SIXTH FRAMEWORK PROGRAMME
PRIORITY 2
"Information Society Technologies"

Project acronym: DESEREC

Project full title: Dependability and Security by Enhanced Reconfigurability

Proposal/Contract no.: IST-2004-026600-DESEREC

Requirements for the fast self-healing module

Project Document Number: DESEREC/D4.1/PU/v1.3

Project Document Date: 06/10/2006

Task Contributing to the Project Document: T4.1, T4.2, T4.3

Deliverable Type and Security: R-PU

Document Editor : Ervin Toth (SEARCH)

Author(s): Jean-Etienne Goubard, Tim Hartog, Katarzyna Nowak, Łukasz Radosz, Marek Woda, M.A.Rónai, G. Szabó, B. Bencsáth, Ervin Toth, Sofoklis Efremidis, Christophe Colas, Luc Paffumi, Pedro Pérez

Abstract: This document sets out the requirements for the fast self-healing system. Models of the event monitoring, serious incident detection and fast reaction modules are presented along with explanations of the expected input and produced output. Both functional and non-functional requirements are listed and categorized.

Keywords: self-healing, event monitoring, incident detection, fast reaction, functional requirements, non-functional requirements.

History

Version	Date	Description, Author(s), Reviewer(s)
1.3	6/10/2006	Version approved by PMT

Executive Summary

The DESEREC project aims to increase the dependability of existing and new networked mission-critical Information Systems (IS) with a Business Services point of view.

The main levers are

- Detect as proactively as possible the incidents and contain them in a limited area of the I.S.
- When an incident cause a failure, handle it smoothly if not seamlessly.
  - ✓ Reconfigure the subpart involved on the same resources or on additional or substituted ones;
  - ✓ When no resource is available to resume performance of a high-priority service in accordance with the business rules, take the risk to stop low-priority services to free needed resources.
- Use a common model of the Information System to build alternate configurations for forecasted under-nominal modes and validate them through simulation, to analyse the events at various levels, to take the decision to react, and to design the response.

To build the DESEREC framework we propose a three-tiered approach.

- 1) Planning of optimal configuration for anticipated operational modes
  - ✓ Modelling of the networked Information systems, validate resilience and performance of the various configurations with simulation
    - Several minutes to deploy policies for an alternate configuration
    - Some hours to define a reliable, deployable and optimised planning (set of configurations rules) and deploy it.
- 2) Fast reconfiguration with priority to critical activities
  - ✓ Online administrator assistance for IS Reconfiguration, topology
  - ✓ Sustain or resume partial operations in less than a minute
- 3) Incident detection and quick containment: self-healing
  - ✓ Detection and response
  - ✓ Reconfigure or cut off the area under suspicion within a few seconds

This document is the first step towards the definition and development of the mechanisms for the third tier (self-healing) to fast detection of incident and fast reaction at local level on the system.

We separated three functions:

- *Event Monitoring* collects monitoring data from multiple sources, aggregates and filters these data and finally converts them into normalized events for other DESEREC functions.
- *Serious Incident Detection* analyses normalized events and identifies serious incidents that would require immediate reaction.
- Upon detection of serious incident, the *Fast Reaction* function verifies if any reaction action has been defined for it and applies this reaction to the IS as an immediate counter-measure.

This document is the D4.1 deliverable of the DESEREC project. This deliverable includes the model and interface description, as well as the formulated requirements, of the fast self-healing system.

First an overall presentation of the project and the fast self-healing module is introduced. Then the methodology used to identify requirements is presented.

The third section presents the model of the event monitoring, serious incident detection and fast reaction components.

The fourth section describes the detailed requirements formulated for each component.

The fifth and sixth sections describe the internal and external interfaces of the fast self-healing module.

The seventh section contains a summary of all requirements with their testing method.

Contents

Figures

1 Introduction

1.1 Situation of this document

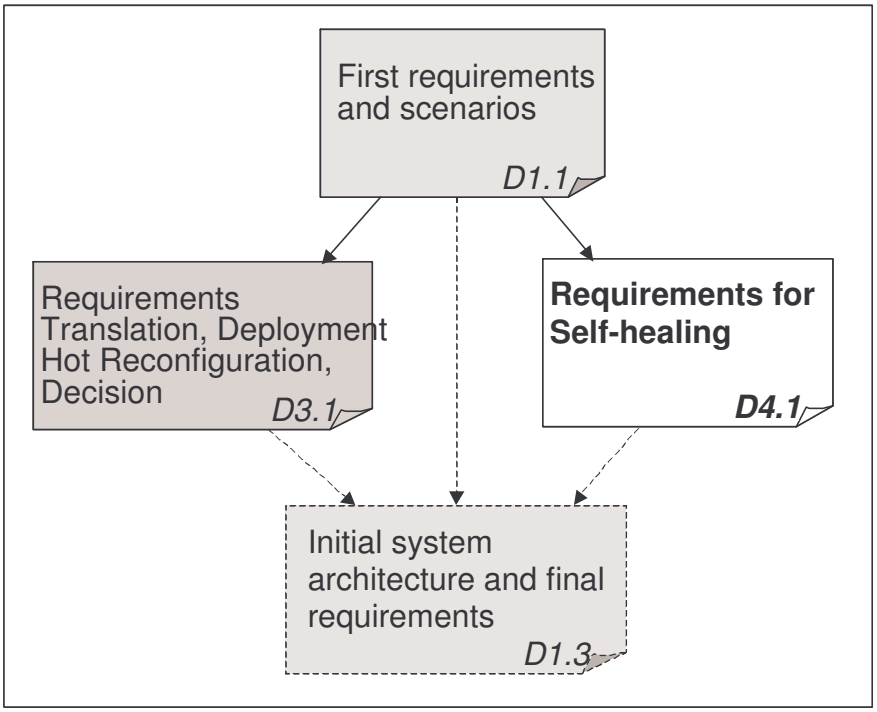

Figure 1. Documentation plan of requirements

Figure 1 presents the situation of this document within the flow of production of the requirements documents down to the Architecture.

***Nota bene**: this deliverable was entitled "Requirements for the fast cicatrisation module" in the Technical Annex of the contract. It has been changed for a better understanding.*

1.2 Purpose of this document

The main goal of this document is to put requirements on the Self-healing tier of the DESEREC framework. As stated in the technical annex of the project, the main modules are Event Monitoring, Serious Incident Detection and Fast Reaction.

1.3 Reminder on DESEREC

1.3.1 Objectives

The approach within the DESEREC project for increasing system dependability comprehends three different areas: modelling and simulation, detection and response, configuration and reconfiguration.

A very innovative aspect of the DESEREC project entails the reaction and response time variables concerning detection and reaction. Specifically, DESEREC distinguishes three different timescales:

- Fast reaction, in the order of seconds after incident detection, focused on ensuring efficient and fast detection techniques for severe incidents and providing means to launch immediate and automated reaction for such kind of incidents. The overall functionality (consisting of monitoring, detection and fast reaction processes) will be referred to by the term "fast self-healing" (it is also called "fast cicatrisation" in the technical annex of the project).
- Reconfiguration could takes some minutes when a suitable configuration is ready to apply to hours when an optimal one is build with the planning and simulation tools to fit a non-forecasted system state.

The *Fast self-healing* module is focused on providing the basic conceptual and technical tools for implementing incident detection and fast reaction. Fast self-healing corresponds to a local response to the incident that includes actions like stopping an application, restarting a computer, blocking a WAN port, to recover the system into a safe state.

