A contract-based system for aerospace aftercare

Felipe Meneguzzi[1], Sanjay Modgil[1], Nir Oren[1], Simon Miles[1], Michael Luck[1], Camden Holt[2], and Malcolm Smith[2]

[1] King's College London, London WC2R 2LS, UK
felipe.meneguzzi@kcl.ac.uk sanjay.modgil@kcl.ac.uk
nir.oren@kcl.ac.uk simon.miles@kcl.ac.uk michael.luck@kcl.ac.uk
[2] Lost Wax, London, UK
camden.holt@lostwax.com malcolm.smith@lostwax.com

Summary. The logistics of the aerospace aftermarket raises a number of very interesting challenges from the perspective of electronic contracting. This is a highly dynamic domain, where contracts are established between airlines and engine manufacturers, as well as between engine manufacturers all the way down the supply lines, providing a particularly illustrative showcase for the technologies developed in the CONTRACT project. In this paper, we describe such a domain, as well as our modelling of it as a multiagent simulator where the CONTRACT framework has been used to monitor for compliance with norms.

Key words: Electronic contracting, norms, contracts, BDI, AgentSpeak(L)

1 Introduction

As more and more business processes are automated and conducted over networks, computational tools have become indispensable in ensuring that businesses remain competitive. In the transition from traditional methods of conducting business among enterprises to automated workflows, one bottleneck in the speed at which business deals are closed lies in the human decision-makers that are still necessary to review the terms of contracts and ensure that once signed, contracts are properly followed. If parts (or even the entirety) of the negotiation and enactment of contracts between businesses are to be automated, the need for a comprehensive framework for electronic contracting becomes apparent.

In this context, the CONTRACT project has developed a comprehensive framework for the creation, management [1] and monitoring [2] of electronic contracts. Electronic contracts comprise sets of *norms*, *i.e.* stipulations regarding expected behaviour, usually expressed in terms of deontic concepts such as obligations, permissions and prohibitions. Development of our framework was informed by a series of case studies described by Jakob *et al.*[3] and led to the various requirements fulfilled by the resulting framework. One of these case studies, which is used in this paper, concerns the aerospace aftermarket, a complex domain in which aircraft engine manufacturers do not simply sell engines, but instead provide a service consisting of maintaining a pool of operational engines available for an airline. This case study is based on Lost Wax's Aerogility [4], an agent-based decision support tool to simulate the aerospace aftermarket.

In this paper, we describe the model used in our simulated aftercare environment, and the way in which the entities (*i.e.* agents) within the model interact with the elements of the CONTRACT framework, with a focus on the monitoring component. We start by describing the aerospace aftermarket case study in Section 2, followed by an overview of the CONTRACT framework in Section 3, its monitoring component in Section 4, and our modelling and implementation of the case study in Section 5. Finally, in Section 6 we conclude.

2 The Aerospace Aftermarket

The aerospace aftermarket is increasingly populated by customers buying a service rather than a product. For example, aircraft engine manufacturers provide long term commitments to make available operational engines for the aircraft of airline operators in order that the aircraft are not grounded while awaiting engines, and thus prevented from flying. Specifically, these commitments[1] consist of having minimum numbers of spare engines available at specific locations (a given engine manufacturer may service an airline operator at multiple sites) and not allowing any aircraft to be idle for greater than an agreed duration.

These minimum service level commitments are stipulated in aftercare contracts. If the commitments are violated (e.g., when an airline operator's aircraft is grounded, awaiting functioning engines for a period of time greater than that agreed with the engine manufacturer), then engine manufacturers receive predetermined financial penalties. In this business model, servicing and maintenance becomes a key driver of long term profitability for the engine manufacturer; aftercare contracts are worth millions of euros and can last several years.

Based on these contracts, engine manufacturers establish repair contracts with service sites (usually located at airports) who are responsible for the actual repair of aircraft engines. Commitments on an engine manufacturer to have engines available for a given airline operator imply commitments in a repair contract, requiring that a service site repairs engines within a given time period.

