RE-TRUST quaternary meeting 18-19 Dec. 2008

Remote entrusting by remote invariants monitoring

Stefano Di Carlo
Politecnico di Torino

Goal

- Detecting software modifications by monitoring automatically inferred invariant properties of an application

Outline

- Invariants overview
- Remote entrusting and invariants
- A practical example
- Conclusions and future activities

Outline

- **Invariants overview**
- Remote entrusting and invariants
- A practical example
- Conclusions and future activities

What is an Invariant?

- An invariant is a property true at a certain point(s) of a program execution
- An invariant is composed of:
  - A **property**: e.g., variable x always contains a value greater than 0;
  - A **location**: the point of the program execution where the property is verified (e.g., before calling the function f())

An example

Program code:

```
for (i=1; i<N; i++)          \\N>1
{
  //code to execute
  …
  return a*2+b*2;
}
```

Invariants:

- *i* is always greater than 0
- the *return* value is always even

- Invariants provide information about the inner logic of the program

Common uses

- Invariants have been introduced in the field of software engineering :
  - Software Testing
  - Software Design
  - Software Optimization
  - Bugs fix
  - …

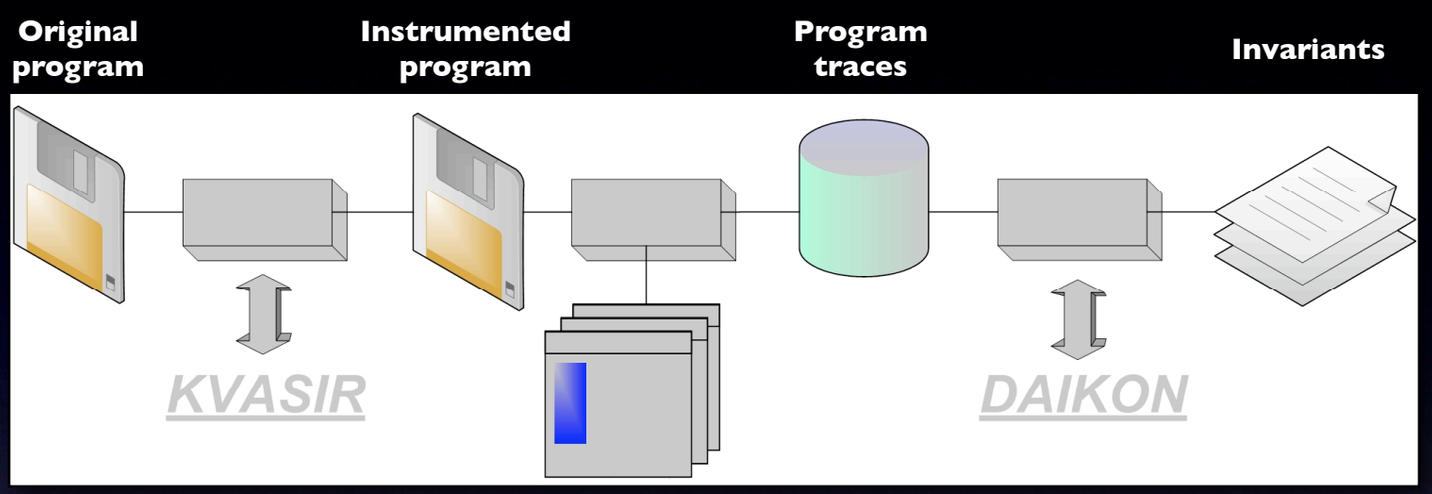
Definition

- **Static analysis**: static analysis of the program code only (no information about code execution is used)
  - Example: analysis of the data-flow
  - Drawback: it provides information about the context of the program only

