, p, d, q, d^{'})$
Freeing pre-conditions	$Pr^C(x, y, p, d, q, d^{'}), \neg d^{'}, Fulfill(z, q) \Rightarrow Pr^{UC}(x, y, p, d, q, d^{'})$
Keeping	$Pr^{UC}(x, y, p, d, q, d^{'}), \neg d, Fulfill(x, p) \Rightarrow Pr^K(x, y, p, d, q, d^{'})$
Withdrawing	$Pr^{UC}(x, y, p, d, q, d^{'}), \neg d, Withdraw(x, y, p) \Rightarrow Pr^W(x, y, p, d, q, d^{'})$ $Pr^C(x, y, p, d, q, d^{'}), \neg d, Withdraw(x, y, p) \Rightarrow Pr^W(x, y, p, d, q, d^{'})$
Not Keeping	$Pr^{UC}(x, y, p, d, q, d^{'}), d, \neg p, nop \Rightarrow Pr^{NK}(x, y, p, d, q, d^{'})$
Dissolving	$Pr^C(x, y, p, d, q, d^{'}), d^{'}, \neg q, nop \Rightarrow Pr^{Dis}(x, y, p, d, q, d^{'})$
Releasing	$Pr^{UC}(x, y, p, d, q, d^{'}), T, Release(y, x, p) \Rightarrow Pr^R(x, y, p, d, q, d^{'})$ $Pr^C(x, y, p, d, q, d^{'}), T, Release(y, x, p) \Rightarrow Pr^R(x, y, p, d, q, d^{'})$
Invalidating	$Pr^{UC}(x, y, p, d, q, d^{'}), Fail(p), nop \Rightarrow Pr^{In}(x, y, p, d, q, d^{'})$

Table 4.1: Rules manipulating the promise states

The explanations below describe the above rules:

- Making conditional promise rule: performing the action $Agree(x, y, p, d, q, d^{'})$ implies the creation of a conditional promise. When this rule is applied, $Pr^C(x, y, p, d, q, d')$ is created.

- Freeing preconditions rule: performing the action $Fulfill(z, q)$ before passing time $d^{'}$ implies the creation of an unconditional promise. When this rule is applied, the conditional promise transforms to an unconditional promise.

- Keeping rule: performing the action $Fulfill(x, p)$ before passing d implies the creation of a kept promise. When this rule is applied, the unconditional promise transforms to a kept promise.

- Withdrawing rule: performing the action $Withdraw(x, y, p)$ before passing d implies the creation of a withdrawn promise. By applying this rule, the conditional/unconditional promise transforms to a withdrawn promise.

- Not keeping rule: not realizing p by the deadline d, implies the creation of a not kept promise. When this rule is applied an unconditional promise transforms to a not kept promise.

- Dissolving rule: not realizing q by passing $d^{'}$ implies the creation of a dissolved promise. When this rule is applied, the conditional promise transforms to a dissolved promise.

- Releasing rule: when x tells y it is no longer needed to fulfill his promise to realize p, it implies the creation of a released promise. When this rule is applied, the conditional/unconditional promise transforms to a released promise.

- Invalidating rule: when p cannot be realized because of an external failure, it implies the invalidation of an unconditional promise. When this rule is applied, the unconditional promise transforms to an invalidated promise.

Joint-promise Formalization. In Section 3.4.1 of Chapter 3, we defined a joint-promise as a tuple $< G, y, p, d, q, d' >$, where G is a set of agents. It means that each member of G make a promise for y to bring about p before d and also each member of G makes a promise to other members to perform its part of the joint-action bringing about p. In VOSAT, it is formalized as follows:

$$Joint - Pr^{C}(G, y, p, d, q, d^{'}) \equiv$$
$$\bigwedge_{A_i \in G, (A_i, p_i, d_i) \in PL} \Big(Pr^{C}(A_i, y, p, d, q, d^{'}) \wedge \big(\bigwedge_{A_j \in G - \{A_i\}} Pr^{C}(A_i, A_j, p_i, d_i, q, d^{'}) \big) \Big) \tag{4.1}$$

where,

- G is the promiser group

- y is the promisee agent in the joint-promise

- p is a proposition which a group of promisers G should bring about before deadline d
- q is the condition for realizing p by group G that should be fulfilled before d'.
- PL is the plan negotiated for distributing tasks among agents, during the VO operation phase. This plan includes a number of pairs such as (A_i, p_i, d_i) in which A_i is an agent who should bring about p_i before deadline d_i, as a part of the joint-action leading to realize p before d.
- $\bigwedge_{i=1}^{n} p_i \Rightarrow p$, $d_i \leq d$ and n is the number of agents in G.

This formalization means that each member of G makes a conditional promise for y to bring about p before d and also each member of G makes a conditional promise to other members to perform its part of the joint-action bringing about p.

