Monitoring and Explanation of Contract Execution: A Case Study in the Aerospace Domain

Felipe Meneguzzi
Sanjay Modgil
Nir Oren
King's College London
Dept of Computer Science
London, United Kingdom
felipe.meneguzzi@kcl.ac.uk

Simon Miles
Michael Luck
Nora Faci
King's College London
Dept of Computer Science
London, United Kingdom
simon.miles@kcl.ac.uk

Camden Holt
Malcolm Smith
Lost Wax
72 Lower Mortlake Road
London, United Kingdom
camden.holt@lostwax.com

ABSTRACT

In the domain of aerospace aftermarkets, which often has long supply chains that feed into the maintenance of aircraft, contracts are used to establish agreements between aircraft operators and maintenance suppliers. However, violations at the bottom of the supply chain (part suppliers) can easily cascade to the top (aircraft operators), making it difficult to determine the source of the violation, and seek to address it. In this context, we have developed a global monitoring architecture that ensures the detection of norm violations and generates explanations for the origin of violations. In this paper, we describe the implementation and deployment of a global monitor in the aerospace domain of [8] and show how it generates explanations for violations within the maintenance supply chain. We show how these explanations can be used not only to detect violations at runtime, but also to uncover potential problems in contracts before their deployment, thus improving them.

Categories and Subject Descriptors

D.2.10 [**Software**]: Software Engineering; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multi-agent systems*

General Terms

Design, Experimentation

Keywords

Electronic contracting, norms, contracts, monitoring

1. INTRODUCTION

The aerospace aftermarket is a complex and interesting domain, increasingly populated by customers buying a service rather than a product. As described by Jakob *et al.*[8], aircraft engine manufacturers are responsible for providing a specified number of serviceable engines so that aircraft can be kept flying. Engine manufacturers are paid in relation to engine availability, and may face penalties if aircraft are on the ground waiting for serviceable engines. In this model, servicing and maintenance become key drivers of long term profitability for engine manufacturers, with aftercare contracts being worth millions of Euros and lasting for several years.

Cite as: Monitoring and Explanation of Contract Execution: A Case Study in the Aerospace Domain, F. Meneguzzi, S Modgil, N. Oren, S. Miles, M. Luck *et al.*, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

At the same time, in research terms, we have been witnessing a growing interest in the use of norms to regulate and coordinate the behaviours of autonomous agents interacting in open environments. Such regulation is required to mitigate against self-interested behaviour, and thus ensure that the agents act to achieve the overall objectives of the multi-agent systems in which they are deployed. In order to motivate compliance with norms, enforcement measures [5, 11, 14] are applied, such as sanctions threatening some loss of utility for the agents responsible for violation of norms. Appropriate application of such measures requires that agents' actions are *monitored*; that is to say, agent actions must be observed and recognised as complying with or violating norms.

It seems appropriate therefore to consider the application of such normative systems to the kind of scenarios that arise in the aerospace aftermarket. In particular, we have been working on a system in the context of Aerogility, an agent-based decision support tool developed by Lost Wax to simulate aerospace aftercare. Here, we are applying the contracting technology developed in the CONTRACT project to a simulated environment analogous to that provided by the Aerogility simulation system. Rather than automating operation in the application domain, we have been investigating collaboration patterns emerging from contract-driven interaction between parties. By using contract monitoring techniques, aircraft operators and engine manufacturers can investigate the properties of contracts they are involved in, and analyse their impact.

This paper builds on earlier work on requirements [8] and an initial application framework [13] to focus on the development of monitoring mechanisms for detecting and explaining violation and fulfillment of normative specifications of agent behaviours encoded in electronic contracts. Specifically, we describe their usage and utility in an aerospace aftermarket application inspired by Aerogility [12]. The application demonstrates monitoring of aerospace agents deployed in the maintenance, repair and provision of aircraft engines. The agents' obliged, permitted and prohibited behaviours are governed by normative clauses specified in electronic contracts (*e-contracts*) to which the agents are signatories. These contracts instantiate a framework for electronic contract specification, execution and management [14], and the monitoring mechanisms implement a system for monitoring violation and fulfillment of norms (that may or may not be specified in contracts) [4].

In this model, *observers* are explicitly entrusted by contract signatories to accurately relay observations on the state of the world, where these observations are used by monitor agents to determine the violation or fulfillment state of norms. This use of trusted observers provides some measure of assurance that a norm *is reported* as violated if and only if it has *in actuality* been violated, and that sanctions will be applied only as and when appropriate. *Monitors* process observations together with *Augmented Transition Network* (*ATN*) [16] representations of individual normative clauses. The *ATN*s are graphs whose arcs have labels specifying world states, such that if a monitor receives observations indicating that these states hold, then the *ATN* is transitioned across the arcs, to nodes that represent activation, violation, and fulfilment states of the represented norm. The labels, together with additional information such as logged messages, constitute explanations of violation (and fulfilment) that are required to ensure correct assignment of responsibility for violation, and also to identify how prospects for future compliance, and thus realisation of system goals, can be enhanced by modifying business processes and normative specifications

Our implementation of norm based governance and monitoring of aerospace aftermarket agents builds on an earlier simulation model [13], as follows. First, it provides an implementation of the above-mentioned monitoring framework, in which a monitoring agent processes observations received from trusted observers together with *ATN* representations of norms in aftermarket contracts. Second, it shows how rudimentary explanations of norm violation are generated, and how more comprehensive explanations can be generated based on monitoring of *ATN*s representing permitted behaviours that are *not* necessarily contractually specified.

