Formal Analysis of Empirical Traces in Incident Management

Mark Hoogendoorn, Catholijn Jonker, Savas Konur, Peter-Paul van Maanen, Viara Popova, Alexei Sharpanskykh, Jan Treur, Lai Xu, Pınar Yolum

Department of Artificial Intelligence, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands.
Email: {mhoogen, jonker, skonur, pp, popova, sharp, treur, xu, pyolum}@few.vu.nl
URL: http://www.few.vu.nl

Abstract

The project CIM, started in 2003, addresses the problem of automated support for incident management. In this paper some intermediate results are shown, especially on automated support of analysis of errors in traces of incident management. For such traces it can be checked automatically which dynamic properties hold and which fail. The potential of the approach is shown in the formal analysis of a given empirical trace. The approach can also be applied in conjunction with simulation experiments.

1. Introduction

Disasters are unforeseen 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. Incidents are disturbances in a system that can lead to an uncontrollable chain of events, a disaster, when not acted on properly.

Incidents will keep occurring. People can make mistakes and nature can be unpredictable. Typically this causes chaotic situations and the resulting problems are very complex and have to be solved within limited time. Examples of incidents that took on disastrous proportions because of inadequate human intervention are the plane crash in Amsterdam and the Hercules disaster in Eindhoven in the Netherlands.

Research in incident management has been fruitful for the past few years. To manage an incident usually many parties have to cooperate. Because of this multidisciplinary and distributive character of incident management, research is mainly focused on the design and development of information systems to support both multi-party communication and decision making. Systems like IMI [13] and the GIS-based information exchange system for nuclear emergencies in the Netherlands [14] belong to this category.

But incident management has a very dynamic and adaptive character too. Organising multi-party cooperation in a dynamic and adaptive manner, while minimising the number of errors, is one of the main challenges. There have been some attempts. For example COMBINED systems [5] tries to tackle the problem in such an adaptive multi-agent manner.

Finally, of course simulations are very important for the analysis of crisis response and the development of training systems. Such as simulations of strategic management [3].

This paper presents the aims and some intermediate results of the project CIM (Cybernetic Incident Management). It is shown how formal modelling techniques from the area of AI enable automated support for the analysis of empirical or simulated traces (temporal description of chains of events). More in particular, it is shown on the one hand how an empirical trace can be formalised to enable automated analysis, and on the other hand how dynamic properties that are essential for an incident management process can be formalised. Moreover, it is discussed how such dynamic properties can be (and actually have been) checked automatically for a given formalised trace. By pinpointing the failing dynamic properties, this automated analysis shows where things have gone wrong in the incident management process. Currently several incident domains are subject of study, such as plane crashes and environmental disasters, with a focus on local and regional as well as national and international processes. Empirical traces are constructed by hand, by means of the analysis of reports and the interviewing of experts. The formalisation of the traces is done by hand as well. However, for the future there are plans to develop a methodology that supports non-expert users in making formalised traces.

In current practice, procedures for dealing with incidents are mostly on paper and have low accessibility. The execution of procedures is completely dependent on people and one wrong assessment or forgotten protocol can worsen the situation. Experience in dealing with incidents is limited because of the low occurrence leading to the repetition of mistakes. Analysis and reconstruction in retrospect is difficult because of the chaos during the incident and the lack of real time tracking of the actions and decisions of the people involved.

The aim of the 4-year project CIM (2003-2007) is to gather knowledge in order to create a constantly adapting system that encompasses both people and supporting software and that has the ability to process and assess information in an adaptive, interactive and intelligent fashion to support human decisions. As a result, the execution of procedures and the assessment of information can be achieved more effectively. Due to the self-learning abilities of the system, experience is united in the system, not only in people but also in software agents and protocols. Because the system automatically records and even facilitates communication between the parties involved, evaluation in retrospect can be performed based on accurate data.

Knowledge in the system is contained in the communication structures and the supporting software in the form of distributed agents that are able to obtain and weigh information dynamically. The maintenance and evolution of the system is achieved by performing simulations and training sessions, both virtually as well as real-life. Measuring the effectiveness of a response and using this as feedback can improve the quality of the protocols and the system itself.

One specific part of the project deals with development of methods to provide automated support for the analysis of what may have gone wrong in specific (simulated or empirical) traces of incident management. Some first results on this theme are presented in subsequent sections. In Section 2 and 3 an informal analysis of traces of real life case studies is presented. In Section 4 a first categorisation of the types of errors is made. In Section 5 an outline is given of the adopted modelling approach. Section 6 shows a formalisation of the trace of one of the case studies. Section 7 discusses a number of essential dynamic properties that have been formalised and automatically checked for the formalised trace from Section 6. Section 8 is a final discussion.