Obligation Formalization. In VOSAT, obligations are formalized by two simple rules, because they have only two states, i.e. violation and obedience:

1. Obligation Obeying Rule: $Obliged(x, p, d), Fulfill(x, p), \neg d \Rightarrow Obeyed(x, p)$
2. Obligation Violating Rule: $Obliged(x, p, d), \neg p, d \Rightarrow Violated(x, p)$

These formalizations are not shown in a specific logic. The proposition p can be crisp or fuzzy, for example when an obligation is defined for the agent's trustworthiness, p is a fuzzy proposition (e.g. agent's trust level should be more than Low Trust).

Prohibition Formalization. It should be noticed that for prohibitions, two rules similar to obligation rules are considered, as follows:

1. Prohibition Obeying Rule: $Prohibited(x, p, d), \neg p, d \Rightarrow Obeyed(x, p)$
2. Prohibition Violating Rule: $Prohibited(x, p, d), Fulfill(x, p), \neg d \Rightarrow Violated(x, p)$

4.2.2 The Configuration of Norm Monitoring Component

Operational semantics is the means of defining the semantics of the Norm Monitoring Component. It is defined in terms of the configurations of the norm monitoring component and the transition between them [33]. The configuration or state of norm monitoring component is specified based on:

1. a set of propositions denoting environment-related facts
2. a set of Norm Manipulating Rules (NMRs)

3. a set of norms

4. a set of Reaction Rules (RRs)

Environment-related facts describe the environment in which agents interact. NMRs represent the rules introduced in Section 4.2.1 for formalization of promises, obligations and prohibitions. Socio-regulatory and controlling norms have two states, i.e. violated and obeyed, while promises have more than two states, as shown in Table 3.1. Plans that the VO coordinator wants to execute in case of the violation or obedience of norms correspond to the RRs.

Definition 5. Configuration of Norm Monitoring Component is defined by the tuple $< \delta_e, \delta_n, R_{nm}, R_r >$, where δ_e is a consistent set of propositions denoting the environment-related facts, δ_n is a consistent set of norms, R_{nm} is a set of norm manipulating rules, and R_r is a set of reaction rules.

In operational semantics, the transition rules, determine the transitions allowed between configurations. The state of our system can make a transition either when the actions (mentioned in Section 4.2.1) are performed by the agents, or when some external events have occurred.

Algorithm 1 shows that how the state of NMC makes a transition when an action is performed or an event is occurred. In this algorithm, p is a proposition representing an external event or the effect of an agent's action. In Algorithm 1, the set of norms and propositions denoting the environment-related facts is called KB. The proposition p is added to KB. In outer if statement in Algorithm 1, it is checked whether NM-rule r_1 is applicable based on the updated KB. The applicability of r_1 means that the condition of r_1 is satisfied in KB. If r_1 is applicable, its consequent, i.e. q_1, is derived, which shows a violated or obedient norm, or a promise in a certain state, and it is added to KB. For example, if the precondition of a promise is not fulfilled before its specified deadlines, then the *dissolving rule* shown in Table 4.1 is applicable, whose consequent (q_1) is a *dissolved promise*. Furthermore, the inner if statement in Algorithm 1, is related to applicability of the trust-related norms. It means that the trust-related norm has a different level in contrast to other norms, because it is triggered by the violation of the socio-regulatory norms or promises. In a normative environment like a VO, norms can have different levels; norms at level zero are triggered by external events, whereas norms at other levels are triggered in case of a violation of some norm(s) defined at lower levels [94]. If q_1 is a violated promise (not kept or withdrawn) or a violated socio-regulatory norm, and also the condition of NM-rule r_2 specifying the trust-related norm, is satisfied in KB, then r_2's consequent, i.e. q_2 is derived. It should be noticed that q_2 shows the violation or obedience of the trust-related norm, and added to KB. Finally, some Reaction Rules may be applicable, which means that their conditions are satisfied in updated KB. The consequent of the applicable R-rule r_3, shows the plan that is considered for different states of promises, or for violation, or obedience of other norms.

For example, if trust-related norms, communication-related norms or workload-related norms are violated then it is necessary to find the weak points in VO's activities fulfillment. The reaction rule, in this case a risk prediction rule gets triggered, which results in invoking the risk prediction function and sending an alarming message to the VO coordinator.

