An Infrastructure for Maintenance and Evolution of Security and Dependability in Dynamic Computing Scenarios

Antonio Maña, José Ruíz

amg@lcc.uma.es

Safe Society Labs
University of Malaga

St. Petersburg, Sept. 2010

Introduction

- Highly Distributed Systems pose significant challenges for security and dependability.
- In these settings, seamless and dynamic reaction to security problems (threats, intrusions, bugs) becomes a central element for ensuring Security and Dependability (S&D).
- The SERENITY project has developed mechanisms supporting the dynamic deployment, configuration and monitoring of S&D solutions.
- This work presents the extension for supporting seamless dynamic reaction and evolution.

Introduction

- SERENITY focuses on runtime provision and supervision of S&D solutions.
- The main characteristics of SERENITY are:
  - Allows the detection of problems in the operation of individual instances of S&D solutions.
  - Enables automated reconfiguration of the applications using these solutions.

Introduction

- SERENITY Evolution & Maintenance Infrastructure provides a basis for taking actions that can support the maintenance and evolution of S&D solutions.

- Its aim is to analyse problems detected in the operation of S&D solutions on a global level in order to support automated reaction, evolution and maintenance of applications operating in highly dynamic and heterogeneous systems.

Background

- **SERENITY Artefacts**:
  - S&D Patterns represent abstract S&D solutions that provide one or more S&D Properties. The popular *Needham-Schroeder public key protocol* is an example of an S&D solution that can be represented as an S&D Pattern.
  - S&D Classes represent S&D services (abstractions of a set of S&D Patterns characterized for providing the same S&D Properties and being compatible with a common interface). An example of an S&D Class is the *ConfidentialCommunicationClass*, which defines an interface including among others, an abstract method `SendConfidential(Data, Recipient)`.
  - S&D Implementations represent operational S&D solutions, which are in turn called Executable Components. It is important to note that the expression "operational solutions" refers here to any final solution (e.g. component, web service, library, etc.) that has been implemented and tested for compliance with the corresponding S&D Pattern.

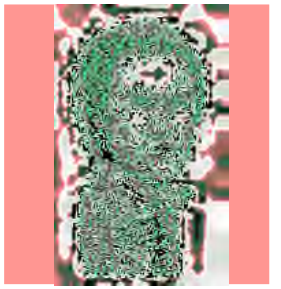
SERENITY actors

- ACTOR: **S&D Solution Developer**
  - Develops S&D Solutions
  - Creates S&D Artefacts