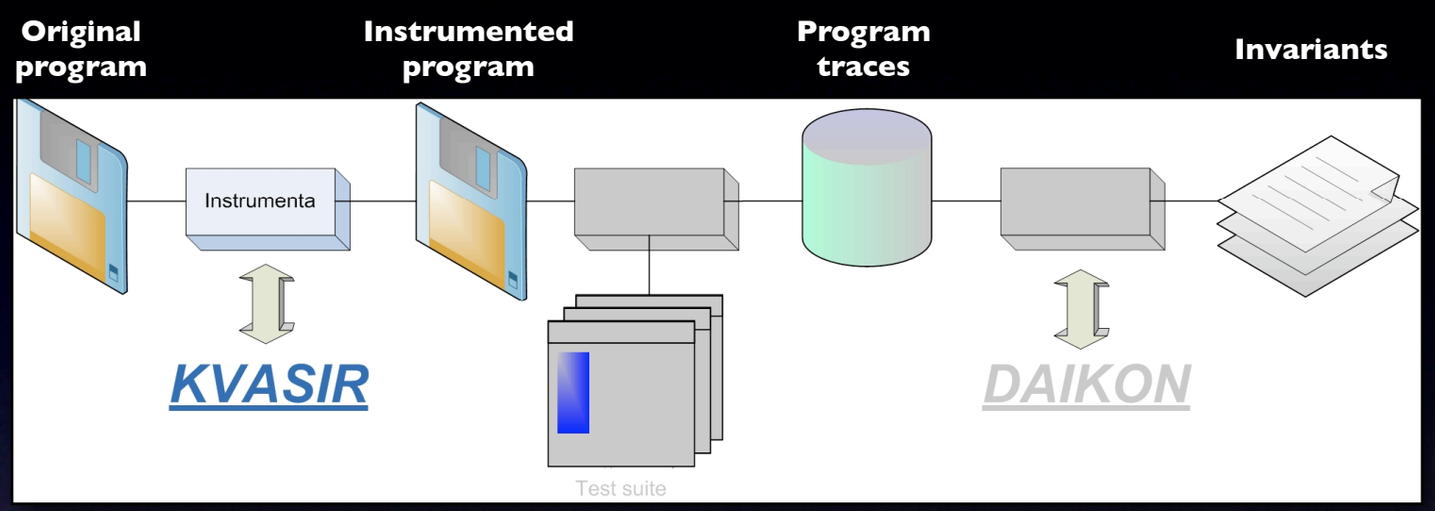
Definition

- **Dynamic analysis**: it uses execution traces to analyze the behavior of a program during its execution
  - Performed through three different phases:
    - Program instrumentation
    - Instrumented program execution
    - Invariant properties search

