



A4.D5.2 – 2nd Version of Diagnosis Prototype

G. Spanoudakis, T. Tsigkritis, C. Kloukinas

Document Number	A4.D5.2
Document Title	2 nd Version of Diagnosis Prototype
Version	1.1
Status	Final
Work Package	WP 4.5
Deliverable Type	Prototype
Contractual Date of Delivery	30 June 2008
Actual Date of Delivery	12 August 2008
Responsible Unit	CUL
Contributors	
Keyword List	S&D violations diagnosis prototype
Dissemination level	PU

Change History

Version	Date	Status	Author (Unit)	Description
0.1	15/07/2008	Draft	T. Tsigkritis (CUL)	First draft
0.2	09/08/2008	Draft	G. Spanoudakis (CUL)	Editions
0.3	11/08/2008	Draft	C. Kloukinas (CUL)	Editing
1.0	12/08/2008	Final	T. Tsigkritis (CUL)	Editing for final version
1.1	08/09/2008	Final	T. Tsigkritis (CUL)	Editing for quality check

Executive Summary

This report is part of the deliverable A4.D5.2 .It provides a description of the second version of the S&D violations diagnosis prototype that has been developed in SERENITY as part of the A4 activity. The code that implements the 2nd version of this prototype is also part of A4.D5.2. The report describes the extensions of the first version of the S&D violations diagnosis prototype, which has been implemented in version 2, and other amendments of the original prototype that were implemented in the second version.

In summary, the new version of the diagnosis prototype realises some modifications in the way in which the belief in the genuineness of runtime events is assessed and an application programming interface (API) that enables the retrieval of diagnostic results from external clients of the tool and, in particular, the SERENITY runtime framework. The new version of the diagnosis prototype provides also a representation of the monitoring and diagnosis results according to a representation schema that has been agreed with the A5, A6 and A7 activities.

The main modification in the computation of beliefs in the new version of the diagnosis prototype is that when it computes the belief in the genuineness of a runtime event that is involved in the violation of an S&D monitoring rule, it takes into account not only the number of other runtime events that are consequences of the possible explanations of this event but also the belief in the genuineness of the runtime events that match these consequences. Furthermore, it takes into account beliefs in the genuineness of events that match with any preconditions that the identified explanations have. The report provides the rationale for the amendments in the event genuineness belief estimation process and gives examples that demonstrate the new approach. It also provides a description of the API of the new version of the diagnosis prototype, and instructions for using them.

Table of Contents

1. Introduction	5
2. Overview of the SERENITY diagnosis process.....	6
3. Modifications in the 2 nd Version of Diagnosis	8
3.1. Overview of modifications.....	8
3.2. Modifications in the computation of beliefs in event genuineness	8
3.3. Example of computation of genuineness beliefs	15
3.4. Schema for presentation of diagnosis results	18
3.5. API for diagnosis tool	26
4. Implementation and Usage Guide.....	28
4.1. Required Software	28
4.2. Installation instructions.....	28
4.3. Usage instructions.....	28
4.3.1. The monitor	28
4.3.2. The Monitoring Manager and Diagnosis Module	29
5. Conclusions	36
Appendix A. 1 st Version Algorithms	37
Appendix B. Proofs of Theorems	40
Appendix C. New Schema for Monitoring Results	43
Appendix D. WSDL of the new API of the Monitor	50

1. Introduction

This report is part of the SERENITY A4.D5.2 deliverable and its purpose is to describe the implementation of the 2nd version of the diagnosis prototype of SERENITY (referred to shortly as “diagnosis tool” in the following). In addition to this report, A4.D5.2 includes the source code of the 2nd version of the diagnosis tool.

The second version of the diagnosis tool extends the first version of the prototype by introducing a new approach for assessing beliefs in the genuineness of events which are involved in violations of S&D monitoring rules. The basic characteristic of the new approach is that, whilst assessing the genuineness of events, it takes into consideration not only the number of the consequences of the possible explanations of the relevant events which appear in the event log of the monitor but also the genuineness of these events. The second version of the diagnosis tool also implements an application programming interface for initiating the diagnosis capabilities of the tool and retrieving the results of the diagnostic analysis performed by it.

The rest of this report is structured as follows. In Section 2, we give an overview of the diagnosis process that was introduced originally in [6] in order to enable the reader understand the overall context and the nature of the modifications introduced in the 2nd version of the prototype. In Section 3, we discuss the modifications made in the new version of the diagnosis prototype, introduce an application programming interface (API) giving access to its functionality, and present the amended version of the schema for reporting monitoring results which enables the notification of diagnostic results to clients of the SERENITY diagnosis framework. In Section 4, we provide installation and usage instructions for the new version of the diagnosis prototype. Finally, in Section 5, we provide some concluding remarks. The report has also four appendices. The first appendix (Appendix A) includes the abductive and deductive reasoning algorithms presented in A4.D5.1 deliverable. The second appendix (Appendix B) gives the proof for the functional form of the new basic probability assignment function for the calculation of basic probability measures in the genuineness of events. The third appendix (Appendix C) lists the new XSD schema for the presentation of monitoring results along with diagnostic information. The fourth appendix (Appendix D) lists the WSDL of the new application programming interface of the monitor, which includes operations for initiating the diagnosis capabilities of the tool and retrieving the results of the diagnostic analysis performed by it.

2. Overview of the SERENITY diagnosis process

The overall process of diagnosing the causes of S&D monitoring rule violations has four main stages, as were presented in [6]. As shown in Figure 1, these stages are:

- Explanation generation
- Explanation effect identification
- Explanation plausibility assessment
- Diagnosis generation

In the first of these stages (i.e., explanation generation), the diagnosis process generates all the *possible explanations* for the individual events which have caused an S&D monitoring rule violation (aka “violation observations”). The possible explanations of violation observations are generated from *assumptions* that have been given to the monitor regarding the operation of the system that is being monitored using abductive reasoning.

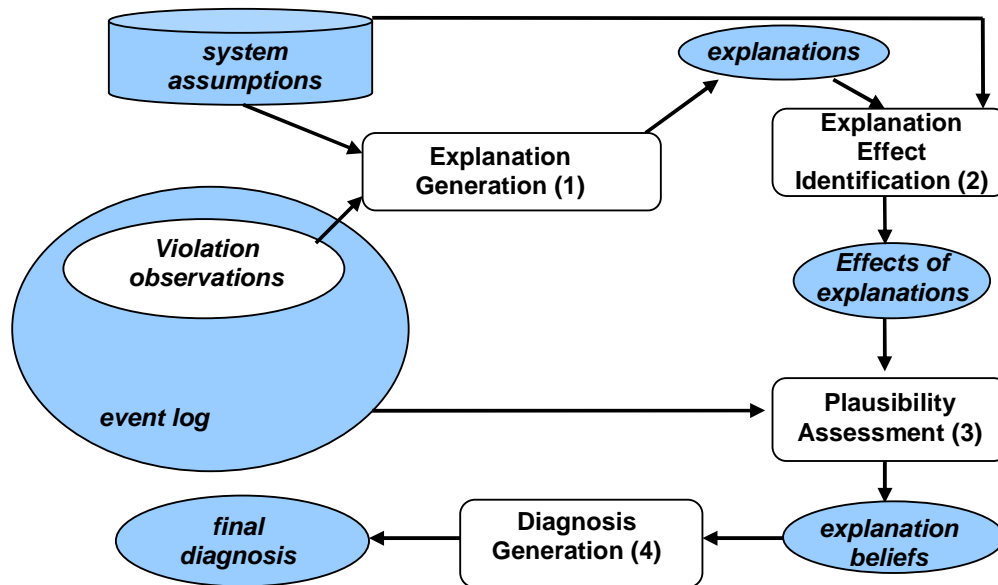


Figure 1 – The SERENITY diagnosis process

After generating explanations for the individual violation observations, the diagnosis process enters its second stage, namely the stage of *explanation effect identification*. This stage is concerned with the identification of all the possible consequences of the explanations of the individual violation observations if these explanations were valid. Whilst the generation of individual explanations from the observation violations are generated by abductive reasoning, the effects of individual explanations are derived by deduction using the assumptions specified in S&D patterns.

Following the identification of the effects of individual explanations, the diagnosis process enters its third stage. At this stage, the process assesses the likelihood of the validity of the individual event explanations. To do so, the expected effects of the individual explanations are checked against the event log of the SERENITY monitoring framework to find if there are events in the log that match the expected effects. Every match that is found between an expected effect and an event in the log casts confirming evidence to the explanation associated with the effect. On the other hand, the absence of a matching event for an effect casts disfavoured evidence to the explanation. Based on the confirming and disconfirming elements of evidence which are identified during this stage, the diagnosis process estimates a belief and a plausibility measure for each individual explanation.

Finally, at the fourth stage of the diagnosis process, i.e., the stage of *diagnosis generation*, the diagnosis framework constructs alternative aggregated explanations for the S&D rule violation from the explanations of the individual violation observations and computes beliefs in the validity of these aggregate explanations. Using these beliefs the framework also identifies the most plausible aggregate explanation for the violation.

3. Modifications in the 2nd Version of Diagnosis

3.1. Overview of modifications

The changes of the diagnosis process that have been introduced in the second version of the diagnosis prototype pertain to the generation and assessment of the validity of explanations. More specifically, in the 2nd version of the prototype, the belief in the genuineness of a violation observation is computed by taking into consideration not only the number of the confirming or disconfirming evidence of its valid explanations, as in the 1st version, but also the genuineness of the gathered evidence itself. Note that the gathered evidence that can confirm or disconfirm the validity of an explanation of a violation observation, and therefore the genuineness of a violation observation, consists of other recorded events that can be found in the event log of the monitor. Thus, the basic characteristic of the 2nd version is that the belief in the genuineness of violation observations depends on the belief in the genuineness of other recorded events that are correlated with the violation observations.

3.2. Modifications in the computation of beliefs in event genuineness

As aforementioned, one of the main amendments that have been made in the 2nd version of the diagnosis prototype is related to the computation of the belief in the genuineness of events. In the first version of the prototype, the computation of the belief in event genuineness was based on the identification of at least one explanation for the event which had additional expected consequences matching with runtime events. This process had two main limitations: (1) it did not take into account the genuineness of the runtime events that matched with the expected consequences of the explanations of the event of interest, and (2) it did not take into account the genuineness of the runtime events that matched with the preconditions (if any) of the explanations of the event of interest. To demonstrate these cases, consider the example of Figure 2. The figure shows an extract of an event log with three events (e_1 , e_2 and e_3) which have occurred in the chronological order indicated by their subscripts. Let us consider that the event which is involved in the violation of an S&D monitoring rule and whose genuineness needs to be assessed in this example is e_3 .

According to the process defined in [6], the computation of the belief in the genuineness of e_3 is based on first identifying the explanation Φ as a possible explanation of the event (Φ is in fact the only possible explanation of e_3 in this example), then identifying all the expected consequences that Φ has in addition to e_3 , and finally counting how many of these consequences match with runtime events in the event log. In this example, Φ has only one more consequence matching with the event e_1 in the log. Thus, according to the belief computation process that was introduced in A4.D5.1, the belief in the validity of Φ would be 1 (i.e., the ratio of the consequences of Φ that match with runtime events) and, therefore, the belief in the genuineness of e_3 would also be 1.

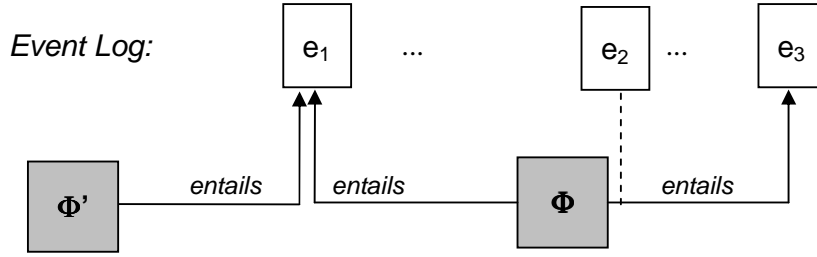


Figure 2 - Events and explanations

The first limitation of the above assessment scheme is that it does not consider the genuineness of the event e_1 . If e_1 is the result of a fault or an attack, then the evidence that it casts for the validity of Φ and, consequently, the genuineness of e_1 should be disregarded. Furthermore, Φ is an explanation of the event e_3 but can logically entail it only if the event e_2 in the event log is also genuine. The abductive reasoning process that the 1st version of the diagnosis prototype uses to generate explanations would have identified the event e_2 as an event that is required for the entailment of e_3 by Φ . However, the computation of the belief in the genuineness of e_3 did not attempt to estimate the belief in the genuineness of e_2 and take it into account.

To overcome the above these limitations, the 2nd version of the diagnosis prototype uses a different function for the computation of the belief in the genuineness of an event. This function also enables the configuration of the computation of event genuineness beliefs by the use of an additional parameter. This parameter is called “diagnosis window” and is used to restrict the part of the event log that can be used during the assessment of the genuineness of a given event. In particular, the diagnosis window is defined as a period of time W and when it is used, only events that have occurred up to W time units prior to the event whose genuineness is assessed can be taken into account in the assessment of this genuineness. The primary role of this parameter is to limit the cost of the computational process required for estimating the event genuineness beliefs.

