Studying Aviation Incidents by Agent-based Simulation and Analysis
A Case Study on a Runway Incursion Incident

Tibor Bosse and Nataliya M. Mogles
Vrije Universiteit Amsterdam, Agent Systems Research Group
de Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

Keywords: Aviation, Incidents, Agent-based Simulation, Verification, Interlevel Relations.

Abstract: This paper introduces an agent-based approach to analyse the dynamics of accidents and incidents in aviation. The approach makes use of agent-based simulation on the one hand, and of formal verification of dynamic properties on the other hand. The simulation part enables the analyst to explore various hypothetical scenarios under different circumstances, with an emphasis on error related to human factors. The formal verification part enables the analyst to identify scenarios involving potential hazards, and to relate those hazards (via so-called interlevel relations) to inadequate behaviour on the level of individual agents. The approach is illustrated by means of a case study on a runway incursion incident, and a number of advantages with respect to the current state-of-the-art are discussed.

1 INTRODUCTION

On May 31, 2009, Air France flight 447 disappeared somewhere over the Atlantic Ocean, during a route from Rio de Janeiro to Paris. The crash was the deadliest accident in the history of Air France, killing all 228 people on board. Whilst currently still under investigation, this accident seems to have been the consequence of a rare combination of factors, like inconsistent airspeed sensor readings, the disengagement of the autopilot, and the pilot pulling the nose of the plane back despite stall warnings[1].

This example illustrates an important problem in the analysis of accidents and incidents in aviation: even if detailed flight data from the 'black box' are available, it is usually difficult to come up with a clear analysis, because the causes of incidents cannot be attributed to a point of failure of one individual entity. Instead, most incidents in aviation are caused by a complex interplay of processes at various levels of the socio-technical system.

The complexity of these processes (and their interplay) poses some difficulties to existing approaches for the analysis of aviation indicents. Traditionally, such analyses are done via fault and event trees, graphical representations of Boolean logic relations between success and failure types of events. However, although widely used, there is an increasing awareness that fault and event trees have serious limitations, especially when it comes to analysing dynamic systems with time-dependent interactions (see Everdij (2004) for a more extensive argumentation). More recently, alternative approaches have been developed, such as FRAM (Hollnagel, 2004) and STAMP (Leveson, 2004). While these approaches have proved successful in various case studies, they still have some drawbacks. In particular, FRAM lacks a formal semantics, which makes a computational analysis of complex non-linear processes impossible. STAMP does have a formal basis, but takes an aggregated, organisational perspective (based on system dynamics), which hinders an analysis at the level of individual agents (such as pilots and air traffic controllers), and their underlying mental processes.

As an alternative, the current paper presents an approach for analysis of aviation incidents that takes a multi-agent perspective, and is based on formal methods. The approach is an extension of the approach introduced in the work of Bosse and Mogles (2012), which was in turn inspired by Blom, Bakker, Blanker, Daams, Everdij and Klompstra (2001). Whereas this approach mainly focuses on the analysis of existing accidents (also called *accident analysis*), the current paper also addresses analysis of potential future accidents (called *risk*

[1] http://en.wikipedia.org/wiki/Air_France_Flight_447

Bosse T. and M. Mogles N..
Studying Aviation Incidents by Agent-based Simulation and Analysis - A Case Study on a Runway Incursion Incident.
DOI: 10.5220/0004229800470057
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 47-57
ISBN: 978-989-8565-38-9

analysis). This is done by means of a multi-agent simulation framework that addresses both the behaviour of individual agents (operators, pilots) as well as their mutual communication, and interaction with technical systems. By manipulating various parameters in the model, different scenarios can be explored. Moreover, by means of automated checks of dynamic properties, these scenarios can be assessed with respect to their likelihood of the occurrence of accidents. The approach is illustrated by a case study on a runway incursion incident at a large European airport in 1995.

The remainder of this paper is structured as follows. In Section 2, the modelling approach used in the paper is presented. In Section 3, the scenario used within the case study is described. Section 4 introduces the agent-based model to simulate this (and similar) scenarios, and Section 5 presents the simulation results. Section 6 addresses formal analysis of the model and its results, and Section 7 concludes the paper with a discussion.

2 MODELLING APPROACH

To model the different aspects of aviation operations from an agent perspective, an expressive modelling language is needed. On the one hand, qualitative aspects have to be addressed, such as observations, beliefs, and actions of human operators. On the other hand, quantitative aspects have to be addressed, such as the locations and speeds of aircraft. Another requirement of the chosen modelling language is its suitability to express on the one hand the basic mechanisms of aviation operations (for the purpose of simulation), and on the other hand more global properties of these operations (for the purpose of logical analysis and verification). For example, basic mechanisms of aviation operations involve decision functions for individual agents (e.g., an operator may decide to give runway clearance, and a pilot to abort a take-off procedure in case of an emergency). On the other hand, examples of global properties address the overall safety of an operation, such as "no collisions take place".

The predicate-logical Temporal Trace Language (TTL) introduced in the work of Bosse, Jonker, van der Meij, Sharpanskykh and Treur (2009) fulfils all of these desiderata. It integrates qualitative, logical aspects and quantitative, numerical aspects. This integration allows the modeller to exploit both logical and numerical methods for analysis and simulation. Moreover it can be used to express dynamic properties at different levels of aggregation, which makes it well suited both for simulation and logical analysis.

