Formal Analysis of Empirical Traces
in Incident Management[1]

Mark Hoogendoorn[a], Catholijn M. Jonker[b],
Peter-Paul van Maanen[a,c], and Alexei Sharpanskykh[a]

[a] Department of Artificial Intelligence, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
E-mail: {mhoogen, sharp}@cs.vu.nl
URL: http://www.cs.vu.nl/~{mhoogen, pp, sharp}

[b] Department of Man-Machine Interaction
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail: C.M.Jonker@tudelft.nl
URL: http://www.mmi.tudelft.nl/

[c] Department of Human in Command, TNO Human Factors
P.O. Box 23, 3769 ZG, Soesterberg, The Netherlands
E-mail: peter-paul.vanmaanen@tno.nl

Abstract

Within the field of incident management split second decisions have to be made, usually on the basis of incomplete and partially incorrect information. As a result of these conditions, errors occur in such decision processes. In order to avoid repetition of such errors, historic cases, disaster plans, and training logs need to be thoroughly analysed. This paper presents a formal approach for such an analysis that pays special attention to spatial and temporal aspects, to information exchange, and to organisational structure. The formal nature of the approach enables automation of analysis, which is illustrated by case studies of two disasters.

Keywords: incident management, formal analysis, automated evaluation

---

[1] This publication is partly based on work already published in proceedings of AI-2004 [9] and ISCRAM'04 [1].

1. Introduction

Disasters are events that cause great damage, destruction and human suffering. The question that keeps rising is: "Could we have done anything to prevent this?" The key element is the distinction between incidents and disasters. The definition of an incident used throughout this paper is that these are disturbances in a system that can lead to an uncontrollable chain of events, a disaster, when not acted on properly.

Incidents cannot be avoided. People make mistakes and nature is unpredictable. Incidents typically lead to chaotic situations and complex problems that have to be solved within limited time. Examples of incidents that took on disastrous proportions because of inadequate human intervention are the crash of a Boeing 747 in an urban area in Amsterdam and the Hercules disaster in Eindhoven in the Netherlands.

Depending on the type of incident many people of various organisations have to cooperate; fire brigade, police, ambulance services, alarm centres, hospitals, coast guard, local government, national government, army, to name but the most common organisations involved.

From an ICT perspective, research to improve incident management is mainly focused on the design and development of information systems to support both multi-party communication and decision making, thus addressing the multidisciplinary and distributive character of incident management. Systems like IMI [17] and the GIS-based information exchange system for nuclear emergencies in the Netherlands [19] belong to this category.

Some research projects address the major problem in incident management of adaptively organising multi-party cooperation in a dynamic context, while minimising the number of errors. For example, the COMBINED project [7] tries to tackle the problem using adaptive multi-agent technology.

Finally, some projects focus on simulation tools and techniques to support analysis of crisis response and to support the development of training systems. An example of such an approach for simulations of strategic management is presented in [5].

Not addressed in the research mentioned above, is the ability to analyse the progress of incident management after the fact or while the incident unfolds. Such an analysis is useful for two reasons. First of all, it allows for the evaluation of historic cases whereby errors can be detected and learned from in order to avoid them from occurring again. Secondly, if such an analysis could actually be performed at runtime, this would enable the detection of errors before they result in a disaster. The present approach focuses mainly on the first, since the initial process of formalisation is much easier working from official reports that are published *after* an incident than applying an automated means of such analyses. This is also the proper order for any possible future development of support *during* an incident.

To enable analysis of incident management scenarios a number of formal techniques based on mathematical logic have been developed during the last decade. In particular, modal logic methods (e.g., propositional modal, epistemic, temporal, deontic logics, etc.) gained a special popularity. For example in [16] *Why-Because Analysis* is introduced to reconstruct an incident and to identify its causes. Epistemic logic is used in [12] to model the (diverging) attitudes of incident investigators. In [8] deontic logic is used to identify conflicts that often occur when formal work guidelines are violated.

Although modal logic has clearly defined syntax and semantics, it's expressive power is limited. For example, numbers and arithmetic operations cannot be easily expressed. In comparison, variants of predicate logics allow much more expressivity. For example, Johnson and Holloway [13] show how classical first order predicate logic can be used for the specification of causal relations in the models of incidents.However, predicate logics typically suffer from the undecidability and the lack of analytical tools. Furthermore, standard predicate logic does not provide means to specify and reason about temporal properties of systems.

This paper introduces a formal, logic-based approach for modelling and analysis of incident management scenarios. For the formalisation the *Temporal Trace Language* (TTL) is used, which is a variant of an order-sorted predicate logic [18]. TTL allows representing dynamic (temporal) properties of systems. Particularly for TTL dedicated analysis techniques and tools have been developed.

Furthermore, the paper shows how such automated techniques can be applied for the analysis of particular incident management cases. More specifically the paper shows how domain specific properties can be specified in a sublanguage of TTL, how a trace (a temporal description of chains of events) can be formally specified, and, finally, how automated verification can be performed upon such a trace. Properties that can be analysed include spatial, temporal, information exchange, as well as organisation structure properties. These properties can originate from a variety of sources, such as disaster plans, disaster prevention plans, and laws. Two disasters that occurred in the Netherlands have been analysed in this way, of which the results are presented in this paper.

In Section 2 of this paper an informal analysis of traces of two real-life case studies is presented. In Section 3 an outline is given of the adopted modelling approach. Section 4 shows a formalisation of the trace of the first case study. A description of the formalisation of the second trace is omitted, since it is based on a similar approach. In Section 5 both the empirical traces described in Section 2 are validated by means of a number of essential dynamic properties that have been formalised and automatically verified against the traces. Section 6 is a final discussion.

2.   Case Studies

In order to illustrate the methodology, a number of cases have been studied; the Hercules disaster and the Dakota disaster which both occurred in the Netherlands. These two examples have been chosen due to the different nature of the plans that are applicable in the particular cases. In the Hercules disaster very detailed plans were available in the form of disaster prevention plans, whereas for the Dakota disaster merely high level disaster plans were applicable.

2.1   The Hercules Disaster

The informal analysis of the Hercules disaster presented here is based on [10]. A formalisation of the occurrences during this disaster can be found in Section 4.