Aftercare and repair contracts also contain other interdependent commitments that are secondary to the core service level and repair commitments. For example, a given airline operator may place restrictions on the provenance of engines. Thus, an engine manufacturer may be committed to not using engines previously mounted on the aircraft of one of the airline's competitor airlines. Similarly, an engine manufacturer may be committed to not using parts for engine repair supplied by specific part suppliers. These restrictions may in turn need to be stipulated in the repair contracts, so that if an engine manufacturer is committed to not using parts from a given part supplier, then the repair contract between the engine manufacturer and the service site responsible for the actual repair must also prohibit the service site from ordering parts from that part supplier.

The aerospace aftermarket use case, first introduced by Jakob *et al.*[3], and further developed by Meneguzzi *et al.*[5], is therefore an ideal application for evaluating

[1] The CONTRACT semantics and language make the notion of a commitment more concrete, by distinguishing between, and giving structure to obligations, prohibitions and permissions.

and validating CONTRACT[2] concepts and technologies. 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.

3 The Contract Architecture

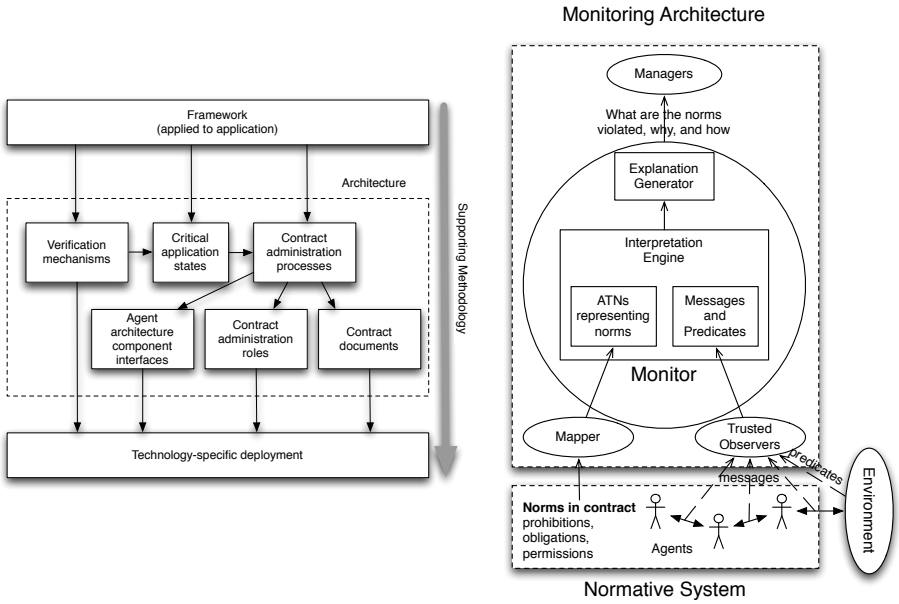
The models and procedures comprising the CONTRACT *framework* and *architecture* are shown in Figure 1(a). The primary component of this is the framework itself, depicted at the top of the figure, which is the conceptual structure used to describe a contract-based system, including the contracts and the agents to which they apply. From the framework specification of a given application, other important information is derived. First, understanding the contractual obligations of agents allows us to specify the *critical states* that an application may reach. A critical state of a contract-based system with regard to an obligation essentially indicates whether the obligation is fulfilled or fulfillable, e.g. achieved, failed, in danger, etc. A state-based description, along with the deontic and epistemic implications of the specified contracts, can then be used to verify a system either off-line or at run-time [6] (though we do not discuss this further here).