In seeking to demonstrate the applicability of this work, we provide two scenarios illustrating deployment of a monitor to detect and explain violations of norms by implemented *AgentSpeak(L)* [15] aerospace agents that are signatories to contracts. A key feature is the interdependency of contracts between agents at different points in the supply chain, from the supply of parts for engine maintenance to the supply of the repaired engines for aircraft. Thus, the scenarios illustrate how normative violations in one contract may imply normative violations in another contract, and how the explanations account for such interdependencies. In addition, we provide the ability to run simulations in which explanations of normative violations can inform subsequent revisions to contracts. Such revised contractual specifications can then be tested to determine whether they better ensure that norms are not violated (so ensuring that system-wide goals are met).

The paper is organised as follows. Section 2 gives an overview of the aerospace aftermarket domain and the contractual governance of agent behaviours in that domain. In Section 3 we briefly review prior work on e-contract specification of norms and monitoring of norms. The main contributions of the paper are then described in Sections 4 and 5. In Section 4 we describe the generation of explanations of violations, and in Section 5 we illustrate the use of monitoring and explanations in scenarios of the aerospace domain. Section 6 concludes and discusses future work.

2. THE AEROSPACE AFTERMARKET

The deployment of our contract-based monitoring architecture is situated in the context of aerospace aftercare contracts. In particular, we adopt an application scenario first introduced by Jakob *et al.*[8], and further developed by Meneguzzi *et al.*[13]. This scenario, in turn, was inspired by Lost Wax's Aerogility software [12], and simplified to better illustrate the use of the monitor. Aerogility is a decision support system, based on a simulation of aircraft engine aftercare contracts, aimed at identifying possible problems, such as supply bottlenecks and the effects of restrictions in a contract. In the simulation performed by Aerogilty, an engine comprises a number of modules on which wear and tear is simulated in order to foresee the need to perform maintenance in the engine, on an individual module basis. Aircraft engine manufacturers are contracted by airlines not only to supply working engines for their

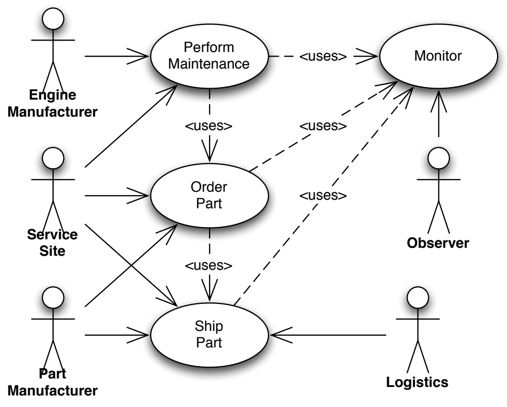


Figure 1: Use cases and interactions (in UML).

aircraft, but also to care for these engines throughout their lifetime following any number of conditions imposed by the airline. Among these conditions, aircraft operators may require that engines have a restricted provenance, (*e.g.* for used engines not to have been previously mounted by a competing airline), that replacement parts may only be supplied by certain approved part suppliers, and that engines must be serviced within a particular time frame. In this paper, we simplify the application domain to focus on a particular set of norms, and how these norms are monitored and explanations for the violation of these norms generated. As a consequence, we do not model the composition of an engine in modules, and assume that maintenance consists simply of replacing certain parts that need to be sourced from particular part suppliers. This simplification allows us to focus on simpler norms that, nevertheless, capture the general constraints that are simulated by Aerogility.

Thus, our application domain comprises, at the top level, *aircraft operators* signing aftercare contracts with *engine manufacturers*, including conditions of delivery deadlines and restrictions on the part suppliers allowed to supply replacement parts. Engine manufacturers, in turn, subcontract *service sites* at various airport hubs to service engines as they require maintenance. Service sites are obliged to deliver repaired engines within a maximum of 5 days, and thus need to make sure that all pre-requisites of the repair are performed well before this deadline, including the delivery of any required parts. During maintenance at service sites, parts may need to be ordered from *part suppliers* and shipped through *logistics* agents. When a service site needs to order parts, it sends requests for bids to all its known part suppliers, each of which then sends their bids to supply parts. In general, a service site will accept the bid with the lowest price and earliest delivery date estimate, and will turn down bids from part suppliers whose delivery estimate would render its own deadline commitments inviable, as well as bids from part suppliers forbidden in the contract terms. After a bid is selected and the parts order is placed, part suppliers ship the new parts to the service site. Once the parts arrive, the service site resumes repairing the engine, and readies it in the designated aircraft, notifying the engine manufacturer that the repair is complete. A summary of the use cases involved in our simulation is shown in the use case diagram of Figure 1, including monitoring (explained in Section 3.2).

Given the complexity of modern aircraft engines, their production and maintenance involves complex supply chains, with parts sometimes coming from a very limited range of suppliers. As a consequence, problems at the bottom of the supply chain (part suppliers) may easily cascade to the top (aircraft operators). In particular, part suppliers may experience delays in delivering parts to the service site, which in turn may prevent the service site from repairing an engine on time, and result in the engine manufacturer violating its contract with the aircraft operator. These delays may occur for a number of reasons: a part supplier may take longer than expected to fabricate a new part; the logistics agent may delay shipping; or the service site may not find a permitted part supplier to supply parts.

In this paper, we focus on the supply chain from the engine manufacturer down to the part suppliers, so the simulations of Section 5 do not deal with the aircraft operators. Instead we focus on the interactions (and normative restrictions) between engine manufacturer, service site, part suppliers and logistics agents. In these simulations, we use the monitor of Section 3.2 to detect contract violations and generate explanations for the origin of the violations. The explanations thus generated guide the improvement of the contracts and the simulation of the application domain.