Algorithm 1: Norm Monitoring Algorithm

Input:
- KB, a set of propositions denoting norms & environment-related facts
- NM-rules, a set of Norm Manipulating rules
- R-rules, a set of Reaction rules
- p, a proposition denoting an environment-related fact

Add p to KB
if *the condition of the NM-rule* r_1*, is satisfied in KB* **then**
  q_1:=Consequent(r_1)
  Add q_1 to KB
  if q_1 *is a violated promise or a violated socio-regulatory norm and the condition of the NM-rule* r_2*, is satisfied in KB* **then**
    q_2:=Consequent(r_2)
    Add q_2 to KB
  end
  foreach r_3 *as a member of R-rules, whose condition is satisfied in KB* **do**
    b:=Consequent(r_3)
    Add b to KB
  end
end

We use Organization Oriented Programming Language (2OPL) [1] to implement our Norm Monitoring Component. The use of 2OPL is justified when there is an environment within which agents interact to achieve some common goals, and where they need to be organized for it. This language is introduced to specify norms, violations and sanctions within a multi-agent system. The 2OPL has been implemented as a JAVA software project within which a Prolog is used to implement the inference engine to keep and reason about the brute facts and institutional facts. In 2OPL, brute facts describe the states of the environment in which agents perform their actions and institutional facts describe the violation state of norms. The artifact defined by 2OPL is responsible to monitor the violation of norms and impose the related sanctions. All fundamental concepts of the 2OPL including brute facts, effect rules, counts-as rules and sanction rules should be written in a .2opl file, which is then interpreted by the 2OPL [1]. Definitions of the main 2OPL concepts are as follows:

- Brute facts are represented as Prolog facts and are initialized before the execution of 2OPL organization.

- Counts-as rule are represented by $\rho \Rightarrow \phi$ in which ρ is brute fact and ϕ is an institutional fact. This rule should be read as "ρ counts as ϕ". It should be mentioned that in 2OPL, institutional facts are not manually programmed, because they are obtained by applying Counts-as rules.

- Sanction rules relate institutional facts to brute facts expressed by rules like $\phi \Rightarrow \rho$ where ϕ consists of institutional facts and brute facts and ρ is a brute fact.

- Effect rules consist of pre-condition, action name, and post-condition parts. When an action is performed, the post-conditions of its related effect rule is realized which in turn changes the brute facts base.

The base for 2OPL is sufficiently similar to what we need to implement our supervisory artifact, and specifically the NMC of the VO Supervisory Assisting Tool (VOSAT). Although there are a number of differences in features of a 2OPL organization and our setting, this language can be easily extended in order to properly implement the norm monitoring component. We have implemented our environment-related facts, δ_e, as Brute Facts in 2OPL. For example, $curTime(0)$ is a brute fact which sets the current time before starting of the VO operation phase. The actions (Fulfill, Withdraw, etc.) are programmed as Effect Rules with pre- and post-conditions. For example, the effect rule below specifies that the action $fulfill(X, P)$ can be performed if P is not yet realized, and then the fact P will be realized after the performance of the action.

```
{not realized(P)}
 fulfill(X,P)
{realized(P)}
```

Our norm manipulating rules, R_{nm}, are implemented as Counts-as rules. For example, Figure 4.2 illustrates the implementation of the promise manipulating rules, as shown in Table 4.1. The first rule shown in Table 4.1, in which there is only one action, is implemented by an effect rule, while other rules are implemented as counts-as rules. It should be noticed that for each promise, at most one of the mentioned states in Table 3.1 is true at any point in time. The norm manipulating rules determine which state should remain true forever and which state should be removed. The termination states, i.e. kept, not kept, withdrawn, invalidated, released and dissolved will in principle remain true in the VOSAT framework, once they are true. Not kept and withdrawn states are considered as violation states. The VO coordinator may however decide to remove them once the sanctions (e.g. charging some damage costs to an agent or adding the agent to some blacklists) are applied.

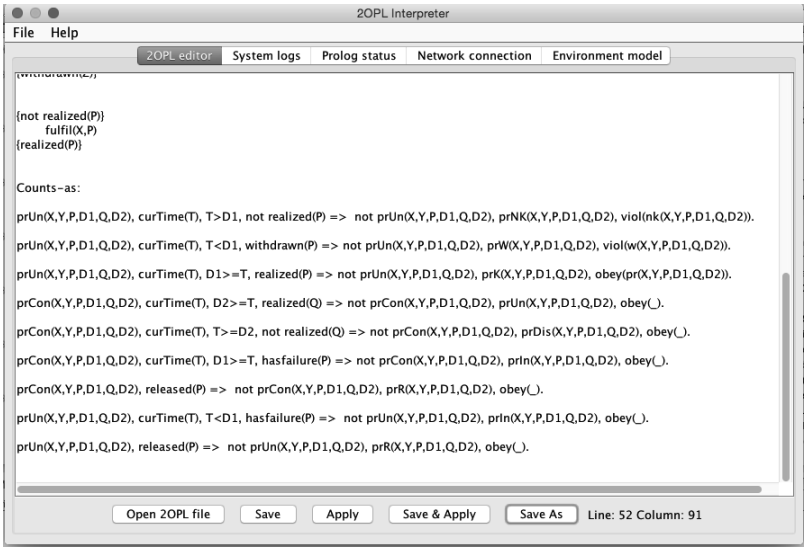

Figure 4.2: Implementation of Manipulating Promise Rules in 2OPL

Ultimately, we implement our reaction rules, R_r, as sanction rules in 2OPL. For example, the sanction rule below adds the promiser of a not kept promise X, to a specific black list:

```
viol(nkept,X,Y), blakList1(V), add (X,V,V2) => not viol (
   nkept,X,Y), not blakList1(V), blakList1 (V2).
```