Dynamic analysis

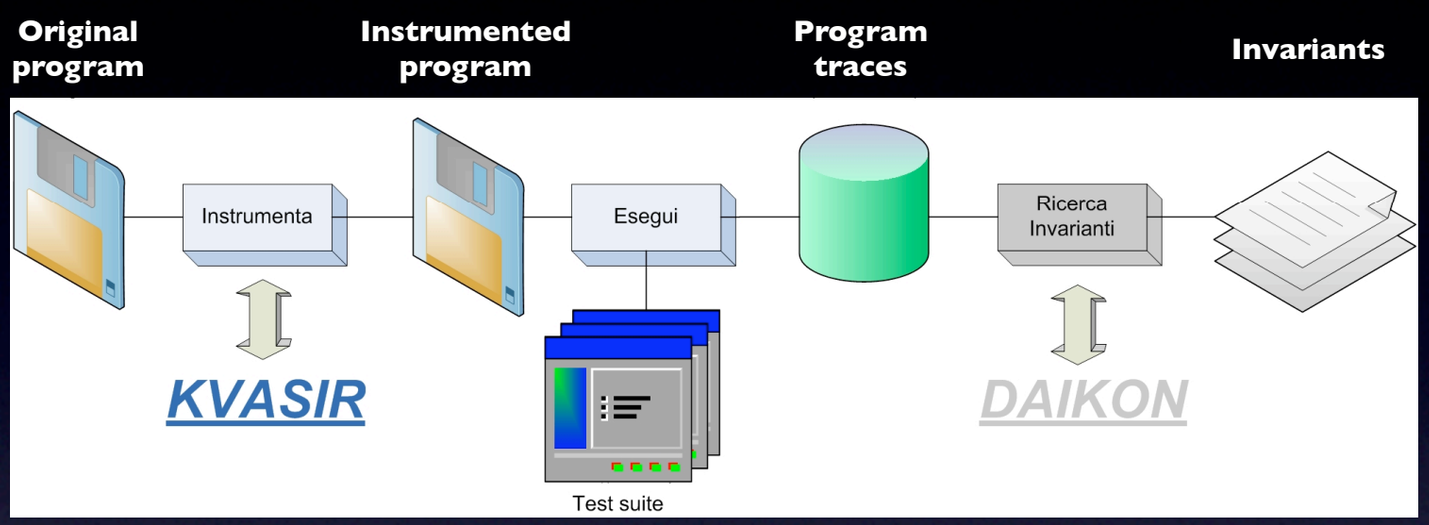


Dynamic analysis



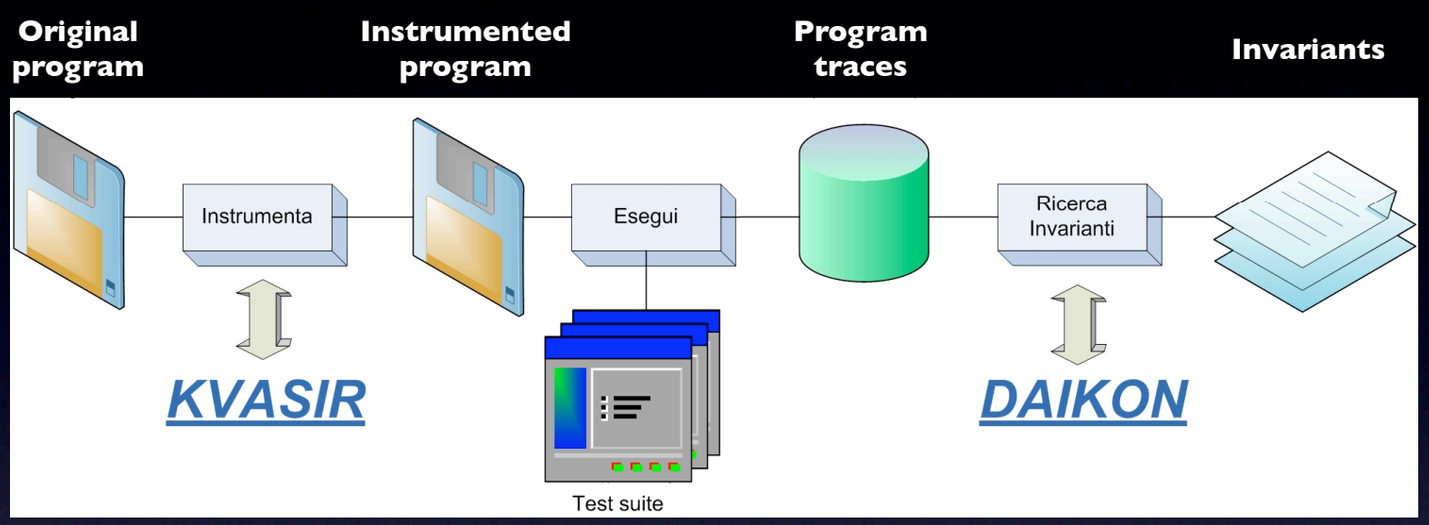
- The program is instrumented to trace the content of each variable during its execution

Dynamic analysis



- The instrumented program is executed under a meaningful workload

Dynamic analysis



- Patterns and relations are searched over the program traces to define invariant properties