On October 16th, 1996 at 6:03 p.m. a Hercules military airplane crashed at the airport of Eindhoven, in the Netherlands. This disaster involves key examples of miscommunications and lack of communication and is therefore a well known example of a non-optimal working disaster prevention organisation. An informal description of the events that took place during the rescue phase is presented below.

The Air-Traffic Control Leader on duty anticipated an accident and activated the so-called crash bell at 6:03 p.m. and hereby initiated the alarm phase. Through intercom he announced that a Hercules plane had landed with an accident and pointed out the location of the plane. The Assistant Air-Traffic Control Leader at the same time contacted the Emergency Centre of the Fire department at the Airbase and reported the situation. The Fire department immediately took action.

The Airbase Fire department must, when reporting to external authority, report which scenario is applicable. There are three different types of scenarios: Scenario 1: A maximum of 2 people involved, Scenario 2: More than 3 and less than or equal to 10 people. Scenario 3: More than 10 people. This all can be found on a checklist and also has consequences for the activities that should take place and the amount of authorities that need to be informed.

The Air-Traffic Control Leader on duty knew that at least 25 people were on board of the plane; this was due to a private source. He called the Emergency Centre of the Fire department at the Airbase around 6:04 p.m. with the order to call 06-11 (the national emergency number at that time).

The Chief of the Airbase Fire department ('On Scene Commander', OSC) asked Air-Traffic Control for the number of people on board of the plane at 6:04 p.m. According to the OSC, the answer was 'nothing known about that'. As a consequence the OSC reported Scenario 2 through the walkie-talkie. The Emergency Centre operator says not to have heard Scenario 2 being declared but does not want to state that this has not been said.

At 6:06 p.m. the Emergency Centre operator calls 06-11 and is connected to the Central Post for Ambulances (CPA). From that point on, the Emergency Centre operator got help from a fire fighter. Together they tried to inform several governmental officials.

At 6:12 p.m. the Regional Emergency Centre of the Fire department (RAC) Eindhoven phoned air-traffic control with the question whether backup was needed, the response was 'negative'. At 6:12 p.m. the Emergency Centre employee and the aforementioned fire fighter decided to follow Scenario 2 of the disaster plan (there were at least 4 people on board of the Hercules because that is the usual crew for this type of plane). At 6:15 p.m. the first civil fire trucks pulled out.

Besides alarming and informing all parties, actions on the scene were taken during that same period. Immediately after the announcement of the Air-Traffic Control Leader the Airbase Fire department went to the scene with a Land Rover Command vehicle (LARO) with the OSC and two Major Airport Crash Tenders (MAC's) each manned with a crew of 3 people. The OSC thought that only the crew was on board of the plane and till the moment passengers had been found he handled accordingly.

At 6:05 p.m. the LARO arrived at the scene and directed the MAC's to the plane. At 6:07 p.m. the MAC's took their position of attack, the plane was on fire across the full length of the body. According to the procedures, the extinguishing was aimed at making the body fire-free. At 6:09 p.m. this was the case and the rest of the fire did not spread anymore. In this situation, the survivors could escape from the plane by themselves.

Around 6:10 p.m. one of the MAC's was empty and the other one only had a quarter of the water-supply left. The OSC decided to have a fire fighter switch the empty one for another one that was still full. After 6 minutes the fire fighter was back with a full MAC.

At 6:19 p.m. there was complete control over the fire at the right wing and engine. Thereafter, at 6:25 p.m. the first civil fire trucks arrived on the scene. After their arrival the OSC contacted the chief of the first fire truck who was told that probably

four people were on board of the plane. After pumping water to the MAC's at 6:38 p.m. they started extinguishing the left engine.

6:33 p.m. was the exact time point when the decision was made to go inside the plane and use a high-pressure hose to extinguish some small fires inside the plane. After that, at 6:37 p.m. the fire fighters were in the plane for the first time and shortly thereafter the first casualty was discovered. Almost at the same time 20 to 30 other casualties were discovered.

2.2 The Dakota Disaster

The informal analysis of the Dakota disaster presented here is based on [11].

The plane crash of a Dakota PH DDA in 1996 in The Netherlands is another examined disaster. The plane had 6 crew members and 26 passengers on board and crashed into the Wadden Sea.

In the Dakota disaster, other factors influenced the emergency rescue process. For instance, some officers were not familiar with emergency procedures and protocols for the disaster. Therefore the wrong procedures and protocols were picked up and consequently inefficient procedures were followed. Another example is that an overload of some of the partners potentially caused some mistakes during the rescue process. However, miscommunications and inappropriate decisions were also involved in the rescue process.

On September 25, 1996 a Dakota PH DDA of the Dutch Dakota Association (DDA) left Texel International Airport Holland. The plane had 6 crewmembers and 26 passengers on board. Shortly after take-off the crew reported engine trouble to Texel International Airport Holland (TIA). Around 4:36 p.m. the crew contacted the Navy airbase The Kooy (MVKK) and stated that it wanted to make an emergency landing on The Kooy. After a short while, The MVKK observed the Dakota disappear from the radar screen.

The MVKK immediately sent a helicopter, initiated a team of rescue helicopters and alarmed the coast guard centre (KWC). At 4:46 p.m. the KWC passed the correct information of the disaster to Regional Alarm Centre northern part of

Noord-Holland (RAC) and asked the RAC to alarm the relevant partners. Unfortunately, the RAC only organised the rescue boats and vessels and did not alarm other parties that should be warned in the disaster.

At 4:55 p.m., the KWC reported the disaster to *Noord Hollands Dagblad* (a Dutch newspaper) and RTL TV station. Consequently, the KWC got many requests for information from the ANP (Dutch press office). The KWC was thus under a lot of pressure.

Through the ANP, the National Centre for Coordination (LCC) got the message that the Dakota had crashed. At 5:03 p.m. the LCC contacted the KWC, the KWC asked the LCC to help by providing a trauma team.