3. NORMS AND MONITORING

In this section we briefly review our previous work on specification of [14], reasoning about, and monitoring [7, 4] *norms* by *norm aware* agents, which serves as a base on which we have developed our aerospace aftermarket application.

Norms are essentially rules that can be categorised as *permissions*, *obligations*, and *prohibitions*, that respectively describe what may be done, should be done, and should not be done. While agents typically comply with norms, they may decide not to do so if it serves their self-interest, or simply if they cannot do so. For example, an optimistic agent may over-commit on a manufacturing contract, agreeing to provide more goods than it is capable of manufacturing, after estimating that typically, large orders will not arrive simultaneously. If the latter does occur, it may be unable to meet one of its orders without incurring extra expense (e.g. by sub-contracting), thus violating a norm. In such scenarios, culpable agents are subject to sanctions (that may themselves be specified as *contrary-to-duty* obligations that come into force when other obligation are violated); in the above example, the manufacturer may be obliged to pay a fine if an order is not delivered on time.

3.1 Representing Norms

Philosophers and computer scientists have carried out much research into normative systems (e.g. [6, 9, 10]), most of which focus on deontic logic formalisations. While deontic logics provide formal models for reasoning about norms, their representations are somewhat abstract and do not readily lend themselves to implementation. To address this drawback, we have proposed a more readily implementable framework for representation and reasoning about norms [14]. While the framework makes no assumptions about the context in which the norms are specified, it aims to easily handle norms appearing within contracts. In [14], a norm $\mathcal{N}$ is a tuple:

$$\begin{aligned}\langle &Type,\\ &ActivationCondition,\\ &NormCondition,\\ &ExpirationCondition,\\ &Target\rangle\end{aligned}$$

where $Type$ identifies whether the norm is an obligation or permission[1], and the next three parameters are all predicates expressed in some logical language (which we leave unspecified). The activation condition describes some state of the world in which $\mathcal{N}$ comes into force; the normative condition describes some world state in which $\mathcal{N}$ is being complied with (in the case of an obligation), or being made use of (in the case of a permission). The expiration condition describes some world state in which $\mathcal{N}$ no longer has normative force, and $\mathcal{N}$'s target set specifies which agents are responsible for, or allowed to make use of, the norm. Consider the following example obligation on a service site SS to repair an aircraft engine E at some time T' that is within 7 days of receiving a request from an engine manufacturer EM at time T:

$$\begin{aligned}\langle &Obligation,\\ &received(engine\_repair\_request, EM, E, T),\\ &engine\_repaired(E, T') \vee (T' \leq T + 7),\\ &engine\_repaired(E, T') \vee (T' > T + 7),\\ &SS\rangle\end{aligned}$$

It is important to note that the above norm can be considered to exist as an *abstract* norm that comes into force, and so only exists in its *instantiated* form when a specific repair request for a specific engine is received by a specific service site from a specific engine manufacturer. Thus, an abstract norm serves as a template for instantiated norms; when an abstract norm's activation condition evaluates to true, an instantiated norm is created and specialised to the situation (i.e., the variables EM, E, T and SS are instantiated). Multiple instantiated norms may be spawned from a single abstract norm, and may exist at the same time. Once instantiated, a norm persists until its expiration condition is met, regardless of the state of its activation condition.

3.2 Monitoring for Compliance

In this subsection, we build on our representation, and outline our monitoring framework [4, 7] in order that we can subsequently apply it to the aerospace aftermarket domain. Now, the goal of our monitoring framework is to be able to identify the status of a norm at any point in time. Typically, such statuses include whether a norm is abstract or instantiated, as well as whether it has been violated or expired. The ability to determine and reason about such statuses is useful not only to agents interacting with each other within the system, but also for activation of other norms. For example, a sanction may take the form of a contrary-to-duty norm obliging an agent to pay a penalty, which may come into effect if the status of some other norm is "violated".

In our monitoring framework, monitors receive observations from trusted observers that are explicitly entrusted by all contract parties to accurately report on the state of the world. These observations are then processed, together with *Augmented Transition Network* (*ATN*) [16] representations of norms, to determine their status.

The use of trusted observers ensures some degree of certainty that a norm will be *reported* as violated if and only if it has in *actuality* been violated, and so some assurance that sanctions will only be applied as and when appropriate. Such assurances are important in order to encourage deployment of agents in electronic contracting environments. For example, consider an obligation on a service site to pay for engine parts within a certain time after receipt of these parts. Recognition that the payment has been made may be based on an observer reporting that the monies for payment have been deposited in the bank account of the part supplier, where the observer may be the bank itself, and where the bank has been explicitly entrusted by the service site and part supplier (in the relevant supply contract) to report that the monies have been paid if

[1]We assume prohibitions are obligations to ensure some state of affairs does not hold.

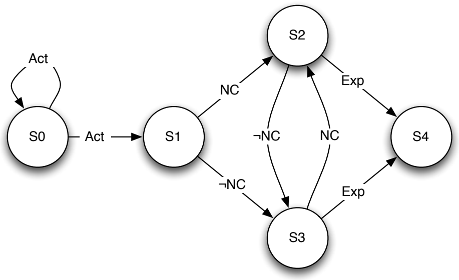


Figure 2: The ATN used to represent norms.

and only if they have in actuality been paid. Thus, if no such report of payment is received by a monitor, then there is some degree of certainty that the monitor's reporting that the obligation has been violated is accurate.[2]

Observations relayed by trusted observers to monitors may either be messages observed as having been sent to and from contract parties (e.g., a message received by a service site requesting repair of an engine), or predicate logic descriptions of properties holding in the world (e.g., that an engine has been repaired, or an action has occurred). In either case, these observations are processed, together with *ATN*s, to determine the status of the represented norms.