This list can be used for some future decisions about agents, such as the selection of suitable VO partners for creation of a new VO. Imposing related sanctions against the norms' violations is very important in Virtual Organizations. For instance, if a promiser notifies the VO coordinator, before reaching the deadline, that it cannot fulfill its promise on time, that agent should then be punished less than if it were in the situation in which the deadline has passed and the promise is not fulfilled. But clearly, the sanction policies are usually not the same in different VOs. Typically, two kinds of sanctions are applied to incentivize the norm compliance by agents and to discourage deviation from norms: one affects agents resources (e.g. financial punishment), and the other one affects agents reputation (e.g. black listing). Sanction rules are either mentioned in the consortium agreement or specified by the VO coordinator.

4.2.3 An Example Case

To provide more details on how NMC works for promise monitoring in virtual organizations, consider a VO established to produce canned tomato paste. Jobs

to be performed in this VO are divided into a set of tasks. Let us consider one such task in this VO responsible for producing and sending 72500 tons of chopped tomato to the main factory, that in turn can produce 50000 tons of canned tomato paste.

Further assume that, in this task, three tomato producer companies (Tp_1, Tp_2, Tp_3) and two delivery enterprises (De_1, De_2) are involved, which jointly prepare and deliver 72500 tons of chopped tomato to the main factory to be processed into canned tomato. After negotiations, a number of sub-tasks are determined, as addressed below. It should be noticed that to simplify the model we assume that the results of the negotiation between the partners and the task leader constitutes a set of non-conflicting sub-tasks. The Task Leader requires agreements from the involved companies. The following agreements are thus made:

- Agreement 1: Tp_1 agrees with the Task Leader that: it fulfils q_1 (25000 Tn chopped tomato is ready) before January 10.
- Agreement 2: De_1 agrees with the Task Leader that: if q_1 is fulfilled before January 10 then it fulfils q_2(25000 Tn chopped tomato is delivered to the main factory) within 2 days.
- Agreement 3: Tp_2 agrees with the Task Leader that: it fulfils q_3 (21000 Tn chopped tomato is ready) before January 20.
- Agreement 4: De_2 agrees with the Task Leader that: if q_3 is fulfilled before January 20 then it fulfils q_4 (21000 Tn chopped tomato is delivered to the main factory) within 1 day.
- Agreement 5: Tp_3 agrees with the Task Leader that: it fulfils q_5 (26500 Tn chopped tomato is ready) before January 21.
- Agreement 6: De_1 agrees with the Task Leader that: if q_5 is fulfilled before January 21 then it fulfils q_6 (26500 Tn chopped tomato is delivered to the main factory) within 1 day.

The formalization of the above mentioned actions in our VO supervisory approach are shown below.

- Act 1: $agree(Tp_1, TaskLeader, q_1, 10, T, 0)$.
- Act 2: agree $(De_1, TaskLeader, q_2, 12, q_1, 10)$.
- Act 3: agree $(Tp_2, TaskLeader, q_3, 20, T, 0)$.
- Act 4: agree $(De_2, TaskLeader, q_4, 21, q_3, 20)$.
- Act 5: agree $(Tp_3, TaskLeader, q_5, 21, T, 0)$.

- Act 6: agree $(De_1, TaskLeader, q_6, 22, q_5, 21)$.

Please note that we use 0 to show the absence of a deadline and T as true when there is no condition for a promise. Also assume that all the above agreement actions are performed on the first day of January, i.e. $newTime(1)$ is January 1, and $newTime(n)$ represents date January n. Based on the effect rule implemented in our .2opl file for VOSAT, each action "agree" leads to creation of a promise. For example, performing an action Act2: $agree(De_1, TaskLeader, q_2, 12, q_1, 10)$ results in $prCon(De_1, TaskLeader, q_2, 12, q_1, 10)$, which is shown as Pr_2^C in Table 4.2.

The trace in Table 4.2 begins with the six actions mentioned above. According to the first and second rules shown in Table 4.1, three conditional and three unconditional promises are derived, as shown in the third column of Table 4.2. The promiser of an unconditional promise does not need to wait for fulfillment of any other promise. However, after fulfilling a promise or generally realizing a fact, a related conditional promise is converted to unconditional which will be further monitored. Consider that in this scenario, on January 5, when Tp_1 informs the TaskLeader that it is impossible to prepare 25000 Tn till January 10 (because of an external failure which is out of the promiser's control), and it can just prepare 15000 chopped tomato (q_7) instead at that time. Consequently, its first promise to prepare 25000 Tn tomato gets invalidated (last rule in Table 4.1). To assist with handling of this case, Tp_3 agrees with the TaskLeader to prepare 10000 Tn (labeled as q_8) till January 20, while De_1 also promises to deliver these two amount of chopped tomato prepared by Tp_1 and Tp_3 to the main factory (q_9, and q_{10}, respectively). Now we have new agreements:

- Act 7: $agree(Tp_1, TaskLeader, q_7, 10, T, 0)$.
- Act 8: $agree(Tp_3, TaskLeader, q_8, 20, T, 0)$.
- Act 9: $agree(De_1, TaskLeader, q_9, 12, q_7, 10)$.
- Act 10: $agree(De_1, TaskLeader, q_{10}, 22, q_8, 20)$.

