DCS DAYS

FORMAL METHODS AND SOFTWARE (LEGAL) LIABILITY:

A formal approach to build systems that generate digital evidence

Eduardo Mazza

Advisors:

Marie-Laure Potet

Daniel Le Métayer (INRIA - Monbonnot)

Outline

- About me
- Project LISE
- Objective
- Study case
- Challenges
  - Specification
  - Analysis
- Approach

About me

- Master's degree in Computer Science – Brazil, 2008
  - Malware detection using data mining techniques
- Internship at INRIA – Grenoble, June 2008
  - SIMple – language to help to define privacy policies.
  - Automate privacy policy manipulation
- PhD at VERIMAG – Grenoble, Nov. 2008
  - Project LISE

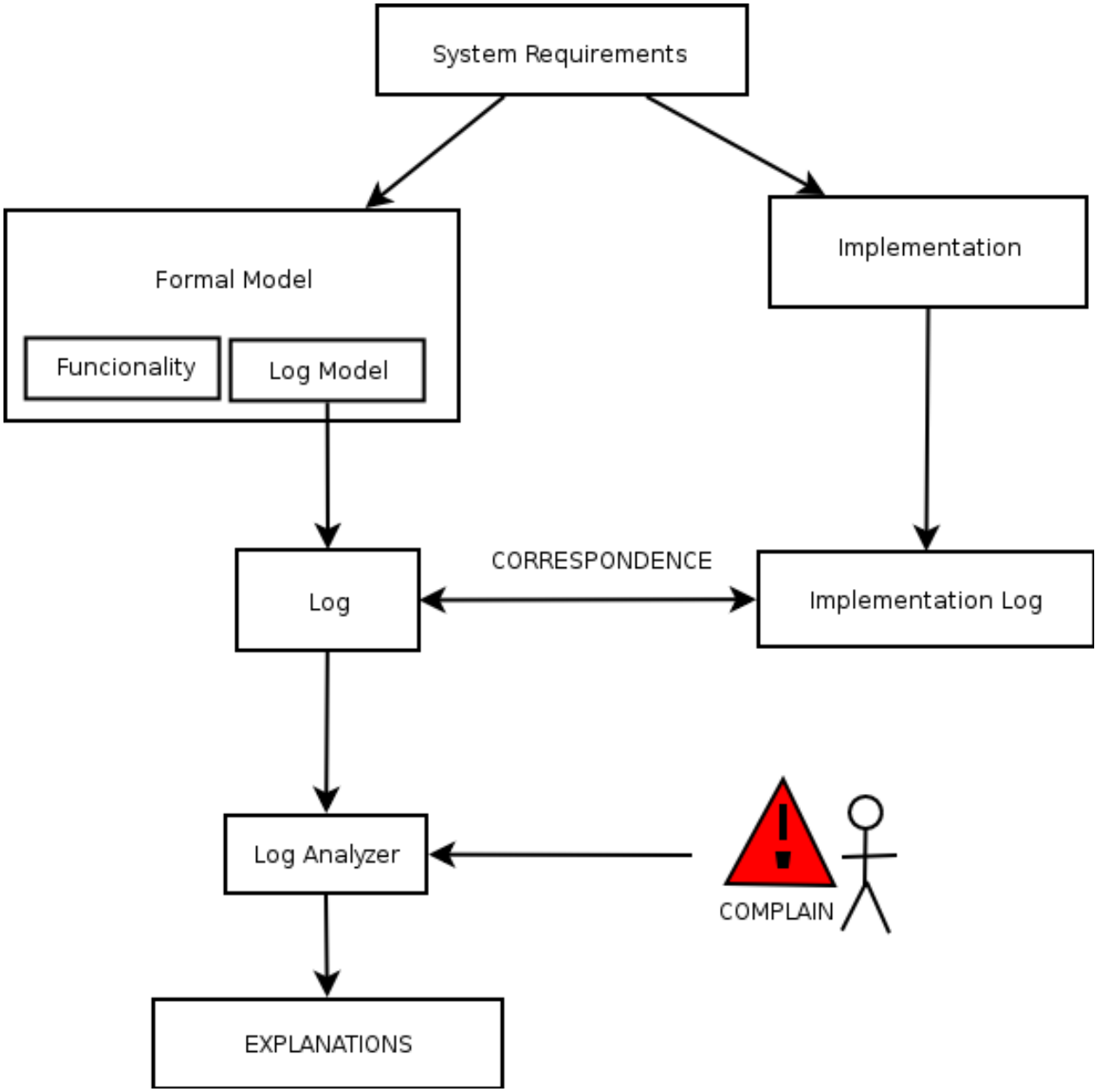
LISE: **L**iability **I**ssues in **S**oftware **E**ngineering