2. The Hercules Disaster

In this section an informal analysis of the Hercules disaster [8] is presented.

On October 16th, 1996 at 6:03 p.m. a Hercules military airplane crashed at the airport of Eindhoven, in the Netherlands. This incident involves many 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 events during the alarm phase are presented, after that the emergency assistance during that period is described.

Alarm phase.
The Air-Traffic Control Leader on duty anticipated an accident and activated the so-called crash bell at 6:03 p.m. Trough the intercom installation 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 this person, the answer was ‘nothing known about that’. Following from this the OSC reported Scenario 2 through the walkie-talkie. The Emergency Centre operator says not to have heard this 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.

Emergency Assistance
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.

3. The Dakota Incident

In this section an informal analysis of the Dakota incident [9] is presented.

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

In the Dakota incident, other factors are involved in the emergency rescue process. For instance, some officers are not familiar with emergency procedures/protocols for the incident. The wrong procedures/protocols are picked up. An inefficient rescue procedures/protocols consequently is followed. Another example is that an overload of some of the partners can potentially cause some mistakes during the rescue process. However, miscommunications and inappropriate decisions are 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 that the Dakota disappears from the radar screen.

The MVKK immediately sent a helicopter, initiated a team of rescue helicopters and alarmed the coast guard centre (KWC). At 16:46 the KWC passed the correct information of the incident 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 incident.

At 16:55, the KWC reported the incident 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 is thus under a lot of pressure.

Through the ANP, the National Centre for Coordination (LCC) got the message that the Dakota had crashed. At 17:03 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 17:05 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 17:07 the LCC made an appointment with the Ministry of Public Health, Wellbeing, and Sports (VWS), VWS finally arranged the trauma team.

At 17:17 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 17:20 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 the arrangements of a crisis centre finally set up at the Navy.

At 18:44 all bodies are found and transported. There is only one survivor of the incident.

4. Categorisation of Error Types

Based on the above informal traces of the Hercules and Dakota incidents, in this section a first attempt is made to categorize the probable causes of the mistakes made during the incident managing phase after the crashes.

Incomplete Information
First of all a property of urgent situations would be that a lot of decisions are made based on incomplete information. There may not be enough time or resources to gather all relevant information to support a decision, and therefore a wrong decision might be made.

For example, in the Hercules case the operator of the Airbase Fire department has no knowledge of the amount of people on board of the plane, while he has to decide on who to call and what kind of backup to request, without this information. An approach that can be used for planning (which is a typical task in incident management) using incomplete information is for example presented in [6].

Contradictory Information
A second property is that a lot of decisions are made based on contradictory information. One might think of urgent situations in which a decision is made, in spite of a lacking sound support for it, which causes mistakes. For example, in the Dakota incident two numbers are mentioned to be the main information telephone number for relatives of the casualties.

Incorrect Information
Similar to the above properties, information can also be incorrect. Incorrect information obviously misleads incident management, and may cause errors. This incorrectness might be caused by, among others, ill communication, accident, or misinterpretation.

For example, in the Hercules disaster the air-traffic control leader knew how many people were on board of the plane, however he replied a request of the on scene commander for the amount of passengers with a denial of the fact that he knew the amount of people on board.

Decision making with incorrect information can be incorporated in reflective agents, an example of the modelling of reflective agents can be found in [2]. In [7] a constraint satisfaction framework for decisions under uncertainty is presented.

Use of Different Protocols
Another property is that in larger scale incidents a lot of parties are involved, and therefore it becomes more probable that different rules or protocols are used in situations where the same should have been used. In these situations parties might expect others to have different behaviours than they have in reality.

For example, in the Hercules case the operator of the Airbase Fire department had in mind that the protocol involved calling 06-11. This was however another protocol in case of a less severe accident. This caused unnecessary delays. In the

Dakota incident, for example the coast guard did not act according to the prescribed policy for disasters in the Wadden Sea, but instead used that of the North Sea, and hence the national level organizations got involved in the regional ones, which caused a lot of delay and misunderstandings.

Exception Handling
If disaster plans do not deal with different scenarios, or parties are not familiar enough with such plans, it is possible that exceptions are not handled well. For instance, in the Dakota incident the commander of the regional fire department is surprised when he hears from the Regional Alarm Centre about the incident and that the region is not involved in its management. Because it was not asked, the commander did not take any further action. If he had, it probably would have been very helpful. In these cases a back-up plan should be available.