These *ATN*s are essentially 5-node directed graphs in which each node represents a distinct state of the same norm. Based on messages received from observers describing the states of interest specified by the norm's components, $Activation$, $NormCondition$ and $Expiration$, the monitor matches the messages with the labels of the *ATN*'s arcs that describe the corresponding states of interest, so as to transition the *ATN* from one node to the next. Figure 2 shows a generic *ATN* representation of a norm, as follows.

- $S0$ is the node indicating that the norm is in its abstract form.

- If the monitor receives observations indicating that the activation condition holds, then the *ATN* transitions across the arc labelled by the activation condition, to $S1$, in which case the norm is said to be activated. Thus, in our example obligation, if an observer relays a message received by a specific service site from an engine manufacturer requesting repair of a specific engine, then the *ATN* for the obligation is transitioned to $S1$.

- If, subsequently, the monitor immediately receives observations indicating that the normative condition holds, then the *ATN* is transitioned to $S2$, otherwise it transitions to $S3$. $S2$ thus represents the non violated state of an obligation, or the 'made use of' state of a permission, and $S3$ the violation state of an obligation, or the 'not made use of' state of a permission. In our example obligation, the normative condition can be expressed as $engine\_repaired(E, T') \lor (T' \leq T+7)$ so, if at the current time T', either of these disjuncts is observed as holding (if the engine is repaired at time T', or if T' is within the seven day time window), then the *ATN* transitions to the non-violated state $S2$. If the normative condition does not hold, i.e., neither disjunct is observed as holding (the engine is not repaired and the current time is greater than seven days after activation), then the ATN transitions to the violated state $S3$. Notice that in general, norms may toggle between $S2$ and $S3$. For example, if an obligation to drive on the left is activated once a driver begins to drive then, at each subsequent time point, the driver may be observed to be driving on the left, in which the case the corresponding *ATN* is in state $S2$, or the driver may be observed to be driving on the right, in which case the *ATN* is in state $S3$.

- Finally, if at any time point T' the monitor receives observations indicating that the normative expiration holds (either $engine\_repaired(E, T')$ or $T' > T + 7$ is observed as holding) then the *ATN* transitions to $S4$. Since this is effectively a terminal state, the *ATN* is no longer processed.

[2]Contrast this with either the service site or part supplier reporting that the monies have been paid, where it may be in the interest of these parties to mis-report the truth of the matter.

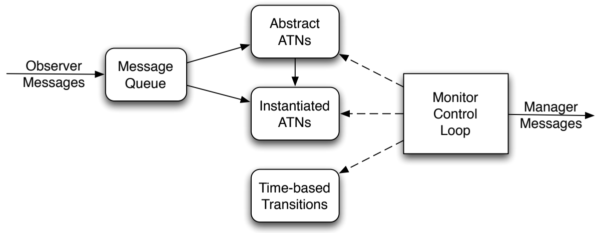


Figure 3: Overview of the monitor control loop.

3.3 Monitoring Algorithm

The monitor itself fulfils a number of functions. First, it acts as a store for all *ATN*s within the system. Second, it is responsible for routing messages to the appropriate *ATN*s. Third, it must send out notifications to the appropriate agents when a norm is instantiated, violated, fulfilled or expires (i.e. when a norm's status changes in a specific way).

Storing *ATN*s is relatively trivial, and will not be discussed in depth, but the monitor is able to add, remove, and retrieve *ATN*s. Message routing requires sending any received messages to the appropriate *ATN*s, causing them to transition between states as described above. The monitor is responsible for passing one other type of message to *ATN*s, namely time messages. These are sent to all *ATN*s at regular intervals, indicating the passing of time. *ATN*s process these messages differently, transitioning only when a timeout condition on the edge successfully processes the time message. These interactions are illustrated in Figure 3. Moreover, since the monitoring algorithm handles each norm ATN independently (both abstract and instantiated), it is easy to distribute it across multiple computers, each handling a set number of ATNs.

We may conceptually view the monitoring process as following the algorithm described in Table 1, in which the monitor repeatedly dispatches messages to ATNs. If an abstract *ATN*'s activation condition is satisfied (line 7), the *ATN* is instantiated (lines 8 to 10), and added to the set of instantiated *ATN*s. Messages are then dispatched to all instantiated *ATN*s in the system. If the message satisfies the requirements on an arc leaving the current node (line 15), a transition takes place (line 17), and agents are notified of the transition as appropriate (lines 18 to 21). Finally, if the transition leads to an expired state, the instantiated norm is removed from the instantiated norm set (line 22). We assume that timestep messages are treated as any other message.

4. EXPLANATIONS OF VIOLATIONS

Given the *ATN* representation and the monitoring algorithm above, we can now consider how the monitor generates explanations

Algorithm 1 Monitor control loop

```
Require: Message queue $Q_{msg}$
Require: Message store $M_{St}$
Require: Set of abstract norm ATNs $\mathcal{X}_{Abs}$
Require: Set of instantiated norm ATNs $\mathcal{X}_{Inst}$
 1: while Monitor is active do
 2:   while $Q_{msg}$ is not empty do
 3:     Retrieve $Msg$ from head of $Q_{msg}$
 4:     Add $Msg$ to $M_{St}${First, deal with messages}
 5:     for all Abstract norm ATN $A$ in $\mathcal{X}_{Abs}$ do
 6:       for all Arcs $Act$ in $\mathcal{A}$ do
 7:         if $satisfied(M_{St},Act)$ then
 8:           create $I$, an instantiated version of the ATN $I$ of $A$
 9:           add $I$ to $\mathcal{X}_{Inst}$
10:           move $I$ to state $S1$
11:         end if
12:       end for
13:     end for
14:     for all Instantiated norm ATN $I$ in $\mathcal{X}_{Inst}$, with current state $C$,
        do
15:       for all Arcs $a$ leaving $C$, going to $T$, do
16:         if $satisfied(M_{St},a)$ then
17:           move $I$ to state $T$
18:           if $T$ is $S3$ then
19:             Notify manager of violation
20:           else if $T$ is $S4$ then
21:             Notify manager of expiration
22:             remove $I$ from $\mathcal{X}_{Inst}$
23:           end if
24:         end if
25:       end for
26:     end for
27:   end while
28: end while
```

for contract clause (i.e. norm) violations.