Furthermore, consider if in this scenario De_1 does not deliver 10000 Tn prepared by Tp_3 on January 22. Therefore, $newTime(22)$ triggers the fifth rule in Table 4.1 resulting in $pr^{NK}(De_1, TaskLeader, q_{10}, 22, q_8, 20)$, which is a violation state, triggering the sanction rule. In this current scenario, everything except the two above mentioned situations goes well, i.e. all other promises except for Pr1, Pr2 and Pr10 are kept, as indicated in Table 4.2. The task therefore gets successfully completed on January 22.

Step	Action	Promise
1	$Act1, Act2, Act3, Act4,$ $Act5, Act6, newTime(1)$	$Pr_1^{UC}, Pr_2^C, Pr_3^{UC},$ $Pr_4^C, Pr_5^{UC}, Pr_6^C$
2	$Fail(q_1), Act7, Act8,$ $Act9, Act10, newTime(5)$	$Pr_1^{In}, Pr_2^D is, Pr_3^{UC}, Pr_4^C,$ $Pr_5^{UC}, Pr_6^C, Pr_7^{UC}, Pr_8^{UC}, Pr_9^C, Pr_{10}^C$
3	$Fulfill(Tp_1, q_7),$ $newTime(10)$	$Pr_1^{In}, Pr_2^{Dis}, Pr_3^{UC}, Pr_4^C, Pr_5^{UC},$ $Pr_6^C, Pr_7^K, Pr_8^{UC}, Pr_9^{UC}, Pr_{10}^C$
4	$Fulfill(De_1, q_9),$ $newTime(12)$	$Pr_1^{In}, Pr_2^{Dis}, Pr_3^{UC}, Pr_4^C, Pr_5^{UC}$ $Pr_8^{UC}, Pr_9^K, Pr_{10}^C$
5	$Fulfill(Tp_2, q_3), Fulfill(Tp_3, q_8),$ $newTime(20)$	$Pr_1^{In}, Pr_2^{Dis}, Pr_3^K, Pr_4^{UC}, Pr_5^{UC},$ $Pr_6^C, Pr_7^K, Pr_8^K, Pr_9^K, Pr_{10}^{UC}$
6	$Fulfill(De_2, q_4), Fulfill(Tp_3, q_5),$ $newTime(21)$	$Pr_1^{In}, Pr_2^{Dis}, Pr_3^K, Pr_4^K, Pr_5^K,$ $Pr_6^{UC}, Pr_7^K, Pr_8^K, Pr_9^K, Pr_{10}^{UC}$
7	$Fulfill(De_1, q_6), Fulfill(De_1, q_{10}),$ $newTime(22)$	$Pr_1^{In}, Pr_2^{Dis}, Pr_3^K, Pr_4^K, Pr_5^K,$ $Pr_6^K, Pr_7^K, Pr_8^K, Pr_9^K, Pr_{10}^{NK}$

Table 4.2: Execution Trace in Scenario of Tomato Sauce industry.

4.3 Norm Abidance

As Figure 4.1 illustrates, the Norm Abidance Component receives the promises' states, and the violation or obedience of socio-regulatory norms from NMC as inputs, and then measures two degrees: Committing Norm Obedience Degree (CNOD), and Socio-regulatory Norm Obedience Degree (SNOD), which are explained in the following sections.

The obedience degrees, (*CNOD* and *SNOD*) are used to determine the agent's *trust level*, task reassignment, and reward distribution. As mentioned before, joint-promises are decomposed into several individual promises, so the violation and obedience of co-working norms are considered in evaluation of the committing norms. There is also no obedience degree for controlling norms, because the violation of these norms are directly considered in risk prediction, while the violation and obedience of socio-regulatory norms, co-working norms and committing norms are needed for evaluation of trust.

4.3.1 CNOD - Committing Norm Obedience Degree

CNOD indicates how much the behavior of each agent is compliant with its committing norms (promises). In a VO, typically, based on the type of promised sub-task, some Quality Specification Criterion (QSC) are defined at the time of agreement between promiser and the VO coordinator or task leaders. For example, if a partner is the farmer who delivers tomato in the tomato paste production-example, then for instance the tomato's quality grades (e.g. G_1, G_2 or G_3) or color (e.g. red, orange, or green) are the QSCs for its delivered product, which is usually agreed between this partner and the promisee, e.g. the VO coordinator. Therefore, QSCs should also be evaluated when measuring the CNOD

for an agent.