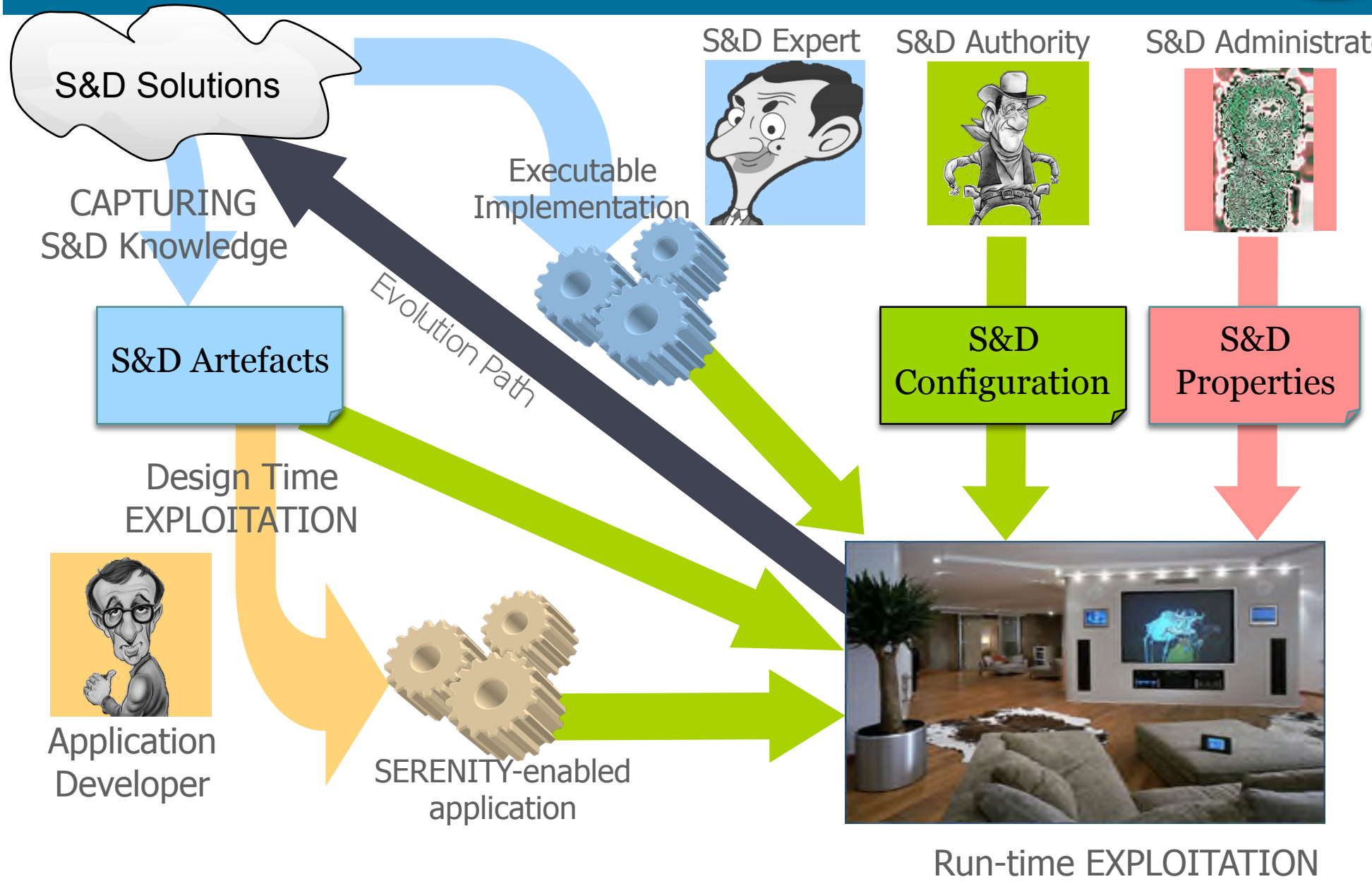
- ACTOR: **Application Developer**
  - Identifies S&D Requirements
  - Develops applications