Coincidentally, a big drill for ambulances was ready to start. The Drill leader asked the president of the Dutch health service (GGD) whether the drill should still go on. At 5:05 p.m. the president of the GGD called RAC to inquire if the accident is for real. The RAC responded that neither the KWC nor the harbour office (HK) knew what was going on. The GGD even agreed to start the drill.

At almost the same time, the KWC asked the MVKK to take care of the wounded and told the LCC that the trauma team should be sent to MVKK. At 5:07 p.m. the LCC made an appointment with the Ministry of Public Health, Wellbeing, and Sports (VWS), VWS finally arranged the trauma team.

At 5:17 p.m. the first helicopter with casualties landed at Gemini Hospital (Gemini), the Gemini called the RAC to ask what the purpose of this is. The RAC replied that they only knew a plane had crashed and did not know anything more.

At 5:20 p.m. the RAC asked the KWC to get a trauma team from Gemini to MVKK. Meanwhile the centre for ambulances (CPA) of Amsterdam, the mayors of Den Helder and Wieringen, and the commander of the regional fire department are notified. After a while a crisis centre was finally set up at the Navy. At 6:44 p.m. all bodies were found and transported. Only one person survived the disaster.

3.   Modelling Approach

To formally specify dynamic properties that are essential in incident management processes, an expressive language is needed. To this end the *Temporal Trace Language* (TTL) is used as a tool; cf. [15]. TTL is a variant of an order-sorted predicate logic [18]. The possibility to define arbitrary sorts, functions and relations in TTL provides wide means for the conceptualisation of the incident management domain. Furthermore, whereas standard multi-sorted predicate logic is meant to represent static properties only, TTL is an extension of such language with explicit facilities to represent dynamic (temporal) properties of systems. The temporal expressivity is particularly important for the representation of and reasoning about incidents over time. Moreover, TTL allows expressing quantitative (or numerical) aspects of systems (e.g., precise timing relations, amounts of objects). TTL has the semantics of order-sorted predicate logic that is defined by interpretation of sorts, constants, functions and predicates, and a variable assignment.

All these characteristics of TTL ensure its applicability for specifying a great variety of concepts and relations that can be found in real-life incidents. In the following the most essential elements of the syntax of TTL are introduced.

To specify state properties for system components, ontologies are used. A *state ontology* is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state 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 set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T 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), or any other form, as long as it has a linear ordering. The sort TIME comprises the set of all linearly ordered time points. To perform arithmetical operations on time, a number of

functional symbols are introduced: −, +, /, •:TIME x VALUE→ TIME. Here VALUE is the sort containing all numbers. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of a role r within an incident management process (e.g., mayor, or fire fighter) at time point t is denoted by

$$state(\gamma, t, input(r))$$

analogously

$$state(\gamma, t, output(r))$$
$$state\ (\gamma, t, internal(r))$$

denote the output state, internal state and external world state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r)) |= p denotes that state property p holds in trace γ at time t in the output state of role r. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as ¬, ∧, ∨, ⇒, ∀, ∃. In trace descriptions, notations such as

$$state(\gamma, t, output(r)) |= p$$

are shortened to

$$output(r) | p$$

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or

the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \twoheadrightarrow_{e, f, g, h} \beta$, 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 of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [4]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can be depicted graphically.

Examples of properties expressed in TTL are given in Section 5.

Despite the high expressivity TTL can also be used for performing automated analysis. To this end the dedicated verification algorithm for checking TTL properties on empirical and simulated traces has been developed and implemented [3]. More details on the automated analysis of traces are provided in Section 5.

4.   Formalisation of an Empirical Trace

Informal traces of events, such as the informal trace presented in Section 2 of the Hercules disaster, can be formalised using the formal language TTL as briefly described in Section 3; see also [15]. The translation from an informal trace of events to a formal trace is currently done by hand. However, for the future there are plans to develop a methodology that supports non-expert users (i.e., incident management domain experts) in making this translation. This translation can be supported by offering an off the shelf ontology which is possibly extended with domain specific elements, as well as a tool in which this ontology can easily be used to formalise the occurrences during incidents. Figure 1 shows a screenshot of the tool which has already been developed based for the TTL language (cf. [4]) and could potentially be used for this purpose. Non-experts could use the current informal approaches for describing occurrences during incidents and add this formalisation process as an additional step.
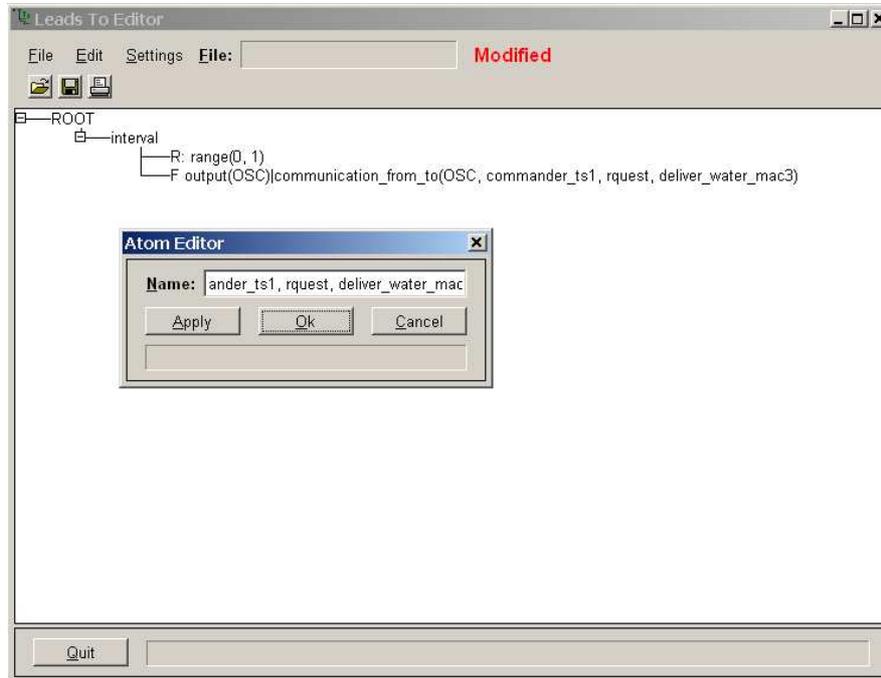
Figure 1: Screenshot of the LEADSTO editor tool.