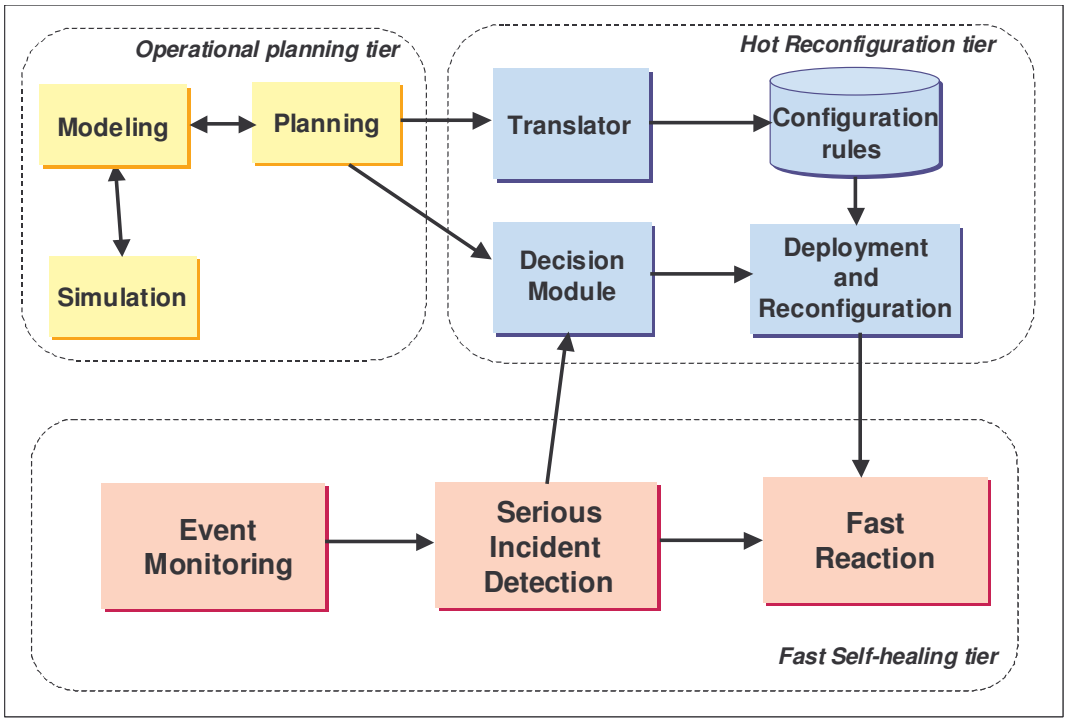

Figure 2. Links between the three tiers

As the priority for the *Fast self-healing* module is being reactive, it has neither the time nor the scope of information to detect the side-effects at the system level of its own actions and, thus, have to report to the *Hot Reconfiguration* tier the incident(s) detected and the action(s) taken in response. Should the dependability of the system is affected under the committed threshold the *Hot Reconfiguration* tier with its *Decision* module will launch the appropriate reconfiguration action to resume the performance of the Information system in accordance with the targeted level of dependability.

Surveillance and monitoring, information gathering and fast event analysis processing are the main tools for addressing incident detection and fast reaction. To perform its task smartly, the *Fast Self-healing* module must carefully qualify the incidents with a

very low percentage of false positive as the reaction aims avoiding escalation and diffusion of incidents and should not generate useless reconfiguration processes.

1.3.2 Functions & modules

Event monitoring, serious incident detection and fast reaction are three main functionalities of the DESEREC *Fast Self-healing* module as depicted in Figure 3.

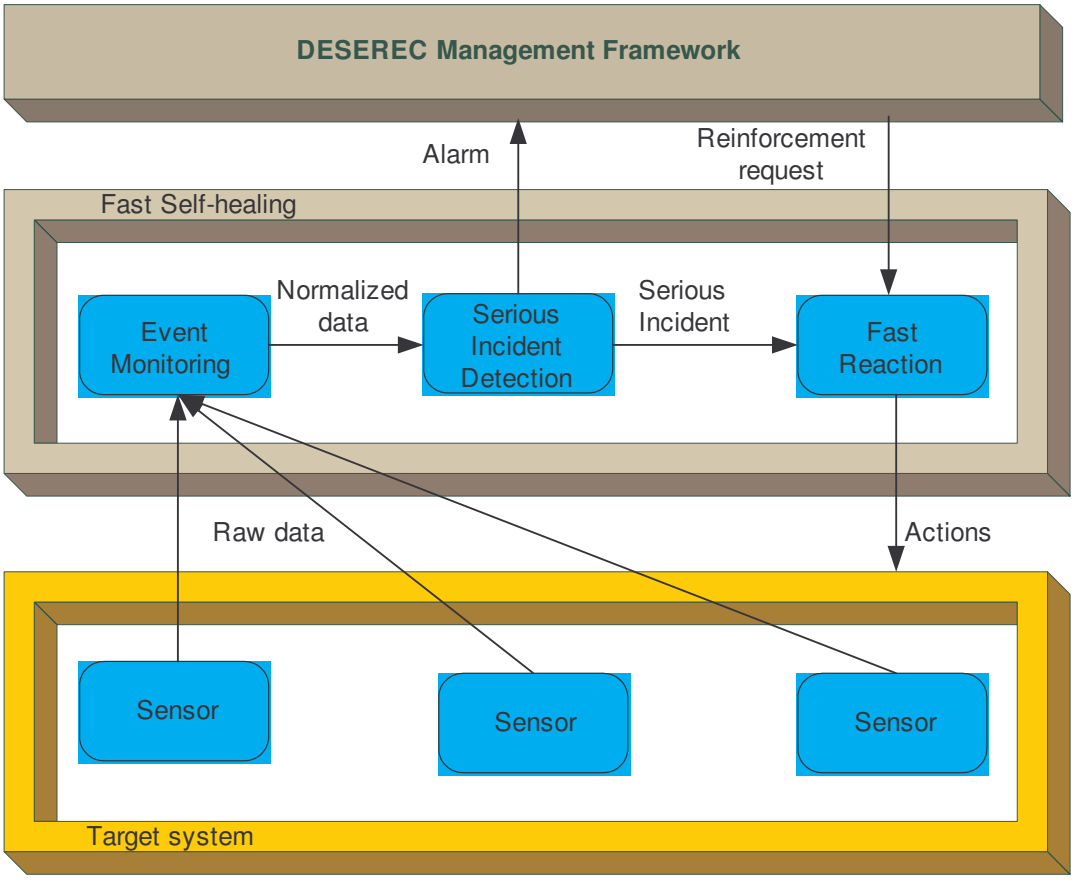

Figure 3. Fast Self-healing module Overview

Please note that, as at time of writing the design of the framework being under process, the atomic level of monitoring and reconfiguration is named with the generic terms "Subpart" or "Component".

1.4 General Constraints

These three functions have to deal with the following constraints:

- The fast reaction shall be executed automatically and as soon as possible after the detection of an incident. Therefore human interaction cannot take place in this process; however, the human administrator must be alerted with high priority.
- To detect serious incident, the analysis of events needs to be very fast and non ambiguous.
- Only upon well-characterised incidents shall be reacted.

- Strong mechanisms must be provided to avoid intrusion.
- Normal system behaviour must not be affected during monitoring.

1.4.1 Work packages

Figure 4 presents the work package breakdown of DESEREC.

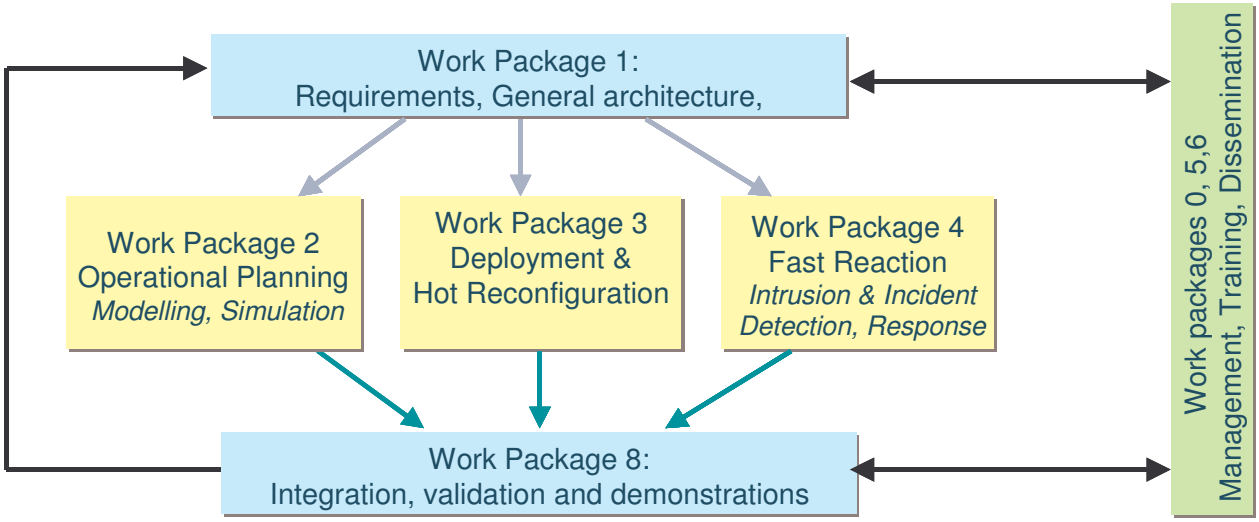

Figure 4. The Work Package breakdown of the DESEREC project

2 Methodology and Notation

2.1.1 Guidelines for the Identification of Requirements

2.1.1.1 Definitions

Fast Self-healing module has considered system capabilities and requirements:

- **Capabilities**, which represent a class of common characteristics or features that a system has to provide, for instance security, privacy, …
- **Requirements**, which are linked to system capabilities, and provide an additional level of detail.