- ACTOR: **S&D Authority**
  - Defines S&D Configuration
  - Manages SRF Library

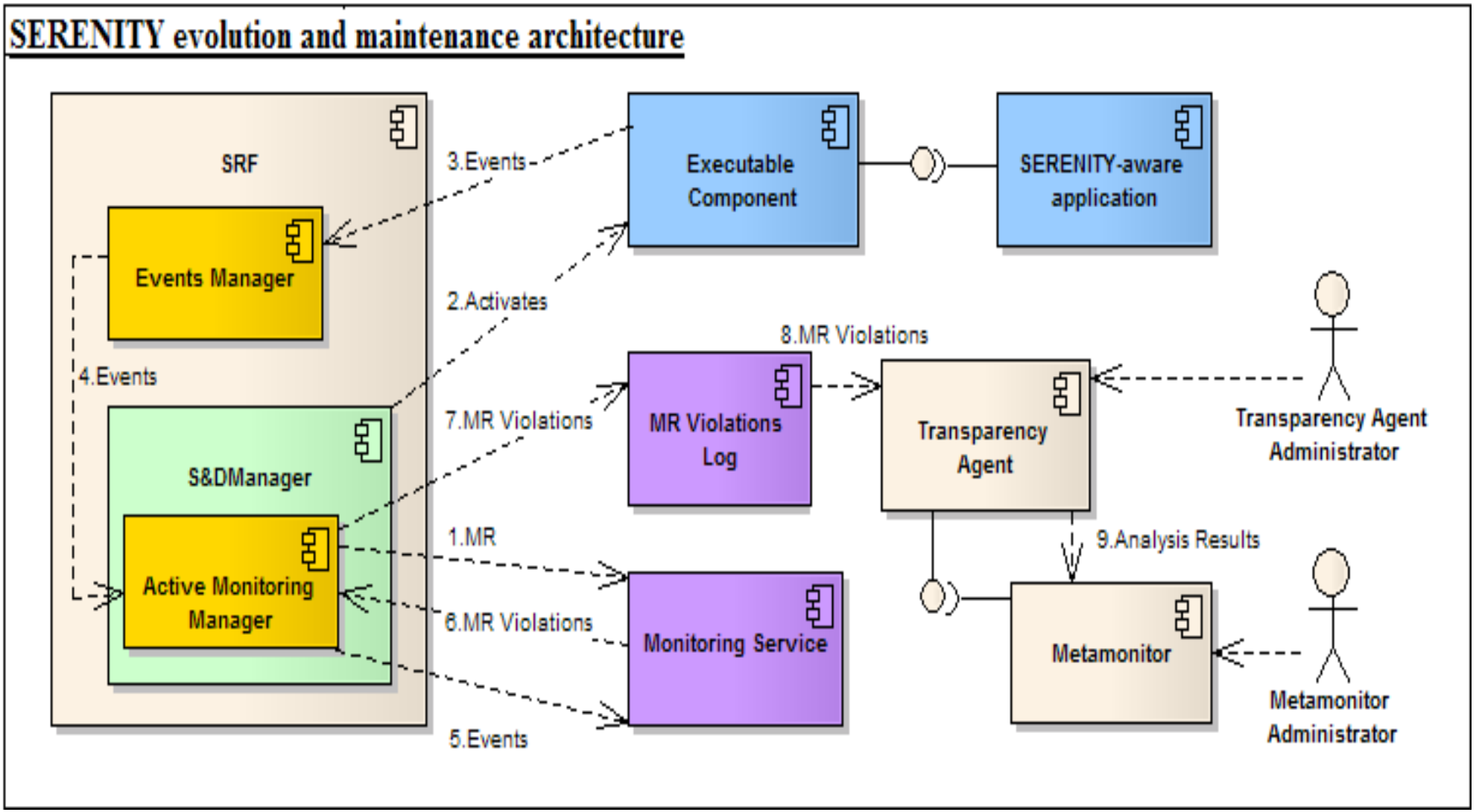
- ACTOR: **Security Officer / S&D Administrator**
  - Defines S&D Properties & S&D Policies



SERENITY Processes Overview



SERENITY maintenance and evolution infrastructure



Monitoring specification

- Monitoring rules are expressed in *EC-Assertion* and have the generic form

$$B \Rightarrow H$$

- stating that when B is True, H must also be True. Both B (Body) and H (Head) are defined as conjunctions of **Event Calculus** predicates.
- The predicates used in monitoring rules express
  - the **occurrence** of an event (***Happens*** predicate),
  - the **initiation** or **termination** of a **fluent** (i.e. condition) by the occurrence of an event (***Initiates*** and ***Terminates*** predicates respectively),
  - or the **validity** of **fluent** (***HoldsAt*** predicate)
- Predicates are associated with time variables

Monitoring specification

- An example of monitoring rule expressed in EC-Assertion is:

Happens(e(_id1,_sender,_receiver,REQ,decrypt(_x,_y),_receiver), t1, R(t1,t1))

$\Rightarrow$

Happens(e(_id2,_receiver,_sender,RES,decrypt(_x,_y),_receiver), t2, R (t1,t1+1000))

- This rule express a bounded availability property for a decryption operation *decrypt(_x, _y)*.
- According to the rule, following an invocation of the operation *decrypt* in the decrypting component _receiver at some time point t1, there should be a response to the caller of the operation (_sender) at some time point t2 which cannot be more than 1000 milliseconds after t1.