- Context
  - **Responsibility**
    - With system more complex is important to know who is responsible
      - Example: system that use open-source or third party components
  - **Digital evidences**
    - What can be legally used as digital evidence? How to formalize it?
  - **Contract** made between legal parts
    - The main object of LISE
    - it should contains agreements about **responsibility** and **digital evidences**

Specific objective for VERIMAG

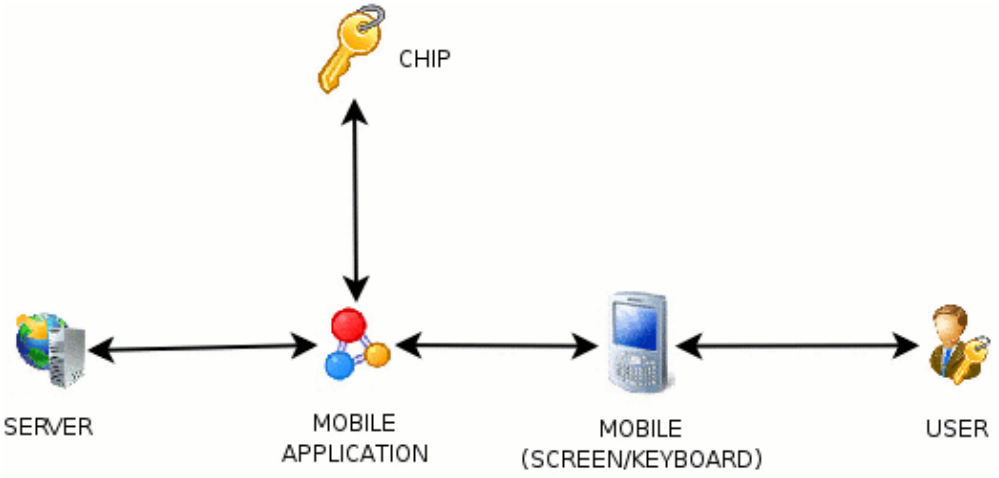
- Propose a language for formally describe the **responsibility** of legal parts in contracts
- Formal specification of **log** as digital evidence
- Create a **Log Analyzer**, to determine the responsibility, based on the log, when a dysfunction occurs
- Approach:
  - System composed with actors
  - Log: communication registry between actors
  - The log can never be corrupted – the information registered corresponds exactly what it happened

Approach


System Requirements
Formal Model
Funcionality
Log Model
Implementation
Log
CORRESPONDENCE
Implementation Log
Log Analyzer
COMPLAIN
EXPLANATIONS

Study case

- Signature system in mobile



- Examples of problems:
  - User alleges that he has never signed any document
  - User alleges that he has signed document by the signature is not in server

Challenges - Specification

- Formal specification of:
  - responsibility
    - how to formally define responsibility?
    - how to formalize responsibility with so many different situations? Example: casual responsibility
  - dysfunction
    - how define dysfunctions and make the link with its responsible actors
  - log (and log properties)
    - completeness – logs should contain all information necessary to establish the responsibility
    - minimality – logs should contain the minimum information possible without loose the completeness

Challenges - Analysis

- Trace Analyzer
  - Procedure to determine responsibilities
  - Production of digital evidence
- Distributed registered log
  - each actor has its own log
- Partial concrete log
  - log always has a limited size
  - not all actors can be logged
    - hardware
    - protected software

Approach

- Functional model
  - General specification of the system using the B formal language
    - Actors (User, Mobile, Server, ...)
    - Events (Server send document to mobile application, ...)
    - Trace (server_log, mobile_log, ...)
  - Automatic generation of the log based on communication between actors
- Global dysfunction
  - **Collusion** – the application do not ask the PIN and send the document direct to chip
  - **Independent dysfunctions** – mobile shows a different document and the user agree but enter a incorrect PIN
  - **Complementary dysfunctions** – the user accept and put the correct PIN, but the application do not send the user the message that the PIN was incorrect