There are two types of requirements, technical and non-technical:

- **Technical Requirements** describe both the function of the system and the operational constraints within which this function is performed and include functional, performance, interface and quality requirements,
- **Non-Technical Requirements** cover other aspects such as contractual understandings, conditions and/or clauses, rules, ….

2.1.1.2 Characteristics of Requirements

Requirements should describe desired characteristics of the specified system functionality. It illustrates a characteristic of the problem not a solution. Solution will be studied later during the design phase. The major qualities of a requirement are:

- **Simple**: A requirement must have an elementary structure and state a basic need. It cannot, at a given level, be broken down into several simpler requirements. At system level, a simple requirement can be broken down into several requirements at Component level.
- **Concise**: A requirement must be expressed in a brief and clear manner. It contains neither an explanation nor a justification.
- **Unambiguous**: A requirement must have only one possible interpretation. A requirement is said ambiguous if it can be semantically interpreted in several ways, implying an uncertainty with regard to the design to be elaborated.
- **Verifiable**: As much as possible, it should be possible to verify requirements: if a test can not be associated, the validity of that requirement should be further investigated, requirements should be testable either by Inspection or Demonstration or Analysis or Test points, requirement can be refined in various (testable) sub-requirements to be testable, an effective finite procedure must make it possible to check that the system complies to the requirement(s).
- **Feasible**: It should be considered if a realistic and satisfactory technical solution can be implemented or can be elaborated in the timeframe of the project.
- **Non-redundant**: A requirement must have no overlap with another requirement. In case of redundancy, requirements must be divided and refined until suppression of the overlap.
- **Non-incompatible**: No conflict with another requirement (see above).
- **Classifiable**: It can be associated with an attribute belonging to a previously adopted classification system.
- **Necessary**: A requirement reflects a need.
- **Traceable**: Identifiable by a unique identifier.

2.1.1.3 Identification of a Requirement

Each requirement is identified in the following way:

Capability_Requirement_XX [Priority/TestMethod]

where **Capability** is one word dedicated to express the corresponding capability, and **Requirement** is one word dedicated to express the corresponding requirement; **XX** is a sequential number used to differentiate requirements with the same capability and requirement if more than one. If a section only contains one capability, then the name of the section will be used as the capability identification. **Priority** and **Test Method** information are also included for each requirement, according to the classification included in section 7.

Some explanations, descriptions, examples, and schemas are added to improve the understanding of the capability or the requirement and support the definition.

2.1.1.4 Levels of Priority

Three levels of priority are defined for the requirements: mandatory, recommended or optional. These levels are indicated by the key words SHALL, SHOULD and MAY respectively. The meaning of the key-words is taken from RFC-2119 and quoted below:

- **SHALL** means "that the definition is an absolute requirement of the specification".
- **SHOULD** means "that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course."
- **MAY** means "that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item."

2.1.2 Classification of the Requirements

Based on the definitions given, the requirements may be grouped in a table to indicate the relevant information:

- Priority: level of priority
  - M stands for Mandatory (SHALL),
  - O stands for Optional (MAY),
  - R stands for Recommended (SHOULD).
- Test Method: anticipated test method for the validation of the requirement:
  - I stands for Inspection (compliance with requirements is determined by formal examination; think of a building inspection for compliance with the local code),
  - A stands for Analysis (compliance with requirements is determined by models or by interpreting results using established principles; for example, a design's performance might be analyzed by running simulations),
  - D stands for Demonstration (compliance with requirements is validated by observing the item in operation),
  - T stands for Test Point (compliance with requirements is validated by evaluating or executing an item under controlled conditions, configurations, and inputs in order to observe the response. Results are quantified and analyzed).

Identification	Priority	Test method
REQ_Collection_Patterns_01	M	Demonstration
REQ_Collection_Patterns_02	O	Analysis
REQ_Collection_Events_01	M	Demonstration
REQ_Collection_Events_02	M	Demonstration

3 Objectives and Model description

This section contains the logical models and descriptions of the modules making up the fast self-healing functions as well as data flows between these modules (external, for upper level functions of DESEREC, and internal) are also included.

3.1.1 Event Monitoring Model Description

The Event Monitoring module (EV) will gather and aggregate relevant information from the target system using data collectors. This information (raw data) will be filtered and normalised into a common format, generating events. Such events will be sent to the Serious Incident Detection (SID) in order to detect incidents (Figure 2).

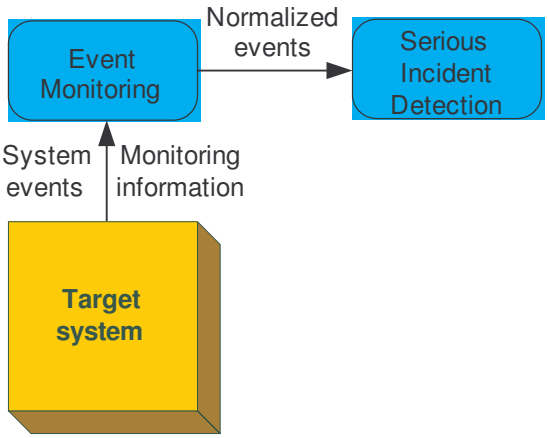

Figure 5: Data flow between the Event Monitoring Module and its related modules.

The Event Monitoring module uses data collectors to obtain information on the monitored system elements (hardware, software, network etc..). This information (raw data) is filtered and, if required, aggregated prior to its storage in an internal repository as events. The delivery of the normalised events in a standardized format is performed by an internal element, also responsible for the communication with the Serious Incident Detection module (Figure 3).