4.1 Explaining Contract Violations

We consider first how a simple explanation could be formulated by the monitor. As we have seen, the monitor detects that a norm is violated when the *ATN* associated with it transitions to a particular state, S3. Since every *ATN* is associated with a norm, an initial approach to explaining a contract violation is to state which norm in the contract has been violated.

However, an explanation consisting solely of the norm instance that was violated is often inadequate for diagnosing any deeper causes for a violation, as it expresses only the *fact* and not the *causes* of the violation. A refinement of the explanation would be to include the *ATN* transition that led to the violating state, i.e. the observation of a message or state by which the monitor detected the obligation or prohibition had not been fulfilled. This refined explanation, which we will refer to as a *single-ATN explanation* below, would allow a designer to pinpoint the *immediate cause* of a norm being violated. However, it does not help to determine the overall circumstances relevant to violation, for which we would also need to explain the *indirect causes* leading up to the violation.

We therefore need to consider ways to recognise other observations of relevance to a violation, thus building a richer picture of the surrounding circumstances. Notice that, as shown in the monitoring algorithm of Section 3.3, each observation arriving at the monitor is supplied to *every ATN* being monitored, allowing the monitor to detect if a single observation causes more than one *ATN* to activate, expire, or transition between violation and non-violation. Multiple norms being affected (i.e. multiple *ATN*s being activated, violated, unviolated, expired) by the *same* observation, means that violations or fulfilments of those norms have a *common* cause.

Therefore, we can enhance our explanations of one violation by reporting the observations causing transitions in other *ATN*s with the same arc labels. For example, an engine requiring repair is an activation condition for both returning the engine to working order, and for ordering parts necessary to make repairs, each monitored separately. If the engine is not repaired in time, violating the first obligation, then observations relating to the ordering of parts may help explain why this was the case. An explanation using this enhancement can thus take the form of a set of single-ATN explanations *chained* together by the observations common to each ATN. It is important to note here, that we assume a completely observable environment, and thus the explanations will be deterministic, and based on the ATNs available to the monitor. We will see an example of a chained ATN explanation in Section 5.

4.2 Improving Explanations

The primary idea of our monitor is to detect violations of norms stated in a contract agreed between the contract parties, e.g., engine manufacturers and site services. Now, the observations that the monitor receives (through subscribing to trusted observers) are just those which indicate those norms are activated, violated or not, or expired. However, as described above, our explanations of violations of the norms comprise the relevant observations received by the monitor. Therefore, the explanations that the monitor can give are limited to observations directly relevant to determining the fulfilment of the contract clauses.

This is often inadequate for good explanations. For example, a contract may place no obligations, prohibitions or permissions on how engine parts are supplied, and so the monitor will receive no observations regarding this supply, but it may be exactly this factor that has led to the violation of the obligation to repair an engine (i.e. something wrong with the part supply chain).

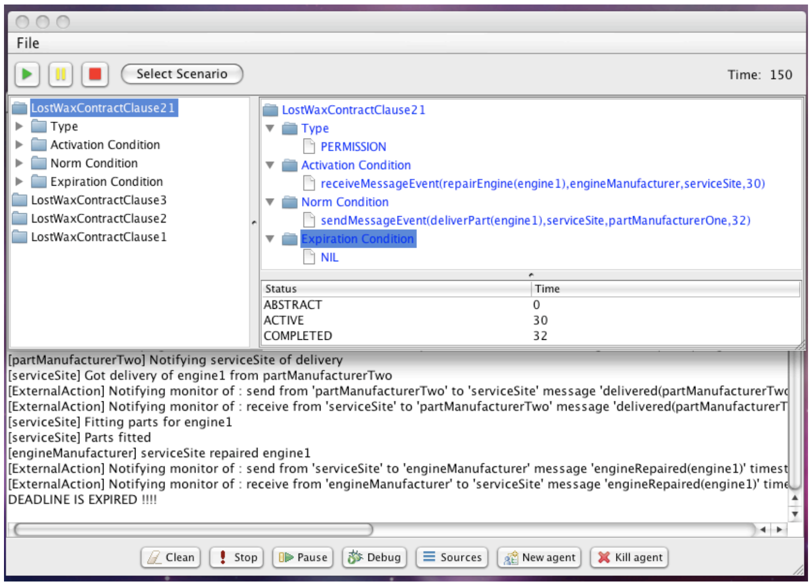
In our approach, we aim to improve explanations generated by the same monitoring machinery described above, by adding *ATN*s representing norms not present in the contract but helpful to monitor, purely to build better explanations: *explanation ATNs*. So, in the example above, we would want to add explanation *ATN*s for monitoring the part supply chain, to ensure the monitor had records of observations relating to part supply and could determine where part supply problems and repair violations had a common cause.