QSC Evaluation

The design of our approach to evaluate QSCs for an agent is partially rooted in [88]. In our approach two rating factors for QSC are defined as follows:

1. Fulfillment of QSC (FQ) shows the degree of fulfillment of each QSC. The maximum value of FQ for a QSC is the value stated in the contract, which is mutually agreed between each agent and the task leader or the VO coordinator.

2. Influence of QSC (IQ) is defined in [0,1] by the task leader or the VO coordinator, to determine the importance and influence of each QSC in quality evaluation of results produced by the agent.

We use these two factors to calculate the quality of the results produced by the agent after performing its sub-task against the ideal quality, which are mutually agreed in the contract. In other words, this is defined as a measure of how good the agent delivers his agreed promise with the task leader or the VO coordinator.

At first, we define function C, $C : S \rightarrow P(V)$, where S is the set of all sub-tasks defined in a VO, the set of all quality specification criteria defined for results of sub-tasks in a VO, is denoted by V, and $P(V)$ is the *power set* of V. The quality of delivered sub-task st, if at least one criterion is specified for it (i.e. $|C(st)| > 0$), is calculated as follows:

$$Quality(st) = \sum_{i=1}^{|C(st)|} FQ(c_i, st) * IQ(c_i, st) \tag{4.2}$$

where, st shows a sub-task, $FQ : V \times S \rightarrow [0, 1]$ and $IQ : V \times S \rightarrow [0, 1]$, V is the set of all quality specification criteria, $|C(st)|$ is the number of quality criteria for the sub-task st, and c_i shows the i^{th} criterion of the $C(st)$.

The $Max\_Quality$ value for a sub-task shows the maximum possible quality occurrence to perform the sub-task. It is calculated as follows:

$$Max\_Quality(st) = \sum_{i=1}^{|C(st)|} max(FQ(c_i, st)) * IQ(c_i, st) = \sum_{i=1}^{|C(st)|} IQ(c_i, st) \tag{4.3}$$

where, $max(FQ(c_i, st))$ indicates the ideal and maximum value of the fulfillment of the c_i, which is here equal to 1, because in our approach, FQ values are normalized into [0,1].

The relative quality between what the agent actually delivered and maximum possible quality value (expected value) is calculated as follows:

$$Relative\_Quality(st) = \frac{Quality(st)}{Max\_Quality(st)} \tag{4.4}$$

It should be noticed that relative quality for each sub-task is measured after it is performed, i.e. when the state of the promise made to perform the sub-task is kept. It means that relative quality is ignored for promises, which are not kept, invalidated and withdrawn.

Since different criteria have different FQ values, we normalize all received values into [0,1]. Assume that the promisee (service client) for which the sub-task is performed, sends a fulfilment value for a criterion, called x, in [a,b], then the formulas below show how it is normalized into an interval [0,1], and named $x^{'}$.

For those criteria that have positive connotations (e.g. the Availability, Throughput, and Reliability for a developed software service), their values will be scaled as follows [28]:

$$x^{'} = \left\{ \begin{array}{cc} \frac{x-a}{b-a} & if\ a \neq b \\ 1 & otherwise \end{array} \right\} \tag{4.5}$$

For criteria with negative connotation (e.g. the Response time for a software service) the formula below is used [28]:

$$x^{'} = \left\{ \begin{array}{cc} \frac{b-x}{b-a} & if\ a \neq b \\ 1 & otherwise \end{array} \right\} \tag{4.6}$$

Promise Evaluation Considering the Quality Specification Criteria

To measure the committing norms obedience degree (CNOD), three measures are considered and explained in the following paragraphs.

Promise Fulfillment (PF). The evaluation of a promise in its termination state (i.e. kept, not kept, withdrawn, invalidated, released or dissolved) can be different based on the VO in which the promise is made. An example of values assigned by the VO coordinator to different states of a promise is shown in Table 4.3. These values show the fulfillment degree of an agent's promise. In fact, the promise fulfillment values are determined for a promiser who made a promise to perform a sub-task. The value should be positive, or negative, e.g. promise fulfillment degrees in Table 4.3 are defined in {-2,-1,-0.3,2}. Withdrawn and Not Kept states of a promise have negative effects on the performance evaluation of promiser, so their values are also negative, while kept promise has positive effect on the performance evaluation of the promiser. Invalidated promise can negatively influence the promiser performance, however the reason is out of the control of the promiser, so we consider a very little negative value for it (e.g. -0.3 as shown in Table 4.3). Realised promise shows that promisee cancels the promise, so it is not negative or positive for promiser's performance. Moreover, when preconditions of a promise are not fulfilled, the state of a promise is dissolved, which is not

negative or positive for the promiser. Therefore, released and dissolved promises are not considered in CNOD measurement.