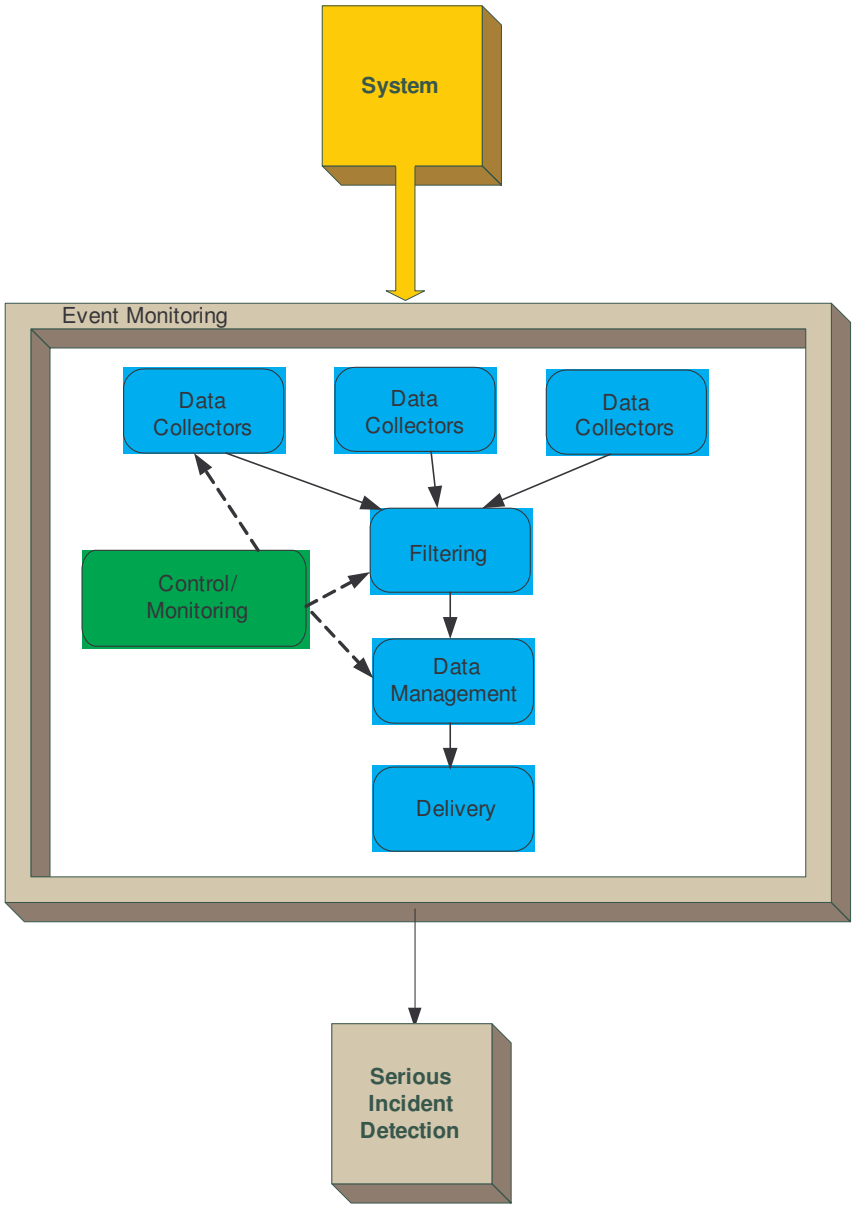

Figure 6: Event Monitoring Logical Model

3.1.2 Serious Incident Detection Model Description

The Serious Incident Detection (SID) module is a correlation and detection module. It receives normalized events from the Event Monitoring module. The SID module shall be used to find relations between these events by correlating them on certain properties. This will result in the detection of incidents. An incident is an event, or series of events, which are not part of the standard operation of a service and which causes, or may cause, an interruption to, or reduction in, the quality of that service. Examples are system failures, service failures, system misuse or attacks on certain systems. These incidents will cause the SID module to send alarms to the Fast Reaction Module where action(s) can be taken. An alarm is a notification that an incident has occurred.

Figure 7 shows a possible communication path between the different modules which are/can be related to the SID module.

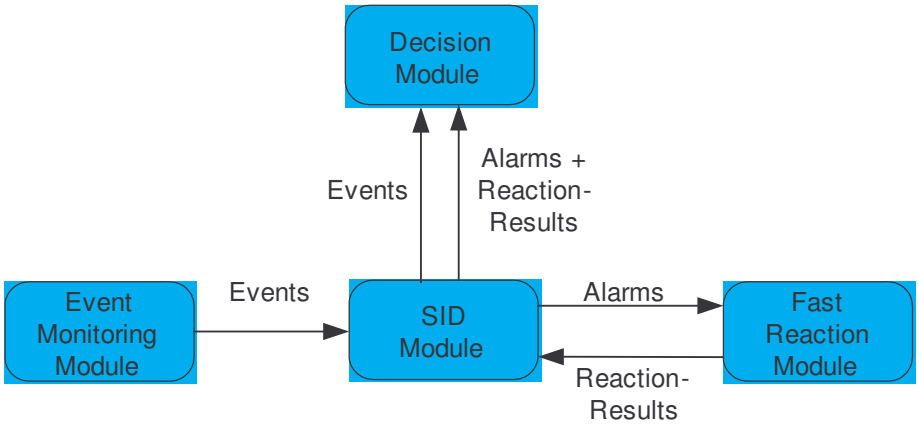

Figure 7: Data flow between the Serious Incident Module and its related modules.

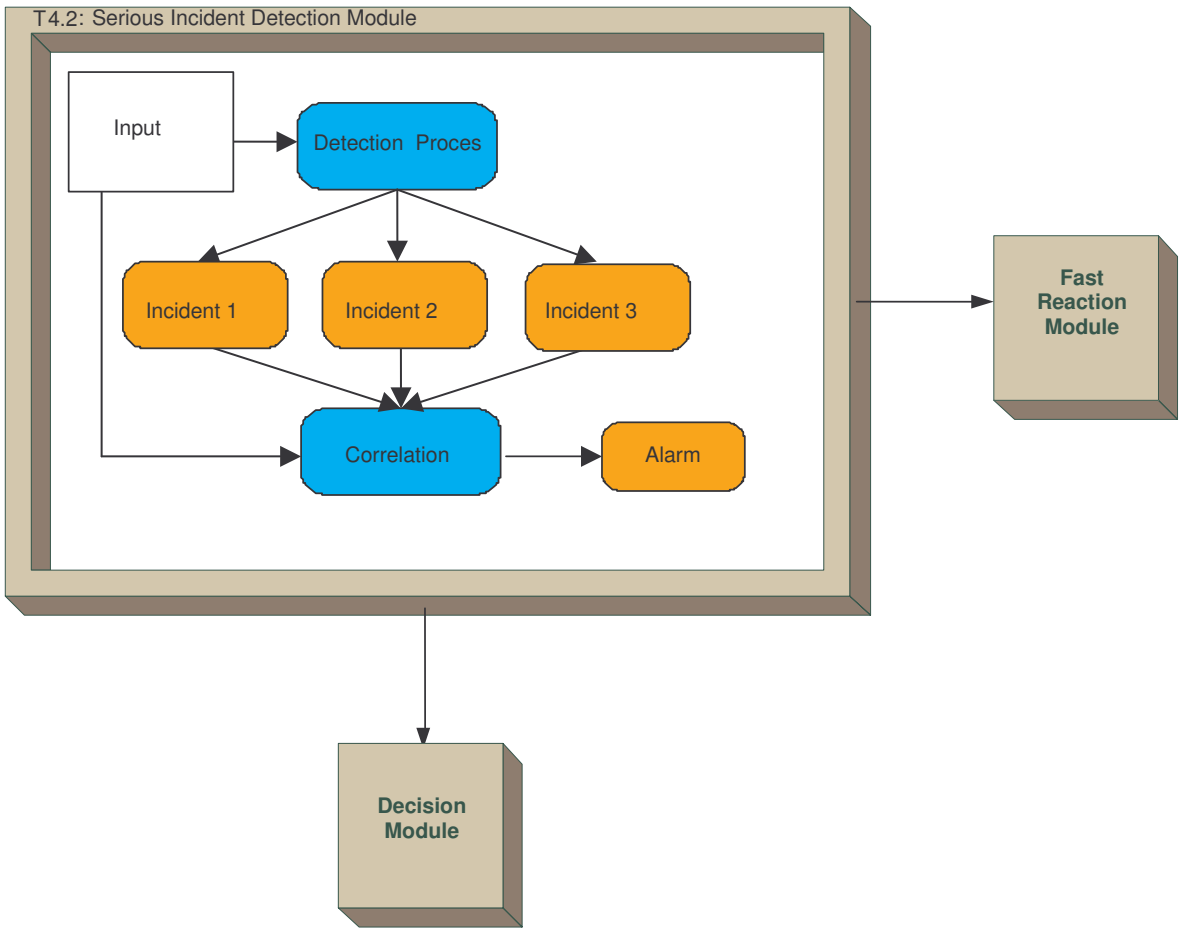

Figure 8: Serious Incident Detection Module - Logical Model

The Serious Incident Detection module will perform local analysis of normalized events received from the Event Monitoring module. The timeframe window for detection and correlation are small and the correlation rules are kept simple. This way the alarms that are sent to the Fast Reaction module, resulting from the detection of a serious incident, will trigger a reaction as fast as possible. As a result the incidents are kept local and cascading effects are prevented.

3.1.3 Fast Reaction Module Model Description

The Fast Reaction module (FRM) is an intervention module. It receives alarms from the Serious Incident Detection (SID) module, evaluates the possible reactions and deploys them. Furthermore, its reactions shall be executed in synchronization with the ones decided by the Hot Reconfiguration module. The aim of the FRM is to deploy reconfigurations to the appropriate components of the system administered by DESEREC in order to cut the possibility of incident propagation, thus locally limiting the impact of the detected incident. Figure 9 shows the possible communication paths between the different modules which are be related to the FRM.