Given the above considerations, the genuineness of an event has been re-defined as follows:

Definition 1: The genuineness of an event E_i within a diagnosis timewindow $[T_{min}, T_{max}]$ and given a set of explanations E_o that may have already been generated is defined as

$$\text{Genuine}(E_i, [T_{min}, T_{max}], E_o) = \{EXP \neq \emptyset\} \wedge$$

$$\{(\exists E_w. E_w \in \text{EventLog}[T_{min}, T_{max}]) \wedge \text{mgu}(E_i, E_w) \neq \emptyset \wedge (t_i^{LB} \leq t_w \leq t_i^{UB})\} \wedge$$

$$\{\forall (\Phi_{ij}, \Phi_{ij}^c) \in \text{EXP}_i \text{ Valid}(\Phi_{ij}, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\})\}$$

where,

- EXP is the set of the alternative explanations that can be generated for the event E_i , i.e., $EXP_i = \{(\Phi_{i1}, \Phi_{i1}^c), \dots, (\Phi_{iN}, \Phi_{iN}^c)\}$, with the explanations which are already in E_o removed, i.e.: $EXP = EXP_i - E_o$.
- $\text{EventLog}[T_{min}, T_{max}]$ is a subset of the events in the log of the monitor which includes only events which have occurred in the time range $[T_{min}, T_{max}]$

- $mgu(E_i, E_w)$ is the function that returns the most general unifier of two atomic formulas E_i and E_w [2]
- t_i^{LB} and t_i^{UB} are the lower and upper boundary of the time range within which E_i has occurred¹
- $Valid(\Phi_{ij}, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\})$ denotes that the explanation Φ_{ij} is valid within the diagnosis range $[T_{min}, T_{max}]$ and is formally defined as

$$Valid(\Phi_{ij}, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\}) = \bigwedge_{E_u \in Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}])} Genuine(E_u, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\}) \wedge \{ \bigvee_{E_w \in Consequences(E_i, \Phi_{ij}, [T_{min}, T_{max}])} Genuine(E_w, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\}) \}$$

- $Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is the set of recorded events that are preconditions in the deductive path from Φ_{ij} to E_i , and have occurred within the diagnosis range $[T_{min}, T_{max}]$. This set is defined as:

$$Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}]) = \{ E_w \mid E_w \in EventLog[T_{min}, T_{max}] \wedge \exists c, f. (f \in DeductivePath(\Phi_{ij}, E_i)) \wedge (c \in Body(f)) \wedge mgu(c, E_w) \neq \emptyset \}^2$$

- $Consequences(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is the set of the expected consequences of the explanation Φ_{ij} that have occurred within the diagnosis range $[T_{min}, T_{max}]$. This set is defined as:

$$Consequences(E_i, \Phi_{ij}, [T_{min}, T_{max}]) = \{ E_k \mid E_k \in \Phi_{ij}^C \wedge E_k \in EventLog[T_{min}, T_{max}] \}$$

According to the above definition, an event E_i that has occurred or is expected to occur at time t_i is genuine, given a diagnosis range $[T_{min}, T_{max}]$, if

- It exists in the part of the event log of the monitoring framework that falls within the diagnosis range, and
- There is at least one valid explanation Φ_{ij} of it, i.e., an explanation Φ_{ij} for which: (i) all the preconditions in the deductive path from Φ_{ij} to E_i match with genuine runtime events that have occurred before E_i and within the diagnosis range $[T_{min}, T_{max}]$, and (ii) at least one of the expected consequences of Φ_{ij} which fall into the diagnosis range $[T_{min}, T_{max}]$ (if any) matches with another genuine runtime event that has occurred within the same period.

The above definition of genuineness is recursive. It should also be noted that in Definition 1, the time period over which the genuineness of an event is assessed is defined by the absolute time range $[T_{min}, T_{max}]$. This range is determined by the length of the diagnosis window W that is required in a particular monitoring setting and the timestamp of the original event E_i whose genuineness is to be

¹ t_i^{LB} and t_i^{UB} are both equal to the timestamp t_i of E_i if E_i is an event in the log of the monitor.

² The set $Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is determined during the abductive reasoning process that determines the explanation Φ_{ij} (see step 20 in the *Explain* algorithm in Appendix A).

assessed using the formulas: $T_{max} = t_i + W/2$ and $T_{min} = t_i - W/2$. Furthermore, since during the recursive assessment of event genuineness there is a possibility of reaching an event having an explanation which has already been taken into account for another event, Definition 1 excludes from the possible explanations of a given event E_i any explanations which might have been generated earlier in the assessment process (see set E_o). In the example shown in Figure 2, for instance, the assessment of the genuineness of the event e_3 will require the assessment of the genuineness of the event e_1 since the latter event is an expected consequence of the explanation Φ of e_3 . When assessing the genuineness of e_1 , the explanation Φ should not be taken into account despite being an explanation of e_1 , since Φ is the explanation which led to the assessment of e_1 in the whole process. In this case, only the second explanation of e_1 , i.e., Φ' , should be taken into account according to Definition 1. It should, however, be appreciated that for the initial events which are involved in the violations of S&D monitoring rules, the set E_o is empty, hence setting no initial restrictions as to the explanations that may be taken into account in the assessment of event genuineness.

Based on the above definition of event genuineness, the basic probability assignment³ to it is defined as follows:

Definition 2: The basic probability assignment to the genuineness of an event E_i , within a diagnosis range $[T_{min}, T_{max}]$, given a set of explanations E_o that may have already been generated, is defined as:

$$m^{GN}(\text{Genuine}(E_i, [T_{min}, T_{max}], E_o)) = m^{IL} \oplus m^{EX} (\text{InLog}(E_i) \wedge \text{Explainable}(E_i, [T_{min}, T_{max}], E_o))$$

$$m^{GN}(\neg \text{Genuine}(E_i, [T_{min}, T_{max}], E_o)) = m^{IL} \oplus m^{EX} (\neg \text{InLog}(E_i) \vee \neg \text{Explainable}(E_i, [T_{min}, T_{max}], E_o))$$

where

- $m^{IL} \oplus m^{EX}$ denotes the combination of the basic probability assignments m^{IL}_i and m^{EX}_i according to the DS theory [3]
- $\text{Explainable}(E_i, [T_{min}, T_{max}], E_o)$ is a proposition, which denotes whether E_i is explainable and is defined as:

$$\text{Explainable}(E_i, [T_{min}, T_{max}], E_o) = \bigvee_{(\Phi_{ij}, \Phi_{ij}^c) \in \text{EXP}} \text{Valid}(\Phi_{ij}, [T_{min}, T_{max}], E_o \cup \{\Phi_{ij}\})$$

where

- $\text{EXP} = \text{EXP}_i - E_o$ if $\text{EXP}_i - E_o \neq \emptyset$
- $\text{EXP} = \{E_{NULL}\}$ if $\text{EXP}_i - E_o = \emptyset$

According to Definition 2, the basic probability assignment m^{GN} in the genuineness of an event E_i is defined as a combination of two other distinct basic probability assignments, namely the assignment m^{IL}_i which computes the basic probability of the existence or potential existence of the event E_i in the event log of the monitoring framework, and the assignment m^{EX} which computes the basic

³ Basic probability assignments have a special meaning in the context of the Dempster Shafer theory of evidence (*DS theory*) which underpins the diagnosis framework of SERENITY. An overview of this theory and its axiomatic foundation has been provided in [6] and is not repeated in this report. A full description of DS theory is also available in [4].

probability of the existence of a valid explanation for E_i (i.e., an explanation with genuine consequences and preconditions).

It should be noted that the definition of m^{GN} allows the assignment of a basic probability of genuineness to events that have no explanation since the proposition $Explainable(E_i, [T_{min}, T_{max}])$ is defined as a disjunction over the validity of all possible explanations that can be generated by the abductive reasoning process in [6] (i.e., the explanations which are members of the set EXP) and a special explanation called *null explanation* that is denoted by E_{NULL} in the above definition. This is because an event E_i that has no explanations of its own should still be able to cast some evidence for another event E_j as either a precondition or a consequence of one of the explanations of E_j . Thus, the assignment of a zero basic probability to the explainability of E_i in such cases would result in zero basic probability for its genuineness and, could, therefore, reduce or even make equal to zero the basic probability in the genuineness of E_j (the latter effect would happen if E_i is a precondition of the only explanation that E_j may have). Note also that during the assessment process an event may end up having no explanations because its own explanations may have been already considered as explanations of previous events in the reasoning path. The probability that is assigned to events with no explanations is discussed below following the introduction of the basic probability m^{EX}_i . Also the formula that results from the combination of the basic probability assignments m^{IL} and m^{EX} is established by Theorem 1 below, following the formal definitions of these assignments which are given next.

Definition 3: The basic probability assignment m^{EX} in the existence of a valid explanation for an event E_i is defined as:

$$\begin{aligned}
m^{EX}(Explainable(E_i, [T_{min}, T_{max}], E_o)) &= 1 && \text{if } E_i = P_{NULL} \\
m^{EX}(Explainable(E_i, [T_{min}, T_{max}], E_o)) &= \alpha_1 && \text{if } E_i = C_{NULL} \\
m^{EX}(Explainable(E_i, [T_{min}, T_{max}], E_o)) &= \alpha_2 && \text{if } E_i \neq P_{NULL}, E_i \neq C_{NULL}, \text{ and} \\
&&& EXP = \{E_{NULL}\}
\end{aligned}$$

$$\begin{aligned}
m^{EX}(Explainable(E_i, [T_{min}, T_{max}], E_o)) &= \\
\sum_{I \subseteq EXP \text{ and } I \neq \emptyset} (-1)^{|I|+1} &\left\{ \prod_{E_u \in Conds(E_i, I)} m^{GN}(Genuine(E_u, [T_{min}, T_{max}], E_o \cup I)) \times \right. \\
&\left. \left\{ \sum_{S \subseteq Conseq(E_i, I) \text{ and } S \neq \emptyset} (-1)^{|S|+1} \left\{ \prod_{E_w \in S} m^{GN}(Genuine(E_w, [T_{min}, T_{max}], E_o \cup I)) \right\} \right\} \right\}
\end{aligned}$$

where

- EXP is the explanation set of E_i that has been defined in Definition 2
- $Conds(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is a set that includes the preconditions in the deductive path from an explanation Φ_{ij} to the event E_i or a special element P_{NULL} , called *NULL precondition* which denotes the absence of any precondition in the deductive path. $Conds(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is defined as:
$$\begin{aligned}
Conds(E_i, \Phi_{ij}, [T_{min}, T_{max}]) &= \{P_{NULL}\} && \text{if } Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}]) = \emptyset \\
Conds(E_i, \Phi_{ij}, [T_{min}, T_{max}]) &= Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}]) && \text{if } Conditions(E_i, \Phi_{ij}, [T_{min}, T_{max}]) \neq \emptyset
\end{aligned}$$
- $Conseq(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is the set of the recorded events which are consequences of the explanation Φ_{ij} of E_i and have occurred within the diagnosis range $[T_{min}, T_{max}]$. If no such events exist, then $Conseq(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ contains a single special event C_{NULL} , called *null*

consequence, which denotes the absence of any consequence of Φ_{ij} in the relevant time period. The set $Cons(E_i, \Phi_{ij}, [T_{min}, T_{max}])$ is defined as:

$$\begin{aligned}
Cons(E_i, \Phi_{ij}, [T_{min}, T_{max}]) &= \{C_{NULL}\} && \text{if } ExCons(E_i, \Phi_{ij}, [T_{min}, T_{max}]) = \emptyset \\
Cons(E_i, \Phi_{ij}, [T_{min}, T_{max}]) &= ExCons(E_i, \Phi_{ij}, [T_{min}, T_{max}]) && \text{if } ExCons(E_i, \Phi_{ij}, [T_{min}, T_{max}]) \neq \emptyset
\end{aligned}$$

The basic probability assignment m^{EX}_i is used to calculate the basic probability of the existence of at least one explanation with genuine preconditions and genuine consequences for the event E_i . The definition of m^{EX}_i is recursive as it is based on the basic probability assignment functions m^{GN}_u of all the events E_u which are preconditions of the different explanations of E_i or consequences of these explanations. It should be noted, however, that the definition of m^{EX} relaxes the logical definition of explanation validity. In particular, whilst the logical definition of event genuineness requires the event to have at least one valid explanation for being genuine, m^{EX} assigns a small probability to the explainability of events even if they have no explanation. This is because, as we discussed earlier, an event E_i with no explanations may be a precondition of the only explanation of another event E_j and if this is the case, the assignment of a zero basic probability to its explainability would also reduce to zero the basic probability in the genuineness of E_j . The probability that is assigned to events with no explanations by m^{EX} is determined by the parameter α_2 . The value of this parameter should be set very close to zero, in order to provide a close approximation of the logical definition of explainability (see Definition 1) in cases where an event does not have any explanation.

It should also be noted that m^{EX} assigns pre-determined basic probability measures to null preconditions and consequences. These measures are 1 and α_1 , respectively. The former assignment reflects the fact when a NULL precondition denotes that an explanation has no preconditions and therefore its validity should not be affected by the criterion of precondition genuineness. The assignment of a basic probability measure that is equal to α_1 to a NULL consequence reflects a different consideration. More specifically, whilst the reasoning principle underpinning the diagnosis framework of SERENITY favours explanations which are confirmed by the fact that they have genuine consequences other than the events that they have been generated for, it would be unfair to disregard entirely explanations which have no other such consequences. Cases of such explanations are more likely to arise given that in the new version of the diagnosis framework, the diagnosis range is restricted and, therefore, it may be possible to end up with explanations with no further consequences, merely because these consequences correspond to events outside the given diagnosis range. Thus, it is important to assign some basic probability measure in the validity of such explanations but at the same time keep this measure low to reflect the absence of consequences. The framework introduces the parameter α_1 to define the probability measure that should be used in such cases and leaves the choice of the value of this parameter to the user of the framework with an expectation that this value will be a small number close to zero to ensure that explanations with no consequences cannot affect significantly the beliefs in the genuineness of events. However, α_1 must be greater than α_2 to ensure that explanations with no consequences affect more the beliefs in the genuineness of events than null explanations do.