When formalising a trace four key elements need to be represented within such a trace: (1) temporal aspects; (2) spatial aspects; (3) information exchanges, and (4) organisation structure.

Formalising a trace has several benefits. First of all, specific properties which should hold for a trace can be verified. An example of such a property in the case of an airplane crash is that a fire truck should be at the disaster area within 3 minutes according to the International Civil Aviation Organisation (ICAO). Some properties (such as the above mentioned example) can often easily be checked by hand, but in more complex cases, a mistake may have been caused by a wrong chain of events. These types of causes are usually difficult to determine, and the formalisation can help for this purpose.

Another benefit of the formalisation is in the case where the protocol for the disaster prevention organisation was incorrect. After the protocol has been rewritten it can be formalised by means of executable properties and the scenario in

which the previous protocol failed can be used as an input. Resulting from this, a simulation can be performed which in turn will result in a trace of the functioning of the disaster prevention organisation when using the new protocol. By means of this trace the properties that failed with the previous protocol can again be verified to see whether the new protocol has indeed improved. In case the properties are again not satisfied the cause of this failure can be determined and the protocol can be revised until the desired properties are all satisfied.

An example of a formalisation of a trace is shown in Figure 2. It models the events that occurred during the Hercules disaster. Only a part of the trace is shown for the sake of brevity. On the left side of the figure the atoms are shown that are present in the trace. All atoms have the format

output('role')|communicated_from_to('src', dst', 'type', 'information')

The 'role' indicates the role that outputs this information, whereas the 'src' and 'dst' model the source and destination role (notice that 'role' = 'src' always holds). A list of all the abbreviations used for the roles is shown in Table 1.

The types of communication are based on speech acts [2]. In the full trace also atoms containing input are present. Behind the atom there is a time line, indicating when the atom is true in the trace.
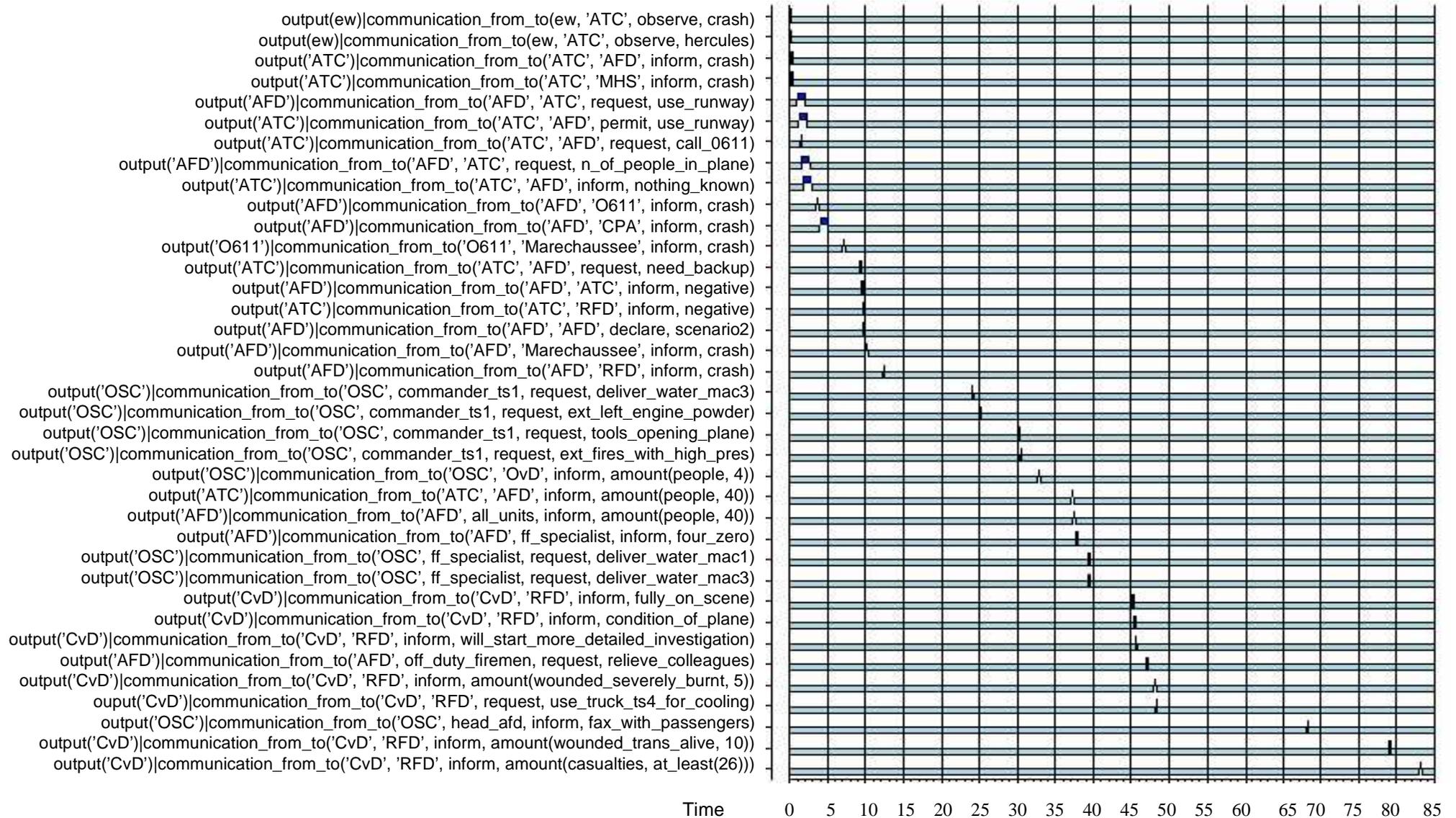
Figure 2: Hercules disaster's formal empirical trace. Vertical axis depicts atoms and horizontal axis depicts them as true (dark) or false (light), per minute.

For example, the first atom

output(ew)|communication_from_to(ew, 'ATC', observe, crash)

states that the external world outputs a crash of a Hercules to air-traffic control, is true during the first minute after the crash, as he observes the crashed plane during that period. This is a means to formalise the observation of events.