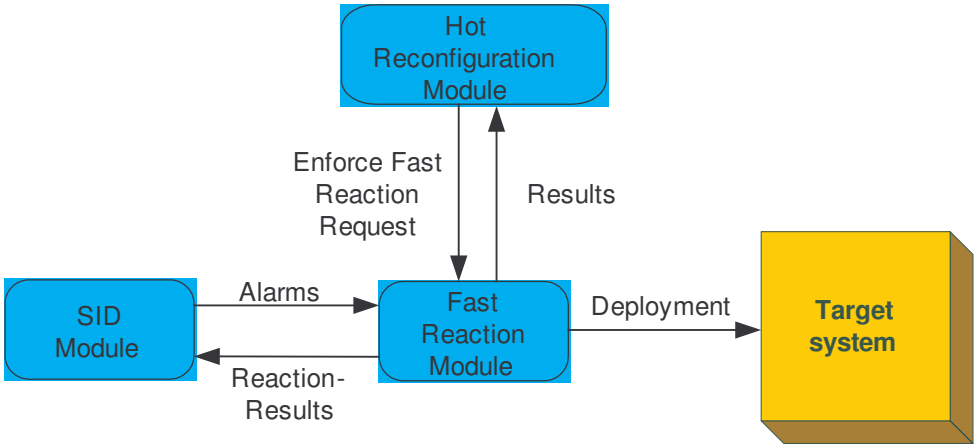

Figure 9: Data flow between the Fast Reaction Module and its related modules.

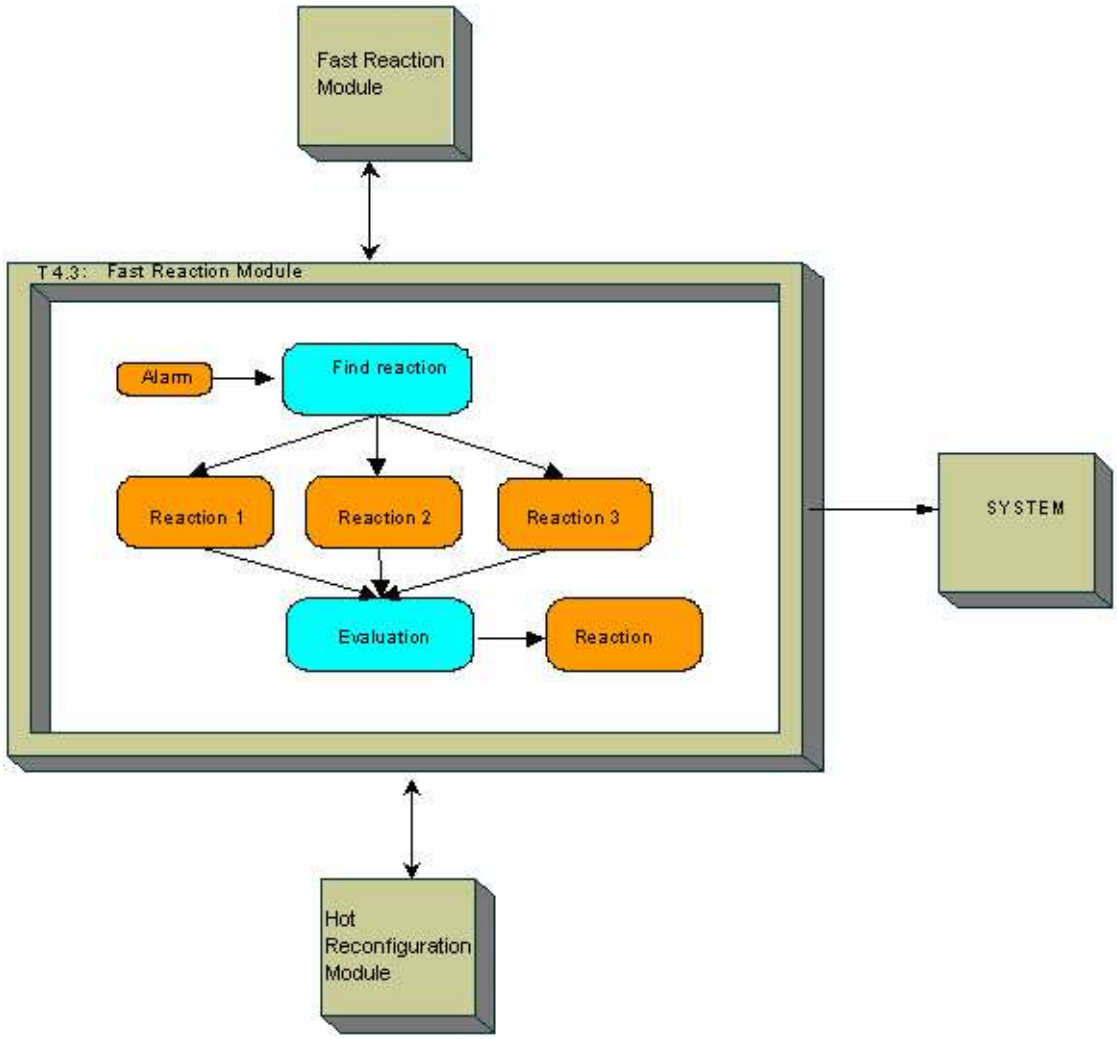

Figure 10: Fast Reaction Module - Logical Model

4 Specific Requirements

This chapter describes the requirements for the three modules within the fast self-healing functionality. Both functional and non-functional requirements are described.

4.1 Event Monitoring

4.1.1 Functional Requirements

4.1.1.1 Event Data Collection

The functionality required collecting data from data sources.

Capability: **Collection**

Requirement identification: **Sources**

- **REQ_Collection_Sources_01[M/D]**: The Event monitoring module SHALL be able to collect information from different data sources.

  Note: Data sources will be software application logs, Simple Network Management Protocol (SNMP) traps, syslog, information from different monitoring devices (Intrusion Detection systems (IDS), Network monitoring system), ...

- **REQ_Collection_Sources_02[M/D]**: The Event monitoring module SHALL have an inventory of monitored sources.
- **REQ_Collection_Sources_03[M/D]**: The Event monitoring module SHALL be able to create events from data collected from data sources.

Requirement identification: **Patterns**

- **REQ_Collection_Patterns_01[M/D]**: The Event monitoring module SHALL be able to use predefined patterns from which events should be created.

  Note: These patterns should be manually defined.

- **REQ_Collection_Patterns_02[O/A]**: The Event monitoring module MAY be able to propose event patterns based on data collected from data sources.

Requirement identification: **Events**

- **REQ_Collection_Events_01[M/D]**: The Event monitoring module SHALL be able to normalize events created from data collected from data sources.
- **REQ_Collection_Events_02[R/D]**: All the collected events SHOULD be characterized by a number of fixed attributes.
- **REQ_Collection_Events_03[R/D]**: It SHOULD be possible to define dynamic attributes to events depending on the event type.
- **REQ_Collection_Events_04[R/D]**: The event monitoring module SHOULD give value to fixed and dynamic attributes when the event is created.
- **REQ_Collection_Events_05[M/D]**: The Event monitoring module SHALL use a mechanism to ensure the synchronization of the events creation time.

Requirement identification: **Store**

- **REQ_Collection_Store_01[M/I]**: The event monitoring module SHALL keep a repository that holds all events that took place in the last pre-defined number of (configurable parameter) minutes.

4.1.1.2 Filtering

Describes low-level filtering functionality that applies to collected raw data.

Capability: **Filtering**

Requirement identification: **Events**

- **REQ_Filtering_Events_01[M/A]**: The Event monitoring module SHALL screen for only crucial events.
- **REQ_Filtering_Events_02[M/A]**: The crucial events SHALL be divided into critical (alerts) and vital (warnings).
- **REQ_Filtering_Events_03[M/T]**: The Event monitoring module SHALL be able to filter/aggregate the collected data.
- **REQ_Filtering_Events_04[M/D]**: Collecting data filtering SHOULD be configurable.
- **REQ_Filtering_Events_05[M/D]**: The Event monitoring module SHALL be able to filter duplicated events generated in predefined time windows.