The TTL language is based on the assumption that dynamics can be described as an evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. These properties are often called *state properties* to distinguish them from dynamic properties that relate different states over time. A specific state is characterised by dividing the set of state properties into those that hold, and those that do not hold in the state. Examples of state properties are 'aircraft A moves with speed S', or 'Air Traffic Controller C provides runway clearance to aircraft A'. Real value assignments to variables are also considered as possible state property descriptions.

To formalise state properties, ontologies are specified in a (many-sorted) first order logical format: an *ontology* is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts (sometimes also called signatures). The examples mentioned above then can be formalised by n-ary predicates (or proposition symbols), such as, moves_with_velocity(A, S) or communicate_from_to(C, A, runway_clearance). Such predicates are called *state ground atoms* (or *atomic state properties*). For a given ontology Ont, the propositional language signature consisting of all ground atoms based on Ont is denoted by APROP(Ont). One step further, the *state properties* based on ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. Thus, an example of a formalised state property is moves_with_velocity(A, S) & communicate_from_to(C, A, runway_clearance). Moreover, a *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont).

To describe dynamic properties of complex processes such as in aviation, explicit reference is made to *time* and to *traces*. A fixed time frame T is assumed which is linearly ordered. Depending on the application, it may be dense (e.g., the real numbers) or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers). Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following (informally stated) dynamic property about the absence of collisions:

For all traces γ,
there is no time point t
on which a collision takes place.

A *trace* γ over an ontology Ont and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in trace γ at time t property p holds' is formalised by state(γ, t) |= p. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. *Dynamic properties* are expressed by temporal statements built using the usual first-order logical connectives (such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$) and quantification ($\forall$ and $\exists$; for example, over traces, time and state properties). For example, the informally stated dynamic property introduced above is formally expressed as follows:

$\forall\gamma$:TRACES $\neg\exists$t:TIME
state(γ, t) |= collision

In addition, language abstractions by introducing new predicates as abbreviations for complex expressions are supported.

To be able to perform (pseudo-)experiments, only part of the expressivity of TTL is needed. To this end, the executable LEADSTO language described by Bosse, Jonker, van der Meij and Treur (2007) has been defined as a sublanguage of TTL, with the specific purpose to develop simulation models in a declarative manner. In LEADSTO, direct temporal dependencies between two state properties in successive states are modelled by *executable dynamic properties.* The LEADSTO format is defined as follows. Let α and β be state properties as defined above. Then, $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g,
then after some delay between e and f
state property β will hold for a certain time interval with duration h.

Based on TTL and LEADSTO, two dedicated pieces of software have recently been developed. First, the LEADSTO Simulation Environment (Bosse, Jonker, van der Meij and Treur, 2007) takes a specification of executable dynamic properties as input, and uses this to generate simulation traces. Second, to automatically analyse the resulting simulation traces, the TTL Checker tool (Bosse et al., 2009) has been developed. This tool takes as input a formula expressed in TTL and a set of traces, and verifies automatically whether the formula holds for the traces.

3 CASE STUDY

Based on the modelling languages TTL and LEADSTO, our model for flight operations will be introduced in Section 4. Although this is a generic model, it will be illustrated (in Section 5) by applying it to a specific case study. To this end, a simple scenario is used in the context of a runway incursion incident that occurred in 1995 (Bosse and Mogles, 2012). This scenario was obtained by performing a semi-structured interview with an available expert, a two years retired pilot of a European civil aviation company.

The runway incursion incident took place during the departure of an Airbus A310 of a civil aviation company from one large airport in Europe. Although the details of the interview and the case study are not shown here (see Bosse and Mogles (2012) for this purpose), a summary of the scenario is provided below. A schematic overview of the situation is provided in Figure 1.

The Airbus was preparing for the departure: the pilot-in-command was sitting on the left and the co-pilot on the right seat in the cockpit and they were ready to start taxiing. They were supposed to taxi to runway 03 in the north-east direction. The Airbus received permission to taxi and started taxiing to its runway. Approximately at the same time, a military Hercules aircraft that was ready for the departure as well received permission to taxi in the north-west direction from its parking gate. The Hercules was supposed to take off from runway 36 that crossed with runway 03 that was designated for the Airbus. Both aircraft were taxiing to their runways. During the taxiing, the Airbus received its flight route from the air traffic controllers. Some time tater, when the Airbus was near the runway designated for taking off, it switched from the taxiing radio frequency to the frequency of the Tower and received permission to line up on the assigned runway. The Hercules was still at the taxiing radio frequency and also received permission to line up, while at the same time the Airbus received permission to take off at the radio frequency of the Tower. However, due to unknown reasons[2], the Hercules pilot interpreted his permission for lining up as permission for taking off and started taking off on runway 36. As a result of this mistake of the pilot of the Hercules, two aircraft were taking off simultaneously on crossing runways, and none of the crews were aware of that. The air traffic controllers in the Tower observed the conflicting situation and communicated a 'STOP' signal to the pilot-in-command of the Airbus, while the Airbus was still on the ground (but at high speed). The pilot had to make a quick decision about the

[2] This misinterpretation might be explained by the fact that the pilot of the Hercules got used to the routine procedure of taxiing from the same military parking place at this airport and perhaps also of taking off from the same runway. And in many past cases, the line up procedure was often immediately followed by taking off, as permissions for lining up and taking off were sometimes given simultaneously.

termination of the take-off as there is a point in this process that one cannot safely do this anymore. After having analysed the situation, the pilot-in-command of the Airbus gave a command to the co-pilot (who controlled the aircraft) to abort the take-off and start braking on the runway. During braking, the crew of the Airbus saw the Hercules flying close in the air above their own aircraft at a distance of about 5 meters. A serious collision was prevented.