The basic probability assignment m^{LL} that is used in the definition of m^{GN} is itself defined as follows:

Definition 4: m^{LL} is the basic probability assignment in the existence or potential existence of an event E_i in the event log of the monitoring framework, defined as:

$$m^{LL}(\text{InLog}(E_i)) = 1 \quad \text{if} \quad \begin{array}{l} E_i \in \text{EventLog or} \\ E_i \text{ is a } \textit{ HoldsAt}, P_{NULL} \text{ or } C_{NULL} \text{ predicate} \end{array}$$

$$\begin{aligned}
m^{IL}(\neg \text{InLog}(E_i)) &= 1 && \text{if } E_i \notin \text{EventLog}, \\
&&& E_i \text{ is not a } \textit{HoldsAt}, P_{\text{NULL}} \text{ or } C_{\text{NULL}} \\
&&& \text{predicate, and} \\
&&& \textit{lastTimestamp}(\textit{captor}(E_i)) \geq t_i^{UB} \\
m^{IL}(\text{InLog}(E_i) \vee \neg \text{InLog}(E_i)) &= 1 && \text{if } E_i \notin \text{EventLog}, \\
&&& E_i \text{ is not a } \textit{HoldsAt}, P_{\text{NULL}} \text{ or } C_{\text{NULL}} \\
&&& \text{predicate, and} \\
&&& \textit{lastTimestamp}(\textit{captor}(E_i)) < t_i^{UB}
\end{aligned}$$

The function m^{IL} assigns a basic probability of 1 to $\text{InLog}(E_i)$ for any event E_i that has been recorded in the log or denotes a null precondition (P_{NULL}) or a null consequence (C_{NULL}). This basic probability reflects the complete certainty about the existence of the event in the log (note that, by virtue of the axiomatic foundation of the DS theory, the basic probability assigned to $\neg \text{InLog}(E_i)$ is in this case zero). m^{IL} assigns also a basic probability measure of 1 to $\text{InLog}(C_{\text{NULL}})$ and $\text{InLog}(P_{\text{NULL}})$ as these are special events denoting the absence of explanation consequences and preconditions for particular explanations and when they are established by the reasoning processes of the diagnosis framework they can be assumed to be always true⁴.

m^{IL} assigns a basic probability of 1 to $\neg \text{InLog}(E_i)$ for any event E_i (other than P_{NULL} or C_{NULL}) that has not been recorded in the log when m^{IL} is invoked and, at the time of the invocation, the timestamp of the latest event which has been generated by the captor that is expected to produce E_i and recorded in the log (i.e., $\textit{lastTimestamp}(\textit{captor}(E_i))$) is greater than or equal to the maximum time boundary until when E_i should have occurred (i.e., t_i^{UB}). If, however, at the time of the invocation of m^{IL} , the event in question (E_i) is not in the log and the timestamp of the latest event that has been received from the captor that is expected to produce E_i is less than t_i^{UB} , m^{IL} assigns a basic probability of 1 to $\text{InLog}(E_i) \vee \neg \text{InLog}(E_i)$ and, as a consequence of the axiomatic foundation of the DS theory, a basic probability of 0 to each of $\text{InLog}(E_i)$ and $\neg \text{InLog}(E_i)$. The assignment of basic probabilities, in the latter case represents the complete uncertainty about the occurrence or not of E_i within the time period that it is expected to occur, as we have discussed in [7][8].

Given the logical definitions of m^{GN}_i , m^{IL}_i , and m^{EX}_i the basic probability measures of m^{GN}_i are computed according to the following theorem:

Theorem 1: The basic probability assignments to the genuineness of events are calculated by the formulas:

$$\begin{aligned}
m^{GN}(\text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) &= m^{IL}(\text{InLog}(E_i)) \times m^{EX}(\text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o)) \\
m^{GN}(\neg \text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) &= \\
& m^{IL}(\neg \text{InLog}(E_i)) + m^{EX}(\neg \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o)) - \\
& (m^{IL}(\neg \text{InLog}(E_i)) \times m^{EX}(\neg \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o)))
\end{aligned}$$

The functional form of m^{GN}_i is derived from the definitions of m^{GN} , m^{IL} , and m^{EX} and the axiomatic foundation of the DS theory, as we prove in Appendix B.

⁴ The need to provide an m^{IL} basic probability measure of such events may arise due to the recursive definition of m^{EX} in terms of m^{GN} and m^{IL} .

3.3. Example of computation of genuineness beliefs

In the following, we give an example of the computation of the belief in event genuineness based on the basic probability assignments m^{GN} , m^{IL} , and m^{EX} . Our example is based on the diagnosis example that was used in [6] in order to enable a direct comparison with the belief measures generated by the first version of the diagnosis prototype for the example. This example is related to an air traffic management system (ATMS) that uses different radars to monitor the trajectories of airplanes in different air spaces. The operations of ATMS may be monitored at runtime to ensure the integrity of its components and the information generated by them. One of the rules monitored for ATMS checks the integrity of the information that is provided by the different radars that cover an airspace. More specifically, the rule states that if one of the radars used by the ATMS first sends a signal indicating that an airplane is in a particular airspace, every other radar that covers the same space should also send a signal indicating the presence of the plane in the given space within a certain time period of time after the receipt of the initial signal. This rule is specified in the monitoring language of the SERENITY monitoring framework as follows:

Rule 1:

```
Happens(signal(_r1, _a1, _s1), t1, R(t1, t1))  $\wedge$  HoldsAt(covers(_r1, _s1), t1)  $\wedge$ 
( $\exists$  _r2) (_r2  $\neq$  _r1)  $\wedge$  HoldsAt(covers(_r2, _s1), t1)  $\Rightarrow$ 
Happens(signal(_r2, _a1, _s1), t2, R(t1+1, t1+5))
```

In the same example, let us also assume that:

— The following assumptions are valid for ATMS.

- (A0) **Initiates**(_e1, _f), t1, R(t1, t1)) \wedge $\neg \exists$ _e2, t2: **Terminates**(_e2, _f), t2, R(t1, t2)) \Rightarrow **HoldsAt**(_f, t2)
- (A1) **Initially**(_f) \wedge $\neg \exists$ _e2, t2: **Terminates**(_e2, _f), t1, R(0, t2)) \Rightarrow **HoldsAt**(_f, t2)
- (A2) **Happens**(inspace(_a, _s), t1, R(t1, t1)) \wedge **HoldsAt**(covers(_r, _s), t1) \Rightarrow **Happens**(signal(_r, _a, _s), t2, R(t1, t1+5))
- (A3) **Happens**(inspace(_a, _s), t1, R(t1, t1)) \Rightarrow **Happens**(permissionRequest(_a, _s), t2, R(t1-20, t1-1))
- (A4) **Happens**(inspace(_a, _s), t1, R(t1, t1)) \Rightarrow **Initiates**(inspace(_a, _s), inairspace(_a, _s), t1)
- (A5) **Initiates**(inspace(_a, _s), inairspace(_a, _s), t1) \wedge **HoldsAt**(landing_airspace_for(_s, _airportX), t1) \Rightarrow **Happens**(landingRequest(_a, _airportX), t2, R(t1-10, t1))
- (A6) **Happens**(changeOfLandingApproach(_airportX, _s), t1, R(t1, t1)) \Rightarrow **Initiates**(changeOfLandingApproach(_airportX, _s), landing_airspace_for(_s, _airportX), t1)

— The following events have been recorded in the log of the monitor:

- (E1) **Initially**(covers(R1, S1), 0) [captor-0]
- (E2) **Initially**(covers(R2, S1), 0) [captor-0]
- (E3) **Happens**(changeOfLandingApproach(AR-a, S2), 0, R(0, 0)) [captor-AR-a]
- (E4) **Happens**(signal(R2, A2, S2), 1, R(1, 1)) [captor-R2]

- (E5) **Happens**(changeOfLandingApproach(AR-a,S1),2,R(2,2)) [captor-AR-a]
(E6) **Happens**(permissionRequest(A1,S1),3,R(3,3)) [captor-0]
(E7) **Happens**(signal(R1,A1,S1),7,R(7,7)) [captor-R1]
(E8) **Happens**(signal(R2,A5,S1),13,R(13,13)) [captor-R2]

Given the above event log, *Rule-1* would be violated when the event $E7:Happens(signal(R1,A1,S1),7,R(7,7))$ arrives at the monitor. This is because at this point, the monitor can deduce that the time of the clock of the event captor *captor-R1* is at least 7 and, therefore, it has not received a signal from radar R1 to notify the presence of the aircraft A1 in S1 within the period from $t=2$ to $t=6$ or, equivalently the negated predicate $\neg Happens(signal(R2,A2,S2),t, R(2,6))$ is true. The expectation for receiving such a signal is established by Rule-1 following the receipt of the event $E4:Happens(signal(R2,A2,S2),1, R(1,1))$ in the log which can be unified with Rule-1. At the point when the violation of Rule-1 is confirmed, the event $E4$ is identified as being involved in an S&D rule violation and the diagnosis process for computing a belief in its genuineness can start. The results of this process are described below assuming that the diagnosis range is determined by the time boundaries $T_{min}=0$ and $T_{max}=7$. T_{max} in this case is the timestamp of the event $E7$ that triggered the detection of the violation and T_{min} is determined by the parameter W which in this instance is assumed to take the value 7.

The set of possible explanations that can be derived for $E4:Happens(signal(R2,A2,S2),1, R(1,1))$ includes the predicate $E9:Happens(inspace(A1,S1),t1,R(2,7))$, that is:

$$\Phi_{E4,1} = \{E9\}$$

This explanation is generated by an amended implementation of the *Explain* algorithm in [6] using the assumption (A2) and the preconditions that must be met for the explanation to be valid include the predicate $H1: HoldsAt(covers(R1,S1),t1)$ for $t1$ taking values in the time range $R(2,7)$. Thus,

$$Conditions(E4, \Phi_{E4,1}, [0,7]) = \{ H1/t1 \in [2,7] \} \text{ and}$$

$$Conds(E4, \Phi_{E4,1}, [0,7]) = \{ H1/t1 \in [2,7] \}$$

Also, the set of the expected consequences of the explanation $E9:Happens(inspace(A1,S1),t1,R(2,7))$ includes the atomic formulas $E10:Happens(landingRequest(A1,AR-a), t2, R(0,6))$ and $E11:Happens(permissionRequest(A1,S1), t2, R(0,7))$. Thus:

$$\Phi_{E4,1}^C = Conseq(E4, \Phi_{E4,1}, [0,7]) = \{ E10, E11 \}$$

The atomic formula $E10$ is derived from the explanation $E9:Happens(inspace(A1,S1),t1,R(2,7))$ and the assumptions (A4) and (A5), whilst the atomic formula $E11$ is derived from $E9:Happens(inspace(A1,S1),t1,R(2,7))$ and the assumption (A3).

Thus, the computation of the basic probability in the genuineness of $E4:Happens(signal(R2,A2,S2),1, R(1,1))$ will be based on the formula:

$$\begin{aligned} m^{GN}(\text{Genuine}(E4, [0,7], \emptyset)) &= \\ m^{IL}(\text{InLog}(E4) \times m^{EX}(\text{Explainable}(E4, [0,7], \emptyset))) &= \\ m^{IL}(\text{InLog}(E4) \times \sum_{\{\Phi_{E4,1}, \Phi_{E4,1}^C\}} \{ \prod_{E_u \in \text{Conds}(\Phi_{E4,1})} m^{GN}(\text{Genuine}(E_u, [0,7], \{\Phi_{E4,1}\})) \times \\ \{ \sum_{S \subseteq \text{Conseq}(E4, \Phi_{E4,1}, [0,7]) \text{ and } S \neq \emptyset} \{ \prod_{E_w \in S} m^{GN}(\text{Genuine}(E_w, [0,7], \{\Phi_{E4,1}\})) \} \} \} &= \end{aligned}$$

$$\begin{aligned}
& 1 \times \{ m^{\text{GN}}(\text{Genuine}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\}) \times \\
& \{ \sum_{S \subseteq \{E10, E11\} \ \& \ S \neq \emptyset} \\
& \{ \prod_{E_w \in S} m^{\text{GN}}(\text{Genuine}(E_w, [0, 7], \{\Phi_{E4,1}\})) \} \} = \\
& 1 \times m^{\text{GN}}(\text{Genuine}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\}) \times \\
& \{ m^{\text{GN}}(\text{Genuine}(E10), [0,7], \{\Phi_{E4,1}\}) + \\
& m^{\text{GN}}(\text{Genuine}(E11), [0,7], \{\Phi_{E4,1}\}) - \\
& m^{\text{GN}}(\text{Genuine}(E10), [0,7], \{\Phi_{E4,1}\}) \times \\
& m^{\text{GN}}(\text{Genuine}(E11), [0,7], \{\Phi_{E4,1}\}) \}
\end{aligned}$$

The individual basic probability measures in the above formula can then be computed as follows:

— For $m^{\text{GN}}(\text{Genuine}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\})$ we have:

$$\begin{aligned}
& m^{\text{GN}}(\text{Genuine}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\}) = \\
& m^{\text{IL}}(\text{InLog}(HI/tI \in [2,7])) \times \\
& m^{\text{EX}}(\text{Explainable}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\}) = \\
& 1 \times m^{\text{EX}}(\text{Explainable}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\})
\end{aligned}$$

HI: HoldsAt(covers(R1, S1), t1) / t1 ∈ [2, 7] has the conjunction

Initially(covers(R1, S1), 0) ∧ ¬∃_e, t1: Terminates(_e, covers(R1, S1), t1, R(0, 7))

as its single explanation (this explanation is generated by the assumption (A1) and this explanation has no preconditions and further consequences. Thus,

$$\begin{aligned}
& m^{\text{EX}}(\text{Explainable}(HI/tI \in [2,7]), [0, 7], \{\Phi_{E4,1}\}) = \\
& m^{\text{GN}}(\text{Genuine}(P_{\text{NULL}})) \times m^{\text{GN}}(\text{Genuine}(C_{\text{NULL}})) = \\
& (m^{\text{IL}}(\text{InLog}(P_{\text{NULL}})) \times m^{\text{EX}}(\text{Explainable}(P_{\text{NULL}}))) \times \\
& (m^{\text{IL}}(\text{InLog}(C_{\text{NULL}})) \times m^{\text{EX}}(\text{Explainable}(P_{\text{NULL}}))) = \\
& (1 \times 1) \times (1 \times \alpha_1) = \alpha_1
\end{aligned}$$

$$\text{Thus, } m^{\text{GN}}(\text{Genuine}(HI/tI \in [2,7]), [0, 7]) = \alpha_1 \quad (\text{I})$$

— For $m^{\text{GN}}(\text{Genuine}(E10), [0, 7], \{\Phi_{E4,1}\})$ we have:

$$\begin{aligned}
& m^{\text{GN}}(\text{Genuine}(E10), [0, 7], \{\Phi_{E4,1}\}) = \\
& m^{\text{IL}}(\text{InLog}(Happens(E10), [0, 7])) \times \\
& m^{\text{EX}}(\text{Explainable}(E10), [0, 7], \{\Phi_{E4,1}\})
\end{aligned}$$

Since, however, there is no event in the log matching the formula

E10: Happens(landingRequest(A1, AR-a), t2, R(0, 6)), $m^{\text{IL}}(\text{InLog}(E10), [0, 7])$ will be equal to zero and, consequently, $m^{\text{GN}}(\text{Genuine}(E10), [0, 7], \{\Phi_{E4,1}\}) = 0$ (II)

— For $m^{GN}(\text{Genuine}(E11,[0,7], \{\Phi_{E4,1}\}))$ we have:

$$m^{GN}(\text{Genuine}(E11,[0,7], \{\Phi_{E4,1}\})) = m^{IL}(\text{InLog}(E11,[0,7])) \times m^{EX}(\text{Explainable}(E11,[0,7], \{\Phi_{E4,1}\}))$$

In this case, the event $E6:\text{Happens}(\text{permissionRequest}(A1,S1),3,R(3,3))$ in the log matches $E11:\text{Happens}(\text{permissionRequest}(A1,S1),t2,R(0,7))$ and, thus:

$$m^{IL}(\text{InLog}(E11,[0,7])) = 1$$

Note, however, that $E11:\text{Happens}(\text{permissionRequest}(A1,S1),t2,R(0,7))$ has no other explanation than the explanation $E9:\text{Happens}(\text{inspace}(A1,S1),t1,R(2,7))$ from which it was identified as an expected consequence. Thus, the set of its possible explanations will include only the NULL explanation E_{NULL} and, therefore, by virtue of Definition 2,

$$m^{EX}(\text{Explainable}(E11,[0,7], \{\Phi_{E4,1}\})) = \alpha_2.$$

$$\text{Thus, } m^{GN}(\text{Genuine}(\text{Happens}(E11,[0,7], \{\Phi_{E4,1}\}))) = \alpha_2 \quad (\text{III})$$

Given (I), (II) and (III),

$$m^{GN}(\text{Genuine}(E4, [0, 7], \{\Phi_{E4,1}\})) = 1 \times \alpha_1 \times \alpha_2$$

3.4. Schema for presentation of diagnosis results

Following the introduction of diagnostic and threat detection capabilities in the SERENITY monitoring framework, it has become necessary to update the XML schema for reporting the results of the monitoring process to the clients of the monitoring framework so as to include diagnostic and threat related information.