The framework specification is used to determine suitable processes for administration of the electronic contracts through their lifetimes, including establishment, updating, termination, renewal, and so on. Such processes may also include observation of the system, so that contractual obligations can be enforced or otherwise effectively managed, and these processes depend on the critical states identified above. Once suitable administration processes are identified, we can also specify the roles that agents play within them, the components that should be part of agents to allow them to manage their contracts, and the contract documents themselves.

4 Monitoring

Based on the case study outlined in Section 2, we have developed a prototype agent system in which we employ the CONTRACT architecture from Section 3. Our prototype models aerospace agents, and observes message exchanges between agents, where these messages are indicative of the *interactions* between agents. These messages are gathered by an observer agent, which makes use of a monitor component, and sends compliance data to a manager agent that takes action when warranted. The observed

[2] http://www.ist-contract.org/



(a) The CONTRACT framework. (b) Monitoring architecture.

Fig. 1. The overall structure of the CONTRACT architecture and framework.

messages are processed by a monitor component, together with transition network representations of the norms in the contract that the aerospace agents are signatories to. Based on this processing, the monitor reports on the norms that are activated, their fulfilment status, and the norms that expire.

In the CONTRACT monitoring framework, monitors receive observations from 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) [7] 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 =ed, and so provides some assurance that sanctions will only be applied as and when appropriate. Once the status of a norm is ascertained through the monitoring process, the decision of what actions are to be taken is delegated to *manager* agents, which might apply sanctions for violations and rewards for fulfilment, as appropriate. This flow of information from the interacting agents through the monitor and to managers is illustrated in the diagram of Figure 1(b).

Observations relayed by 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 ATNs, to determine the status of the norms that these ATNs represent. In particular, we use the semantics of Modgil *et al.*[8] for translating norms into ATNs and processing them to determine norm status.

Briefly, the resulting norm ATNs are 5-node directed graphs in which each node represents a distinct state of the norm represented by the ATN (illustrated in Figure 2(b)). Based on messages received from observers describing the states of interest specified by the norm's components, *ActivationCondition* (Act), *NormCondition* (NC), and *ExpirationCondition* (Exp), 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. Thus the node in which the ATN is in, and the messages labelling the arcs that have been transitioned, respectively indicate the status of the norm and provide a rudimentary explanation as to why the norm has that status.

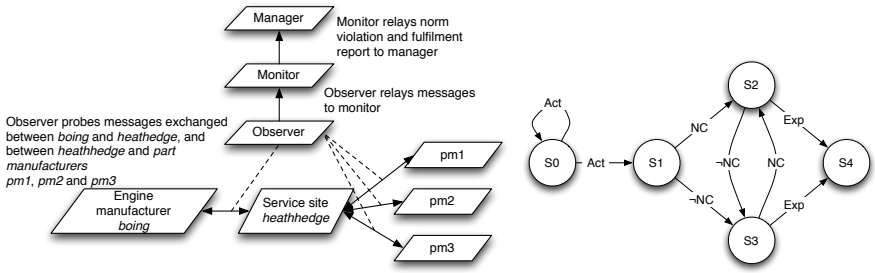
5 Prototype Implementation and Monitoring Scenario

The prototype developed for the CONTRACT framework is driven by events in the environment that are associated with the normative conditions specified in contract clauses. These events drive complying agents to adopt plans to fulfil their obligations. Such a mechanism lends itself very well to implementation through reactive-planning BDI agents, such as PRS [9], and AgentSpeak(L) [10]. In consequence, our CONTRACT prototype was implemented using Jason [11], which is a Java-based AgentSpeak(L) interpreter. More specifically, AgentSpeak(L) is an agent language, as well as an abstract interpreter for the language, and follows the *beliefs, desires and intentions* (BDI) model of practical reasoning [12]. In simple terms, a BDI agent tries to realise the *desires* it *believes* are possible by committing to carrying out certain courses of action through *intentions*, and in AgentSpeak(L), this is simplified in that an agent chooses plans of action that are considered possible by the agent's beliefs, making the notion of desires implicit in the plan representation. The language of AgentSpeak(L) allows the definition of *reactive procedural plans*, so that plans are defined in terms of events to which an agent should react by executing a sequence of steps (*i.e.* a procedure).

