Applications of log and trace analysis to industrial projects

Wahab Hamou-Lhadj, PhD., ing.
Software Behaviour Analysis Research Lab
Concordia University
Montreal, QC, Canada
http://www.ece.concordia.ca/~abdelw

Université de Montréal
08/10/2015

Software engineering: current challenges

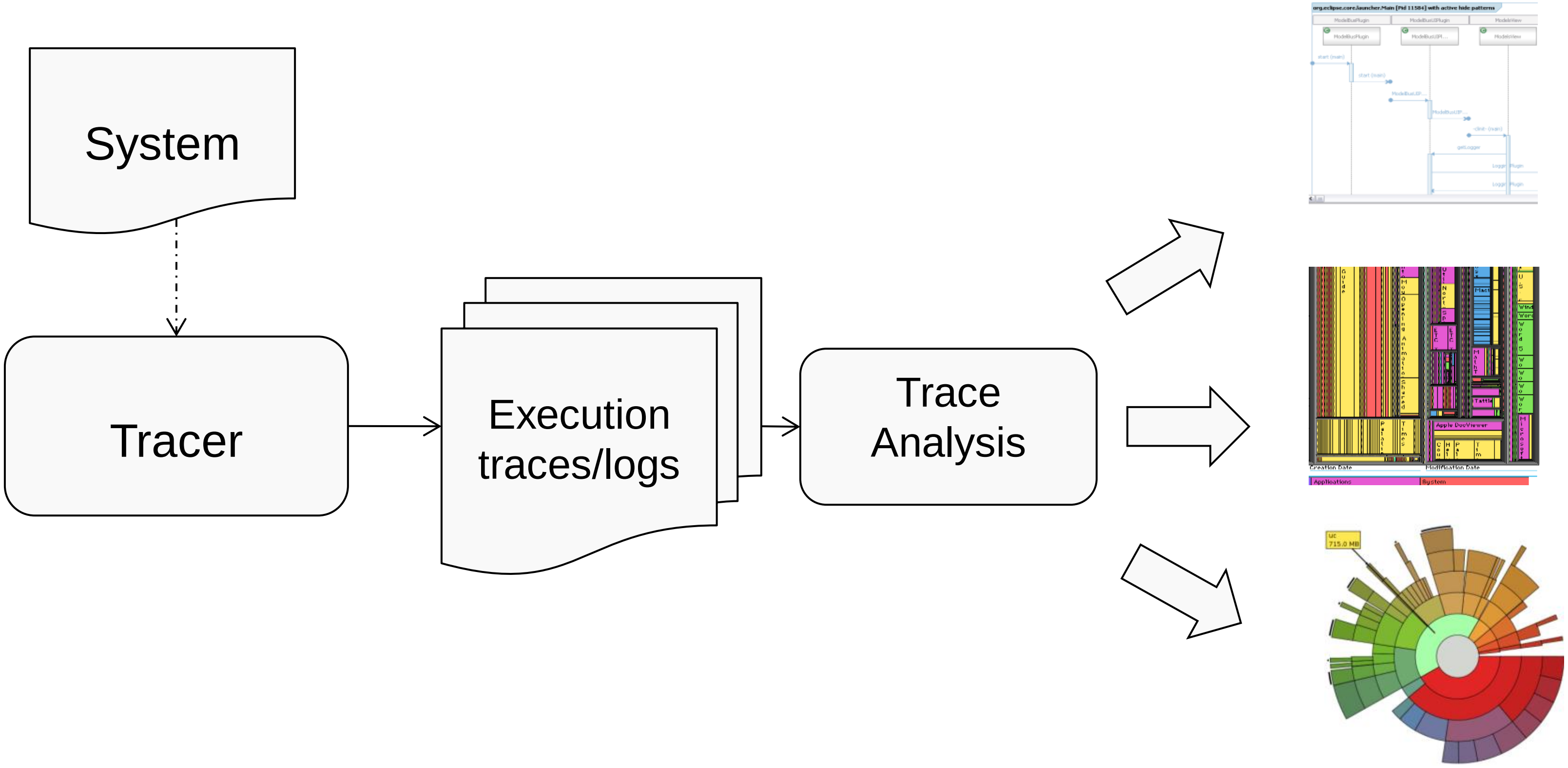
- Software systems are inherently complex
- Many of them are poorly structured
- The development effort is human-intensive
- Software industry tends to be poorly regulated
- As a direct consequence: Maintenance, security, and other software engineering activities are challenging and costly