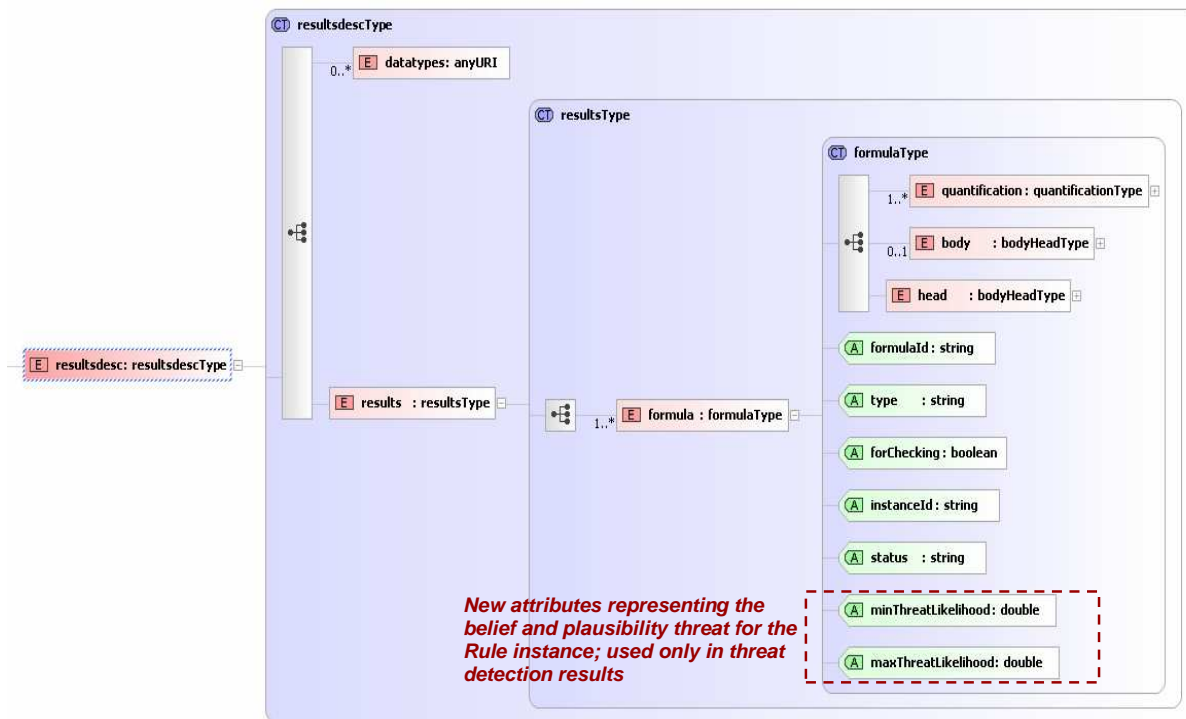


Figure 3 - Additions to *resultType* in the new schema for monitoring results

The original schema that the monitoring framework used in order to return its results back to the SRF was the same as the schema used for specifying the monitoring rules. The upper level part of this schema is shown in Figure 3. As shown in the figure, the schema enables the recording of information about rule instances which are created following the unification of rules with specific events during the monitoring process.

In particular, a rule instance is described by the element *formula* in the schema which is of type *formulaType*. The basic monitoring result that was returned by versions 1 and 2 of the monitoring framework was recorded as the value of the attribute *status* of the element *formula*. This value could be one of the following:

- *Satisfied*: This value indicates that the particular rule instance is satisfied.
- *Inconsistency_WRT_Recorded_Behaviour*: This value indicates that the particular rule instance has been violated by runtime events only (i.e., without taking into consideration any events which are derived by assumptions)
- *Inconsistency_WRT_Expected_Behaviour*: This value indicates that the particular rule instance has been violated by runtime and derived events.
- *No_Decision*: This value indicates that no decision can be made yet regarding the satisfiability of the particular rule instance.

Following the incorporation of diagnostic and threat detection capabilities into the SERENITY monitoring framework, in addition to the above information, the monitoring results should return the minimum and maximum threat likelihood measures for rule instances in all cases where the S&D rules are monitored with activated diagnosis and/or threat detection capabilities. To enable the

recording of this information, the original schema has been extended by two attributes, namely *minThreatLikelihood* and *maxThreatLikelihood*. These attributes represent the belief and plausibility of a potential violation of the rule instance, respectively, as shown in Figure 3.

Furthermore, the diagnosis and threat analysis which can be carried out by the extended monitoring framework of SERENITY, generate belief and plausibility measures for the genuineness of the individual events that have been unified with specific predicates in a rule instance and belief and plausibility measures for the possibility of occurrence of events that could be unified with such predicates in the future. To record these measures and enable their return back to the clients of the monitoring framework, the results schema of the monitor has been extended by adding two new attributes to the type *predicateType*, which is used in the schema to represent the events that have been unified or could be unified with the rule.

The new attributes (see Figure 4) are called *minThreatLikelihood* and *maxThreatLikelihood* and are used to record the belief and plausibility of the genuineness of the event unified with the predicate in the case of diagnostic results. In the case of threat detection results, these attributes are used to record the belief and plausibility of the genuineness of events that have already been unified with the particular predicate in the rule instance or the belief and plausibility of a potential occurrence of an event that could be unified with the particular predicate in the rule instance if no such event has occurred yet, respectively. To represent diagnostic information, we have also extended the type *predicateType* in the results schema with one more Boolean attribute, called *confirmed*. This attribute represents whether, given the belief and plausibility measures calculated for a particular event that has been unified with the predicate in the rule, it is possible to treat the event as a genuine event. The value of this attribute is set to *True* in cases where the belief in the genuineness of an event exceeds the belief in the non genuineness of it, i.e., when $\text{Bel}(\text{Genuine}(E)) > \text{Bel}(\neg\text{Genuine}(E))$ as we discuss in [6].

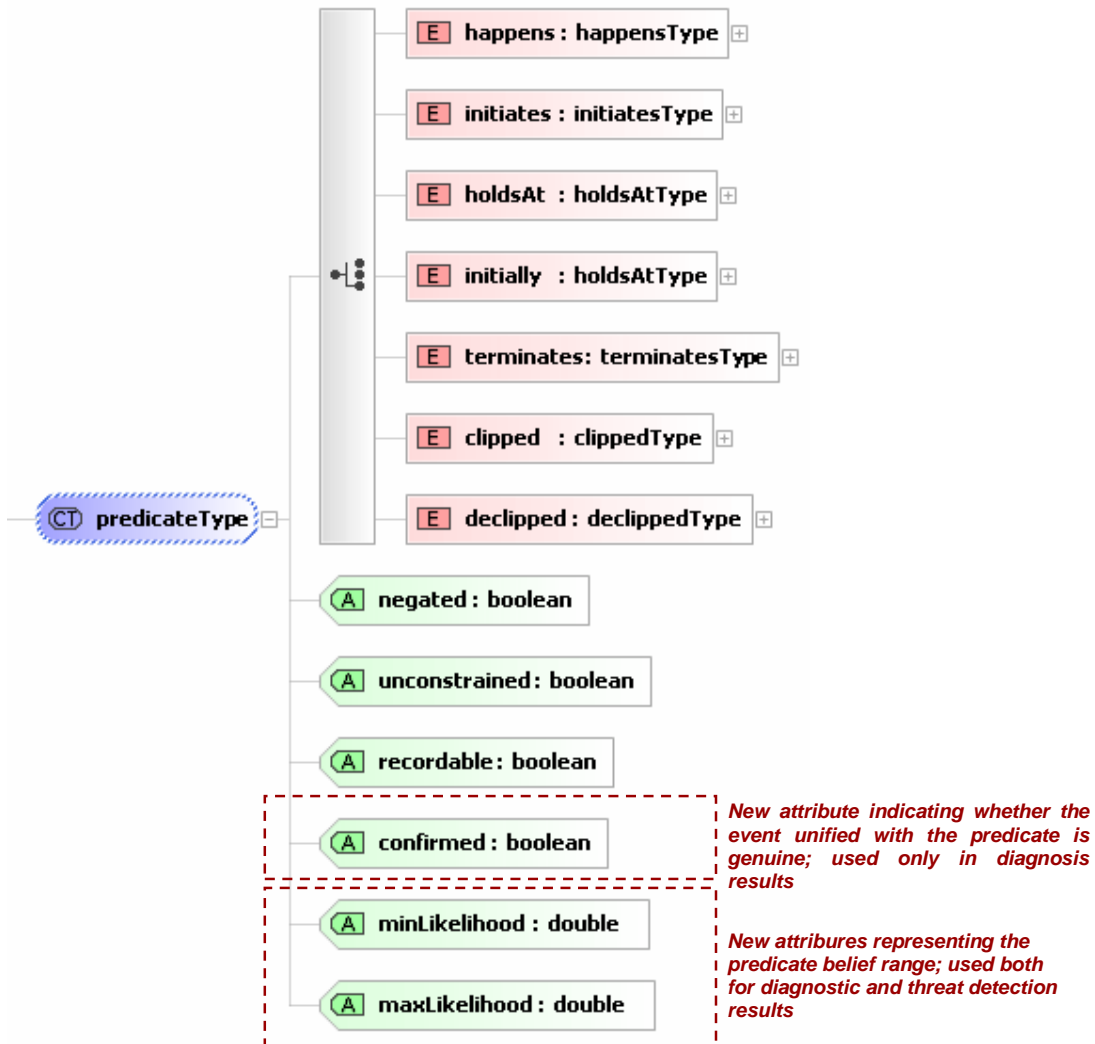


Figure 4 - Additions to *predicateType* in the new schema of monitoring results

The full new XSD result schema is given in Appendix C and the new elements in it have been highlighted.

Table 1 presents an example of monitoring results represented according to the new XML schema. The example presents a violation of Rule-1. As we discussed earlier, this rule specifies an integrity condition about the *Air Traffic Control Management Systems (ATMS)* stating that when the ATMS gets a signal event from a radar that covers a specific airspace (signified by the variable *radarID1* in the rule) that an airplane (*airplaneID1*) is in a particular airspace (*airspaceID1*) at some time point *t*, it should also get a signal event from any other radar (*radarID2*) that covers the same airspace within 5 time units. The XML results document that is listed in Table 1 shows the violation caused by an instantiation of this rule where the radar *r1001* has sent a signal indicating that the airplane *BA.1001.1001* is in the airspace *HRW.2002.2002* at the time point 1000001 but the radar *r1002* which covers the same airspace did not send a similar signal in the time range from 1000001 to 1005001 as required by the rule.

The results file also includes the minimum and maximum likelihood measures that have been calculated by the diagnosis tool of SERENITY for the genuineness of the two signal events which have caused the violation. In particular the minimum and maximum likelihood measure for *eventID1* (i.e. the event representing the signal of *radar1*) are recorded as 0.7 and 0.9, respectively. Also, the minimum and maximum likelihood measure for the absence of an event representing a signal that should have been sent from *radar2* which covers the same airspace as *radar1* are recorded as 0.1 and 0.2, respectively (i.e. the absent event is deduced by the principle of negation as failure as we have discussed in A4.D3.1 and is referred to as *eventID2* in the results file).