Promise State	**Promise Fulfillment Value**
Not Kept	-2
Withdrawn	-1
Invalidated	-0.3
Kept	2

Table 4.3: An example of Promise Fulfillment (PF) values

Promise Importance (PI). Next to the Promise Fulfillment (PF) value addressed above, it is also needed to consider Promise Importance (PI) in the measuring of CNOD, because some promises are related to the sub-tasks that have more important roles in the collaboration success of the VO. The values of PIs are considered in [0,1].

Q-Factor (QF). To show the role of Relative-Quality in evaluation of CNOD, we define the Q-Factor (QF) for promise p_j made for performing sub-task st_{p_j}, as follows:

$$QF(p_j) = \begin{Bmatrix} Relative\_Quality(st_{p_j}) & if\ \ st_{p_j}\ \ is\ \ performed \\ 1 & otherwise \end{Bmatrix} \tag{4.7}$$

where, st_{p_j} shows the sub-task for performing which promise p_j is made. The value of $QF(p_j)$ is in [0,1], because the value of $Relative\_Quality(st_{p_j})$ is [0,1]. If the st_{p_j} is performed then its related $PF(p_j)$ in Formula 4.8 has the maximum value (i.e. 2, considering Table 4.3). Moreover, if all quality criteria agreed for this sub-task, are ideally fulfilled, $Relative\_Quality(st_{p_j})$ is 1, and consequently $QF(p_j)$ is 1. When promise p_j is withdrawn, not kept or invalidated, its $PF(p_j)$ is negative. In this situation, $QF(p_j)$ is 1, because 1 is the identity element under multiplication, which shows that $Relative\_Quality$ is ignored for withdrawn, not kept and invalidated states.

We define the function f_1, $f_1 : \delta_A \rightarrow P(\delta_p)$, where δ_A is the set of agents involved in a VO, δ_p is the set of promises made so far in the VO, and $P(\delta_p)$ is the *power set* of δ_p. Similar to the formula for Relative_Quality, the CNOD for agent A is calculated as follows:

$$CNOD(A) = \frac{\sum_{j=1}^{|f_1(A)|} PF(p_j) * PI(p_j) * QF(p_j)}{\sum_{j=1}^{|f_1(A)|} max(PF(p_j)) * PI(p_j)} \tag{4.8}$$

where, $|f_1(A)|$ shows the number of promises that agent A has made so far in the VO (for each sub-task only one promise is made), $PI : \delta_p \rightarrow [0,1]$, $QF : \delta_p \rightarrow [0,1]$ and $max(PF(p_j))$ indicates the ideal and maximum value of the fulfillment

of the promise p_j. It should be noticed that considering the example values for promise fulfilment in Table 4.3, $PF : \delta_p \rightarrow \{-2, -1, -0.3, 2\}$.

This formula returns a value for each agent, which shows the ratio of agent's success in fulfilling its promises and their QSCs in relation to the ideal state.

Below, it is shown that CNOD is a value in [-1,1]. Since $PI(p_j)$ and $QF(p_j)$ are values in [0,1], and PF is defined in {-2,-1,-0.3, 2}, we have:

$$|PF(p_j) * PI(p_j) * QF(p_j)| \leq max(PF(p_j)) * PI(p_j), \quad \forall j = 1 \ldots |f_1(A)|$$

and then:

$$\sum_{j=1}^{|f_1(A)|} |PF(p_j) * PI(p_j) * QF(p_j)| \leq \sum_{j=1}^{|f_1(A)|} max(PF(p_j)) * PI(p_j)$$

Based on the rule ($|a + b| \leq |a| + |b|$), we have:

$$|\sum_{j=1}^{|f_1(A)|} PF(p_j) * PI(p_j) * QF(p_j)| \leq \sum_{j=1}^{|f_1(A)|} |PF(p_j) * PI(p_j) * QF(p_j)|$$

and consequently,

$$|\sum_{j=1}^{|f_1(A)|} PF(p_j) * PI(p_j) * QF(p_j)| \leq \sum_{j=1}^{|f_1(A)|} max(PF(p_j)) * PI(p_j)$$

It is also clear that:

$$\frac{|\sum_{j=1}^{|f_1(A)|} PF(p_j) * PI(p_j) * QF(p_j)|}{\sum_{j=1}^{|f_1(A)|} max(PF(p_j)) * PI(p_j)} \leq 1$$

This means that $|CNOD| \leq 1$ (see Formula 4.8).

If PI for all promises of an agent A is 1, PF is defined in {-2,-1,-0.3, 2}, and *all* its promises are *not kept* then $CNOD(A) = \frac{\sum_{i=1}^{|f_1(A)|}(-2)*1*1}{\sum_{i=1}^{|f_1(A)|} 2*1} = -1$, if *all* its promises and their related QSCs are *kept*, CNOD(A)=1 and if half of its promises and their related QSCs are *kept* and half of its promises are *not kept*, CNOD(A)=0. According to the definition of joint-promise (see Formula 4.1), the creation of a joint-promise generates several individual promises. These individual promises will be counted in the CNOD measurement. It should be noticed that the promises of an agent for performing a specific sub-task are counted only once. CNOD is an internal measure, which is invisible for stakeholders, and only used as a criterion for trust evaluation and other future decisions.