4.1.1.3 Monitoring and Control

Configuration, reporting and status monitoring requirements

Capability: **Monitoring**

Requirement identification: **Patterns**

- **REQ_Monitoring_Patterns_01[R/I]**: The Event monitoring module SHOULD provide an interface to define or modify event patterns.

  Note: A pattern describes how an event matches into a predefined event category or type.

Requirement identification: **Events**

- **REQ_Monitoring_Events_01[R/I]**: The Event monitoring module SHOULD provide an interface to define or modify types of events.
- **REQ_Monitoring_Events_02[M/I]**: The Event monitoring module SHALL provide an interface to define or modify fixed or dynamic attributes.
- **REQ_Monitoring_Events_03[R/I]**: The Event monitoring module SHOULD provide an interface to monitor the events collected in real time along with their attributes.

Requirement identification: **Reports**

- **REQ_Monitoring_Reports_01[R/I]**: The Event monitoring module SHOULD provide reports with information of evolution of events over time.
- **REQ_Monitoring_Reports_02[O/I]**: The Event monitoring module MAY provide an interface to configure the reports based on some event attributes.

Requirement identification: **Statistics**

- **REQ_Monitoring_Statistics_01[R/I]**: The Event monitoring module SHOULD provide statistical information about the collected events.

4.1.1.4 Data Management

Data storage and retrieval.

Capability: **Data Management**

Requirement identification: **Input**

- **REQ_Data_Management_Input_01[M/D]**: The Event monitoring module SHALL accept raw events data.

Requirement identification: **Storage**

- **REQ_Data_Management_Storage_01[M/D]:** Events SHALL be stored for a pre-defined (configurable) period
- **REQ_Data_Management_Storage_02[R/I]:** Events SHOULD be available for post analysis
- **REQ_Data_Management_Storage_03[O/I]:** Non-crucial events MAY be ignored and stored at the source of the raw events data.

Requirement identification: **Output**

- **REQ_Data_Management_Output_01[M/A]:** The Events SHALL be converted into a standard format

  Note: IODEF and IDMEF should be considered as the basis for elaborating such standard format.

4.1.1.5 Event Delivery

Requirements dealing with event provisioning to other modules.

Capability: **Delivery**

Requirement identification: **EventManagement**

- **REQ_Delivery_EventManagement_01[R/A]**: The Event monitoring module SHOULD be able to distinguish between events needed by the Serious Incident Detection module or Decision module.
- **REQ_Delivery_EventManagement_02[R/A]**: It SHOULD be possible to define different filter or aggregation rules for events depending on the receiver.

4.1.2 Non-Functional Requirements

4.1.2.1 Performance Requirements

- **REQ_Performance-HandleEvents[R/I]:** The event monitoring module SHOULD be able to handle events concurrently in real time.
- **REQ_Performance-StoreSystemStatus[M/I]:** Monitored system element status SHALL be stored with the event.

  Note: This way at a later moment it can be checked why a given event/incident occurred.

- **REQ_Performance-EventsAggregation[R/I]:** The sensors SHOULD be able to aggregate events to make processing faster and to avoid network and node overload.
- **REQ_Performance-EventsOverload[R/I]:** There SHOULD be a mechanism to tune the behaviour of event monitoring functionality.

- **REQ_Performance-RealTime[R/I]:** The event monitoring module SHOULD perform the pre-processing of raw data from monitored systems in less than 20 seconds.
- **REQ_Performance-EventProcessing[R/I]** Every event SHOULD be processed completely by the module within 30 seconds.

4.2 Serious Incident Detection module

4.2.1 Functional Requirements

4.2.1.1 Detection

This section defines the requirements for the detection of serious incidents.

- **REQ_Detection_Algorithm[M/A]:** There SHALL be a detection engine which executes detection algorithms on the received input.
- **REQ_Detection_Incident_Types[R/A]:** The detection engine SHOULD have a list of vulnerabilities of the components, which fall under the incident detection process of this detector.
- **REQ_Detection_Incident_Confidence_Value[O/D]:** A confidence value MAY be assigned to the detected incidents.
- **REQ_Detection_Fast_Reaction_Trigger[M/A]:** If a serious incident has been detected, an alarm SHALL be sent to the fast reaction module.
- **REQ_Detection_Logging[M/A]:** Logs of detected incidents and logs of alarms sent to the Fast Reaction module SHALL be maintained.

4.2.1.2 Correlation

This section defines requirements concerning the correlation of events in order to detect serious incidents.

- **REQ_Correlation_Rules[M/D]:** The serious incident detection module SHALL maintain appropriate sets of rules that will allow it to correlate monitored events and draw conclusions on the occurrence(s) of incidents.
- **REQ_Correlation_Notification[M/D]:** The serious incident detection module SHALL send an alarm to the fast reaction module if a serious incident has been detected by the correlation of events.

4.2.2 Non-Functional Requirements

4.2.2.1 Performance Requirements

This section defines the requirements concerning the performance of the serious incidents detection process.

- **REQ_Performance_False_Positive[R/I]:** the number of false positive alarms and incidents SHOULD be kept low.
- **REQ_Performance_False_Negative[R/I]:** the number of false negative alarms and incidents SHOULD be kept low.
- **REQ_Performance_Time_01[R/I]:** The detection module SHALL detect incidents within a few seconds.
- **REQ_Performance_Time_02[R/I]:** Incident detection SHOULD not exceed 30 seconds/minutes.

- **REQ_Performance_Availability[R/I]:** The detection module SHALL exhibit a high level of availability.

4.2.2.2 Security Requirements

This section defines the requirements concerning the security of the serious incident detection module and it's communication paths.

- **REQ_Security_ClockSync[M/I]:** The events which are sent to the detection module SHALL be included with reliable timestamps.

4.2.2.3 Other Requirements

This section defines all other requirements.

- **REQ_Configuration_Sens[M/D]:** The serious incident detection module SHALL be able to be configured for operation at various sensitivity levels.
  Note: Sensitivity with respect to the drawing of conclusions for alerting for incidents when monitored events are correlated.

- **REQ_Configuration_Enable[M/D]:** There SHALL exist a way for the operator to modify/configure the detection process, including being able to enable/disable detection for a specific kind of incident.
- **REQ_Configuration_Threshold[O/I]:** The Operator MAY be able to customize the confidence value threshold which regulates the flow of alarms to the Fast Reaction module.
  Note: this way the operator can define the confidence value needed for an incident. Confidence values can/may be use judge if the alarm should be sent to the Fast Reaction module. This may be used to minimize the number of false positives.

- **REQ_Configuration_Finetune[O/I]:** The detection process MAY be tested and fine-tuned using stimulated data.
- **REQ_Investigation_Information[R/A]:** The detection module SHOULD be able to provide the information that leads to an alarm.
  Note: when the operator found a false positive, he would like to be able to avoid it in the future.

4.3 Fast reaction

4.3.1 Functional requirements

4.3.1.1 Decision

The functionalities required taking a fast reaction decision. The user/operator of DESEREC has created rules to map alarms from the Serious Incident Detection Module to specific reactions. This decision process entails checking if there is a mapping between the received alarm and the predefined reactions. If such a mapping exists a sanity check should be executed to check if the predefined reaction is possible.

- **REQ_Decision_FindReaction[M/D]:** There SHALL exist a mechanism to retrieve a list of reactions, corresponding to an alarm.
- **REQ_Decision_EvaluateReaction[M/D]:** There SHALL exist a mechanism to check the applicability of the reaction in the context of the subject of the received alarm.

4.3.1.2 Pre-configuration

The requirements listed in this section deal with the functionalities needed to allow the execution of a fast reaction.

- **REQ_Preconfiguration_ReactionsPermissions[M/D]:** In case of fast reactions applied by script execution, the scripts SHALL have enough access permissions to be executed on target systems.

  Note: It implies the target local system pre-configuration in order to assign the appropriate user / folder / file access rights that allow the scripts be executed.
  In case of fast reactions applied by SNMP write *commands execution, SNMP community strings SHALL be made known to SNMP manager (server).*

4.3.1.3 Reaction Management

The requirements listed in this section deal with the general management of the overall reaction process.