Software engineering: current challenges

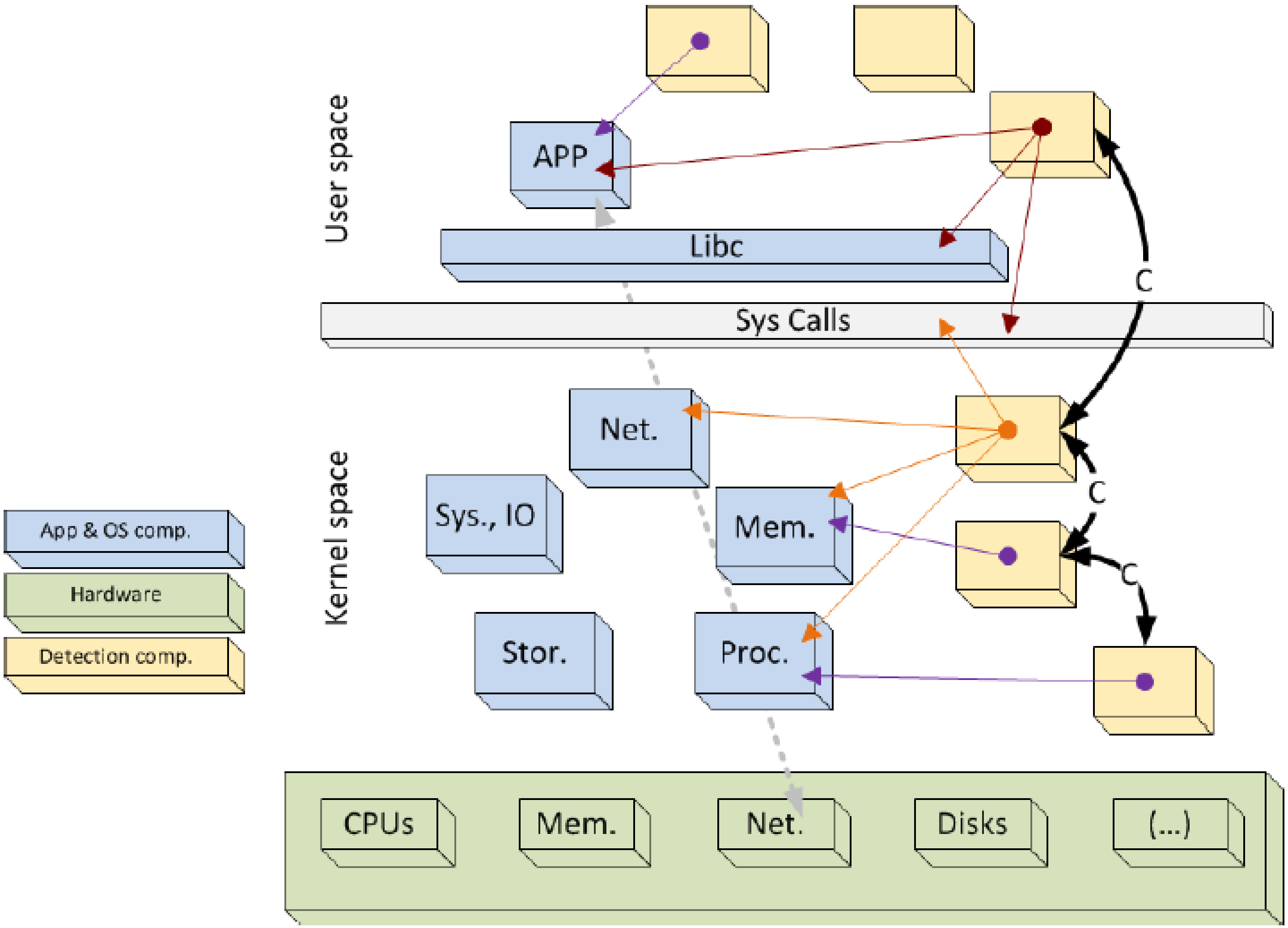
- Software systems are inherently complex
- Many of them are poorly structured
- The development effort is human-intensive
- Software industry tends to be poorly regulated
- As a direct consequence: Maintenance, security, and other software engineering activities are challenging and costly