Example

Considering the case of partners in an R&D project introduced earlier in the thesis, assume that agent A agrees to perform the following two sub-tasks for agent C that represents a task leader or the VO coordinator:

- st_1: providing a manual report document within 3 weeks from now.
- st_2: developing a software service within 6 weeks from now.

Consequently, two promises are made by agent A to agent C as follows:

1. Pr 1: $Pr^{UC}(A, C, st_1, 3weeks, T, 0)$.
2. Pr 2: $Pr^{UC}(A, C, st_2, 6weeks, T, 0)$.

For the report document, assume that, based on their agreement, the content of the manual document should be fully compliant with agreed TOC (Table Of Content) with the attachment of 25 screen shots, i.e. $C(st_1) = \{content, attachment\}$. For the software service, assume that it is agreed that response time of the service should not be greater than one minute, availability of the service should be 24h, and reliability should be greater than 90%, $C(st_2) = \{ResponseTime, Availability, Reliability\}$. Obviously, QSCs for these sub-tasks would be relevant and measured if they are performed, as explained before. Assume also that, the first promise is kept, with 70% compliant with agreed TOC, and accompanied with 12 screen shots, while the second promise is invalidated, since the development of software could not go on due to reasons beyond the control of the agent. Therefore, two QSCs for the first promise, i.e. content, and attachment need to be measured. For example, if the content is 70% compliant with agreed TOC, then $FQ(content, st_1)$ is $\frac{0.7-0}{1-0}$ and if 12 screen shots are delivered then the normalized $FQ(attachments, st_1)$ is $\frac{12-0}{25-0}$ (considering the Formula 4.5). Furthermore, if the IQ of the content is 0.9, and the IQ of the attachment is 0.5, then we can compute the quality of delivered sub-task, st_1, as follows:

$$Quality(st_1) = FQ(content, st_1) * IQ(content, st_1) + FQ(attachment, st_1) * IQ(attachment, st_1) = 0.87$$

$max(FQ(content, st_1))$ is 1, because the ideal result for content is 100 % compliant with agreed TOC. The ideal result for attachment is 25 screen shots, so $max(FQ(attachment, st_1))$ is $\frac{25}{25} = 1$; therefore, we have: $Max\_Quality(st_1) = 1.4$ and finally $Relative\_Quality(st_1) = 0.621$. Assuming that the Promise Importance (PI) of the first promise (which is fulfilled) is 1 and the PI of the second promise (which is invalidated) is 0.5, we then have CNOD(A)=0.247, considering the Formula 4.8 and Table 4.3.

4.3.2 SNOD - Socio-regulatory Norm Obedience Degree

Socio-regulatory Norm Obedience Degree shows to which extent an agent has complied with the socio-regulatory norms, as specified in the consortium agreement. We define function f_2, $f_2 : \delta_A \rightarrow P(\delta_s)$, where δ_A is the set of agents involved in a VO, δ_s is the set of all socio-regulatory norms defined for agents in the VO, and $P(\delta_s)$ is the *power set* of δ_s. SNOD is calculated as follows:

$$SNOD(A) = \frac{\sum_{i=1}^{|f_2(A)|} NF(n_i)}{|f_2(A)|} \tag{4.9}$$

where, $NF(n_i)$ shows the Norm Fulfilment (NF) degree of socio-regulatory norm n_i ($NF : \delta_s \rightarrow \{-1, 1\}$). $NF(n_i)$ is 1 if norm n_i is kept (obeyed), while $NF(n_i)$ is -1 if n_i is violated. It is clear that SNOD is a value in [-1,1]. These norms have two states, i.e. violated and obeyed (kept), so in this formula only the number of violated norms and obeyed norms are taken into account.

4.4 Conclusion

This chapter addresses the VO Supervisory Assisting Tool (VOSAT). It includes norm monitoring component, norm abidance component, trust evaluating component, risk predicting component and partner selecting component. The first two component are discussed in this chapter. Norm Monitoring Component is responsible for monitoring and controlling the agents' behavior against their defined norms, and imposing related sanctions against the norms' violations. This component is implemented using Organization Oriented Programming Language (2OPL). Norm abidance component is responsible for measuring the committing norms obedience degrees (CNOD) and socio-regulatory norms obedience degrees (SNOD) for each agent based on the information calculated in norm monitoring component. CNOD and SNOD are used to evaluate trust level of VO partners during the VO operation phase, to select best-fit partner for reassignment of risky tasks, and reward distribution, which are discussed in Chapters 5 and 6.