- **REQ_Management_Initiate[R/A]:** The operator SHOULD be able to initiate a reconfiguration manually.
- **REQ_Management_ResultNotification[M/D]:** A mechanism SHALL exist to notify an entity of the results of the fast reaction process it triggered.
- **REQ_Management_Automatic[M/I]:** The reaction process SHALL be fully automatic during runtime.
- **REQ_Management_Atomicity[M/A]:** The reaction process SHALL be executed in atomic steps.

  Note : "Atomic" in this context means that an action of the reaction process, cannot be interrupted, nor cancelled during its execution. It is the same kind of atomicity as for the databases transaction.

- **REQ_Management_Priority[M/D]:** There SHALL exist a mechanism allowing to assign a priority to the pending fast reaction requests.
- **REQ_Management_NoInterferences[O/D]:** There MAY exist a mechanism allowing to automatically cancel pending requests with less priority that would interfere with a higher priority fast reaction.

4.3.1.4 Enforcement

The requirements listed in this section deal with the fast reaction enforcement on a target device.

- **REQ_Enforcement_Transparent[R/D]**: The reaction execution SHOULD be transparent to the IS user.

  Note : In this context, "transparent" means that during the reaction process, the Information system User won't be notified of the progress of the fast reaction, and won't be aware of it. He just may notice the border effect changes.

- **REQ_Enforcement_Deployment_01[M/D]**: There SHALL exist a mechanism allowing deploying target system component's specific configuration.
- **REQ_Enforcement_Deployment_02[M/D]**: There SHALL exist a mechanism allowing ensuring that the configuration is correctly applied.
- **REQ_Enforcement_Deployment_03[R/A]**: There SHOULD exist a mechanism allowing to rollback the concerned part of the system after a fast reaction occurred or at minimum re-initialise it to a known configuration.

4.3.1.5 Reporting

The requirements listed in this section describe the mechanisms used to send reports to external entities.

- **REQ_Reporting_System[R/D]:** the reporting engine SHOULD be able to list the elements and components under control of the system.
- **REQ_Reporting_SystemConfiguration[R/D]:** the reporting engine SHOULD be able to list the different configuration parameters of the system.
- **REQ_Reporting_SystemReconfiguration[M/D]:** the reporting engine SHALL be able to list all the configuration changes applied to the system.
- **REQ_Reporting_Scripts[M/D]:** the reporting engine SHALL be able to list all the fast reaction scripts likely to be enforced on the system.
- **REQ_Reporting_Reactions[M/D]:** the reporting engine SHALL be able to list the past applied reactions, with their result, the related alarm information, the target system part information, the starting time of the fast reaction and the end time of the reaction processing.

  Note: action result is mainly (success or failure)

4.3.2 Non-Functional Requirements

4.3.2.1 Performance Requirements

- **REQ_Performance_DeploymentTime[M/A]**: The fast reaction module SHALL deploy the triggered reaction as fast as possible (less than a minute).
- **REQ_Performances_ReactionTime[M/A]**: The time from the incident detection report to the end of the reaction processing SHALL be less than a few minutes.
- **REQ_Performance_ReactionScope[R/A]**: The reaction process SHOULD only have effects at the target's local system level, avoiding to affect other subsystems.

  Note: If an action that is supposed to be a fast reaction, can affect other subsystems, it may be necessary to analyze it and to decide at a higher level (hot reconfiguration module)

4.3.2.2 Security Requirements

- **REQ_Security_AuthenticationRequester[M/D]**: Reaction requester SHALL be authenticated
- **REQ_Security_IdentificationRequester[M/I]**: In the reaction process, all parties involved SHALL be able to identify themselves to each other.
- **REQ_Security_AuthorizationRequester[M/I]**: The agent SHALL be authorized to perform the reaction process.

- **REQ_Security_PrivacyRequester[M/I]**: Stored information used for fast reaction SHALL be protected from unauthorized access.
  Note: This requirement mainly concerns scripts and configuration files.

4.4 Other non-functional requirements

4.4.1 Security Requirements

- **REQ_Security-Authentication[M/I]:** Data source and destination SHALL be securely identified
- **REQ_Security-Integrity[M/I]**: There SHALL exist mechanisms to ensure the integrity of exchanged information.
- **REQ_Security-Privacy[M/I]**: The information stored in the modules SHALL be protected from unauthorized access
- **REQ_Security_Confidentiality[M/I]:** All communications with other Deserec subsystems, that are reached over the network and are not running on the same node, SHALL be encrypted and protected against replays.
- **REQ_Security_RightManagement[M/I]:** Only authenticated users and system elements that have the enough rights SHALL be able to change the configuration of the system.

5 Internal interfaces

This chapter enumerates the data to be exchanged between the three modules of the fast self-healing functionality. The functionality is provided by the following modules:

- Event Monitoring module (EV)
- Serious Incident Detection module (SID)
- Fast Reaction module (FRM)

5.1 EV - SID Interface

Capability: **EventMonitoring-SeriousIncidentDetection_Interface**

Requirement identification: **Information**

- **REQ_EV-SID_Interface_Information_01[M/D]**: Event monitoring module SHALL send normalized events to Serious Incident Detection module.
- **REQ_EV-SID_Interface_Information_02[M/D]**: The information provided by the event SHALL describe such event.

  Note: This information should contain the type, source, time, severity and description of the event at least.
- **REQ_EV-SID_Interface_Information_03[R/D]**: Similar events SHOULD be aggregated before send to Serious Incident Detection module.

Requirement identification: **Communication**

- **REQ_EV-SID_Interface_Communication_01[M/A]**: The Event monitoring module SHALL use a specific communication protocol to send events to the Serious Incident Detection module.
- **REQ_EV-SID_Interface_Communication_02[M/D]**: The Event Monitoring module SHALL provide a safe and reliable event transfer mechanism to the Serious Incident Detection module.
- **REQ_EV-SID_Interface_Communication_03[R/T]**: The Event monitoring module SHOULD keep the events available for the Serious Incident Detection module in case of communication interruption (limited to a few minutes).

5.2 FRM - SID Interface

Capability: **FRM-SID_Interface**

- **REQ_FRM-SID_Interface_Ask[M/D]**: The Serious Incident Detection module SHALL send an *alarm* information to the Fast reaction module, when detected.
- **REQ_FRM-SID_Interface_Result[M/D]**: The Fast reaction module SHALL send information describing the result of the fast reaction to the Serious Incident Detection module, in the case where it has been triggered by it.

  Note: This information may include (the reaction applied, the target system, the current configuration, the success, or cancel status)

- **REQ_FRM-SID_Interface_Priority[R/I**]: Priority levels SHOULD be included with a fast reaction request..

6 External interfaces

In this chapter the data to be exchanged between the fast self-healing system and the external modules are listed.

6.1 Deployment and Hot Reaction Interface

This section describes the interface between hot reaction and fast self-healing elements.

6.1.1 SID-Decision Module Interface

This section describes the requirements for Events and Alarms (Serious Incidents) formats, protocols and non-functional requirements of this external interface.
The Decision Module is responsible for the detection of a wide range of incidents and by providing clear elements helping the decision process and for making the right decision in order to cure the system by launching quickly appropriated reactions, which is done by the Hot Reconfiguration Module (called HRM).

Capability: **Decision-FastSelf-healing-Interface**

Requirement identification: **Decision-FastSelf-healing-Interface_Events**
- **REQ_Decision-FastSelf-healing-Interface_Events_01[M/I]**: The Decision module SHALL receive normalized events from the *Serious incident detection* module.
- **REQ_Decision-FastSelf-healing-Interface_Events_02[M/I]**: The events received SHALL contain information about the type of the event source.
- **REQ_Decision-FastSelf-healing-Interface_Events_03[M/I]**: The events received SHALL contain information about the type of the event data.
- **REQ_Decision-FastSelf-healing-Interface_Events_04[R/T]**: The Detection module SHOULD receive information about the source of events which performs the event detection.