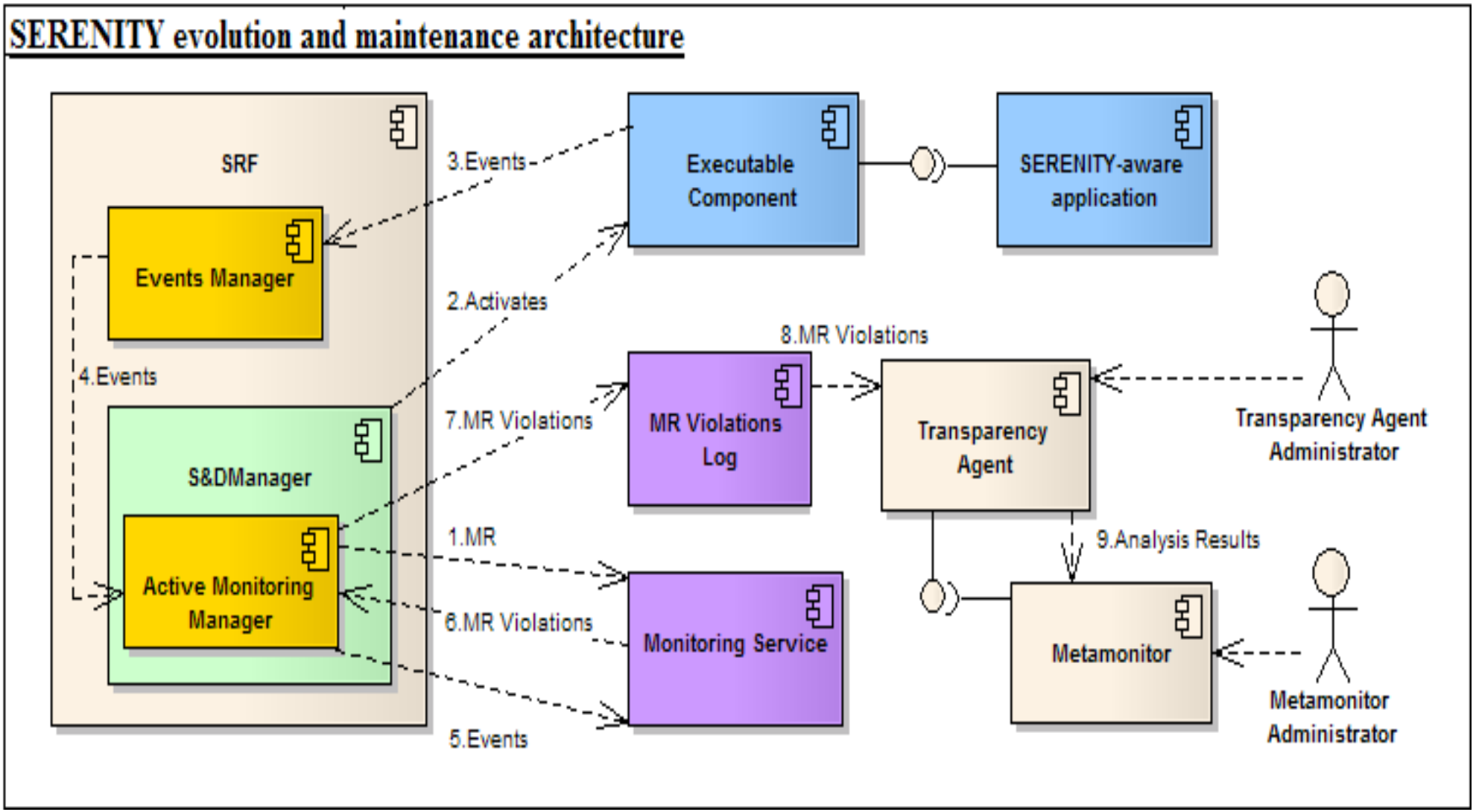
Monitoring specification

- Time constraints are indicated as time ranges e.g. R(t1,t1+1000)
- The monitoring rules in S&D Patterns must be designed to provide the information required in order to assess the correct functioning of the pattern and executable components that realise it.
- In a running system, the basic building blocks are the **Executable Components** (**ECs**), which are implementations of the S&D Patterns.
- ECs must include appropriate Event Capturers in order to inform their clients about their internal operation.
- All implementations of an S&D Pattern must include code to capture the events used in the monitoring rules of the pattern and to notify the events to the application through the SRF.