Abbreviation	Abbreviates
AFD	Airbase Fire Department
ATC	Air Traffic Control
CPA	Central Post Ambulances
MAC	Major Airport Crash tender
OSC	On Scene Commander
OSO	On Scene Operations
MHS	Medical Health Servies
OvD	Officer On Duty
CvD	Commander on Duty
0611	The national emergency number

Table 1: A list of all abbreviations for the Hercules trace.

A verification of properties that should hold for the disaster prevention organisation of the Hercules disaster case is presented in the next section.

Also for the Dakota incident a formalized trace has been created which of which a fragment is shown in Figure 3. Table 2 provides an overview of the abbreviations used within this trace. Furthermore, the same ontology as presented for the Hercules disaster has been used to express communications within the trace.
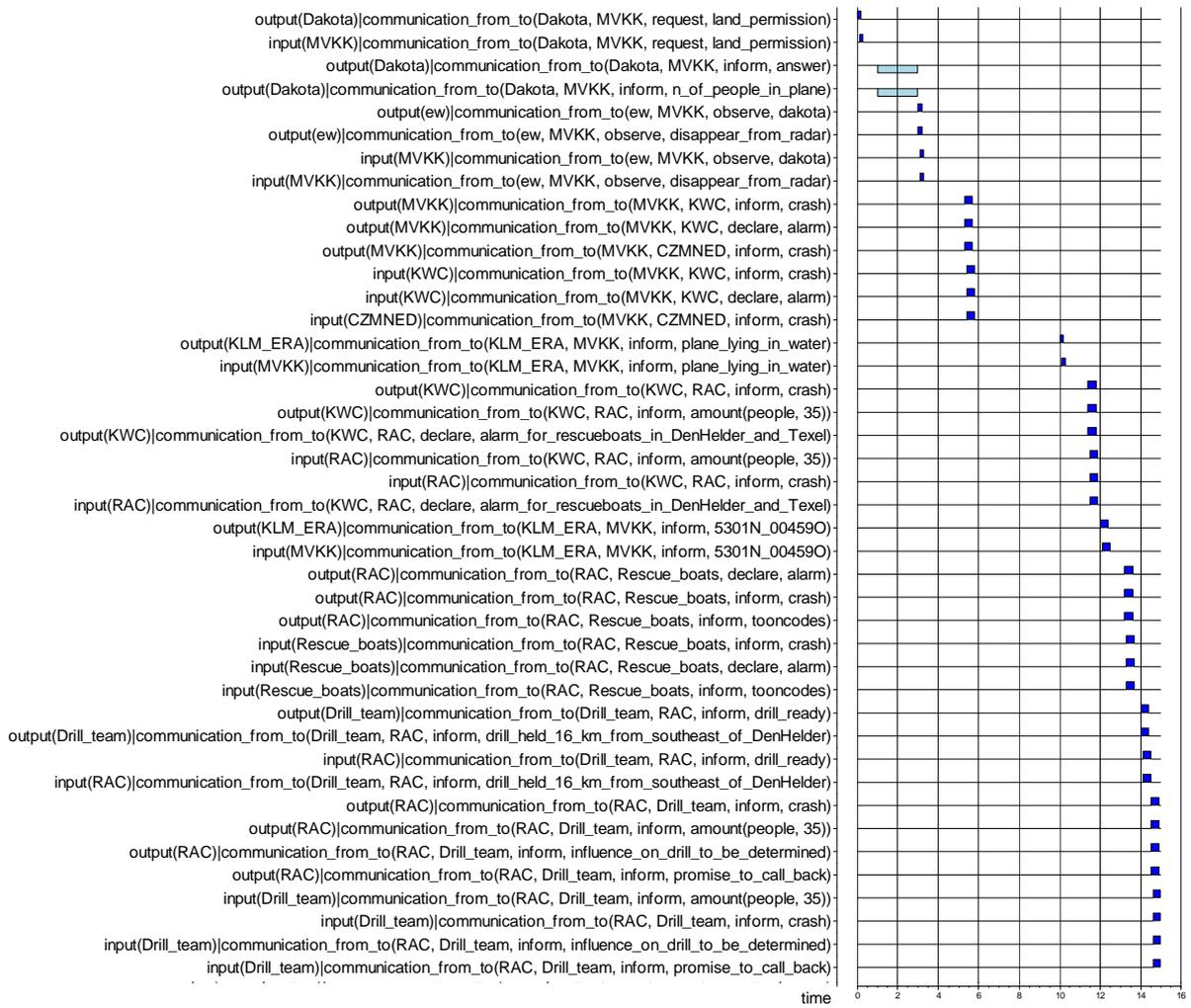
Figure 3: Partial Dakota incident formal empirical trace. Vertical axis depicts atoms and horizontal axis depicts them as true (dark) or false (light), per minute

CPA	Center for ambulances
CZB	Central nursing barracks on navy base
CZMNED	Commander naval forces
GGD	Health service
Gemini	Hospital
HK	Harbor office
KWC	Coast guard center
LCC	National center for coordination
MVKK	Navy airbase
RAC	Regional alarm center

RIAGG	Regional institute for ambulant mental health care
SAR	Search and Rescue team
VWS	Ministry of public health, wellbeing, and sports
V&W	Ministry of traffic and water affairs
DDA	Dutch Dakota Association
TIA	Texel International Airport Holland
Dakota	Dakota PH DDA airplane
ANP	Dutch press office

Table 2: A list of all abbreviations for the Dakota trace.

5.   Validation of a Trace

After having obtained a formalised trace, either by formalisation of an empirical trace or by a simulated trace, it is useful to verify essential dynamic properties of an incident management process for the trace. By means of this verification one can determine in a precise manner what exactly went wrong in the incident management process described by the trace.

5.1   Validation Methodology

Such dynamic properties can be verified using a special software environment *TTL Checker* that has been developed for the purpose of verifying dynamic properties specified in the Temporal Trace Language TTL (cf. [15]) against traces.