Requirement identification: **Decision-FastSelf-healing-Interface_Alarms**
- **REQ_Decision-FastSelf-healing-Interface_Alarms_01[M/T]**: The Decision module SHALL receive alarms from the *Serious incident detection* module.
- **REQ_Decision-FastSelf-healing-Interface_Alarms_02[M/I]**: The alarms received SHALL contain information about the impact on the system.
- **REQ_Decision-FastSelf-healing-Interface_Alarms_03[M/I]**: The events received SHALL contain information about the alarm data.

Requirement identification: **Decision-FastSelf-healing-Interface_Communication**
- **REQ_Decision-FastSelf-healing-Interface_Communication_01[M/I]**: A communication protocol SHALL exist between Event and Serious incident detection modules with the Decision module.

6.1.2 FRM-HRM Interface
- **REQ_FRM-HRM_Interface_Ask[O/I]:** The Hot Reconfiguration Module MAY ask the Fast reaction module to enforce a fast reaction.

- **REQ_FRM-HRM_Interface_Result[O/A]**: The Fast reaction module MAY send some information describing the result of the fast reaction to the Hot Reconfiguration Module, when it has been triggered by it.

  Note: This may include (the reaction applied, the target system, the current configuration, the success, or cancel status)

- **REQ_FRM-HRM_Interface_Priority[O/I]**: There MAY exist a mechanism allowing Hot Reaction Module to request a specific level of priority for the enforcement of a fast reaction.

7 List of requirements

This section contains a summary of the requirements, as well as their priority and test method according to section 2.1.2

Identification	Priority	Test method
REQ_Collection_Sources_01	M	Demonstration
REQ_Collection_Sources_02	M	Demonstration
REQ_Collection_Sources_03	M	Demonstration
REQ_Collection_Patterns_01	M	Demonstration
REQ_Collection_Patterns_02	O	Analysis
REQ_Collection_Events_01	M	Demonstration
REQ_Collection_Events_02	R	Demonstration
REQ_Collection_Events_03	R	Demonstration
REQ_Collection_Events_04	R	Demonstration
REQ_Collection_Events_05	M	Demonstration
REQ_Collection_Store_01	M	Inspection
REQ_Filtering_Events_01	M	Analysis
REQ_Filtering_Events_02	M	Analysis
REQ_Filtering_Events_03	M	Test Point
REQ_Filtering_Events_04	M	Demonstration
REQ_Filtering_Events_05	M	Demonstration
REQ_Monitoring_Patterns_01	R	Inspection
REQ_Monitoring_Events_01	R	Inspection
REQ_Monitoring_Events_02	M	Inspection
REQ_Monitoring_Events_03	R	Inspection
REQ_Monitoring_Reports_01	R	Inspection
REQ_Monitoring_Reports_02	O	Inspection
REQ_Monitoring_Statistics_01	R	Inspection
REQ_Data_Management_Input_01	M	Demonstration
REQ_Data_Management_Storage_01	M	Demonstration
REQ_Data_Management_Storage_02	R	Inspection
REQ_Data_Management_Storage_03	O	Inspection
REQ_Data_Management_Output_01	M	Analysis
REQ_Delivery_EventManagement_01	R	Analysis
REQ_Delivery_EventManagement_02	R	Analysis
REQ_Performance-HandleEvents	R	Inspection
REQ_Performance-StoreSystemStatus	M	Inspection
REQ_Performance-EventsAggregation	R	Inspection
REQ_Performance-EventsOverload	R	Inspection
REQ_Performance-RealTime	R	Inspection
REQ_Performance-EventProcessing	R	Inspection
REQ_Detection_Algorithm	M	Analysis
REQ_Detection_Incident_Types	R	Analysis
REQ_Detection_Incident_Confidence_Value	O	Demonstration
REQ_Detection_Fast_Reaction_Trigger	M	Analysis
REQ_Detection_Logging	M	Analysis
REQ_Correlation_Rules	M	Demonstration
REQ_Correlation_Notification	M	Demonstration
REQ_Performance_False_Positive	R	Inspection
REQ_Performance_False_Negative	R	Inspection

REQ_Performance_Time_01	R	Inspection
REQ_Performance_Time_02	R	Inspection
REQ_Performance_Availability	R	Inspection
REQ_Security_ClockSync	M	Inspection
REQ_Configuration_Sens	M	Demonstration
REQ_Configuration_Enable	M	Demonstration
REQ_Configuration_Threshold	O	Inspection
REQ_Configuration_Finetune	O	Inspection
REQ_Investigation_Information	R	Analysis
REQ_Decision_FindReaction	M	Demonstration
REQ_Decision_EvaluateReaction	M	Demonstration
REQ_Preconfiguration_ReactionsPermissions	M	Demonstration
REQ_Management_Initiate	R	Analysis
REQ_Management_ResultNotification	M	Demonstration
REQ_Management_Automatic	M	Inspection
REQ_Management_Atomicity	M	Analysis
REQ_Management_Priority	M	Demonstration
REQ_Management_NoInterferences	O	Demonstration
REQ_Enforcement_Transparent	R	Demonstration
REQ_Enforcement_Deployment_01	M	Demonstration
REQ_Enforcement_Deployment_02	M	Demonstration
REQ_Enforcement_Deployment_03	R	Analysis
REQ_Reporting_System	R	Demonstration
REQ_Reporting_SystemConfiguration	R	Demonstration
REQ_Reporting_SystemReconfiguration	M	Demonstration
REQ_Reporting_Scripts	M	Demonstration
REQ_Reporting_Reactions	M	Demonstration
REQ_Performance_DeploymentTime	M	Analysis
REQ_Performances_ReactionTime	M	Analysis
REQ_Performance_ReactionScope	R	Analysis
REQ_Security_AuthenticationRequester	M	Demonstration
REQ_Security_IdentificationRequester	M	Inspection
REQ_Security_AuthorizationRequester	M	Inspection
REQ_Security_PrivacyRequester	M	Inspection
REQ_Security-Authentication	M	Inspection
REQ_Security-Integrity	M	Inspection
REQ_Security-Privacy	M	Inspection
REQ_Security_Confidentiality	M	Inspection
REQ_Security_RightManagement	M	Inspection
REQ_EV-SID_Interface_Information_01	M	Demonstration
REQ_EV-SID_Interface_Information_02	M	Demonstration
REQ_EV-SID_Interface_Information_03	R	Demonstration
REQ_EV-SID_Interface_Communication_01	M	Analysis
REQ_EV-SID_Interface_Communication_02	M	Demonstration
REQ_EV-SID_Interface_Communication_03	R	Test Point
REQ_FRM-SID_Interface_Ask	M	Demonstration
REQ_FRM-SID_Interface_Result	M	Demonstration
REQ_FRM-SID_Interface_Priority	R	Inspection
REQ_Decision-FastSelf-healing-Interface_Events_01	M	Inspection
REQ_Decision-FastSelf-healing-Interface_Events_02	M	Inspection
REQ_Decision-FastSelf-healing-Interface_Events_03	M	Inspection
REQ_Decision-FastSelf-healing-Interface_Events_04	R	Test Point
REQ_Decision-FastSelf-healing-Interface_Alarms_01	M	Test Point
REQ_Decision-FastSelf-healing-Interface_Alarms_02	M	Inspection
REQ_Decision-FastSelf-healing-Interface_Alarms_03	M	Inspection
REQ_Decision-FastSelf-healing-Interface_Communication_01	M	Inspection

REQ_FRM-HRM_Interface_Ask	O	Inspection
REQ_FRM-HRM_Interface_Result	O	Analysis
REQ_FRM-HRM_Interface_Priority	O	Inspection

8 Conclusion

This document presented the requirements for the *Fast Self-healing* module (event monitoring, serious incident detection and fast reaction functions). The requirements expressed in this deliverable have been derived from the project's objectives and from general constraints that were briefly reminded respectively in sections 1 and 3. Then, we have classified the requirements based on their priority and test method to apply.

We have expressed 105 requirements including 21 interface requirements (internal to *Fast Self-healing* module or with the other modules in the infrastructure). Moreover, 61 requirements were defined as mandatory (34 recommended and 10 optional).

These requirements will serve as a basis for the next steps which consist in defining the products architecture and specification for the *Fast Self-healing*. This task will deliver the D4.2 (*Architecture and specifications for the Fast Self-healing system*) document.