5.1 Prototype Overview

In order to demonstrate the utility of deploying CONTRACT's monitoring framework in the aerospace aftermarket, a proof of concept prototype was implemented (see Meneguzzi *et al.*[13] for a more detailed description of the prototype). The prototype simulates five aerospace agents (*boing*, *heathhedge* and three part suppliers *pm1*, *pm2* and *pm3*) trying to fulfil an aerospace aftercare contract. Messages exchanged between these agents are observed by a trusted observer and then relayed to the monitor component from Section 4. The monitor processes these messages together with ATN representations of the norms following the format described in Section 4, and thus determines the activation, fulfilment and violation status of norms.

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 screenshots of this are shown in Figure 3. The screen of Figure 3(a) shows a log of the actions taken by the agents as well as the messages exchanged between them, while that of Figure 3(b) shows the formalised contract (top left), instantiated norms (top right) and the current monitored status of an instantiated norm (mid right).



(a) Information flows in prototype. (b) General ATN format.

Fig. 2. ATNs and information flows.

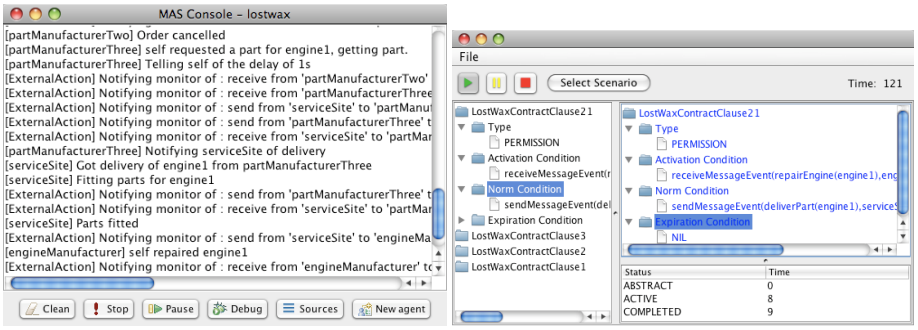
The repair contract that was modelled, represented and monitored, specifies commitments on an engine manufacturer *boing* (that also has an aftercare contract with airline operator *hardjet*) and a service site *heathhedge*. The service site is contractually obliged to repair engines for *boing* within a given time period. Satisfaction of this obligation in turn enables *boing* to fulfil its aftercare contract obligation to *hardjet* to ensure that that no *hardjet* plane is grounded for greater than a certain period of time awaiting a repaired or serviced engine. The aftercare contract also stipulates *hardjet's* provenance restrictions on parts or modules for use in the engines made available by *boing*. These restrictions are in turn encoded in *boing's* contract with *heathhedge*, as permissions and prohibitions on *heathhedge's* ordering of engine parts from named part suppliers, namely, *heathhedge* is permitted to source parts from *pm1* and *pm2*, but prohibited from sourcing parts from *pm3*.

5.2 Monitoring Scenario

In this section we describe a scenario that was tested on our prototype in order to validate the use of observers, ATN representation of norms, and the processing of these ATNs and observations by a monitor component. We also discuss how the results of monitoring (detection and explanation of violations) could be used to suggest logistical and contractual changes in order that the core commitments (on *heathhedge* and *boing* respectively) of repairing engines within a given time, and thus ensuring availability of minimum numbers of operational engines, could be met.

The scenario is described below. It begins with *hardjet* sending a request to *boing* for an engine for one of its planes *P*. It is *P*'s engine itself that is then removed for repair since no other engine is available. *Boing* then orders a repair of the engine from *heathhedge*. Notice that each event in a scenario is recognised as having occurred, based on an observed message exchange. Thus, event 1 is recognised as having occurred based on a message sent by *heathhedge* to *pm1*.