The software environment takes a dynamic property and one or more (empirical or simulated) traces as input, and checks whether the dynamic property holds for the trace(s). Using this environment, the formal representation relations presented below have been automatically checked against the trace depicted in Figure 2. Traces are represented by sets of PROLOG facts of the form

    holds(state(m1, t(2)), a, true).

where m1 is the trace name, t(2) time point 2, and a is a state formula in the ontology of the component's input.

The above holds-statement, indicates that state formula a is true in the component's input state at time point 2. The programme for temporal formula checking uses PROLOG rules that reduce the satisfaction of the temporal formula to the

satisfaction of atomic state formulae at certain time points, which can be read from the trace representation. Examples of such reduction rules are:

```
sat(and(F,G)) :- sat(F), sat(G).

sat(not(and(F,G))) :- sat(or(not(F), not(G))).

sat(or(F,G)) :- sat(F).

sat(or(F,G)) :- sat(G).

sat(not(or(F,G))) :- sat(and(not(F), not(G))).
```

5.2   Validation of the Hercules Disaster Trace

Below a number of properties are expressed that are particularly relevant for the Hercules case (Section 3.1). The properties are represented in structured semi-formal format, and a formalisation in TTL is available from the authors. The properties are categorised into four types, namely spatial properties, temporal properties, properties concerning the information exchange, and finally, organisation structure properties. Properties can originate from disaster plans, disaster prevention plans, and, for instance, also from laws. In principle, properties originating from such sources are known in advance and can be seen as a standard set of properties to be checked. These properties can also aid the design of protocols and procedures. For specific incidents , additional properties might need to be identified (e.g., aviation laws in the Hercules disaster). Currently, the formalisation of properties is done by hand, by logical experts. In order to allow non-experts (i.e., incident management domain experts) to formalise such properties, a three step process can be followed whereby first of all informal properties are distinguished following the currently available approaches for non-experts. Thereafter, such properties are translated into so-called semi-formal form that provide more structure to the properties. Finally, the semi-formal properties can be translated into formal properties using (a possible extension of) the ontology mentioned in Section 4. Specification of these formal properties can be done in the dedicated checking tool developed for TTL [3] of which a screenshot is shown in Figure 3. Examples of such a formalisation process can be found in [6]. Following

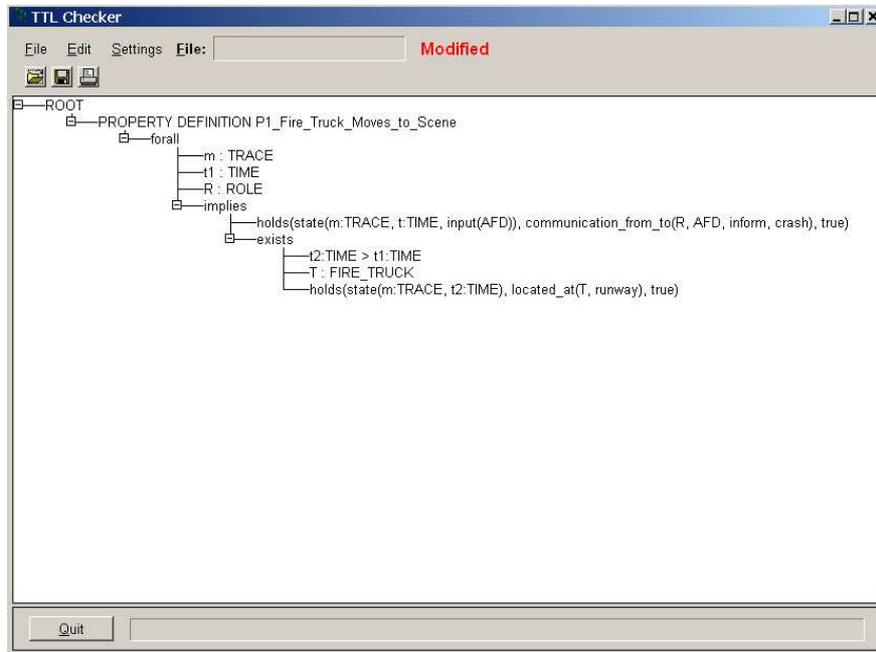this procedure minimises the probability that errors are made in the formalisation process.

Figure 3: Screenshot of the TTL checker tool.

5.2.1 Spatial Properties

Spatial properties include the movement and position of both people as well as material involved in the disaster prevention process. An example of such a property for the Hercules disaster is shown below.

P1: Fire Truck Moves to Scene

At any point in time t1,

if the AFD is informed about a plane crash on the runway

then at a later point in time t2 a fire truck T will be located on the runway.

Formalisation of the property P1:

$\forall$t1, R:ROLE

[state(γ, t1, input('AFD')) |= communication_from_to (R, 'AFD', inform, crash) &

$\Rightarrow$ $\exists$t2 > t1, T:FIRE_TRUCK   state(γ, t2)|= located_at(T, 'runway')]

This property is satisfied for the given trace.

5.2.2 Temporal Properties

Temporal properties include notions such as timely arrival of information and resources. The essence of a property falling into this category is a restriction on the amount of time something is allowed to take. Three properties related to the Hercules disaster have been specified within this category:

P2: Timely Information Delivery

At any point in time t1,

if ATC generates information for AFD about the plane crash,

then at a later point in time t2, t2 ≤ t1+2 AFD will communicate this information to RFD.

Formalisation of the property P2:

∀t1 [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
⇒ ∃t2≤ t1+2   state(γ, t2, output('AFD'))|= communication_from_to('AFD','RFD', inform, crash)]

This property is not satisfied for the given trace.

P3: MAC Timely Arrival at the Disaster Area

At any point in time t1,

if AFD receives information from ATC about the plane crash,

then at a later point in time t2 MAC will join AFD, and at a still later point in time t3 will come to the disaster area in less than 3 minutes upon the plane crash information reception.

Formalisation of the property P3:

∀t1 [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
⇒ ∃t2>t1 state(γ, t2) |= member_of('MAC', 'AFD') &
    ∃t3≤ t1+3 & t3>t2 & state(γ, t3)|=member_of('MAC', 'OSO')]

This property is satisfied for the given trace.

P4: Sufficient Number of Ambulances, Called Immediately

At some time point t1,

if CPA generates information about the number of ambulances, sent to the disaster area to RFD,

then at no later point in time t2 CPA will ask for additional ambulances.