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<resultsdesc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ">
  <datatypes>http://www.liquid-technologies.com</datatypes>
  <results>
    <formula status="Inconsistency_Wrt_Runtime_events" forChecking="true"
formulaId="Rule-1" type="Future_Formula" instanceId="Rule-1_100">
      <quantification>
        <quantifier>forall</quantifier>
        <timeVariable>
          <varName>t1</varName>
          <varType>TimeVariable</varType>
        </timeVariable>
      </quantification>
      <quantification>
        <quantifier>existential</quantifier>
        <timeVariable>
          <varName>t2</varName>
          <varType>TimeVariable</varType>
        </timeVariable>
      </quantification>
      <body>
        <predicate negated="false" unconstrained="true" recordable="true"
          minLikelihood="0.7" maxLikelihood="0.9">
          <happens>
            <ic_term>
              <operationName>signal</operationName>
              <partnerName>127.0.0.1</partnerName>
              <id>eventID1</id>
              <variable persistent="false" forMatching="true">
                <varName>status</varName>
                <varType>OpStatus</varType>
                <value>REQ</value>
              </variable>
              <variable persistent="false" forMatching="true">
                <varName>sender</varName>
                <varType>Entity</varType>
                <value>127.0.0.1</value>
              </variable>
              <variable persistent="false" forMatching="true">
                <varName>receiver</varName>
                <varType>Entity</varType>
                <value>0.0.0.0</value>
              </variable>
              <variable persistent="false" forMatching="true">
                <varName>source</varName>
                <varType>Entity</varType>
                <value>0.0.0.0</value>
              </variable>
              <variable persistent="false" forMatching="true">
                <varName>radarID1</varName>
                <varType>string</varType>
                <value>r1001</value>
              </variable>
            </ic_term>
          </happens>
        </predicate>
      </body>
    </formula>
  </results>
</resultsdesc>
```

```

<variable persistent="false" forMatching="true">
  <varName>airplaneID1</varName>
  <varType>string</varType>
  <value>BA.1001.1001</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>airspaceID1</varName>
  <varType>string</varType>
  <value>HRW.2002.2002</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>senderID1</varName>
  <varType>string</varType>
  <value>127.0.0.1</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>receiverID1</varName>
  <varType>string</varType>
  <value>0.0.0.0</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>evSourceID1</varName>
  <varType>string</varType>
  <value>0.0.0.0</value>
</variable>
</ic_term>
<timeVar>
  <varName>t1</varName>
  <varType>TimeVariable</varType>
  <value>1000001</value>
</timeVar>
<fromTime>
  <time>
    <varName>t1</varName>
    <varType>TimeVariable</varType>
    <value>1000001</value>
  </time>
</fromTime>
<toTime>
  <time>
    <varName>t1</varName>
    <varType>TimeVariable</varType>
    <value>1000001</value>
  </time>
</toTime>
</happens>
</predicate>
<operator>and</operator>
<predicate negated="false" unconstrained="false" recordable="true">
  <holdsAt>
    <fluent name="CoversFluent">
      <variable persistent="false" forMatching="true">
        <varName>radarID1</varName>
        <varType>string</varType>
        <value>r1001</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>airspaceID1</varName>
        <varType>string</varType>
        <value>HRW.2002.2002</value>
      </variable>
    </fluent>
  <timeVar>
    <varName>t1</varName>
    <varType>TimeVariable</varType>

```

```

    <value>1000001</value>
  </timeVar>
</holdsAt>
</predicate>
<operator>and</operator>
<predicate negated="false" unconstrained="false" recordable="true">
  <holdsAt>
    <fluent name="CoversFluent">
      <variable persistent="false" forMatching="true">
        <varName>radarID2</varName>
        <varType>string</varType>
        <value>r1002</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>airspaceID1</varName>
        <varType>string</varType>
        <value>HRW.2002.2002</value>
      </variable>
    </fluent>
    <timeVar>
      <varName>t1</varName>
      <varType>TimeVariable</varType>
      <value>1000001</value>
    </timeVar>
  </holdsAt>
</predicate>
</body>
<head>
<predicate      negated="false"      unconstrained="false"      recordable="true"
  minLikelihood="0.1" maxLikelihood="0.3">
  <happens>
    <ic_term>
      <operationName>signal</operationName>
      <partnerName>127.0.0.2</partnerName>
      <id>eventID2</id>
      <variable persistent="false" forMatching="true">
        <varName>status2</varName>
        <varType>OpStatus</varType>
        <value>REQ</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>sender2</varName>
        <varType>Entity</varType>
        <value>127.0.0.2</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>receiver1</varName>
        <varType>Entity</varType>
        <value>0.0.0.0</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>source1</varName>
        <varType>Entity</varType>
        <value>0.0.0.0</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>radarID2</varName>
        <varType>string</varType>
        <value>r1002</value>
      </variable>
      <variable persistent="false" forMatching="true">
        <varName>airplaneID1</varName>
        <varType>string</varType>
        <value>BA.1001.1001</value>
      </variable>

```



```

<variable persistent="false" forMatching="true">
  <varName>airspaceID1</varName>
  <varType>string</varType>
  <value>HRW.2002.2002</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>senderID2</varName>
  <varType>string</varType>
  <value>127.0.0.2</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>receiverID1</varName>
  <varType>string</varType>
  <value>0.0.0.0</value>
</variable>
<variable persistent="false" forMatching="true">
  <varName>evSourceID1</varName>
  <varType>string</varType>
  <value>0.0.0.0</value>
</variable>
</ic_term>
<timeVar>
  <varName>t2</varName>
  <varType>TimeVariable</varType>
</timeVar>
<fromTime>
  <time>
    <varName>t1</varName>
    <varType>TimeVariable</varType>
    <value>1000001</value>
  </time>
</fromTime>
<toTime>
  <time>
    <varName>t1</varName>
    <varType>TimeVariable</varType>
    <value>1005001</value>
  </time>
</toTime>
</happens>
</predicate>
</head>
</formula>
</results>
</resultsdesc>

```

Table 1 – An example of monitoring results for a specific rule instance

3.5. API for diagnosis tool

The diagnosis tool is an internal module of the monitor component [3] as it is presented in Figure 5. It should be noted that the monitor is deployed as a web service. Thus, this report describes an application programming interface (API) that enables the retrieval of diagnostic results from external clients of the monitor and, in particular, the SERENITY runtime framework. Please note that the provided diagnosis results follow the representation schema presented in Figure 4.

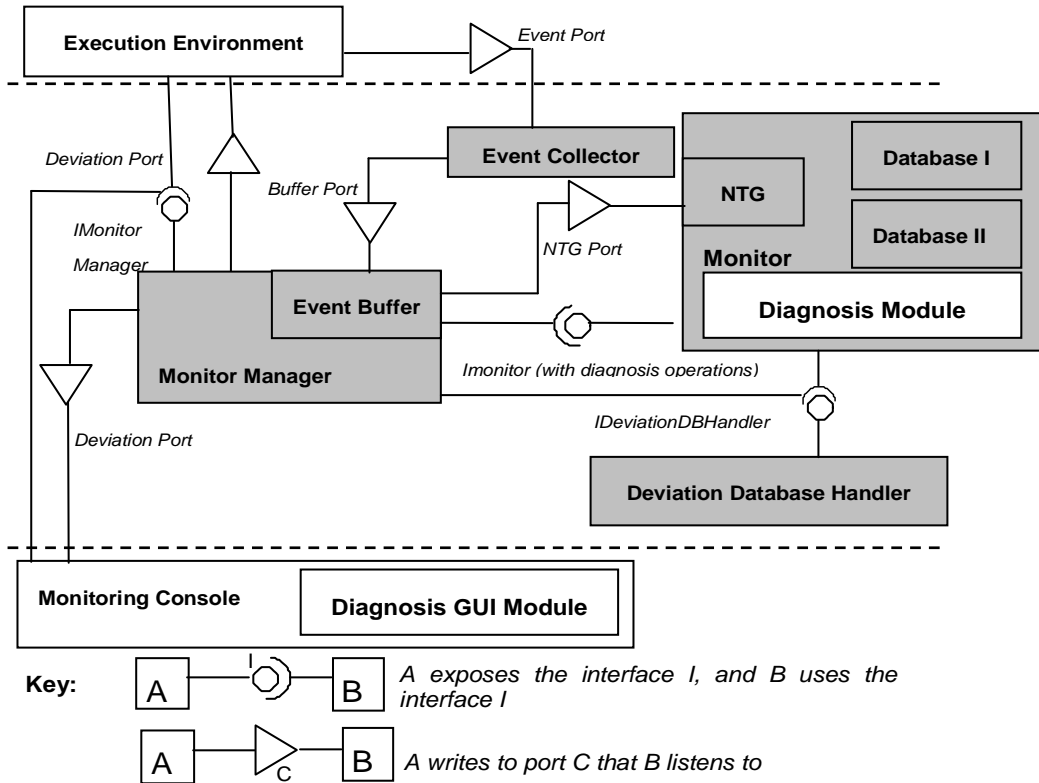


Figure 5 - Architecture of monitor and diagnosis prototype

The API for the monitor and the diagnosis tool is the *NewDataAnalyzerEC* class that is included in the *code* package (*code.NewDataAnalyzerEC.java*) of the delivered code. More specifically, the API for the diagnosis tool consists of the following methods:

— `public String initialiseDiagnoser(String xmlFailedTemplate)`

Return Value: String

Argument: String xmlFailedTemplate

Description: This operation reads a string representation of the xml format of the violated rule and notifies the diagnosis tool of the violations observations. It returns a string indicating whether the diagnosis tool is notified successfully.

— `public String diagnose()`

Return value: String

Argument: N/A

Description: This operation calls the methods that implement the explanation generation algorithm [6], the explanation effect identification algorithm [6] and the new belief assessment process as it is described in Section 3.2. . The return value is a string representation of the xml format of the monitoring and diagnosis results as presented in Table 1.

The full new API of the monitor is given in Appendix D, where the diagnosis tool elements (operations, requests, responses) in it have been highlighted.

4. Implementation and Usage Guide

4.1. Required Software

To use the diagnosis prototype, the user should download and install on his/her machine:

- Version 5.0.14 of the Tomcat server – This server can be downloaded from <http://tomcat.apache.org/>. An installation guide for the server is also available at the same site. Please consult the release note of tomcat for the selection of right XML parser. (*Hint*: During installation, the xalan.jar file should be copied in the tomcat\common\lib folder. The xalan.jar file can be found in the *lib* folder of the manager project)
- Version 1.4 of Axis server – This server can be downloaded from <http://ws.apache.org/axis/>. An installation guide for the server is also available at the same site. (*Hint*: Attention must be given in the copy of the appropriate jar files into the Axis installation folders, as well as, in the classpath setup of Axis server. More details can be found in the Classpath setup section of the installation guide for the server that is provided at the site)

4.2. Installation instructions

The installation of the diagnosis tool is based on the same procedure as the one followed for the monitor. More specifically,

- To install the monitoring manager, extract the files in the archive into the folder C:\Monitor
- To install the data analyzer, copy the folder C:\Monitor\analyzer\code in the classes folder of the axis installation in Tomcat. This prototype assumes that the Tomcat server is deployed on port number 8080 (i.e. default port for tomcat).

4.3. Usage instructions

The functionality of the diagnosis module is available only through the dynamic validation prototype. Thus in the following we give basic instructions on how to start and use the dynamic validation prototype and, as part of it, how to use the diagnostic capabilities that have been integrated into it.

4.3.1. The monitor

To start the monitor (also known as “data analyzer”), the user has to start the Tomcat server by executing the startup

file in the TOMCAT_HOME\bin folder.

To use the analyzer with the monitoring manager, in a command prompt window give the command C:\Monitor\analyzer\deploy The data analyzer is up and the wsdl specification of the data analyzer service can be seen at:

<http://localhost:8080/axis/services/analyzerService?wsdl>

and the analyzer service endpoint is

<http://localhost:8080/axis/services/analyzerService>

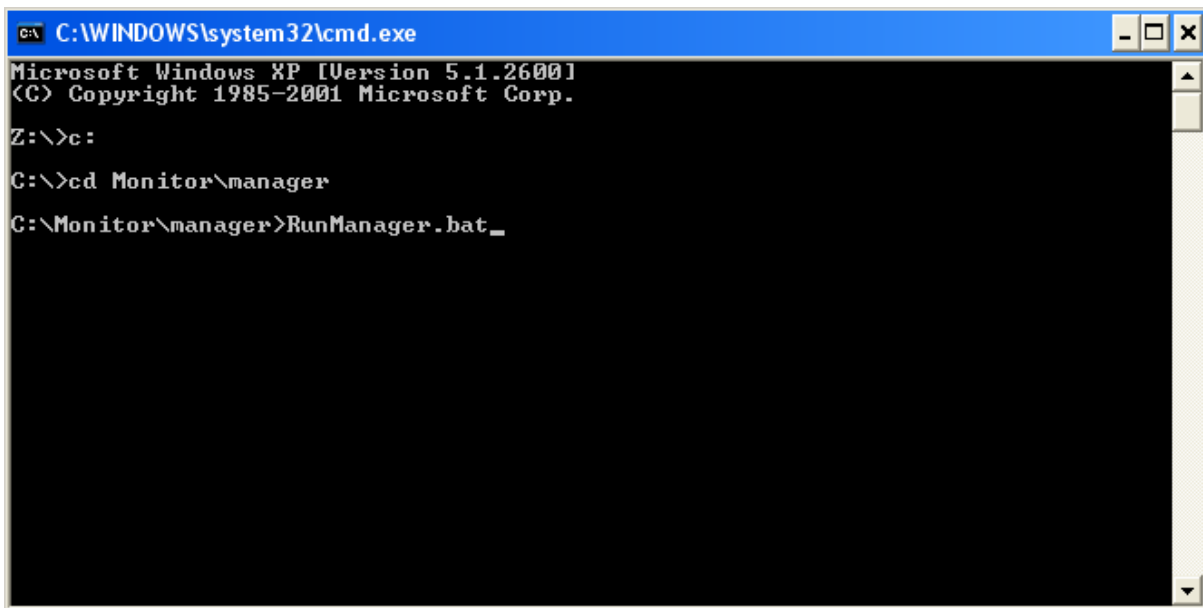
Section 4.3.2. describes how to use the monitoring manager, the diagnostic component of the prototype and the event collector.

4.3.2. *The Monitoring Manager and Diagnosis Module*

The monitoring manager is used to import and select the formulae to be monitored, send the selected formulae to the data analyzer, start the event collector for a monitoring session, initiate a polling process that retrieves possible violations of the properties and view the result of monitoring. To retrieve violations of properties, the monitoring manager polls the data analyzer at regular time intervals that can be specified by the user and shows the results that it retrieves in a formula viewer.

To use the monitor manager, follow the following steps:

1. To start the monitor manager, in a command prompt window execute the command `C:\Monitor\manager\RunManager` as shown in Figure 6 below. Following this, the monitor manager window will pop up. Then, to import the formulae to be monitored, select the option "Import Formulae" from the "File" menu of the manager.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

Z:\>c:
C:\>cd Monitor\manager
C:\Monitor\manager>RunManager.bat_
```

Figure 6 - Command prompt window

2. Then, to import the formulae to be monitored, select the option "Import Formulae" from the "File" menu of the manager.

3. In the file opening dialog box that appears following the selection of this option (see Figure 7), choose the XML file that contains the formulae that you want to monitor.