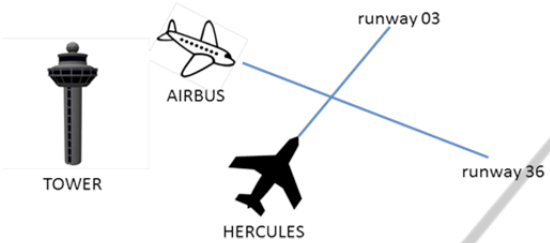

Figure 1: Schematic overview of the case study.

4 AGENT-BASED MODEL

The following subsections describe, respectively, the formal ontology for the case study, the executable dynamic properties (or rules) used to model the scenario, and some simulation results.

4.1 Formal Ontology

As the first step towards the formalisation of the incident identified during the interview, formal domain ontology was developed in TTL. In Table 1 and 2, an overview of the ontology elements is shown, including the relevant sorts and subsorts relations, elements (constants) of sorts, and logical predicates over sorts.

Table 1: Domain ontology: sorts and elements.

SORT	ELEMENTS
AGENT	{tower sub-sorts: PILOT, AIRCRAFT}
PILOT	{airbus_pilot, hercules_pilot}
AIRCRAFT	{hercules, airbus}
ROADWAY	sub-sorts: RUNWAY, TAXIWAY, STARTINGPOINT, CROSSINGPOINT
RUNWAY	{runway_03, runway_36}
TAXIWAY	{taxiway_1, taxiway_2}
STARTINGPOINT	{startingpoint_1, startingpoint_2}
CROSSINGPOINT	{crossing_point(runway_03), crossing_point(runway_36)}
ACTION	{start_taxiing, start_line_up, start_take_off, take_off_from, stop_take_off}
VELOCITY	{low, high, very_high}

As shown in the first three rows of Table 1, the model consists of five active agents that play a role in the scenario (see also Figure 1): Tower, Airbus Aircraft, Hercules Aircraft, Airbus Pilot and Hercules Pilot. In addition, there are elements of the environment that influence the agents' behaviour in the model, such as runways, taxiways and other locations.

Table 2: Domain ontology: logical predicates.

PREDICATE	DESCRIPTION
Communication	
communicate_from_to(A:Agent, B: Agent, C:Action, R:Roadway)	agent A communicates permission for action C on roadway R to agent B
incoming_communication(A: Agent, C:Action, R:Roadway)	agent A receives permission for action C on roadway R
Internal states of agents	
observation(A:Agent, I:Info_El)	agent A observes information element I from the world
belief(A:Agent, I:Info_El)	agent A believes that information element I is true in the world
expectation(A:Agent, C:Action)	agent A has expectation for action C
Actions of agents	
move_from_to(R1: Roadway, R2: Roadway)	action of moving from roadway R1 to roadway R2
performed(A:Agent, C:Action)	agent A performs action C
set_velocity(A:Aircraft, V:Velocity)	aircraft A acquires velocity V
take_off_from(R:Runway)	take-off is performed from runway R
stop_take_off(R:Runway)	take-off from runway R is aborted
Positions of agents	
is_at_position(A:Agent, R:Roadway)	agent A is on roadway R
is_adjacent_to(R1:Roadway, R2:Roadway)	roadway R1 is adjacent to roadway R2
crossing_ways(R1:Roadway, R2:Roadway	roadways R1 and R2 cross
is_half_way(A:Agent,R:Roadway))	agent A is half way on roadway R
in_air(A:Aircraft)	aircraft A is in air
Other information elements used within predicates	
is_available(R:Roadway))	roadway R is available
is_pilot_of(A:Agent, B:Aircraft))	agent A is a pilot of aircraft B
has_role(A:Agent)	an agent has role A
start_taxiing	start taxiing
start_line_up	permission to line up
start_take_off	permission to take off
velocity(A:Aircraft, V:Velocity)	aircraft A has velocity V
has_priority_over(A:Aircraft, B:Aircraft)	aircraft A has priority over aircraft B
not_in_conflict(A1:Agent, A2: Agent)	agent A1 is not in conflict with agent A2
similarity(A1:Action, A2:Action)	action A1 is similar to action A2
velocity(A:Aircraft, V:Velocity)	aircraft A has velocity V
collision(A:Aircraft, B:Aircraft)	aircraft A collides with Aircraft B

4.2 Executable Dynamic Properties

The dynamic relations between the agents are modelled by means of executable dynamic properties (EPs) in LEADSTO. These properties can be subdivided into four different categories, namely properties related to 1) belief formation, 2) communicative action generation, 3) physical action generation, and 4) transfer.