Work Overload
Finally, if tasks are not delegated properly to parties, or a party is not aware of the possibility of delegation, work overload most probably occurs, and might be another cause for errors. For example, in the case of the Dakota incident, during the first period after the crash, the coast guard had a lot to do, and therefore did not pay enough attention to initiating or delegating activities ashore. In these kinds of situations the coast guard should be relieved of the tasks that can be performed by others.

Software agents can be a very useful help in supporting people to cope with all the incoming information and making the right decision, see for example [4].

5. 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* is used as a tool; cf. [11]. In this paper for the properties occurring in Section 6 both informal or semi-formal and formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly defined as follows.

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 γ : 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 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 firefighter) at time point t is denoted by

state(γ, t, input(r))

analogously

state(γ, t, output(r))
state (γ, t, internal(r))
state (γ, t, EW)

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 $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. In trace descriptions, notations such as

state(γ, 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 [12]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

6. Formalisation of an Empirical Trace

Informal traces of events, such as the trace presented in Section 3.1 of the Hercules disaster, can be formalised using the formal language TTL as briefly described in Section 4; see also [11]. 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 in making this translation. 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 (like the example just mentioned) 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 the functioning. In case the properties are again not satisfied the cause of this 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 1. 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 picture 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 [1]. 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.

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 example, the first atom

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

which 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.

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

7. 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 what precisely went wrong in the example incident management process described by the trace.

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 (see Section 4 or [11]) against simulation 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 1. 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.