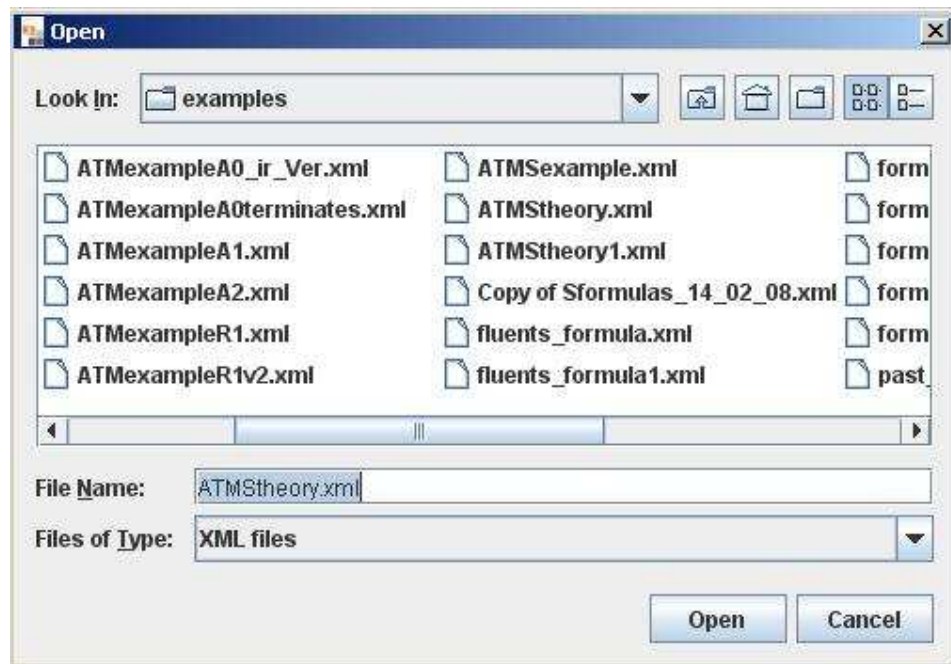


Figure 7 - The file opening dialogue box

4. The monitor manager will then read all the formulae from the file and display the formulae in the display panel of the monitoring manager as illustrated in Figure 8.

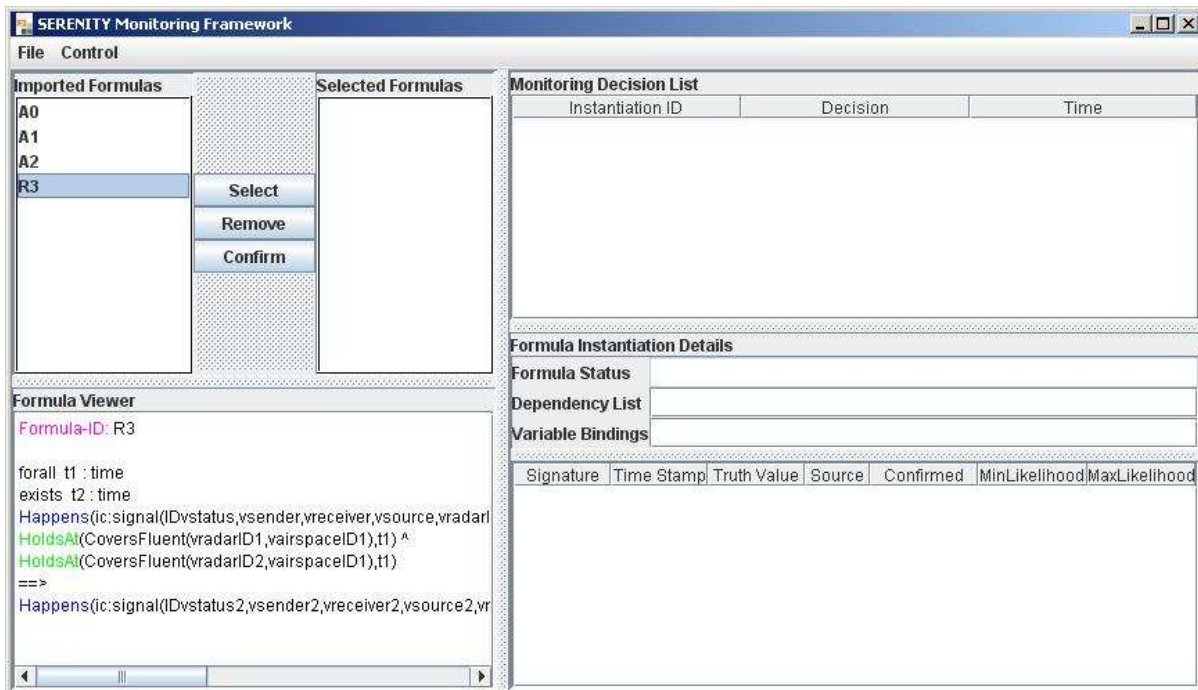


Figure 8 - The Monitoring Manager

The monitoring manager lists the identifiers of the imported formulae in the "Imported Formulae" panel. To view a formula in the event calculus format, the user may select its ID. Following this selection, the formula with the selected ID will be shown in the "Formula Viewer" panel of the manager. If the user wants to select the formula to be monitored, he/she may select its ID in the imported formulae panel and click on the "Select" button. Following this, the selected formula will appear in the "Selected Formulae" panel. The user may repeat the same process to select more formulae. When the selection is complete, the user can click on the "Confirmed" button, to send the formulae to the data analyser. If the submission of formulae to the analyser is successful, the monitor manager will show the following message. The user should press the "Ok" button to continue (see Figure 9).



Figure 9 - Formulas submission message

5. The next step is to provide the analyzer with runtime events. From the Start Data Collection command in the Controls menu of the monitor manager we can activate the collection of the events. The monitor manager can by default accept events in the port number 12345 and report them to the analyzer. The format of those events is based in the event XML schema

described in [1]. Any event collection mechanism that can provide events according to the defined XML schema can be used to report the events. For the purposes of the demo we can use the SOAP event collector which can be located in C:\Monitor\SoapCollector. This application creates a proxy service which listens to a user defined port number, accepts incoming SOAP messages, translates them to the specified event format and then forwards them to the real web service for the execution of the service and to the monitor management tool. For the execution of the collector the user must type `java -cp xercesImpl.jar; commons-net-1.4.1.jar; code.TcpTunnel` followed by the parameters of the listening port, the IP address and port of the real web service and the IP address and port of the monitoring manager. In our case, where the Tomcat is deployed locally in the default port (i.e. 8080) and the manager listens for events in the 12345 port, the execution command should be `java -cp xercesImpl.jar;commons-net-1.4.1.jar; code.TcpTunnel 8081 localhost 8080 localhost 12345`. Now that the collector is activated we must inform the monitor manager to accept any events are send to it by selecting the Start Data Collector option from the Control menu (see Figure 11). Any attempt to invoke a web service in the port 8081 will result to report this event to the monitor manager tool.

6. To set the diagnosis window, the belief value for NULL consequences and the belief value for NULL explanations, the user should select the "parameters" option from the "control" menu in this window and, in the dialog box that pops up (see Figure 10), specify new values for the aforementioned parameters

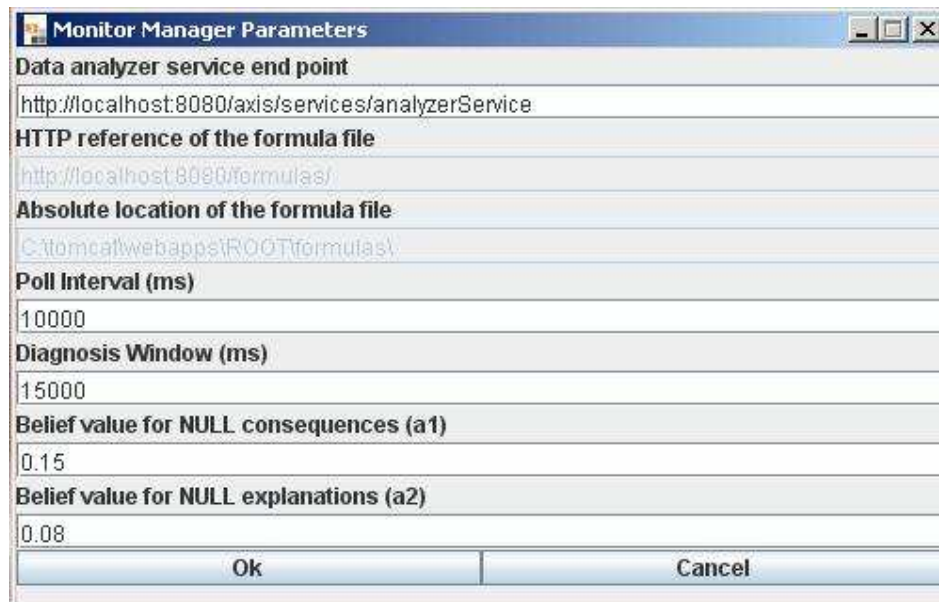


Figure 10 - Monitor Manager parameters dialogue box

7. The user may stop the data collector by selecting the option "Stop Data Collector" in the "Control" menu (see Figure 11)

- To start polling the data analyser in order to view the violations of the formulae being

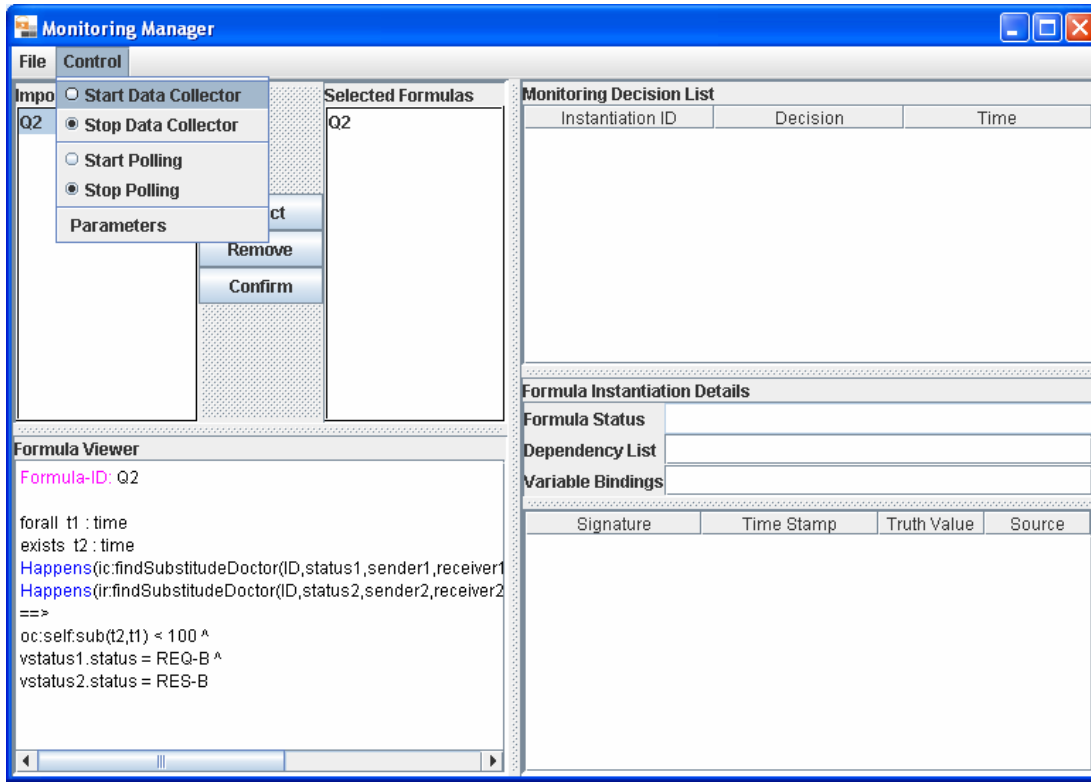


Figure 11 - Starting and stopping data collector in the Monitoring Manager

monitored, the user should select the option "start polling" from the control menu. Following this, the monitor manager will start polling the data analyser at regular time intervals (the default time interval is 10 seconds).

The monitoring manager shows the list of instances of the violated and satisfied formulae in the "Monitoring Decision List" panel as shown in Figure 12. This panel will be updated at the regular intervals. The Monitoring Decision List will show the monitoring summary of each instance of each formula. The left most column in this list shows the unique formula instance ID, the middle column shows the decision for the formula instance, and the right most column shows the time when the decision was made by the data analyzer.

10. To change the polling interval, the user should select the option "stop polling" in the main monitor manager window, then select "parameters" from the "control" menu in this window and, in the dialog box that pops up (see Figure 10), specify a new value for the polling interval. Subsequently, the user should select the option "restart polling" from the control window.
11. To stop the manager, the user should first stop the data collector (if it is running), then stop polling, and finally select the "exit" option from file menu to exit the monitor manager (see Figure 11).

5. Conclusions

This report is part of the deliverable A4.D5.2 and provides a description of the second version of the diagnosis prototype that has been developed as part of SERENITY monitoring framework. In particular, we have described the main amendments that were introduced to the diagnosis approach advocated in SERENITY and given examples demonstrating the results of the amendments. We have not, however, provided a full description of this approach as this can be found in [6].

As we have discussed in [6], the diagnosis of violations of S&D rules in SERENITY is based on generating possible explanations for the runtime events which are involved in violations of S&D monitoring rules through the use of *abductive reasoning*. This process deploys the *assumptions* that have been specified about the system that is being monitored and the S&D patterns which are deployed by it. The possible explanations which are generated by this process are checked against other runtime events to establish if there is further evidence about the validity of the explanations. The input to this search process includes the expected *consequences* of the abduced explanations and – in the new version of the diagnosis tool – any preconditions that need to be satisfied for an explanation to be able to entail the event in question. The new version of the SERENITY diagnosis framework computes beliefs in the validity of the explanations of the events that are involved in S&D violations, based on matches that may be found between the preconditions and consequences of these explanations with other runtime events and a recursive assessment of the genuineness of these events.

It should be noted that the second version of the diagnosis prototype is fully integrated with the SERENITY monitoring framework described in it [3] and the code that implements the new version of the monitoring framework along with the realisation of the diagnostic capabilities described in this report is also part of A4.D5.2.

Ongoing work focuses on conducting an experimental evaluation of the undertaken approach.