Below some examples of properties in formal LEADSTO notation per category are given (for simplicity, the time parameters have been left out). Note that most properties are applied to all agents. Only some of the properties (e.g., EP2, EP6 and EP16) are specific to a particular agent role (e.g., Tower or Pilot).

4.2.1 Belief Formation

Belief formation properties specify how agents create beliefs about the world on the basis of the observations or communications they receive. For instance, EP1 states that, if an agent observes no other agents at a certain roadway, it concludes that this roadway is available.

Belief formation properties may also represent erroneous behaviour, e.g. related to cognitive biases such as the expectation bias (see: http://www.skybrary.aero/index.php/ATC_Expectation_Bias). For example, EP5 states that, if an agent receives an instruction I1, while it has a strong expectation to receive a similar, but slightly different instruction I2, it will believe that it actually did receive I2. This property can be used to model the fact that the Hercules pilot interpreted his permission for lining up as permission for taking off.

EP1 - Belief Formation on Roadway Availability
observation(A:Agent,
not_at_position(B:Agent, R:Roadway))
→ belief(A:Agent, is_available(R:Roadway))

EP5 - Communication Misinterpretation
incoming_communication(A:Agent, I1:Action, R:Roadway)
& belief(A:Agent, similarity(I1: Action, I2: Action))
& I1 ≠ I2
& expectation(A:Agent, I2:Action)
→ belief(A:Agent, I2:Action, R:Roadway)

4.2.2 Communicative Action Generation

These properties specify how agents derive actions to communicate to other agents, based on the beliefs they possess. For instance, EP2 determines when the Tower agent communicates a permission to start taxiing to the different aircraft, whereas EP16 when the Tower communicates a request to abort take-off.

EP2 - Tower: Taxiing request communication
belief(A:Agent, is_at_position(B:Aircraft, S: Startingpoint))
& belief(A:Agent,
is_adjacent_to(T:Taxiway, S: Startingpoint))
& belief(A:Agent, is_available(T:Taxiway))
& belief(A:Agent, has_role(tower))
→ communicate_from_to(A:Agent, B:Aircraft,
start_taxiing(T:Taxiway))

EP16 - Tower: Take-off Abort Request Communication
belief(tower, is_half_way(A:Aircraft, R1: Runway))
& belief(tower, is_half_way(B:Aircraft, R2: Roadway))
& belief(tower, crossing_ways(R1:Runway, R2:Roadway))
& belief(tower, velocity(B:Aircraft, high))
& not collision(A:Aircraft, B:Aircraft)
& B ≠ A
→ communicate_from_to(tower, B:Aircraft,
stop_take_off, R1:Runway)

4.2.3 Physical Action Generation

In addition to communicative actions, agents may also derive physical actions. An example of this is represented by property EP6, which determines that pilot agents may start taxiing when they believe this is appropriate.

EP6 - Pilot: Taxiing Initiation
belief(P:Pilot, start_taxiing(T:Taxiway)
& is_a _pilot_of(P:Pilot, A:Aircraft)
& belief(P:Pilot, is_available(T:Taxiway))
& is_at_position(A:Aircraft, S:Startingpoint)
& belief(P:Pilot, is_adjacent_to(T:Taxiway, S:Startingpoint))
→ performed(P:Pilot,
move_from_to(S:Startingpoint, T:Taxiway))
& performed(P:Pilot, set_velocity(A:Aircraft, low))

4.2.4 Transfer

Finally, transfer properties represent correct transfer of information. For instance, EP3 states that information that is communicated from agent A to agent B is also received as such by agent B (of by the pilot of agent B, if agent B is an aircraft).

EP3 - Communication Transfer
communicate_from_to(A:Agent, B:Agent, I:Action,
R:Roadway)
& is_pilot_of(P:Pilot, B:Aircraft)
→ incoming_communication(P:Pilot, I:Action, R:Roadway)

Due to space limitations, only a number of the executable properties per category have been listed. However, the full specification (using the notation of the LEADSTO simulation tool) can be found at http://www.cs.vu.nl/~tbosse/aviation.

5 SIMULATION RESULTS

This section describes simulation results of the case study across three different scenarios. The first scenario represents the real situation as described in Section 3, and the other two scenarios simulate two hypothetical situations that would occur when the perceptions and the actions of the agents involved would slightly differ from the real case. These hypothetical situations were created by making small changes in some of the relevant parameters.

In the simulation traces depicted in Figures 2-4, a time line is represented on the horizontal axis and the states that hold in the world are represented on the vertical axis. The dark lines on the right indicate time intervals within which the given states are true. For the sake of transparency, the atoms that represent *observations* and *beliefs* of the agents are not depicted in the traces.