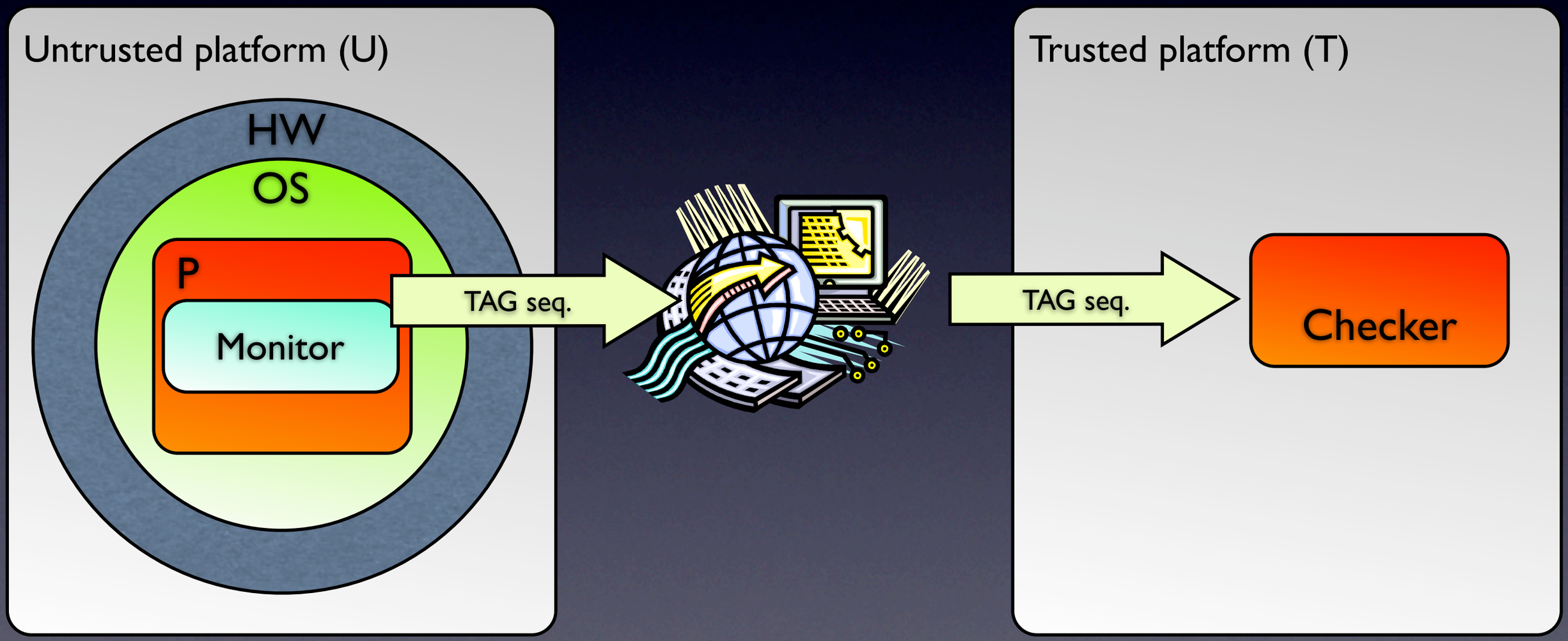
Outline

- Invariants overview
- **Remote entrusting and invariants**
- A practical example
- Conclusions and future activities

Assumption

- Software modifications will probably lead to the modification of some of the invariant properties defined on the original code

Architecture



Monitor & Checker

MONITOR

- Collects variable traces and send them to the server
- Variable traces includes:
  - Variable identifiers
  - Values
  - Program locations

CHECKER

- Receives variables traces
- Based on the identifiers names and locations checks whether invariants are respected or not

Open issues

- Invariants in remote entrusting present three main issues:
  - Reliability
  - Selection
  - Relevance

Reliability

- There is not a strict relationship between invariants violation/integrity and a attacks
  - Invariants violation => Attack
  - Invariants respected => No attack
- **The two conditions are not always true**

Reliability

- Two main causes:
  - **False positive**: invariants are searched over a set of n executions, these may lead to properties not completely specified
  - **False negative**: missing properties due to lacks of the tools used to define the invariants

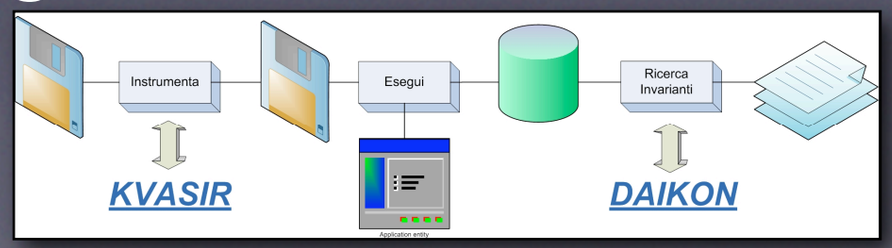
Selection

- **<u>Ideal solution</u>**: using invariants defined on variables critical for the integrity of the program

- **<u>Drawbacks</u>**:
  - Usually no invariants can be defined on these variables
  - If invariants exist their are not relevant