Appendix A. 1st Version Algorithms

Explain(e , $t_{min}(e)$, $t_{max}(e)$, f_{init})

```

1.   $\Phi_e = [ f_{init}; e ]_{OR}$  /* a list keeping the disjunction of possible explanations of  $e$  */
2.  If  $e \in ABD$  Then
3.     $\Phi_e = \text{append}(\Phi_e, (e, t_{min}(e), t_{max}(e)))$ 
4.  Else /*  $e$  is not an abducible atom; find explanations for it */
5.    For all  $f \in AS$  Do /* try all alternative explanations */
6.       $u = \text{mgu}(\text{head}(f), e)$  /* mgu returns the most general unifier of  $e$  and a predicate  $p$  if this unifier exists*/
7.      If  $u \neq \emptyset$  and  $u$  covers all non time variables in  $\text{body}(f)$  Then
8.        Copy  $\text{body}(f)$  into  $CND_f$ 
9.        FormulaFailed = False
10.        $\Phi_f = []_{AND}$  /*  $\Phi_f$  is a list keeping a conjunction of elements explaining the conditions of  $f$  */
11.       While FormulaFailed = False and  $CND_f \neq \emptyset$  Do /* explain all conditions of  $f$  */
12.         Remove some condition  $C$  from  $CND_f$ 
13.         Compute the minimum and maximum possible values  $t_{min}(C)$ ,  $t_{max}(C)$  of  $C$ 
14.         based on  $t_{min}(e)$  and  $t_{max}(e)$ 
15.         If  $t_{min}(C) \neq \text{NULL}$  and  $t_{max}(C) \neq \text{NULL}$  Then /*  $t_{min}(C)$ ,  $t_{max}(C)$  are not undeterminable */
16.            $C_u = \text{ApplyUnification}(u, C)$ 
17.           If  $C \in ABD$  Then /*  $C$  is an abducible so add it to current explanation */
18.              $\Phi_f = \text{append}(\Phi_f, [(f; C_u, t_{min}(C), t_{max}(C))]_{ABD})$ 
19.           Else /*  $C$  is not an abducible condition */
20.             find a derived or recorded event  $e_c$  that can be unified with  $C_u$  and
21.              $t_{min}(e_c) \geq t_{min}(C)$  and  $t_{max}(e_c) \leq t_{max}(C)$ 
22.             If  $e_c = \text{NULL}$  Then /* no logged or derived event matching  $C$  has been found */
23.                $\Phi_C = \text{Explain}(C, t_{min}(C), t_{max}(C), f)$ 
24.               If  $\Phi_C$  is empty Then
25.                 FormulaFailed = True
26.               Else
27.                  $\Phi_f = \text{append}(\Phi_f, \Phi_C)$ 
28.               End If
29.             End If
30.           End If
31.         End If
32.       End While
33.       If FormulaFailed = False Then  $\Phi_e = \text{append}(\Phi_e, \Phi_f)$  End if
34.     End for
35.   End For
36. End If
37. return( $\Phi_e$ )
38. END Explain

```

Figure 13 - Algorithm for generating explanations of atomic events involved in violations

Generate_AE_consequences(AF: Set of Grounded Atomic Formulas, TLIST: List of Assumption Templates, CNS: Set of Consequences)

1. CNS = { }
2. TLIST' = copy of TLIST
3. **For** each atomic formula $P_i \in AF$ **Do**
4. **For** each assumption template T in TLIST' **Do**
5. **For** each predicate $Q \in \text{body}(T)$ **Do**
6. **If** $\text{mgu}(P_i, Q) \neq \emptyset$ and $\text{CompatibleTimeRange}(P_i, Q)$ **Then**
7. $T' :=$ copy of T
8. Apply $\text{mgu}(P_i, Q)$ onto T'
9. Set the truth value of Q in T' to True
10. Update time ranges of other predicates in T' based on the time range of P_i
11. **If** for all predicates $R \in \text{Body}(T')$ such that $R \neq Q$, R is true **Then**
12. **If** $\text{head}(T')$ is fully instantiated **Then**
13. **If** $\text{head}(T')$ is observable **Then**
14. CNS = CNS \cup { (T.id, $\text{head}(T')$) }
15. delete T'
16. **Else** /* $\text{head}(T')$ is a derived predicate */
17. CNS' = { }
18. Generate_consequences({ $\text{head}(T')$ }, TLIST', CNS')
19. CNS = CNS \cup CNS'
20. **End If**
21. **End If**
22. **Else** /* there is a predicate R in $\text{Body}(T')$ whose truth value is unknown */
23. **If** for all predicates $R \in \text{Body}(T')$ such that $R \neq Q$ and R is not true, R is an abducible predicate **Then**
24. TLIST' = append (T' , TLIST')
25. **End If**
26. **End If**
27. **End If**
28. **End For**
29. **End For**
30. Return (CNS)
31. **END Generate_AE_Consequences**

Figure 14 - Algorithm for computing the transitive closure of deductions from abduced predicates

Generate_RE_consequences(AF: Set of Grounded Atomic Formulas, TLIST: List of Assumption Templates, CNS: Set of Consequences)

```

1. CNS = { }
2. For each atomic formula  $P_i \in AF$  Do
3.   For each assumption template T in TLIST Do
4.     For each predicate  $Q \in body(T)$  Do
5.       If  $mgu(P_i, Q) \neq \emptyset$  and  $CompatibleTimeRange(P_i, Q)$  Then
6.          $T' :=$  copy of T
7.         Apply  $mgu(P_i, Q)$  onto  $T'$ 
8.         Set the truth value of Q in  $T'$  to True
9.         Update time ranges of other predicates in  $T'$  based on the time range of  $P_i$ 
10.        If for all predicates  $R \in Body(T')$  such that  $R \neq Q$ , R is true Then
11.          If  $head(T')$  is fully instantiated Then
12.            If  $head(T')$  is observable Then
13.               $CNS = CNS \cup \{ (T'.id, head(T')) \}$ 
14.              delete  $T'$ 
15.            Else /* $head(T')$  is a derived predicate */
16.               $CNS' = \{ \}$ 
17.              Generate_consequences( $\{head(T')\}$ , TLIST,  $CNS'$ )
18.               $CNS = CNS \cup CNS'$ 
19.            End If
20.          End If
21.        Else /* there is a predicate R in  $Body(T')$  whose truth value is unknown */
22.          TLIST = append ( $T'$ , TLIST)
23.        End If
24.      End If
25.    End For
26.  End For
27. Return (CNS)
END Generate_consequences

```

Figure 15 - Algorithm for computing the transitive closure of deductions from recorded events

Appendix B. Proofs of Theorems

Theorem 1: The basic probability assignment to the genuineness of events is calculated by the formulas:

$$\begin{aligned}
m^{\text{GN}}(\text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) &= m^{\text{IL}}(\text{InLog}(E_i) \times m^{\text{EX}}(\text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o))) \\
m^{\text{GN}}(\neg \text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) &= \\
& m^{\text{IL}}(\neg \text{InLog}(E_i) + m^{\text{EX}}(\neg \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o))) - \\
& (m^{\text{IL}}(\neg \text{InLog}(E_i) \times m^{\text{EX}}(\neg \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o))))
\end{aligned}$$

Proof: The combination of the basic probability assignments m^{IL} and m^{EX} requires their mapping on a common frame of discernment, i.e., a set of mutually exclusive propositions representing exhaustively the properties that m^{IL} and m^{EX} cast evidence for. This frame of discernment θ can be defined as a set of vectors consisting of n-pairs pairs of Boolean variables (L_i, V_i) :

$$[(L_1, V_1), \dots, (L_n, V_n)]$$

where

- n is the total number of events within a given diagnosis range
- each of the pairs (L_i, G_i) corresponds to an event E_i .
- the value of the variable L_i in a pair denotes whether or not the event E_i matches with an event that has already or can occur in the log: when $L_i=1$ E_i has already occurred or can occur at some future time point in the log and when $L_i=0$ E_i has neither occurred nor it may occur at some future time point in the log.
- the value of the variable V_i denotes whether or not the event E_i is explainable: when $V_i=1$ E_i 's can be explained and when $V_i=0$ E_i cannot be explained.

Given this definition of θ , the propositions that m^{IL} assigns evidence for (aka *focals* in the context of the DS theory) are:

$$\text{InLog}(E_i) = \{[(L_1, V_1), \dots, (L_i, V_i), \dots, (L_n, V_n)] | L_i=1\}$$

$$\neg \text{InLog}(E_i) = \{[(L_1, V_1), \dots, (L_i, V_i), \dots, (L_n, V_n)] | L_i=0\}$$

Also, the propositions (or, equivalently, focals) that m^{EX} assigns evidence for are:

$$\text{Explainable}(E_i) = \{[(L_1, V_1), \dots, (L_i, V_i), \dots, (L_n, V_n)] | V_i=1\}$$

$$\neg \text{Explainable}(E_i) = \{[(L_1, V_1), \dots, (L_i, V_i), \dots, (L_n, V_n)] | V_i=0\}$$

Thus, the intersections of these focals are not empty sets:

$$\text{InLog}(E_i) \cap \text{Explainable}(E_i) \neq \emptyset$$

$$\neg \text{InLog}(E_i) \cap \text{Explainable}(E_i) \neq \emptyset$$

$$\text{InLog}(E_i) \cap \neg \text{Explainable}(E_i) \neq \emptyset$$

$$\neg \text{InLog}(E_i) \cap \neg \text{Explainable}(E_i) \neq \emptyset$$

Therefore, according to Theorem 3.1 in [3] (p. 60), the basic probability assignments m^{IL} and m^{EX} can be combined using the rule of the orthogonal sum, i.e.:

$$m_1 \oplus m_2 (P) = (\sum_{X \cap Y = P} m_1(X) \times m_2(Y)) / (1 - k_0)$$

where $k_0 = \sum_{V \cap W = \emptyset \text{ and } V \subseteq \theta \text{ and } W \subseteq \theta} m_1(V) \times m_2(W)$

Since, however, the intersections of all the combinations of the focals of m^{IL} and m^{EX} the parameter k_0 in the rule will be equal to zero.

The following matrix shows the different combinations of the focals of the assignments m^{IL} and m^{EX} for a given event E and the basic probability measures that they assign to the intersections of these combinations.

		m^{IL}		
		P1: $m^{IL}(P1)$	$\neg P1$: $m^{IL}(\neg P1)$	$P1 \cup \neg P1 = \theta$: $(1 - m^{IL}(P1) - m^{IL}(\neg P1))$
m^{EX}	P2: $m^{EX}(P2)$	$P1 \cap P2$ $m^{IL}(P1) \times m^{EX}(P2)$	$\neg P1 \cap P2$ $m^{IL}(\neg P1) \times m^{EX}(P2)$	$(P1 \cup \neg P1) \cap P2 = \theta \cap P2 = P2$ $(1 - m^{IL}(P1) - m^{IL}(\neg P1)) \times m^{EX}(P2)$
	$\neg P2$: $m^{EX}(\neg P2)$	$P1 \cap \neg P2$ $m^{IL}(P1) \times m^{EX}(\neg P2)$	$\neg P1 \cap \neg P2$ $m^{IL}(\neg P1) \times m^{EX}(\neg P2)$	$\theta \cap \neg P2 = \neg P2$ $(1 - m^{IL}(P1) - m^{IL}(\neg P1)) \times m^{EX}(\neg P2)$
	$P2 \cup \neg P2 = \theta$: $(1 - m^{EX}(P2) - m^{EX}(\neg P2))$	$P1 \cap (P2 \cup \neg P2) = P1 \cap \theta = P1$ $m^{IL}(P1) \times (1 - m^{EX}(P2) - m^{EX}(\neg P2))$	$\neg P1 \cap (P2 \cup \neg P2) = \neg P1 \cap \theta = \neg P1$ $m^{IL}(\neg P1) \times (1 - m^{EX}(P2) - m^{EX}(\neg P2))$	$\theta \cap \theta = \theta$ $(1 - m^{IL}(P1) - m^{IL}(\neg P1)) \times (1 - m^{EX}(P2) - m^{EX}(\neg P2))$
Where				
— P1 = InLog(E)				
— P2 = Explainable(E)				
— $\neg P_i$ is the complement set of P_i with respect to the powerset of θ				

Thus, as

$$m^{GN}(\text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) = m^{IL} \oplus m^{EX} (\text{InLog}(E_i) \wedge \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o))$$

we have

$$m^{GN}(\text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) = m^{IL} \oplus m^{EX} (P1 \cap P2) = m^{IL}(P1) \times m^{EX}(P2) \quad (\text{I})$$

Also,

$$\begin{aligned} m^{GN}(\neg \text{Genuine}(E_i, [T_{\min}, T_{\max}], E_o)) &= m^{IL} \oplus m^{EX} (\neg(\text{InLog}(E_i) \wedge \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o))) = \\ &= m^{IL} \oplus m^{EX} (\neg \text{InLog}(E_i) \vee \neg \text{Explainable}(E_i, [T_{\min}, T_{\max}], E_o)) = m^{IL} \oplus m^{EX} (\neg P1 \cup \neg P2) \end{aligned}$$

Due to the rule of the orthogonal sum, however, we have:

$$\begin{aligned} m^{IL} \oplus m^{EX} (\neg P1 \cup \neg P2) &= m^{IL} \oplus m^{EX} (\neg P1 \cap P2) + m^{IL} \oplus m^{EX} (P1 \cap \neg P2) + m^{IL} \oplus m^{EX} (\neg P1 \cap \\ &\quad \neg P2) + m^{IL} \oplus m^{EX} (\theta \cap \neg P2) + m^{IL} \oplus m^{EX} (\neg P1 \cap \theta) \\ &= m^{IL}(\neg P1) \times m^{EX}(P2) + m^{IL}(P1) \times m^{EX}(\neg P2) + m^{IL}(\neg P1) \times m^{EX}(\neg P2) + \\ &\quad (1 - m^{IL}(P1) - m^{IL}(\neg P1)) \times m^{EX}(\neg P2) + m^{IL}(\neg P1) \times \\ &\quad (1 - m^{EX}(P2) - m^{EX}(\neg P2)) \\ &= m^{IL}(\neg P1) + m^{EX}(\neg P2) - m^{IL}(\neg P1) \times m^{EX}(\neg P2) \end{aligned} \quad (\text{II})$$

From (I) and (II), it follows that the theorem holds.