5.1 Scenario 1: Interference of Tower

The simulation trace of scenario 1 is shown in Figure 2. This scenario simulates the real events of the case study. It represents the situation that the pilot of the Hercules aircraft misinterprets the information that is communicated to him by controllers in the Tower because of an incorrect expectation (see atom expectation(hercules_pilot

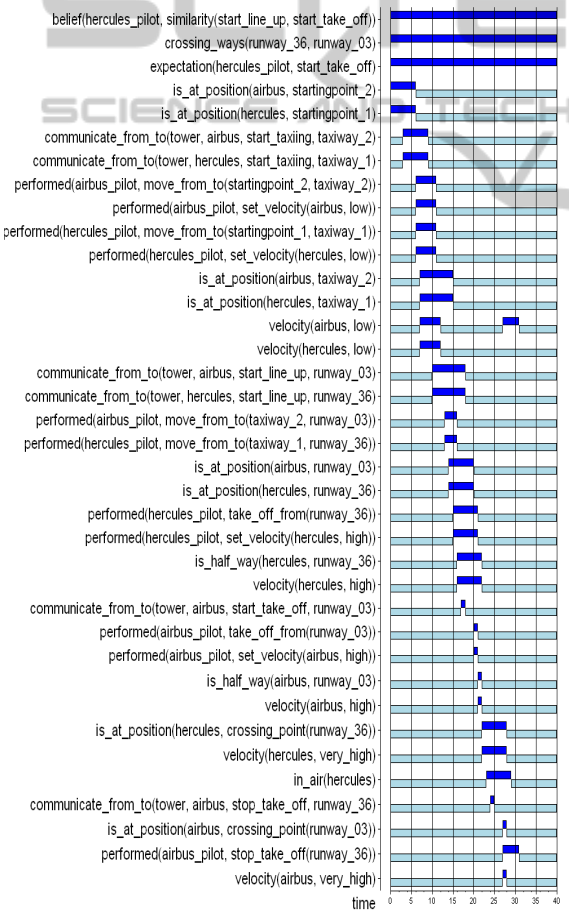

Figure 2: Simulation results of Scenario 1 - Interference of Tower prevents severe collision.

start_take_off) at the top of the trace that is true during the whole simulation), and consequently initiates take-off without take-off clearance (see atom performed(hercules_pilot, take_off_from(runway_36)) that is true from time point 15-21).

There is no atom that states that take-off clearance from the Tower is communicated to the Hercules. At the same time, the clearance for take-off is given to the Airbus aircraft that almost simultaneously initiates take-off from the crossing runway at time point 20; see atom performed(airbus_pilot, take_off_from(runway_03)). Luckily, the Tower observes the conflict situation (this atom is not depicted in the trace) and communicates a "STOP" signal to the Airbus at time point 24. As a result, the pilot of the Airbus aborts the take-off at time point 27 and a severe collision is prevented by this action. This scenario is an example of a case when a hazardous situation created by the wrong decision and action of one agent can be corrected by appropriate intervention of other agents.

5.2 Scenario 2: Nominal Behaviour

The simulation trace of scenario 2 is shown in Figure. 3. This trace represents an ideal scenario where all agents behave properly. In the initial settings of this hypothetical scenario the pilot of the Hercules has no erroneous expectation about the take-off clearance as in scenario 1. As a result, he performs line-up correctly and does not initiate any take-off, as shown in Fig. 3. After both aircraft have performed line-up on their runways at time point 14, permission to take off is communicated only to the Airbus (see atom communicate_from_to(tower, airbus, start_take_off, runway_03))). Hence, in this scenario all agents behave according to the nominal prescriptions of the agent system. Consequently, no collision or hazardous situation occurs.

5.3 Scenario 3: Collision

The simulation of scenario 3 is shown in Figure 4. This scenario represents a situation when the pilot of the Hercules aircraft has erroneous expectations about the take-off clearance and initiates take-off while he should not (like in scenario 1). However, in this case the controllers in the
Tower observe the conflict situation rather late, and therefore they do not have the time to interfere. As a result, both aircraft collide; see atom collision(hercules, airbus) at the end of the trace.

In this scenario the time parameters of the rule that generates the action to take off have been

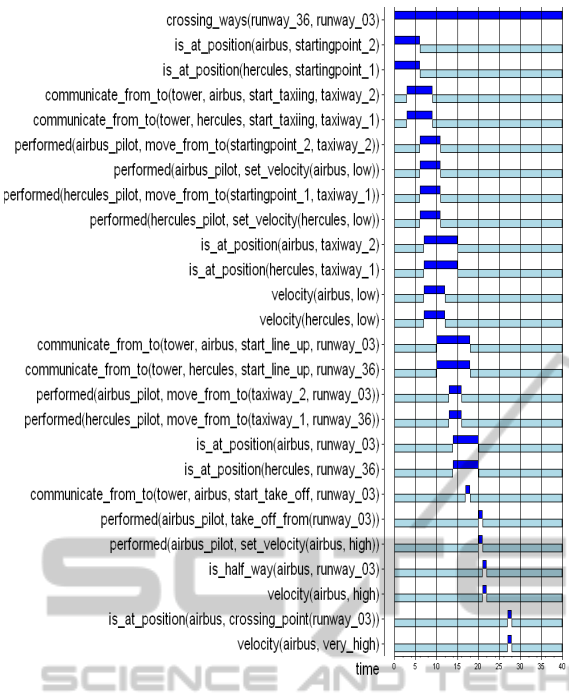

Figure 3: Simulation results of Scenario 2 - Hercules pilot does not make interpretation error.

modified in such a way that this action is performed more quickly. This has important consequences for the opportunity of the Tower to interfere and prevent the collision. As can be seen in Fig. 4, the short duration of the take-off procedure leads to severe consequences as both aircraft perform take-off almost simultaneously on crossing runways.