Formalisation of the property P4:

∀t1, x, y

[state(γ, t1, input('RFD')) |= communication_from_to ('CPA', 'RFD', inform, amount(ambulance_sent, x))

⇒ ¬∃t2>t1

state(γ, t2, output('CPA'))|= communication_from_to('CPA', 'CPA', request, amount(ambulance_needed), y)]

This property is not satisfied for the given trace.

The property P4 is meant to verify if CPA sent a sufficient number of ambulances to the scene immediately.

5.2.3 Information Exchange Properties

The information exchange properties express what information should be exchanged between the various players in the organisation, but also the correctness of this information. An example of such a property is specified below for the Hercules disaster.

P5: Information Correctness

At any point in time t1,

if AFD generates a request for ATC about the number of people on the plane,

then at a later point in time t2 ATC will communicate the correct answer to AFD

Formalisation of the property P5:

∀t1 [state(γ, t1, input('ATC')) |= communication_from_to ('AFD','ATC', request, n_of_people_in_plane)

⇒ ∃t2>t1 state(γ, t2, output('ATC'))|= communication_from_to('ATC','AFD', inform, amount(people, 40))]

Automated verification showed that this property is not satisfied in the given trace.

P6: Choice of Protocols

At any points in time t1 and t2, t2≥t1,

if ATC generates information to AFD about the plane crash at t1,

and that the number of passengers is more than 10 at t2,

then at a later point in time t3 AFD declares Scenario 3.

Formalisation of the property P6:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)

& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]

⇒ ∃t3>t2 state(γ, t3, output('AFD'))|= communication_from_to('AFD','AFD', declare, scenario3)]

This property is not satisfied for the given trace.

5.2.4 Organisation Structure Properties

The final property type is that of properties regarding the organisation structure. Such a structure is usually defined based upon the severity of the incident. Properties that can therefore be checked against the trace check for the correctness of the organisation given the current situation. Two example properties are presented below.

P7: Presence Officer On Duty

At any points in time t1 and t2, t2≥t1,

if ATC generates information to AFD about the plane crash at t1,

and that the number of passengers is more than 10 at t2,

then at a later point in time t3 the OVD role is part of the On Scene Operations

Formalisation of the property P7:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)

& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]

⇒ ∃t3>t2 state(γ, t3)|= member_of('OvD', 'OnSceneOperations')]

This property is satisfied for the given trace.

P8: Presence Police Units

At any points in time t1 and t2, t2≥t1,

if ATC generates information to AFD about the plane crash at t1,

and that the number of passengers is more than 10 at t2,

then at a later point in time t3 police units are part of the On Scene Operations

Formalisation of the property P8:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)

& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]

⇒ ∃t3>t2, P:POLICE_UNIT state(γ, t3)|= member_of(P, 'OnSceneOperations')]

This property is satisfied for the given trace.

As can be seen from the results of properties verification, given above, 4 from 8 properties are not satisfied over the trace. By analysing the obtained results one can get insight in which types of errors occurred in the scenario and which points of the disaster plan, disaster prevention plan, and the law were not fulfilled.

5.3   Validation of the Dakota Disaster Trace

Below a number of properties are expressed that in particular are relevant for the Dakota case (Section 3.2), are represented in structured semi-formal format, and finally have been formalised using TTL. Such properties are categorised into the same four types as the properties specified for the Hercules disaster, namely spatial properties, temporal properties, properties concerning the information exchange, and finally, organisation structure properties.

5.3.1 Spatial Properties

For the spatial properties, one example for the Dakota incident has been verified. This concerns the communication of the correct spatial position of the plan crash site.

P9: Correct Coordinate Information

At any point in time t1,

if the KWC receives a request from the RAC about the coordinates of the airplane,

then at a later point in time t2 the KWC will inform the RAC about the required airplane coordinates

Formalisation of the property:

$\forall$t1

[state(γ, t1, input('KWC')) |= communication_from_to ('RAC', 'KWC', request, plane_coordinates)

  $\Rightarrow$ $\exists$t2 > t1   state(γ, t2, output('KWC'))|= communication_from_to ('KWC', 'RAC', inform, 5301N_00459O)

This property is satisfied for the given trace.

5.3.2 Temporal Properties

As an example of a temporal property for the Dakota incident, a property related to work overload has been specified, as expressed in P10.

P10: Reject Information Request Because of Overload

At any point in time t1,

if the KWC receives a request for information from the ANP,

then at a later point in time t2 the KWC will respond with a rejection due to the work overload

Formalisation of the property:

$\forall$t1

[state(γ, t1, input('KWC')) |= communication_from_to ('ANP', 'KWC', request, information)

$\Rightarrow$ ∃t2 > t1 state(γ, t2, output('KWC'))|= communication_from_to ('KWC', 'ANP', inform, too_many_requests)

This property is satisfied for the given trace.

5.3.3 Information Exchange Properties

Another property which has been verified is an information exchange property. This concerns informing the rescue boats.

P11: Inform Rescue Boats

At any point in time t1,

if the KWC communicates to the RAC the correct information on the incident and a request to alarm the rescue boats in Den Helder and Texel,

then at a later point in time t2 the RAC will inform the rescue boats in Den Helder and Texel about the incident

Formalisation of the property:

∀t1
[state(γ, t1, output('KWC')) |= communication_from_to ('KWC', 'RAC', inform, crash) &
communication_from_to ('KWC', 'RAC', inform, amount(people, 35)) & communication_from_to ('KWC',
'RAC', request, alarm_rescue_boats_in_DenHelder_and_Texel))
  $\Rightarrow$ ∃t2 t2 > t1 state(γ, t2, output('RAC'))|= communication_from_to ('RAC', 'Rescue_Boats', inform, crash)

This property is not satisfied for the given trace.

5.3.4 Organisation Structure Properties

Finally, an organisation structure property has been verified. This property concerns the creation of a search and rescue (SAR) team.