An explanation *ATN* has no qualitative difference from an *ATN* representing a contract clause. Every explanation *ATN* is of type *permission* (rather than obligation or prohibition), because it does not state what should happen, but what could happen, e.g. parts *could* be delivered by this supplier. There is, therefore, no notion of an explanation *ATN* itself being violated. Next, we look at how this explanation mechanism is used in our aerospace case study.

5. SIMULATION SCENARIOS

The CONTRACT project has implemented a range of technologies for supporting contract-based application execution, all available as open source [1]. In this paper, the specific component we have focussed on is the monitor, which behaves in the way described in Section 3. The scenarios themselves are simulated using the AgentSpeak(L) language [15] and run on the Jason [3] platform. A graphical user interface allows users to explore the norms and see the violation or fulfilment states reported by the monitor during run-time, and a screenshot of this is shown in Figure 4. The screen shows the formalised contract (top left), instantiated norms (top right), the current monitored status of an instantiated norm (mid right), and a log of the actions agents are taking (bottom).

In order to show how the explanations generated by our monitor are used to detect both the origin of violations and potential problems in the aerospace aftercare simulation, we have designed two



```
LostWaxContractClause21
  Type
  Activation Condition
  Norm Condition
  Expiration Condition
LostWaxContractClause3
LostWaxContractClause2
LostWaxContractClause1
```

```
LostWaxContractClause21
  Type
    PERMISSION
  Activation Condition
    receiveMessageEvent(repairEngine(engine1),engineManufacturer,serviceSite,30)
  Norm Condition
    sendMessageEvent(deliverPart(engine1),serviceSite,partManufacturerOne,32)
  Expiration Condition
    NIL
```

Status	Time
ABSTRACT	0
ACTIVE	30
COMPLETED	32

```
[partManufacturerTwo] Notifying serviceSite of delivery
[serviceSite] Got delivery of engine1 from partManufacturerTwo
[ExternalAction] Notifying monitor of : send from 'partManufacturerTwo' to 'serviceSite' message 'delivered(partManufacturerTwo
[ExternalAction] Notifying monitor of : receive from 'serviceSite' to 'partManufacturerTwo' message 'delivered(partManufacturerT
[serviceSite] Fitting parts for engine1
[serviceSite] Parts fitted
[engineManufacturer] serviceSite repaired engine1
[ExternalAction] Notifying monitor of : send from 'serviceSite' to 'engineManufacturer' message 'engineRepaired(engine1)' timest
[ExternalAction] Notifying monitor of : receive from 'engineManufacturer' to 'serviceSite' message 'engineRepaired(engine1)' time
DEADLINE IS EXPIRED !!!!
```

Figure 4: Screenshot of the monitoring application.

typical scenarios that demonstrate the use of the monitor explanations in improving the norm-regulated multiagent system. As we have seen in Section 2, violations at the bottom of a complex supply chain can easily cascade to the top. Thus, in both the following scenarios, an obligation on an engine manufacturer, to make available engines to an airline operator, are fulfilled or violated based on whether the sub-contracted service site repairs the engine for the engine manufacturer within a given time period. Furthermore, provenance restrictions on parts, dictated by the airline operator to the engine manufacturer, translate to permissions and prohibitions on the service site's ordering of parts from part suppliers.

In both the following scenarios, we describe monitoring of a contract between the service site and engine manufacturer. The norms encoded include the obligation on the service site to repair engines in a given time frame, as specified in Section 3.1 (its ATN representation is the example in Section 3.2). We identify this obligation as *Ob1* in examples below. In addition, the contract includes the following clauses, all imposed on the service site agent (*SS*).

- A permission to order parts from part supplier 1, *Per1*, represented by an *ATN* as in Figure 2, where:
  - Activation condition (Act) = engine repair request received at time T (the same activation condition as *Ob1*)
  - Normative condition (NC) = order parts from part supplier 1 (so ¬NC = not order part from supplier 1)
  - Expiration condition (Exp) = part ordered from part supplier 1
- A permission to order parts from part supplier 2, *Per2*, represented in the same way as *Per1* except that part supplier 2 replaces part supplier 1 in its specification.
- A prohibition on the service site to order parts from part supplier 3 (expressed as an obligation not to order parts from part manufacturer 3), *Pro1*, represented by an *ATN* with:
  - Activation condition (Act) = engine repair request received at time T (the same activation condition as the obligation)
  - Normative condition (NC) = not order parts from part supplier 3 (¬NC = order part from supplier 3)
  - Expiration condition (Exp) = false (the norm never expires).

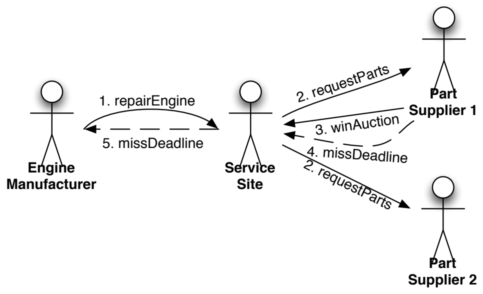


Figure 5: Original scenario representation.

5.1 Scenario 1: Tracing Violations

In the first scenario considered (depicted in Figure 5), the following events are simulated.

1. The service site, *SS*, receives a repair request. This event activates *Ob1*, so the corresponding *ATN* transitions to $S1$ and then immediately to the non-violated state $S2$, given that the 7 days have not elapsed. Similarly, *Per1* is activated and transitions to state $S3$ (indicating that the permission has not been taken advantage of).
2. In a subsequent bidding phase, *SS* makes a request for a required part to part supplier 1 (*PS1*), which in turn responds by indicating that it has the part in stock and the part can be delivered in some given time.
3. *PS1*'s expected delivery time is acceptable to *SS* (in the sense that it leaves enough time for SS to fulfill its time-limited

obligation to the engine manufacturer). *SS* then makes use of *Per1* to order the part from *PS1*, and the corresponding *ATN* (that has also been activated by receipt of the repair request and so is already in $S1$) transitions to $S2$ and then $S4$ (the permission has been made use of and then expires). Notice that in Figure 5, the bidding phase and subsequent use of the permission is indicated by the *winAuction* arrow.