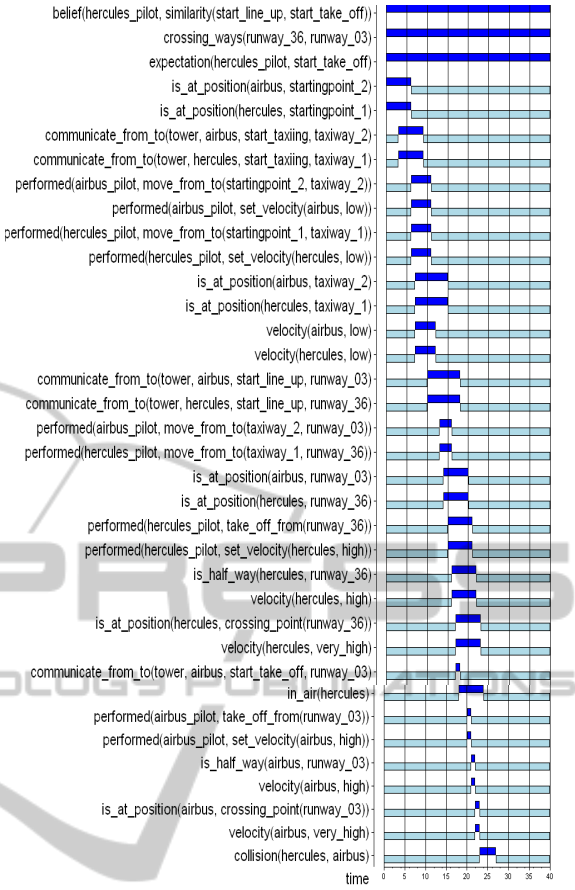

Figure 4: Simulation results of Scenario 3 - Interpretation error by Hercules results in severe collision.

6 FORMAL ANALYSIS

This section addresses formal analysis of the simulated traces. Section 6.1 addresses specification of (global) dynamic properties, Section 6.2 address specification of interlevel relations between dynamic properties at different aggregation levels, and Section 6.3 discusses some results of verification of properties against traces.

6.1 Global Dynamic Properties

Various dynamic properties for the aviation domain have been formalised in TTL, a number of which are introduced below. All of these properties are related in some way to the occurrence of collisions. More specifically, Section 6.1.1 addresses properties that relate to the fact that 'there are never two simultaneous take-offs at crossing runways'. Section 6.1.2 addresses properties that relate to the fact that 'IF any of such simultaneous take-offs occur, THEN they will be corrected on time because one of the aircraft aborts its take-off'. It is easy to see that either one of these cases is sufficient to guarantee that no runway incursions will occur (assuming for simplicity that simultaneous take-offs are the only ways in which runway incursions can possibly occur). All properties in Section 6.1.1 are presented both in semi-formal and in formal (TTL) notation; to enhance readability, the properties in Section 6.1.2 are presented only in semi-formal notation.

Note that the properties presented below address processes at different aggregation levels, thereby distinguishing global properties about the entire scenario (indicated by GP), intermediate properties about input and output states of individual agents (indicated by IP), and local properties about mental processes of agents or about information/communication transfer between agents (indicated by LP).

6.1.1 Absence of Simultaneous Take-Offs

GP1 - No Simultaneous Take-offs at Crossing Runways
There are no trace m, time points t1 and t2, agents a1 and a2, and runway r1 and r2 such that
agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d[3].

¬ [∃m:TRACE ∃t1,t2:TIME ∃a1,a2:AGENT ∃r1,r2:RUNWAY
state(m, t1) |= performed(a1, take_off_from(r1)) &
state(m, t2) |= performed(a2, take_off_from(r2)) &
state(m, t1) |= world_state(crossing_ways(r1, r2)) &
| t1 - t2 | ≤ d]

IP1 - No Simultaneous Permissions to Take off at Crossing Runways
There are no trace m, time points t1 and t2, agents a1 and a2, and runway r1 and r2 such that
the tower gives agent a1 permission for take-off on runway r1 at time t1
the tower gives agent a2 permission for take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d.

¬ [∃m:TRACE ∃t1,t2:TIME ∃a1,a2:AGENT ∃r1,r2:RUNWAY
state(m, t1) |=
communicate_from_to(tower, a1, start_take_off(r1)) &
state(m, t2) |=
communicate_from_to(tower, a2, start_take_off(r2)) &
state(m, t1) |= world_state(crossing_ways(r1, r2)) &
| t1 - t2 | ≤ d]

IP2 - Each Take-off is Preceded by a Corresponding Permission
For all traces m, time points t1, agents a, and runways r
if agent a performs a take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
the tower gave agent a permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= performed(a, take_off_from(r)) ⇒
[∃t2:TIME state(m, t2) |=
communicate_from_to(tower, a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

LP1 - Each Take-off is Preceded by a Corresponding Belief
For all traces m, time points t1, agents a, and runways r
if agent a performs a take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
agent a believed that it had permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= performed(a, take_off_from(r)) ⇒
[∃t2:TIME state(m, t2) |= belief(a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

LP2 - Each Belief about Permissions is Preceded by a Corresponding Communication
For all traces m, time points t1, agents a, and runways r
if agent a believes that it has permission for take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
the tower gave agent a permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= belief(a, start_take_off(r)) ⇒
[∃t2:TIME state(m, t2) |=
communicate_from_to(tower, a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

6.1.2 Correction of Simultaneous Take-Offs

GP2 - All Simultaneous Take-offs are Corrected on Time
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with t1 ≤ t3 ≤ t1+e and t2 ≤ t3 ≤ t2+e on which either agent a1 or agent a2 aborts take-off.