P12: SAR team creation

At any point in time t1,
if the MVKK observes the disappearance of the airplane from the radar,
then at a later point in time t2 the MVKK will initiate the creation of a team of SAR-helicopters

Formalisation of the property:

∀t1
[state(γ, t1, input('MVKK')) |= communication_from_to (w, 'MVKK', observe,
airplane_dissappear_from_radar)
  $\Rightarrow$ ∃t2 > t1   state(γ, t2)|= member_of(SAR_TEAM, MVKK)

This property is satisfied for the given trace.

6. Discussion

This paper shows how empirical traces in incident management can be thoroughly analysed in an automated fashion, by means of formal verification techniques. The approach has been illustrated by the analysis of two historic cases, namely the Dakota disaster and the Hercules disaster. Properties involving spatial, temporal, information exchange and organisational structure that are of particular importance in incident management have been specified and checked for these case studies. The result of the analysis shows that certain errors occurred in these historic cases which can be avoided in future.

Research continues to make the approach applicable for run-time analysis of incident management, possibly intervening in the incident management process in case severe errors are detected that could potentially have dramatic consequences. Such interventions could be in the form of a personal agent for each person involved in incident management which provides feedback to its user in the form of advice. A first step to such assistance could constitute of relieving the effort of human or manual detection of relevant events.

To enable such an analysis, usually one can specify dynamic properties at different aggregation levels, from global properties, to more local properties, and establish hierarchical inter-level relations between the properties. If one of the global properties does not hold for the automatically generated empirical trace of the situation, then verification of properties at intermediate levels can follow to identify were the cause of the problem can be found. The verification process can be continued up to the lowest level, consisting of the simplest local properties. More details of this hierarchical diagnostic approach can be found in [14]. The approach put forward in this paper can be extended to include such a hierarchical diagnostic approach as well.

Eventual application of the present approach is intended to augment the decision making capabilities of incident managers, not to replace them, while decreasing the probability of human error.

Acknowledgements

References

1. Abbink, H., Dijk, R. van, Dobos, T., Hoogendoorn, M., Jonker, C.M., Konur, S., Maanen, P.-P. van, Popova, V., Sharpanskykh, A., Tooren, P. van, Treur, J., Valk, J., Xu, L., and Yolum, P., Automated Support for Adaptive Incident Management, In: Walle, B. van de, and Carle, B. (eds.), *Proceedings of the First International Workshop on Information Systems for Crisis Response and Management, ISCRAM'04*, May 2004, pp. 69-74.

2. Austin, J.L., *How to do things with words*. Oxford University Press, 2nd edition, 1976.

3. Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., Specification and Verification of Dynamics in Cognitive Agent Models. In: *Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06*. IEEE Computer Society Press, 2006, pp. 247-254.

4. Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), *Proc. of the Third German Conference on Multi-Agent System Technologies, MATES'05*, Lecture Notes in Artificial Intelligence, vol. 3550. Springer Verlag, 2005, pp. 165-178.

5. Breuer, K., Satish, U., Emergency Management Simulations-An approach to the assessment of decision making processes in complex dynamic environments, In: Jose J. Gonzalez (eds), *From modeling to managing security: A system dynamics approach*, HoyskoleForlaget, 2003, pp. 145-156.

6. Bruin, B. de, Hoogendoorn, M., A Formal Organizational Modeling Approach to Support Change Processes: A Case Study in Dutch

Municipalities, In: Dignum, V., Dignum, F., Matson, E., and Edmonds, B. (eds.), *Proceedings of the Workshop on Agent Organizations: Models, and Simulation*, 2007, pp. 13-25.

7. Burghardt, P., Combined Systems: The combined systems point of view, In: Carlé, B., Walle, B. van der (eds.), *Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04*, Brussels, Belgium, 2004, pp. 51-56.

8. Burns, C. *Analysing Accidents Using Structured and Formal Methods*, PhD Thesis, Department of Computing Science, University of Glasgow, Scotland, UK, 2000.

9. Hoogendoorn, M., Jonker, C.M., Konur, S., Maanen, P.-P. van, Popova, V., Sharpanskykh, A., Treur, J., Xu, L., Yolum, P., Formal Analysis of Empirical Traces in Incident Management, In: Macintosh, A., Ellis, R., and Allen, T. (eds.), Applications and Innovations in Intelligent Systems XII, *Proceedings of AI-2004, the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer Verlag, 2004, pp. 237-250.

10. Inspectie Brandweerzorg en Rampenbestrijding, *Vliegtuigongeval Vliegbasis Eindhoven 15 juli 1996*, SDU Grafische Bedrijf, The Hague, 1996.

11. Inspectie Brandweerzorg en Rampenbestrijding, *Dakota-incident Waddenzee 1996*, SDU Grafische Bedrijf, The Hague, 1997.

12. Johnson, C.W., *A Handbook for the Reporting of Incidents and Accidents*, Springer Verlag, London, UK, 2002.

13. Johnson C.W., Holloway C.M., A survey of causation in mishap logics, *Reliability Engineering and System Safety*, vol. 80(3), 2003, pp. 271-291.

14. Jonker, C.M., Letia, I.A., Treur, J., Diagnosis of the dynamics within an organisation by trace checking of behavioural requirements, In: Wooldridge, M., Weiss, G., and Ciancarini, P. (eds.), Agent-Oriented Software Engineering, *Proc. of Second Int Workshop AOSE'01*, Lecture Notes in Computer Science, vol. 2222, Springer Verlag, 2002, pp. 17-32.

15. Jonker, C.M., Treur, J., 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.

16. Ladkin, P., Loer, K., Formalism helps in describing accidents, In: *Proceedings of Digital Avionics Systems Conference (DASC)*, IEEE 1999. pp. 6.D.4-1--6.D.4-10.

17. Lee, M.D.E. van der, Vugt, M. van, IMI – an information system for effective multidisciplinary incident management, In: Carlé, B., Walle, B. van der (eds.), *Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04*, Brussels, Belgium, 2004.

18. Manzano, M., *Extensions of First Order Logic*, Cambridge University Press, 1996.

19. Ridder, M. de, Twenhöfel, C., The design and implementation of a decision support and information exchange system for nuclear emergency management in the Netherlands, In: Carlé, B., Walle, B. van der (eds.), *Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04*, Brussels, Belgium, 2004, pp. 3-38.