

Electronic contracting in aircraft aftercare: A case study

Felipe Meneguzzi¹

`felipe.meneguzzi@kcl.ac.uk`

Simon Miles¹ Michael Luck¹

Camden Holt² Malcolm Smith²

and others

¹Department of Computer Science
King's College London

²Lost Wax



- 1 **Aerospace Aftercare**
- 2 **Background on Contracting**
- 3 **The CONTRACT Architecture**
- 4 **A Contract-Based System for the Aerospace Aftermarket**
- 5 **Concluding Remarks**

An Aerospace Aftercare Use Case

- Simplified version of Lost Wax's use case (previous presentation)
- Aircraft engine manufacturers:
 - ▶ Need to maintain an operational engine pool
 - ▶ Receive hourly rates for engine usage
 - ▶ Need to provide minimum service levels
- Electronic contracts established between manufacturers and airlines

Aftercare contracts

- Complex agreements
- Include provisions for:
 - ▶ Restricting provenance of engines
 - ▶ Specifying a minimum number of spare engines
 - ▶ Maximum idle time for aircraft waiting maintenance
 - ▶ Penalties for violations

Background on Electronic Contracting

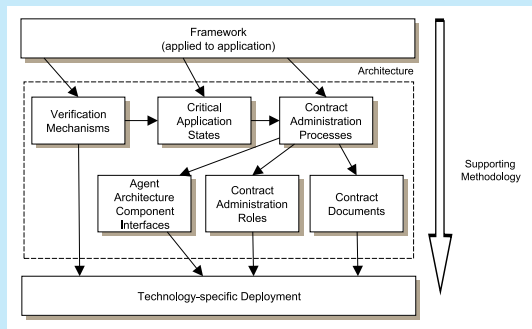
- Systems of self-interested agents:
 - ▶ Inherently unreliable
 - ▶ Require societal control
- We use norms to regulate agent behaviour:
 - ▶ Ensure compliance with societal goals
 - ▶ Usually expressed using deontic concepts
- Norms incorporated into a formal document → *Contract*

The CONTRACT Project

- Explore multiple aspects of contract-based systems
- Aiming at an electronic contracting framework:
 - ▶ Facilitates design, enactment and management of contracts
 - ▶ Includes critical aspects of a contract life cycle
 - ▶ Instantiated here for aerospace aftercare

Structure

- Framework describes:
 - ▶ Contracts
 - ▶ Target agents (contract parties)
- Architecture provides for:
 - ▶ Verification mechanisms
 - ▶ Monitoring of critical states
 - ▶ Administration processes



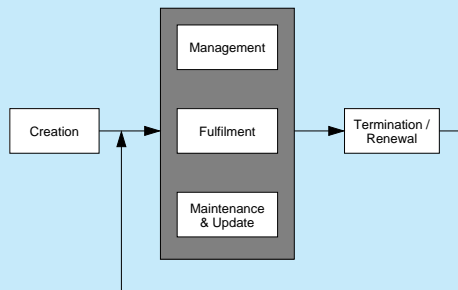
Contracts

- A contract contains clauses:
 - ▶ Obligations
 - ▶ Permissions
 - ▶ Prohibitions
- *Contract parties* bound by clauses
- *Contract roles* are fulfilled by contract parties

Contract Life Cycle

● Five stages:

- ▶ Creation, finding partners, negotiating terms
- ▶ Maintenance and update of a contract in a repository
- ▶ Fulfilment of clauses by participants
- ▶ Management, overseeing fulfillment, taking action
- ▶ Termination or renewal when expired or violated

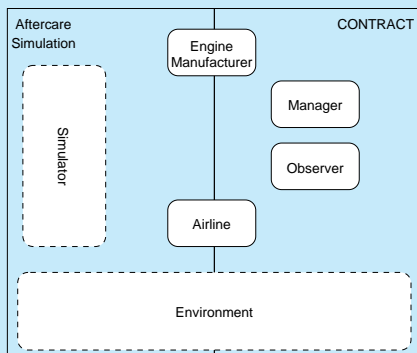


Contract Parties

- Business contract parties:
 - ▶ Agents targeted by the contract
 - ▶ Obligations largely concerned with *business* objectives
- Administrative contract parties:
 - ▶ Required to maintain system integrity:
 - ★ Observer monitors critical state
 - ★ Manager responds to notifications by observer
 - ▶ Obligations concerned with *administering* the system

Agent Roles

- Airline operator
- Engine manufacturer
- Observer
- Manager



Role: Airline Operator

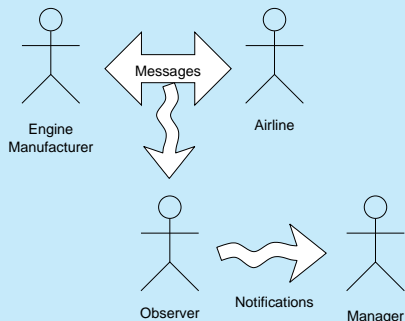
- Goals:
 - ▶ Perform flights according to schedule
 - ▶ Notify manufacturer of unscheduled events
 - ▶ Schedule maintenance ahead of time
- Responsibilities:
 - ▶ Manage a fleet of aircraft
 - ▶ Clock engine cycles as flights are carried out
 - ▶ Inform observer of all communication