IP3 – For all Simultaneous Take-offs that are Observed an Abort Request is Communicated
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if at time t1the tower observes that agent a1 performs a take-off on runway r1
and at time t2 the tower observes that agent a2 performs a take-off on runway r2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with t1 ≤ t3 ≤ t1+e and t2 ≤ t3 ≤ t2+e on which the tower communicates either to agent a1 or to agent a2 a request to abort take-off.

IP4 - All Received Abort Requests are Followed
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 receives from agent a2 a request to abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which agent a1 indeed aborts take-off from r1.

LP3 - All Simultaneous Take-offs are Observed
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there are two time points t3 and t4 with t1 ≤ t3 ≤ t1+e and t2 ≤ t4 ≤ t2+e on which the tower observes both take-offs.

LP4 - All communicated Abort Requests are Received
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 communicates to agent a2 a request to abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which this request is received from a1 by 2.

LP5 - All Observed Take-offs are Converted into Corresponding Beliefs
For all traces m, time points t1, agents a1, and runways r1,
if at time t1 the tower observes that agent a1 performs a take-off on runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which the tower believes that agent a1 performs a take-off on runway r1.

[3] Many of the properties given in this section contain some parameters d and e. These should be seen as constants, of which the value can be filled in by the modeller.

LP6 – For all Beliefs on Simultaneous Take-offs an Abort Request is Communicated
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if at time t1the tower believes that agent a1 performs a take-off on runway r1
and at time t2 the tower believes that agent a2 performs a take-off on runway r2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with $t1 \leq t3 \leq t1+e$ and $t2 \leq t3 \leq t2+e$ on which the tower communicates either to agent a1 or to agent a2 a request to abort take-off.

LP7 - All Received Requests are Converted into Corresponding Beliefs
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 receives from agent a2 a request to abort take-off from runway r1
then there is a time point t2 with $t1 \leq t2 \leq t1+d$ on which agent a1 believes that it should abort take-off from r1.

LP8 - All Believed Requests are Followed
For all traces m, time points t1, agents a1, and runways r1,
if at time t1agent a1 believes that it should abort take-off from runway r1
then there is a time point t2 with $t1 \leq t2 \leq t1+d$ on which agent a1 indeed aborts take-off from r1.

6.2 Interlevel Relations

A number of logical relationships have been identified between properties at different aggregation levels. An overview of all identified logical relationships relevant for GP1 is depicted as an AND-tree in Figure 5.

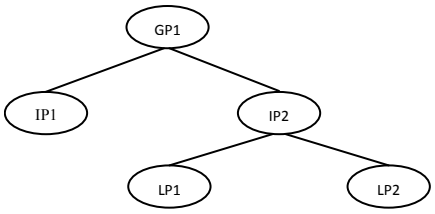

Figure 5: AND-tree of interlevel relations between dynamic properties related to GP1.

The relationships depicted in this figure should be interpreted as semantic entailment relationships. For example, the relationship at the highest level expresses that the implication IP1 & IP2 => GP1 holds, whereas the relationship at the lower level expresses that LP1 & LP2 => IP2 holds. A sketch of the proof for the first implication is as follows (for simplicity reasons abstracting from time constrains):

Suppose that IP1 and IP2 hold. Then, according to IP1, no two permissions to take off at crossing runways will be communicated simultaneously. Moreover, since take-offs are only performed immediately after a corresponding permission has been communicated (as guaranteed by IP2), no simultaneous take-offs are performed at crossing runways. This confirms GP1.

Such logical relationships between dynamic properties can be very useful in the analysis of (both simulated as well as empirical) scenarios, especially when used in combination with the TTL Checker Tool mentioned earlier. For example, for simulation trace 1, checking GP1 pointed out that this property was not satisfied. As a result, by a refutation process (following the tree in Figure 5 top-down) it could be concluded that either IP1 or IP2 failed (or a combination of them). When, after further checking, IP2 was found to be the cause of the failure, the analysis could proceed by focusing on LP1 and LP2. Eventually, LP1 was found satisfied, whereas LP2 failed. Thus, (part of) the source of the incident could be reduced to failure of LP2, i.e., there was an agent (namely the pilot of the Hercules) that believed to have the permission to take off, whilst this was not communicated by the tower. One level deeper, such local properties can even be related to executable properties. For instance, the failure of LP2 can be explained because the Hercules pilot applied property EP5. A full connection of local properties to executable properties is beyond the scope of this paper, but a detailed discussion can be found in Jonker and Treur (2002).

Similar to Figure 5, an AND-tree representing all identified logical relationships relevant for GP2 is shown in Figure 6.

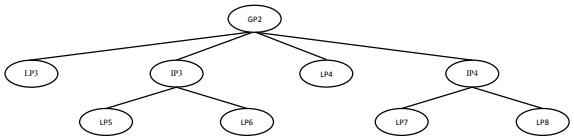

Figure 6: AND-tree of interlevel relations between dynamic properties related to GP2.