4. On the 8th day after activation, no message informing the engine manufacturer of delivery of the engine has been observed. *Ob1*'s *ATN* transitions to the violation state $S3$ and then immediately expires, transitioning to $S4$.

Following this scenario, the monitor will report that *Ob1* has been violated. However, if it only monitors the four contractual norms defined above (*Ob1*, *Per1*, *Per2*, and *Pro1*), the explanation that the monitor can provide in this scenario consists only of the observations that:

1. a repair request was received (activation condition of *Ob1* and *Per1*);

2. a part was ordered from *PS1* (normative and expiration condition of *Per1*); and

3. the engine was not repaired in the time limit (negation of normative condition and expiration condition of *Ob1*).

As discussed in Section 4.2, this explanation is inadequate. To properly understand what has occurred, we would want to know more about what occurred, e.g. what availability information the part supplier provided and thus whether the service site was justified in ordering a part from that supplier on the basis of that information. To do this, we add (for future runs of the simulation) explanation *ATN*s (*ATN*s for possible behaviour not explicitly permitted in the contract) regarding the bidding process. Figure 6 shows the chained explanation that results from the chained *ATN* explanation once explanation *ATN*s are added. Each box depicts a single-*ATN* explanation, with the majority being explanation *ATN*s causally connected to *Per1*.

To further illustrate the point, the user receiving such an explanation may still not be satisfied that enough has been provided to pinpoint the events leading to violation. Therefore, they may add further explanation *ATN*s regarding the logistics agent used to deliver the parts from part supplier to the service site. Such a scenario is depicted in Figure 7. This *drilling down* to include interconnected, relevant details by adding to that part of the system that is monitored can continue indefinitely, but the key aspect is that the full monitoring process only occurs for events deemed potentially relevant to the contractual norms by the user; that is, we are not trying to monitor and connect all events in the system to the violation of contracts.

5.2 Scenario 2: Relaxing Prohibitions

In the second scenario illustrated in Figure 8, SS again violates its obligation to repair the engine within 7 days, but the explanation of the violation above indicates that this is because both the permitted part suppliers *PS1* and *PS2* are not able to provide successful bids. *PS1* cannot supply the part in time, and *PS2* does not have the part in stock. *SS* is prohibited from sourcing parts from the only other available part supplier *PS3*, and so violates its obligation to the engine manufacturer. After analysing the explanation, a designer may conclude that the prohibition *Pro1* should be relaxed in case no other part suppliers are available. Thus *PS3* can be sourced in such exceptional circumstances so that obligation deadlines can be met, as illustrated in Figure 9.

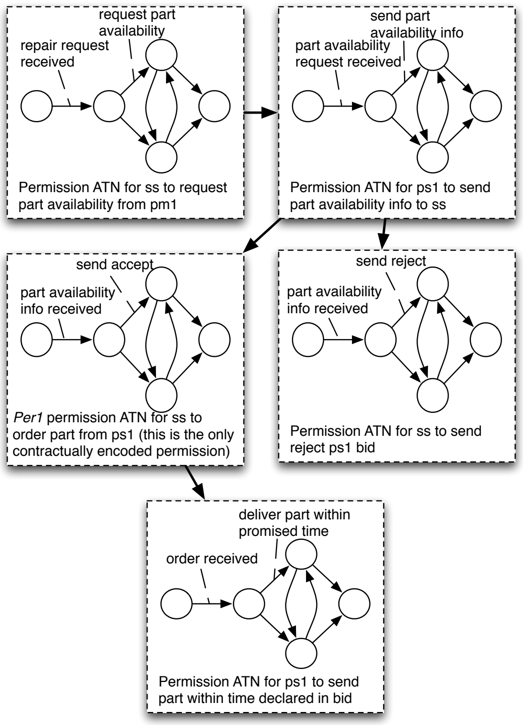

request part availability
repair request received
Permission ATN for ss to request part availability from pm1
send part availability info
part availability request received
Permission ATN for ps1 to send part availability info to ss
send accept
part availability info received
Per1 permission ATN for ss to order part from ps1 (this is the only contractually encoded permission)
send reject
part availability info received
Permission ATN for ss to send reject ps1 bid
deliver part within promised time
order received
Permission ATN for ps1 to send part within time declared in bid

Figure 6: Chained explanation ATNs for Scenario 1

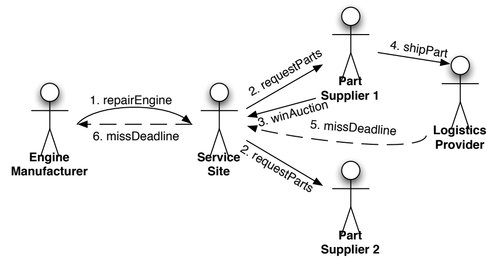


Figure 7: Expanded representation.

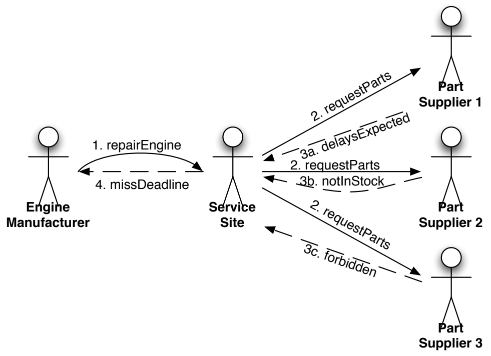


Figure 8: Original scenario leading to failure.