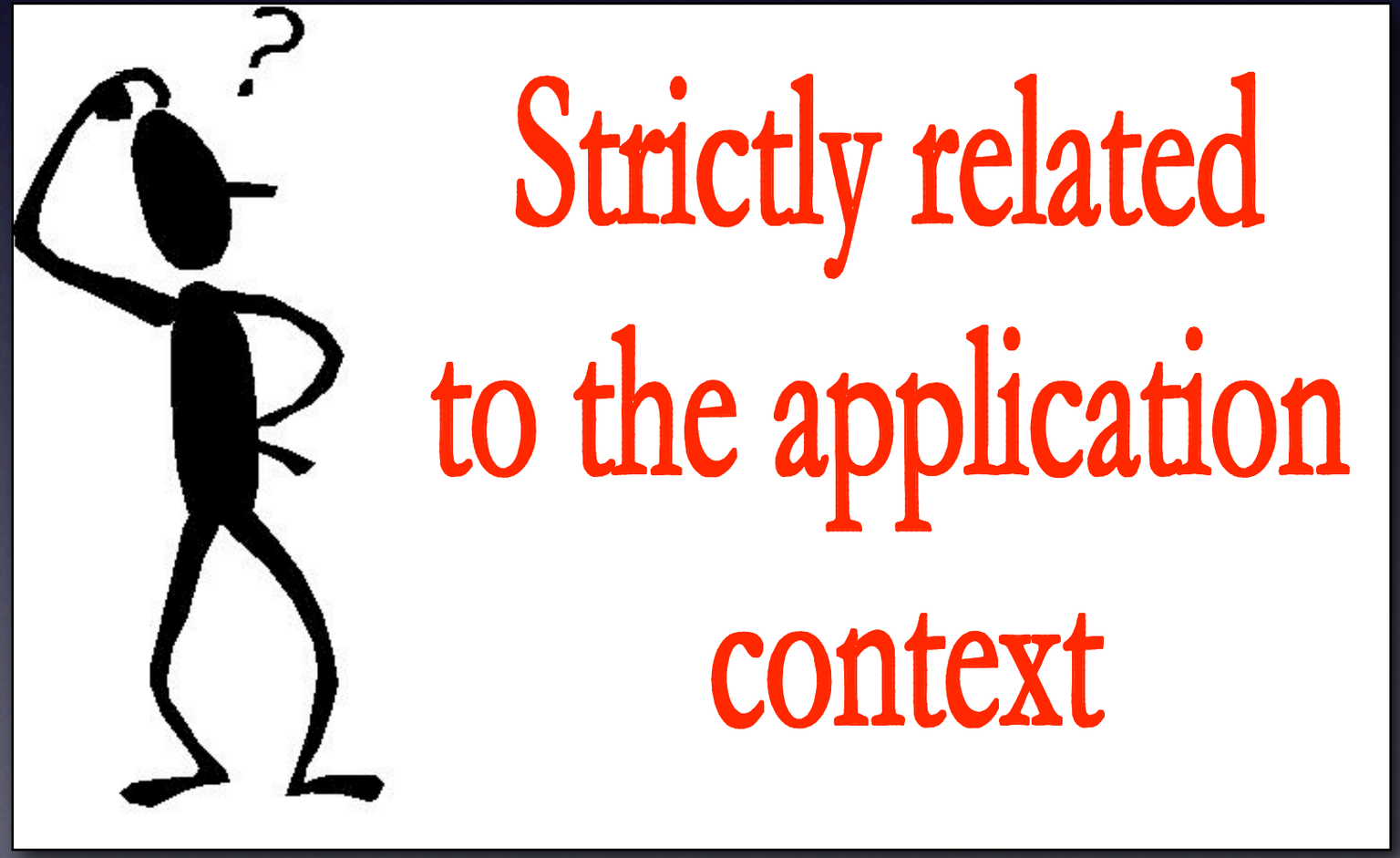
Selection

- Focus on the properties and not on the relative variables
- **Drawbacks**:
  - Number of invariants not manageable
- **Solutions**:
  - Considering only core functions
  - Increasing the number of iterations