(i) heathhedge orders a part for the engine from *pm1*. *(ii) pm1* informs *heathhedge* that the delivery time for the part is 5 days. *(iii)* The delivery time is unacceptable for *heathhedge* since it does not give the service site enough time to complete repair of the engine within the obliged seven day period. *(iv) heathhedge* then orders the part from



(a) Simulation interface. (b) Monitor interface.

Fig. 3. Prototype graphical user interface.

pm2. *(v) pm2* informs *heathhedge* that the delivery time for the part is 3 days. *(vi)* The delivery time is acceptable for *heathhedge*. *(vii)* However because of delays in transport, the part is received from *pm2* after 5 days. *(viii)* Because of the delay in receipt of the part, the engine is repaired and ready 8 days after receipt of the order for repair from *boing*, so *heathhedge* has violated its obligation to repair the engine within 7 days.

As we have seen, monitoring is achieved by observing messages exchanged between the agents. In this scenario, the messages sent from *heathhedge* to *pm1* indicate that the permission on sourcing parts from *pm1* has not been made use of, whereas the messages sent from *heathhedge* to *pm2* indicate that the permission on ordering parts from *pm2* has been made use of by *heathhedge*. Furthermore, the message informing delivery of the part from *pm2* to *heathhedge* indicates a delay in delivery, thus no message from *heathhedge* to *boing* informing the latter of delivery of the repaired engine is observed within 7 days. This results in transitions to the ATN that are then interpreted by the monitor, which subsequently informs that the norm has been violated and is expired.

As discussed earlier, violation of this obligation can cascade to the aftercare contract between *boing* and *hardjet*, in that failure to have an engine repaired in time by *heathhedge* may mean that *boing* cannot honour its obligation to have a minimum number of operational engines available for *hardjet* aircraft. In previous work, Meneguzzi *et al.*[13] use aggregated monitoring information together with additional ATNs to diagnose this type of cascading violations.

6 Conclusions

In this paper we have described the model used in the implementation of the simulator for the aircraft aftercare case study [5] developed for the CONTRACT project[3]. This simulator was used in the development of the CONTRACT framework and in the validation of the monitoring mechanism. Further development of the case study, and assessment of the CONTRACT technology using it, is ongoing

[3] The CONTRACT platform is available for download at http://ist-contract.sourceforge.net/

The use of CONTRACT concepts and monitoring technology will provide more detailed analysis of where and how run-time simulations (which may be based on real data) diverge from the desired behaviours and states, and thus inform revisions to the simulation configuration (and thus potentially *real-world* contractual, logistical specifications and business models).

Acknowledgments: 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.

References

[1] 6th Framework IST CONTRACT, E.: D2.2 - contract based electronic business systems theoretical framework. *http://www.ist-contract.org* (2008)

[2] 6th Framework IST CONTRACT, E.: D5.2.1 - contract monitoring mechanisms and tools. *http://www.ist-contract.org* (2008)

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

[4] LostWax: Aerogility. http://www.aerogility.com/ (2007)

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

[6] Lomuscio, A., Sergot, M.: Deontic interpreted systems. Studia Logica **75**(1) (2003) 63–92

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

[8] Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. In: Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems. (2009)

[9] Ingrand, F.F., Georgeff, M.P., Rao, A.S.: An architecture for real-time reasoning and system control. IEEE Expert, Knowledge-Based Diagnosis in Process Engineering **7**(6) (1992) 33–44

[10] Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In de Velde, W.V., Perram, J.W., eds.: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Volume 1038 of LNCS. Springer-Verlag (1996) 42–55

[11] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. Wiley (2007)

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

[13] Meneguzzi, F., Modgil, S., Oren, N., Miles, S., Luck, M., Faci, N., Holt, C., Smith, M.: Monitoring and explanation of contract execution: A case study in the aerospace domain. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems. (2009)