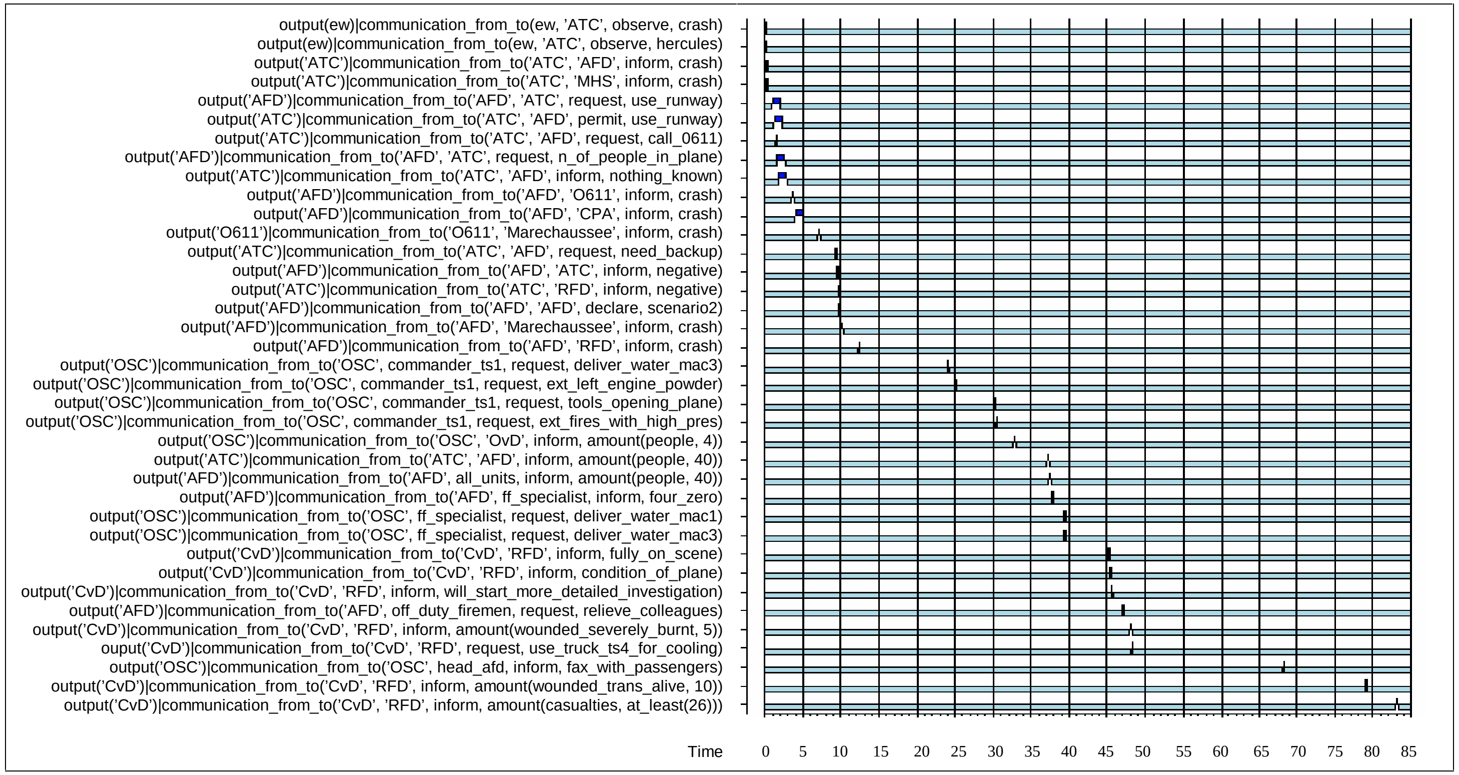

Figure 1: Formalised empirical trace of the Hercules disaster

It is indicated that state formula a is true in the component's input state at time point t2. The programme for temporal formula checking basically uses Prolog rules that reduce the satisfaction of the temporal formula finally 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))).
```

Below a number of properties are expressed that in particular are relevant for the Hercules case (Section 3.1), are represented in structured semi-formal format, and finally have been formalised using TTL.

P1: 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

Formalization of the property P1:

∀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.

P2: 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.

Formalization of the property P2:

∀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.

P3: 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.

Formalization of the property P3:

∀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.

P4: MAC Timely Coming to 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.

Formalization of the property P4:

$\forall$t1 [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
$\Rightarrow$ $\exists$t2>t1 state(γ, t2) |= member_of('MAC', 'AFD') &
$\exists$t3$\leq$ t1+3 & t3>t2 & state(γ, t3)|=member_of('MAC', 'OSO')]

This property is satisfied for the given trace.

P5: 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.

Formalization of the property P5:

$\forall$t1, x,y
[state(γ, t1, input('RFD')) |= communication_from_to ('CPA', 'RFD', inform, amount(ambulance_sent, x))
$\Rightarrow$ $\neg\exists$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 P5 is meant to verify if CPA sent a sufficient number of ambulances to the scene immediately.

As one can see from the results of properties verification, given above, 4 from 5 properties are not satisfied over the trace. By analyzing the obtained results one can get insight in which types of errors occurred in the scenario and which points of the disaster plan were not fulfilled.

8. Discussion

The project CIM started in 2003. In this paper some intermediate results were shown, especially on automated support of analysis of errors in traces of incident management. The potential of the approach was shown in the formal analysis of a given empirical trace. However, the approach can also be applied in conjunction with simulation experiments. If a large number of simulated traces are generated, for example by varying parameters or initial information, for these simulated traces it can be checked automatically which dynamic properties hold or fail for which of the traces.

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, 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. See [10] for more details of this diagnostic approach.

Further potential uses of the checking of dynamic properties is by investigating whether or not certain mistakes would still exist after a modification of the relating protocols. The possibility to check such properties formally, can provide a clue to get to solutions or recommendations in terms of improved protocols.

References

1. Austin, J.L. How to do things with words. Oxford University Press, 2nd edition, 1976
2. Brazier, F.M.T., Treur, J. Compositional modelling of reflective agents. International Journal of Human-Computer Studies, vol. 50, 1999, pp. 407-431
3. Breuer, K., Satish, U. Emergency Management Simulations-An approach to the assessment of decisionmaking processes in complex dynamic environments. In Jose J. Gonzalez (eds), From modeling to managing security: A system dynamics approach, HoyskoleForlaget, 2003, pp. 145-156.
4. Brown, S. M., Santos Jr., E., Banks, S. B., Stytz, M. R. Intelligent interface agents for intelligent environments. In: Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments, 1998, pp. 145-147
5. 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.
6. Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., Williamson, M., An approach to planning with incomplete information. In: Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning, 1992, pp. 115-125
7. Fargier H., Lang J., Martin-Clouraire R., Schiex T. A constraint satisfaction framework for decision under uncertainty. In: Proc. of the 11th Int. Conf. on Uncertainty in Artificial Intelligence, 1995, pp. 167-174
8. Inspectie Brandweerzorg en Rampenbe-strijding, Vliegtuigongeval Vliegbasis Eindhoven 15 juli 1996, SDU Grafische Bedrijf, The Hague, 1996
9. Inspectie Brandweerzorg en Rampenbestrijding, Dakota-incident Waddenzee 1996, SDU Grafische Bedrijf, The Hague, 1997
10. 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
11. 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
12. Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. Cognitive Systems Research Journal, vol. 4, 2003, pp. 191-210.
13. 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.

14. 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.