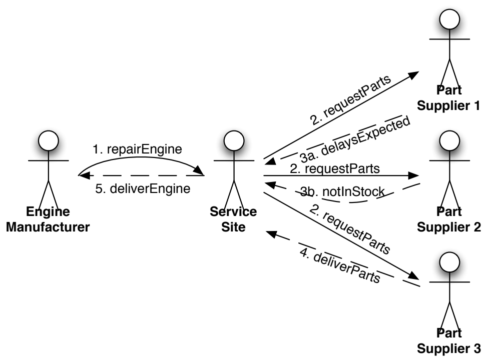


Figure 9: Relaxed restrictions lead to success.

The above illustrates how sophisticated monitoring and explanation of violations feed into a more general methodology for refining contractual specifications. The approach is particularly applicable in complex scenarios in which the reasons why high level goals (such as obligations to have minimum numbers of serviced engines readily available for airline operators) cannot be realised, are not immediately apparent. Such scenarios occur in simulations such as Aerogility's, which also allows parameters affecting norms to be changed (e.g. adding or relaxing prohibitions) in order to test various contractual specifications, and thus arrive at an optimal specification that maximises the chances of realising the high level goals.

6. CONCLUSIONS AND RELATED WORK

The value of explicit contracts between parties comes not only from giving those parties a clear understanding of their obligations, but also in making explicit what constitutes a *violation* of those obligations. However, for violations to have a productive effect, such as to improve business processes to ensure they do not occur again, we have to be able to *detect* that they have occurred and *explain* the circumstances. In an application of any complexity, such as we have in the collaboration between parties in the aerospace domain, this is a non-trivial problem requiring robust and well-specified solutions.

In this paper, we have applied our technologies for *representing* contractual norms in a precise manner, *monitoring* for the violation of these norms, and *explaining* the outcomes of the monitoring process, to aerospace aftermarket scenarios. We have shown that, by using such techniques, critical information is conveyed to those attempting to understand and improve a system. In particular, we have shown that, by monitoring successively detailed parts of a system we can drill down to the root cause (and then fix it), and that by explicitly representing contractual norms and understanding the causes of contract violation we are better placed to alter those contracts to better govern the system. We have presented two scenarios describing how the above benefits are realised in the aerospace aftermarket scenarios.

In future work, we will continue to develop the detection and explanation abilities of the monitoring architecture, with one potential avenue to adapt techniques from truth maintenance systems and belief revision to track the multiple-cause-and-effect links that may exist [2]. Since the current system does not provide any support for users to create explanation-ATNs, we will also consider a methodology for creating them; in particular, by specifying the methodology of the case study scenarios, as robust, re-usable guidelines. We will also look at using the explanations of violations to determine what *danger states* may be detected, i.e. states of the system indicating that violation may be imminent without pre-emptive action.

Acknowledgements: The research described in this paper is partly supported by the European Commission Framework 6 funded project CONTRACT (INFSO-IST-034418). The opinions expressed herein are those of the named authors only and should not be taken as necessarily representative of the opinion of the European Commission or CONTRACT project partners. The first author is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) of the Brazilian Ministry of Education.

7. REFERENCES

[1] IST Contract Open Source Libraries. http://ist-contract.sf.net, 2008.

[2] J. J. Alferes, L. M. Pereira, and T. C. Przymusinski. Belief revision in non-monotonic reasoning and logic programming. *Fundamenta Informaticae*, 28(1-2):1–22, 1996.

[3] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007.

[4] N. Faci, S. Modgil, N. Oren, F. Meneguzzi, S. Miles, and M. Luck. Towards a monitoring framework for agent-based contract systems. In M. Klusch, M. Pechoucek, and A. Polleres, editors, *Cooperative Information Agents XII*, 2008.

[5] D. Grossi. *Designing Invisible Handcuffs*. PhD thesis, Utrecht University, SIKS, 2007.

[6] J. F. Horty. *Agency and Deontic Logic*. Oxford University Press, 2001.

[7] IST CONTRACT project. D5.2.1 - Contract Monitoring Mechanisms and Tools. http://www.ist-contract.org, 2008.

[8] M. Jakob, M. Pěchouček, J. Chábera, S. Miles, M. Luck, N. Oren, M. Kollingbaum, C. Holt, J. Vázquez, P. Storms, and M. Dehn. Case studies for contract-based systems. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[9] C. Krogh. The rights of agents. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Agent Theories, Architectures, and Languages II*, volume 1037, pages 1–16. Springer, 1996.

[10] F. Lopez y Lopez. *Social Power and Norms: Impact on agent behaviour*. PhD thesis, University of Southampton, 2003.

[11] F. Lopez Y Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12(2-3):227–250, 2006.

[12] LostWax. Aerogility. http://www.aerogility.com/, 2007.

[13] F. Meneguzzi, S. Miles, C. Holt, M. Luck, N. Oren, S. Modgil, N. Faci, and M. Kollingbaum. Electronic contracting in aircraft aftercare: A case study. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 63–70, 2008.

[14] N. Oren, S. Panagiotidi, J. Vazquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. In *Proceedings of COIN@AAAI*, pages 61–68, 2008.

[15] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. V. de Velde and J. W. Perram, editors, *Agents Breaking Away; MAAMAW 96*, volume 1038 of *LNCS*, pages 42–55. Springer, 1996.

[16] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970.