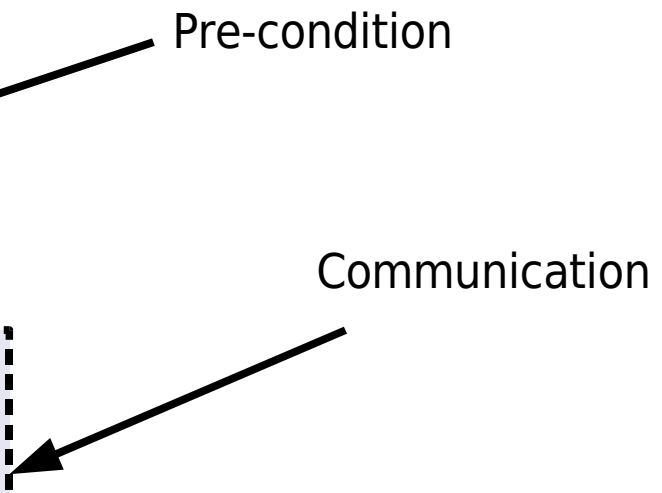
Study case – B model

Communication

```
/* Application sends Document to Mobile */
APP_PASS_DOC =
SELECT
  app_state = AppNormal &
  app_message = TRUE &
THEN

  /* state change */
  app_state := WaitingResponseMobile ||
  app_message := FALSE ||
  /* send Document */
  mobile_message := TRUE ||
  mobile_input := app_input ||

END;
```



Study case – B model

Communication with trace

```
/* Application sends Document to Mobile */
APP_PASS_DOC =
SELECT
  app_state = AppNormal &
  app_message = TRUE &
  app_trace_number < (MAXINT - 1)
THEN
  /* state change */
  app_state := WaitingResponseMobile ||
  app_message := FALSE ||
  /* send Document */
  mobile_message := TRUE ||
  mobile_input := app_input ||
  /* log */
  app_log := app_log <+
  {app_log_number |-> (Receive, Server, App, app_input),
  app_log_number + 1 |-> (Send, App, Mobile, app_input)} ||
  app_log_number := app_log_number + 2
END;
```