Relevance

- An invariant is relevant if it provides useful information



Relevance

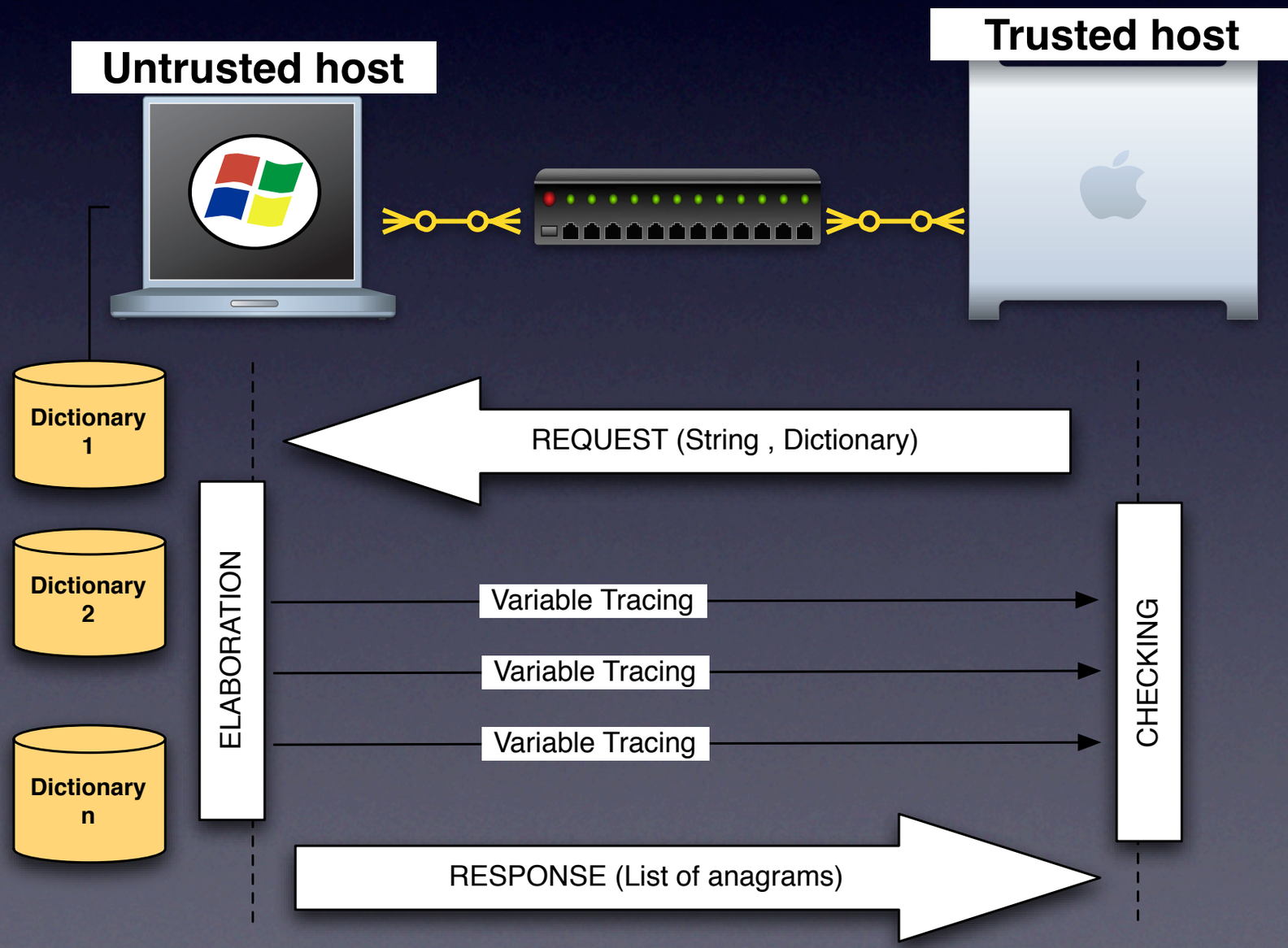
- Using invariants in a different way, selecting those invariants usually considered not relevant in other contexts

Outline

- Invariants overview
- Remote entrusting and invariants
- **A practical example**
- Conclusions and future activities

Target application

- Remote anagrams searching



Monitor & Checker

Version A

- Controlled invariants
  - /ncount == 10
  - /words2 == [DIED, NICK, NECK, DAMIEN, ANTICKED, RICKETY, INACTIVITY, IODINE,