Appendix C. New Schema for Monitoring Results

```

<?xml version="1.0" encoding="utf-8" ?>
<!-- Created with Liquid XML Studio 1.0.8.0 (http://www.liquid-technologies.com) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="resultsdesc" type="resultsdescType" />
  <xs:complexType name="resultsdescType">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="datatypes" type="xs:anyURI" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="resultsType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="formula" type="formulaType" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="formulaType">
    <xs:annotation>
      <xs:documentation>
        ****
        **** ADDED-END
        ****
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="quantification"
type="quantificationType" />
      <xs:element minOccurs="0" name="body" type="bodyHeadType" />
      <xs:element name="head" type="bodyHeadType" />
    </xs:sequence>
    <xs:attribute name="formulaId" type="xs:string" use="required" />
    <xs:attribute name="type" type="xs:string" use="required" />
    <xs:attribute default="true" name="forChecking" type="xs:boolean" />
    <xs:attribute name="instanceId" type="xs:string" use="required">
      <xs:annotation>
        <xs:documentation>
          ****
          **** ADDED-BEGIN
          ****
          The id of the formula instance in the monitor </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="status" type="xs:string" use="required">
        <xs:annotation>
          <xs:documentation>
            whether the formula instance has been satisfied or not:
            &quot;Satisfied&quot;;
            &quot;Undefined&quot;;
            &quot;Inconsistency_WRT_Recorded_Behaviour&quot;;
            &quot;Inconsistency_WRT_Expected_Behaviour&quot;;
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="minThreatLikelihood" type="xs:double">
        <xs:annotation>
          <xs:documentation>The current minimum threat likelihood for this
formula</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="maxThreatLikelihood" type="xs:double">
        <xs:annotation>

```

```

    <xs:documentation>The current maximum threat likelihood for this
formula</xs:documentation>
  </xs:annotation>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="bodyHeadType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="predicate" type="predicateType" />
      <xs:element name="relationalPredicate" type="relationalPredicateType" />
    </xs:choice>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="operator" type="logicalOperatorType" />
      <xs:choice>
        <xs:element name="predicate" type="predicateType" />
        <xs:element name="timePredicate" type="timePredicateType" />
        <xs:element name="relationalPredicate" type="relationalPredicateType" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="predicateType">
    <xs:annotation>
      <xs:documentation>
        ****
        **** ADDED-END
        ****
      </xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:element name="happens" type="happensType" />
      <xs:element name="initiates" type="initiatesType" />
      <xs:element name="holdsAt" type="holdsAtType" />
      <xs:element name="initially" type="holdsAtType" />
      <xs:element name="terminates" type="terminatesType" />
      <xs:element name="clipped" type="clippedType" />
      <xs:element name="declipped" type="declippedType" />
    </xs:choice>
    <xs:attribute default="false" name="negated" type="xs:boolean" />
    <xs:attribute default="false" name="unconstrained" type="xs:boolean" />
    <xs:attribute default="false" name="recordable" type="xs:boolean" />
    <xs:attribute default="false" name="confirmed" type="xs:boolean">
      <xs:annotation>
        <xs:documentation>
          ****
          **** ADDED-BEGIN
          ****
          Whether this predicate is confirmed by the observed events in the
log</xs:documentation>
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="minLikelihood" type="xs:double">
      <xs:annotation>
        <xs:documentation>The minimum likelihood of this predicate to be
genuine</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="maxLikelihood" type="xs:double">
      <xs:annotation>
        <xs:documentation>The maximum likelihood of this predicate to be
genuine</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="timePredicateType">

```

```

<xs:choice>
  <xs:element name="timeEqualTo" type="TimeRelation" />
  <xs:element name="timeNotEqualTo" type="TimeRelation" />
  <xs:element name="timeLessThan" type="TimeRelation" />
  <xs:element name="timeGreaterThan" type="TimeRelation" />
  <xs:element name="timeLessThanEqualTo" type="TimeRelation" />
  <xs:element name="timeGreaterThanEqualTo" type="TimeRelation" />
</xs:choice>
</xs:complexType>
<xs:complexType name="holdsAtType">
  <xs:sequence>
    <xs:element name="fluent" type="fluentType" />
    <xs:element name="timeVar" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="initiatesType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="ic_term" type="icTermType" />
      <xs:element name="ir_term" type="irTermType" />
      <xs:element name="rc_term" type="rcTermType" />
      <xs:element name="as_term" type="asTermType" />
    </xs:choice>
    <xs:element name="fluent" type="fluentType" />
    <xs:element name="timeVar" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="happensType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="ic_term" type="icTermType" />
      <xs:element name="ir_term" type="irTermType" />
      <xs:element name="rc_term" type="rcTermType" />
      <xs:element name="re_term" type="reTermType" />
      <xs:element name="as_term" type="asTermType" />
    </xs:choice>
    <xs:element name="timeVar" type="timeVariableType" />
    <xs:element name="fromTime" type="TimeExpression" />
    <xs:element name="toTime" type="TimeExpression" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="clippedType">
  <xs:sequence>
    <xs:element name="timeVar1" type="timeVariableType" />
    <xs:element name="fluent" type="fluentType" />
    <xs:element name="timeVar2" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="declippedType">
  <xs:sequence>
    <xs:element name="timeVar1" type="timeVariableType" />
    <xs:element name="fluent" type="fluentType" />
    <xs:element name="timeVar2" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="terminatesType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="ic_term" type="icTermType" />
      <xs:element name="ir_term" type="irTermType" />
      <xs:element name="rc_term" type="rcTermType" />
      <xs:element name="as_term" type="asTermType" />
    </xs:choice>
    <xs:element name="fluent" type="fluentType" />
    <xs:element name="timeVar" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>

```

```

</xs:sequence>
</xs:complexType>
<xs:complexType name="fluentType">
  <xs:choice>
    <xs:sequence>
      <xs:element name="target">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="variable" type="variableType" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="source">
        <xs:complexType>
          <xs:choice>
            <xs:element name="variable" type="variableType" />
            <xs:element name="operationCall" type="operationCallType" />
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="variable" type="variableType" />
    </xs:sequence>
  </xs:choice>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="quantificationType">
  <xs:sequence>
    <xs:element name="quantifier">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="forall|existential" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:choice>
      <xs:element name="regularVariable" type="variableType" />
      <xs:element name="timeVariable" type="timeVariableType" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="icTermType">
  <xs:sequence>
    <xs:element name="operationName" type="xs:string" />
    <xs:element name="partnerName" type="xs:string" />
    <xs:element name="id" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="variable" type="variableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="irTermType">
  <xs:sequence>
    <xs:element name="operationName" type="xs:string" />
    <xs:element name="partnerName" type="xs:string" />
    <xs:element name="id" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="rcTermType">
  <xs:sequence>
    <xs:element name="operationName" type="xs:string" />
    <xs:element name="partnerName" type="xs:string" />
    <xs:element name="id" type="xs:string" />
  </xs:sequence>
</xs:complexType>

```

```

<xs:complexType name="reTermType">
  <xs:sequence>
    <xs:element name="operationName" type="xs:string" />
    <xs:element name="partnerName" type="xs:string" />
    <xs:element name="id" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="variable" type="variableType"
  />
</xs:sequence>
</xs:complexType>
<xs:complexType name="asTermType">
  <xs:sequence>
    <xs:element name="operationName" type="xs:string" />
    <xs:element name="id" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="variableType">
  <xs:sequence>
    <xs:element name="varName" type="xs:string" />
    <xs:choice>
      <xs:sequence>
        <xs:element name="varType" type="xs:string" />
        <xs:choice minOccurs="0">
          <xs:element name="value" type="xs:string" />
          <xs:element name="objectValue">
            <xs:complexType>
              <xs:sequence>
                <xs:any />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:sequence>
      <xs:element name="array" type="arrayType" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute default="false" name="persistent" type="xs:boolean" />
  <xs:attribute default="true" name="forMatching" type="xs:boolean" />
</xs:complexType>
<xs:complexType name="expresionType">
  <xs:sequence>
    <xs:element name="varName" type="xs:string" />
    <xs:choice>
      <xs:sequence>
        <xs:element name="varType" type="xs:string" />
        <xs:choice>
          <xs:element name="value" type="xs:string" />
          <xs:element name="fields" type="fieldType" />
        </xs:choice>
      </xs:sequence>
      <xs:element name="array" type="arrayType" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute default="false" name="persistent" type="xs:boolean" />
  <xs:attribute default="true" name="forMatching" type="xs:boolean" />
</xs:complexType>
<xs:complexType name="fieldType">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="field" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="timeVariableType">
  <xs:sequence>
    <xs:element name="varName" type="xs:string" />
    <xs:element fixed="TimeVariable" name="varType" type="xs:string" />
    <xs:element minOccurs="0" name="value" type="xs:string" />
  </xs:sequence>

```

```

</xs:sequence>
</xs:complexType>
<xs:simpleType name="logicalOperatorType">
  <xs:restriction base="xs:string">
    <xs:pattern value="and|or" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="TimeExpression">
  <xs:sequence>
    <xs:element name="time" type="timeVariableType" />
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element name="plusTime" type="timeVariableType" />
        <xs:element name="minusTime" type="timeVariableType" />
        <xs:element name="plus" type="xs:decimal" />
        <xs:element name="minus" type="xs:decimal" />
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="TimeRelation">
  <xs:sequence>
    <xs:element name="timeVar1" type="TimeExpression" />
    <xs:element name="timeVar2" type="TimeExpression" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="varRelationType">
  <xs:sequence>
    <xs:element name="operand1" type="operandType" />
    <xs:element name="operand2" type="operandType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="relationalPredicateType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="equal" type="varRelationType" />
      <xs:element name="notEqualTo" type="varRelationType" />
      <xs:element name="lessThan" type="varRelationType" />
      <xs:element name="greaterThan" type="varRelationType" />
      <xs:element name="lessThanEqualTo" type="varRelationType" />
      <xs:element name="greaterThanEqualTo" type="varRelationType" />
    </xs:choice>
    <xs:element name="timeVar" type="timeVariableType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="operandType">
  <xs:choice>
    <xs:element name="operationCall" type="operationCallType" />
    <xs:element name="expresion" type="expresionType" />
    <xs:element name="constant" type="constantType" />
  </xs:choice>
</xs:complexType>
<xs:complexType name="operationCallType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element minOccurs="0" name="partner" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="expresion"
type="expresionType" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="operationCall"
type="operationCallType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="constantType">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />

```



```
<xs:element name="value" type="xs:string" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="arrayType">
  <xs:sequence>
    <xs:element name="type" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="unbounded" name="value" type="arrayValueType"
  />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="arrayValueType">
  <xs:sequence>
    <xs:element name="indexValue" type="xs:string" />
    <xs:element name="cellValue" type="xs:string" />
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Appendix D. WSDL of the new API of the Monitor

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
= <wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/analyzerService"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/services/analyzerService"
xmlns:intf="http://localhost:8080/axis/services/analyzerService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  + <!-- WSDL created by Apache Axis version: 1.4
  Built on Apr 22, 2006 (06:55:48 PDT)
  -->
  + <wsdl:message name="checkRuleResponse">
  + <wsdl:message name="recordEventResponse">
  + <wsdl:message name="setRulesResponse">
  + <wsdl:message name="initialiseDiagnoserResponse">
  + <wsdl:message name="diagnoseRequest" />
  + <wsdl:message name="recordEventRequest">
  + <wsdl:message name="setRulesRequest">
  + <wsdl:message name="checkRuleRequest">
  + <wsdl:message name="initialiseDiagnoserRequest">
  + <wsdl:message name="diagnoseResponse">
  - <wsdl:portType name="NewDataAnalyzerEC">
  + <wsdl:operation name="setRules" parameterOrder="xmlRules">
  + <wsdl:operation name="initialiseDiagnoser" parameterOrder="xmlFailedTemplate">
  + <wsdl:operation name="diagnose">
  + <wsdl:operation name="recordEvent" parameterOrder="event">
  + <wsdl:operation name="checkRule" parameterOrder="formulaID">
  </wsdl:portType>
  + <wsdl:binding name="analyzerServiceSoapBinding" type="impl:NewDataAnalyzerEC">
  + <wsdl:service name="NewDataAnalyzerECService">
  </wsdl:definitions>
```

References

- [1] Kloukinas C., Ballas C., Presentza D., Spanoudakis G. (2006): “Basic set of Information Collection Mechanisms for Run-Time S&D Monitoring”, Deliverable A4.D2.2, SERENITY Project, http://www.serenity-forum.org/IMG/pdf/A4.D2.2_informationCollectionMechanism_v0.15_final_e.pdf.
- [2] Knight K. (1989): “Unification: a multidisciplinary survey”, ACM Computing Surveys, 21(1):93-124. Available from: <http://www.isi.edu/natural-language/people/unification-knight.pdf>
- [3] Mahbub K., Spanoudakis G., Kloukinas C. (2007): “V2 of dynamic validation prototype”. Deliverable A4.D3.3, SERENITY Project, Available from: [http://www.serenity-forum.org/IMG/pdf/A4.D3.3 - V2_of_Dynamic_validation_Prototype.pdf](http://www.serenity-forum.org/IMG/pdf/A4.D3.3_-_V2_of_Dynamic_validation_Prototype.pdf).
- [4] Shafer G. (1975): “A Mathematical Theory of Evidence”, Princeton University Press.
- [5] Shanahan, M. P. (1999): “The Event Calculus Explained”, In Artificial Intelligence Today, LNAI no. 1600:409-430, Springer.
- [6] Spanoudakis G., Tsigkritis T. (2008): “1st Version of Diagnosis Prototype”. Deliverable A4.D5.1, SERENITY Project, Available from: http://www.serenity-forum.org/IMG/pdf/A4.D5.1_first_version_of_diagnosis_prototype_v1.1_final.pdf
- [7] Tsigkritis T., Spanoudakis G. (2008): “Diagnosing Runtime Violations of Security and Dependability Properties”, Proceedings of 20th International Conference in Software Engineering and Knowledge Engineering, SA, USA
- [8] Tsigkritis T., Spanoudakis G. (2008): “A temporal abductive diagnosis process for runtime properties violations”, ECAI 2008 Workshop on Explanation Aware Computing