6.3 Checking Results

Using the TTL Checker, all dynamic properties introduced in Section 6.1 have been checked against the three simulation traces discussed in Section 5. The results are shown in Table 3 (where 'X' denotes 'satisfied').

As can be seen from the table, scenario 2 is indeed a nominal case in which all expected properties hold. In contrast, in scenario 1, two simultaneous take-offs at crossing runways occur (since GP1 fails), which can eventually be related to an incorrectly derived belief of permission for take-off (failure of LP2).

Table 3: Checking dynamic properties against traces.

property	scenario 1	scenario 2	scenario 3
GP1	-	X	-
IP1	X	X	X
IP2	-	X	-
LP1	X	X	X
LP2	-	X	-
GP2	X	X	-
IP3	X	X	-
IP4	X	X	X
LP3	X	X	+/-
LP4	X	X	X
LP5	X	X	X
LP6	X	X	-
LP7	X	X	X
LP8	X	X	X

However, since the situation is corrected on time (GP2 succeeds), no collision occurs in this scenario. In scenario 3, GP1 also fails, but in addition GP2 fails, which can be related partly to failure of LP3 (the simultaneous take-offs are observed, but too late) and to failure of LP6 (once the tower believes that there are simultaneous take-offs, it is too late to communicate an abort request). As a result, the collision is not prevented.

7 DISCUSSION

For the analysis of accidents and incidents in aviation, roughly two streams can be distinguished in the literature, namely *accident analysis* and *risk analysis*. Whilst the former has the goal to determine the cause of an accident that actually took place, the latter aims to assess the likelihood of the occurrence of future accidents. Hence, although both streams have similar purposes, a main difference is that accident analysis attempts to identify one specific combination of hazardous factors, whereas risk analysis basically explores a whole range of such factors, and the associated risks.

The approach introduced in the current paper in principle addressed both types of analysis. An agent-based method for simulation and analysis of aviation incidents was introduced, and based on a case study on a runway incursion incident it was demonstrated how the approach can be applied both for accident/incident analysis (to examine the causes of the scenario that took place in reality) and for qualitative risk analysis (to determine potential risks for various hypothetical scenarios).

For a more quantitative type of dynamic risk analysis, often Monte Carlo methods are applied; see e.g. the work of Blom et al. (2001); or Stroeve, Blom and Bakker (2004). These methods are very useful for quantitative collision risk estimations, but one of their disadvantages is lack of transparency due to the complex stochastic relations between the elements of the agent-based models that are used. In contrast, the approach presented in this paper is highly transparent; it provides a visible trace of risk related events that can be analysed manually or automatically with the help of special tools. Moreover, the roles of the agents involved in risk creation and reduction (as well as their underlying cognitive processes, like the influence of biased reasoning) are clear from the trace, while in dynamic quantitative risk models used for Monte Carlo simulations this is usually not the case. The complexity of Monte Carlo methods makes it also difficult for the non-specialist to understand the implications of actions and thus makes a public debate of issues a problem. However, a disadvantage of the method proposed in this paper is that it cannot provide a precise risk estimation as is provided by Monte Carlo methods. In follow-up research, we therefore intend to explore the possibilities to combine our approach with elements from Monte Carlo methods.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the SESAR WP-E research network ComplexWorld. It is co-financed by Eurocontrol on behalf of the SESAR Joint Undertaking. The authors are grateful to the retired airline pilot who participated in the interview for his useful input on the case study, and to Jan Treur for a number of fruitful discussions.

REFERENCES

Blom H. A. P., Bakker G. J., Blanker P. J. G., Daams J., Everdij M. H. C., and Klompstra M. B. (2001). Accident risk assessment for advanced air traffic management. In: Donohue, G. L. and Zellweger, A. G. (eds.), Air Transport Systems Engineering, AIAA, pp. 463-480.

Bosse, T., Jonker, C. M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2009). Specification and Verification of Dynamics in Agent Models. *International Journal of Cooperative Information Systems*, vol. 18, 2009, pp. 167-193.

Bosse, T., Jonker, C. M., Meij, L. van der, and Treur, J. (2007). A Language and Environment for Analysis of Dynamics by Simulation. *International Journal of Artificial Intelligence Tools*, volume 16, issue 3, 2007, pp. 435-464.

Bosse, T. and Mogles, N. (2012). Formal Analysis of Aviation Incidents. In: H. Jiang et al. (eds.). *Proceedings of the 25th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, IEA/AIE'12*. Springer Verlag, LNAI volume 7345, 2012, pp. 371-380.

Everdij, M. H. C. (2004). Review of techniques to support the EATMP Safety Assessment Methodology. Report for EEC Safety Methods Survey project, Volume I and II.

Hollnagel, E. (2004). Barriers and accident prevention. Aldershot: Ashgate.

Jonker, C. and Treur, J. (2002). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

Leveson N. (2004). A new accident model for engineering safer systems. Safety Science 42, pp. 237-270.

Stroeve, S. H., Blom, H. A. P., and Bakker, G. J. (2009). Systemic accident risk assessment in air traffic by Monte Carlo simulation. Safety Science 47, pp. 238-449.