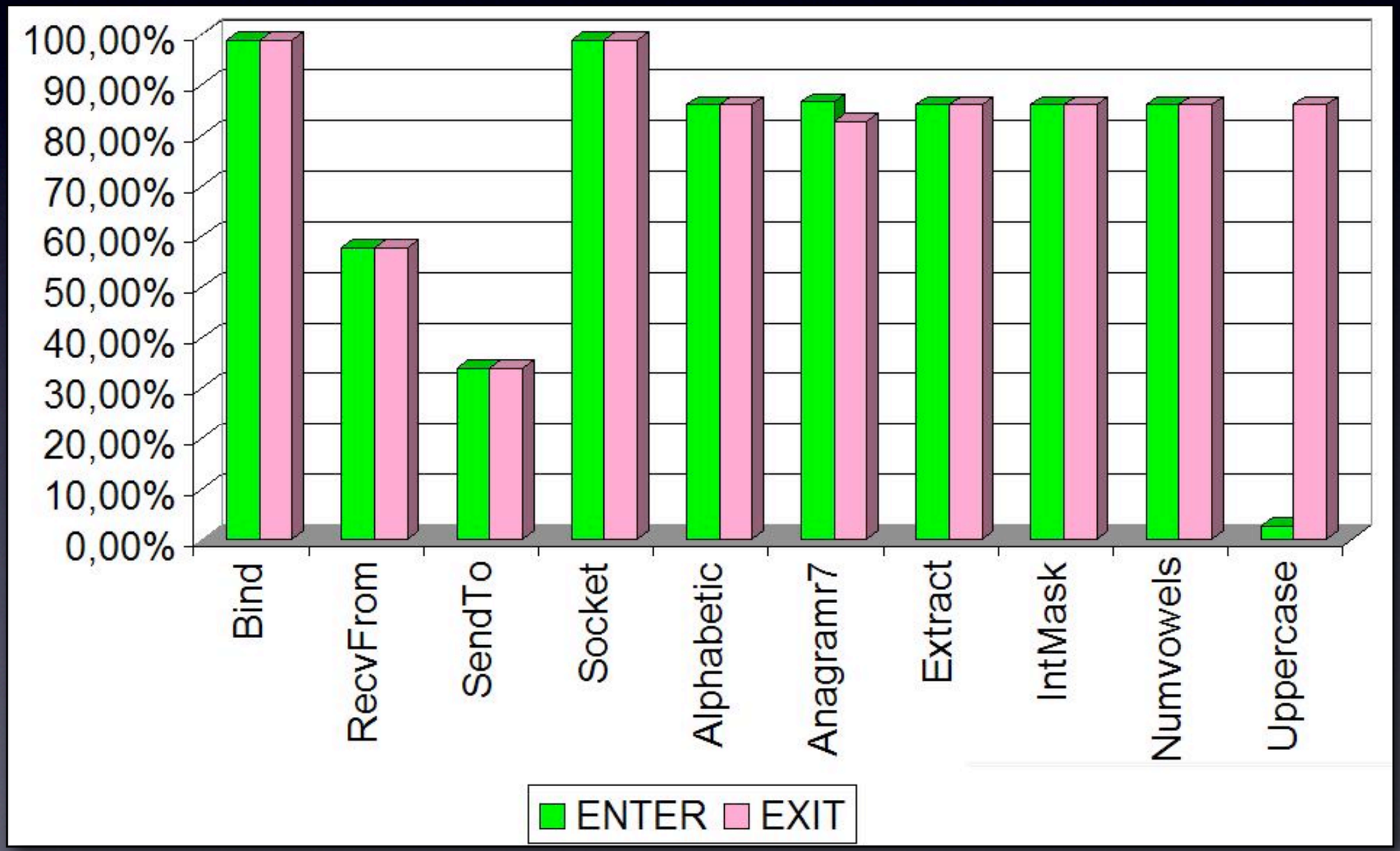
OK

Version B

- Controlled invariants
  - /adjacentdups == 0

KO

Some measures



Outline

- Invariants overview
- Remote entrusting and invariants
- A practical example
- **Conclusions and future activities**

Conclusions

- Remote invariant monitoring can be efficiently included in a remote entrusting architecture
- Not 100% secure, nevertheless it can be combined with different mechanisms

Future works

- Ad-hoc invariants definition tool
- Flow automation
- Mutant code:
  - How to write different versions of a program performing the same functionality but having different invariants?