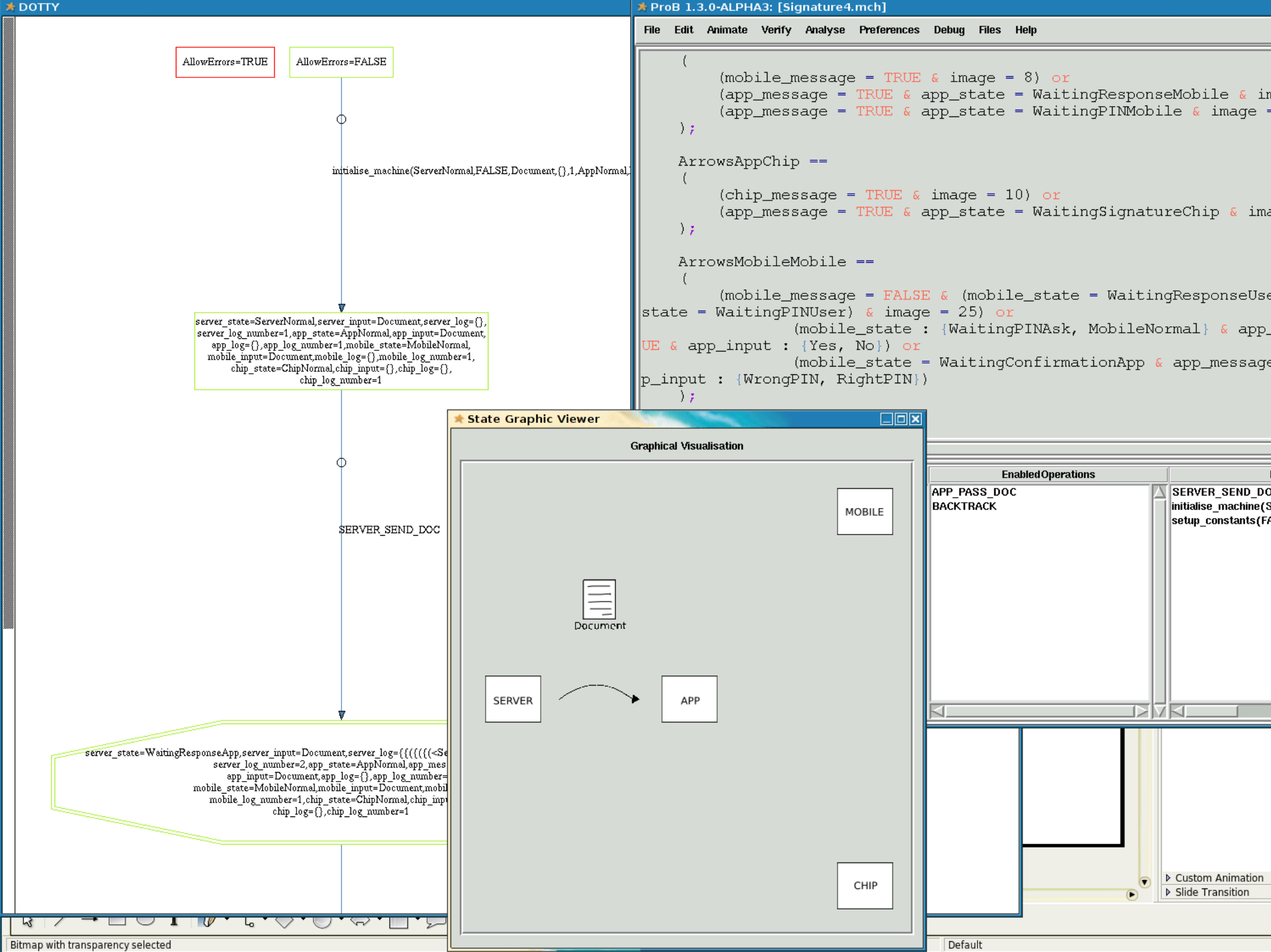
Pre-condition

Communication

Trace

Tools

- Objective
  - create tools graphically visualize:
    - responsible actors
    - digital evidence creation process
- Graphic Animation
  - System events
    - illustrates the how the system works and when communication is passed between actors
  - System states
    - visualization of the system variable through the events of the system as graphs


DOTTY
AllowErrors=TRUE
AllowErrors=FALSE
initialise_machine(ServerNormal,FALSE,Document,{},1,AppNormal,
server_state=ServerNormal,server_input=Document,server_log={},
server_log_number=1,app_state=AppNormal,app_input=Document,
app_log={},app_log_number=1,mobile_state=MobileNormal,
mobile_input=Document,mobile_log={},mobile_log_number=1,
chip_state=ChipNormal,chip_input={},chip_log={},
chip_log_number=1
SERVER_SEND_DOC
server_state=WaitingResponseApp,server_input=Document,server_log={{{{{{{<Se
server_log_number=2,app_state=AppNormal,app_mes
app_input=Document,app_log={},app_log_number=
mobile_state=MobileNormal,mobile_input=Document,mobil
mobile_log_number=1,chip_state=ChipNormal,chip_inp
chip_log={},chip_log_number=1
ProB 1.3.0-ALPHA3: [Signature4.mch]
File Edit Animate Verify Analyse Preferences Debug Files Help
(
(mobile_message = TRUE & image = 8) or
(app_message = TRUE & app_state = WaitingResponseMobile & im
(app_message = TRUE & app_state = WaitingPINMobile & image =
);
ArrowsAppChip ==
(
(chip_message = TRUE & image = 10) or
(app_message = TRUE & app_state = WaitingSignatureChip & ima
);
ArrowsMobileMobile ==
(
(mobile_message = FALSE & (mobile_state = WaitingResponseUse
state = WaitingPINUser) & image = 25) or
(mobile_state : {WaitingPINAsk, MobileNormal} & app_
UE & app_input : {Yes, No}) or
(mobile_state = WaitingConfirmationApp & app_message
p_input : {WrongPIN, RightPIN})
);
State Graphic Viewer
Graphical Visualisation
MOBILE
Document
SERVER
APP
CHIP
EnabledOperations
APP_PASS_DOC
BACKTRACK
SERVER_SEND_DO
initialise_machine(S
setup_constants(FA
Custom Animation
Slide Transition
Bitmap with transparency selected
Default

Utiliser des ordres partiels pour modéliser, vérifier et superviser des systèmes parallèles et répartis

- Thomas Gazagnaire – PhD thesis, 2008
- **Supervision** – to help interpretations of logs with great amount of information
  - with total observation
  - with partial observation - not every event is possible to be observable
    - techniques based on deductions
- **Diagnostic**
  - technique to, from the model and observation, build possible compatible explanations
- **Events correlation**
  - Based on diagnostic results make conclusions about non observable events

Thank you

Questions?
Suggestions?
Comments?