Role: Engine Manufacturer

- Goals:
 - ▶ Perform scheduled maintenance before deadlines
 - ▶ Perform unscheduled maintenance ASAP
- Responsibility:
 - ▶ Inform observer of all communication

Role: Observer

- Monitors activities of contract parties
- Detects whether or not violations take place
- In our system, intercepts communication between parties
- Notifies manager of violations



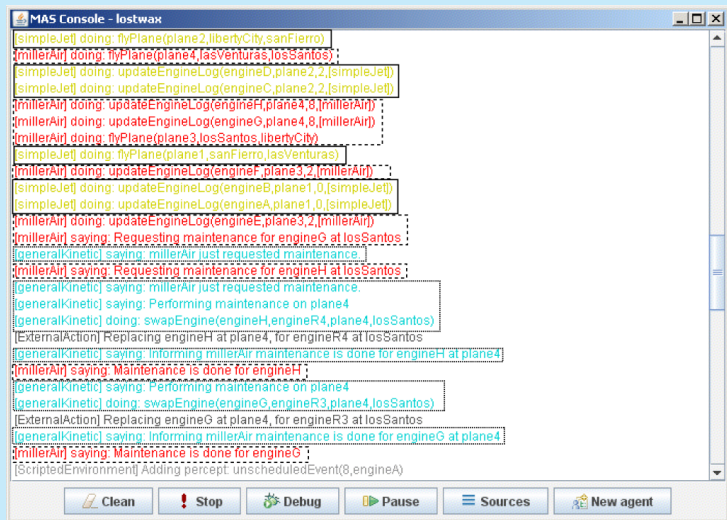
Role: Manager

- Receives violation notifications from Observer
- Takes action to remedy them
- In our system, informs human operator of violation

AgentSpeak(L) and Jason

- AgentSpeak(L) is a *procedural* agent language
- Based on the BDI model
- Designer specifies plans in a library:
 - ▶ Plans encode procedures
 - ▶ Plans are characterised by trigger and context conditions
 - ▶ Goals are implicit in the plans
- Lends itself well to state-based monitoring mechanism
- Prototype implementation in the Java-based *Jason*

Screenshot



```
MAS Console - lostwax
[SimpleJet] doing: flyPlane(plane2,libertyCity,sanFierro)
[MillerAir] doing: flyPlane(plane4,lasVenturas,losSantos)
[SimpleJet] doing: updateEngineLog(engineD,plane2,2,[SimpleJet])
[SimpleJet] doing: updateEngineLog(engineC,plane2,2,[SimpleJet])
[MillerAir] doing: updateEngineLog(engineF,plane4,8,[MillerAir])
[MillerAir] doing: updateEngineLog(engineG,plane4,8,[MillerAir])
[MillerAir] doing: flyPlane(plane3,losSantos,libertyCity)
[SimpleJet] doing: flyPlane(plane1,sanFierro,lasVenturas)
[MillerAir] doing: updateEngineLog(engineF,plane3,2,[MillerAir])
[SimpleJet] doing: updateEngineLog(engineB,plane1,0,[SimpleJet])
[SimpleJet] doing: updateEngineLog(engineA,plane1,0,[SimpleJet])
[MillerAir] doing: updateEngineLog(engineE,plane3,2,[MillerAir])
[MillerAir] saying: Requesting maintenance for engineG at losSantos
[GeneralKinetic] saying: millerAir just requested maintenance
[MillerAir] saying: Requesting maintenance for engineH at losSantos
[GeneralKinetic] saying: millerAir just requested maintenance.
[GeneralKinetic] saying: Performing maintenance on plane4
[GeneralKinetic] doing: swapEngine(engineH,engineR4,plane4,losSantos)
[ExternalAction] Replacing engineH at plane4, for engineR4 at losSantos
[GeneralKinetic] saying: Informing millerAir maintenance is done for engineH at plane4
[MillerAir] saying: Maintenance is done for engineH
[GeneralKinetic] saying: Performing maintenance on plane4
[GeneralKinetic] doing: swapEngine(engineG,engineR3,plane4,losSantos)
[ExternalAction] Replacing engineG at plane4, for engineR3 at losSantos
[GeneralKinetic] saying: Informing millerAir maintenance is done for engineG at plane4
[MillerAir] saying: Maintenance is done for engineG
[ScriptedEnvironment] Adding percept: unscheduledEvent(8,engineA)
```

Clean Stop Debug Pause Sources New agent

lost wax

Summary

- Shown an instantiated system based on the CONTRACT framework
- Examples of concrete Observer, Manager and Contract Parties

Conclusions

- Provide an observation mechanism that can be reused
- Linked a flexible agent model to an explicit contracting mechanism
- Proof of concept for a contracting architecture

Future Work

- Expand the prototype
- Integrate XML contract format
- Incorporate monitoring

Questions?