Monitoring operation

- **Monitoring**
  - When an event from an EC is received by the SRF it is managed by the core monitoring mechanism of the framework.
  - In this case, the event is forwarded to the appropriate monitoring service, which evaluates the state of the EC by applying the monitoring rules defined in the corresponding S&D Pattern.
  - If a violation of one of these rules is detected, this violation is reported to the SRF, which registers it and takes appropriate actions (such as deactivate the pattern, pause, reset, etc.)

SERENITY maintenance and evolution infrastructure



Monitoring operation

- **Need for additional monitoring layers**
  - To deal with potential **problems caused by the interaction between different ECs**, a second monitoring mechanism is in charge of monitoring at the level of one particular SERENITY framework.
  - To **support maintenance and evolution** of specific S&D solutions and detect problems with non-compliant implementations, as well as problems in the modelling, solution-specific elements called Metamonitors perform vertical analysis.

Evolution Maintenance Elements

- **Transparency Agent**
  - Associated and deployed with a specific SRF
  - **Horizontal analysis**: Analyses data from different S&D Solutions in the same environment.
  - Collects information related to the violations of monitoring rules, analyses it and presents it to the TA administrator.
  - The results may be sent to the Metamonitor depending on certain rules, so the administrator of the Transparency Agent can choose what information to send to the Metamonitor and what to keep as confidential (Controlled Transparency).

Evolution Maintenance Elements

- **Metamonitor**
  - Deployed on an external infrastructure.
  - **Vertical analysis**: Analyses data from different machines about the same S&D Solution.
  - Receives information from several Transparency Agents and performs a new analysis on this.
  - The Metamonitor has a global view of what is the behaviour of the S&D Solutions in different contexts, and therefore is able to deduce proper conclusions that are not possible locally (modification of the description of an S&D Pattern, the deactivation of particular S&D solutions, etc.). This benefits both, the user of these solutions and the solution developer.

Application scenario

- Two different enterprises need to exchange confidential messages.
- Each one has a SERENITY node that contains a SRF, S&D solutions, local monitors and a TA.
- ECs are monitored using the events sent to the SRF.
- These events have information about the operating state of the ECs and can keep the SRF informed when a rule is violated.

Application scenario: TA

- The SRF stores events in an event log, which is accessed by the local TA.
- Some of this information is forwarded to the Metamonitors.
- If the frequency of correlation between events in two ECs is statistically significant we may conclude that there is an unforeseen interaction between those ECs and take appropriate measures.

Application scenario: MM

- Suppose we receive in the Metamonitor repeated violations of a monitoring rule for **EC A** indicating an illegal transition between two states.
- This would indicate that **EC A** does not conform with its S&D Pattern model.
- On the contrary if the violations come from different ECs of the same S&D Pattern, these violations would indicate an error in the model of the S&D Pattern.

Application scenario

- All this information can be sent to the developers of the particular S&D solution so they can fix it.
- Once this is done, the SERENITY security manager can notify the client node to deactivate, update or add a new security solution in order to improve the perfomance of another one already used.

Conclusions and further work

- The presented infrastructure supports the detection of problems (intrusions, bugs,…) and the evolution of S&D components and the applications using them, improving the global capability for avoiding intrusions, attacks and other security problems.
- Two monitoring layers are introduced in order to support detection and evolution (**Transparency Agent** & **Metamonitor**).
- Currently, the modifications in the system are done by a human administrator.
- Our ongoing and future work focuses on:
  - Automated Reaction System
  - Automated identification of changes to solve the problems detected

Thanks! Any questions?