This calls for advanced software analysis techniques

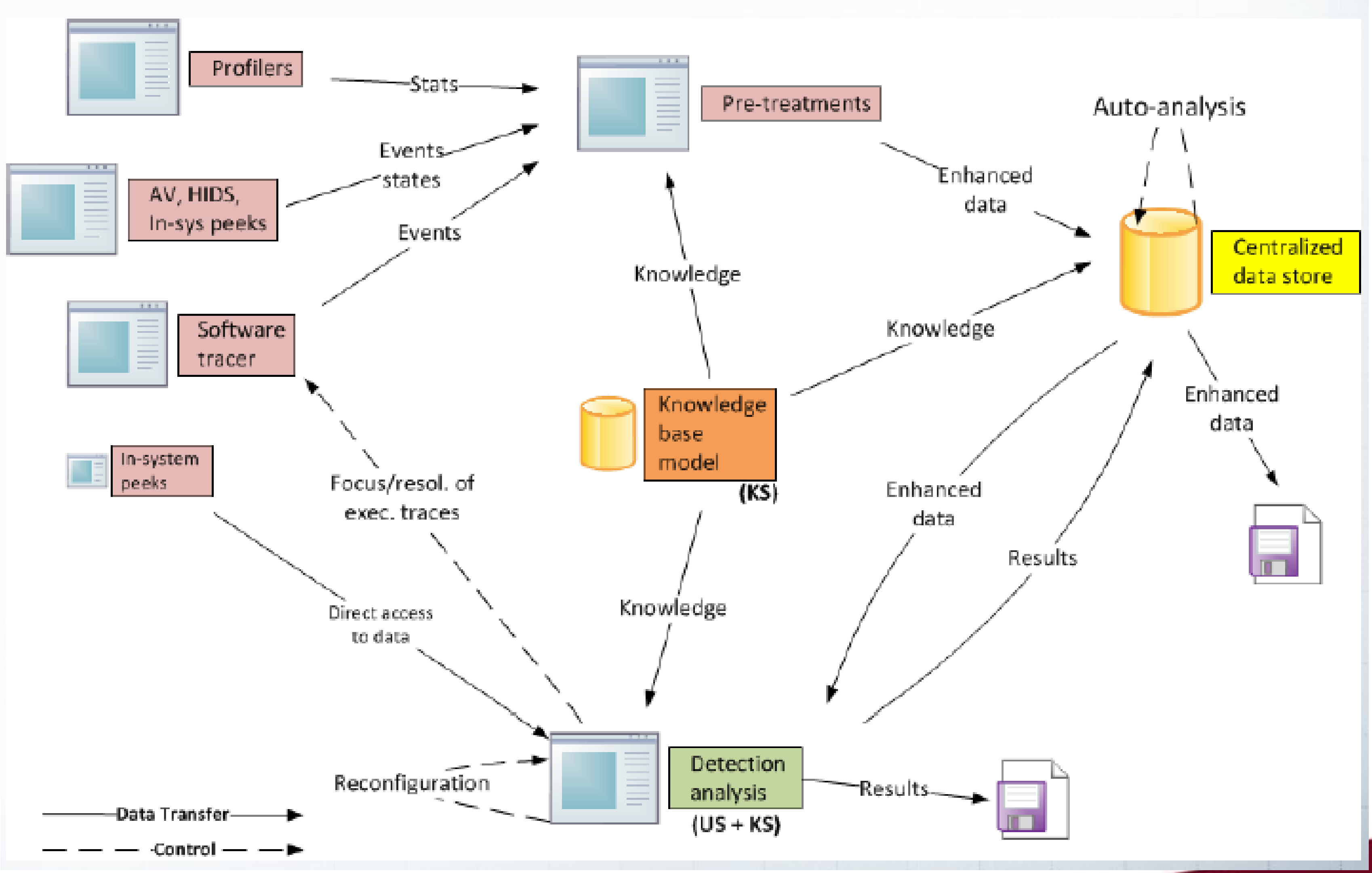
Analysis of the behaviour of software systems: a simplified view



… a bit more complex view



…a very complex tracing infrastructure



M. Couture, A. Hamou-Lhadj, M. Dagenais, A. Goel, "Online surveillance of computerized systems – Analysis of current and future needs,"
In Proc. of the NATO Symposium on Information Assurance and Cyber Defence (IST-112), Quebec City, Quebec, 2012.

Software tracing in industrial projects

Project 1: Tracing and monitoring tools for multi-core systems

Project 2: Host-based anomaly detection systems

Project 3: Tracing, debugging and configuration of avionic systems

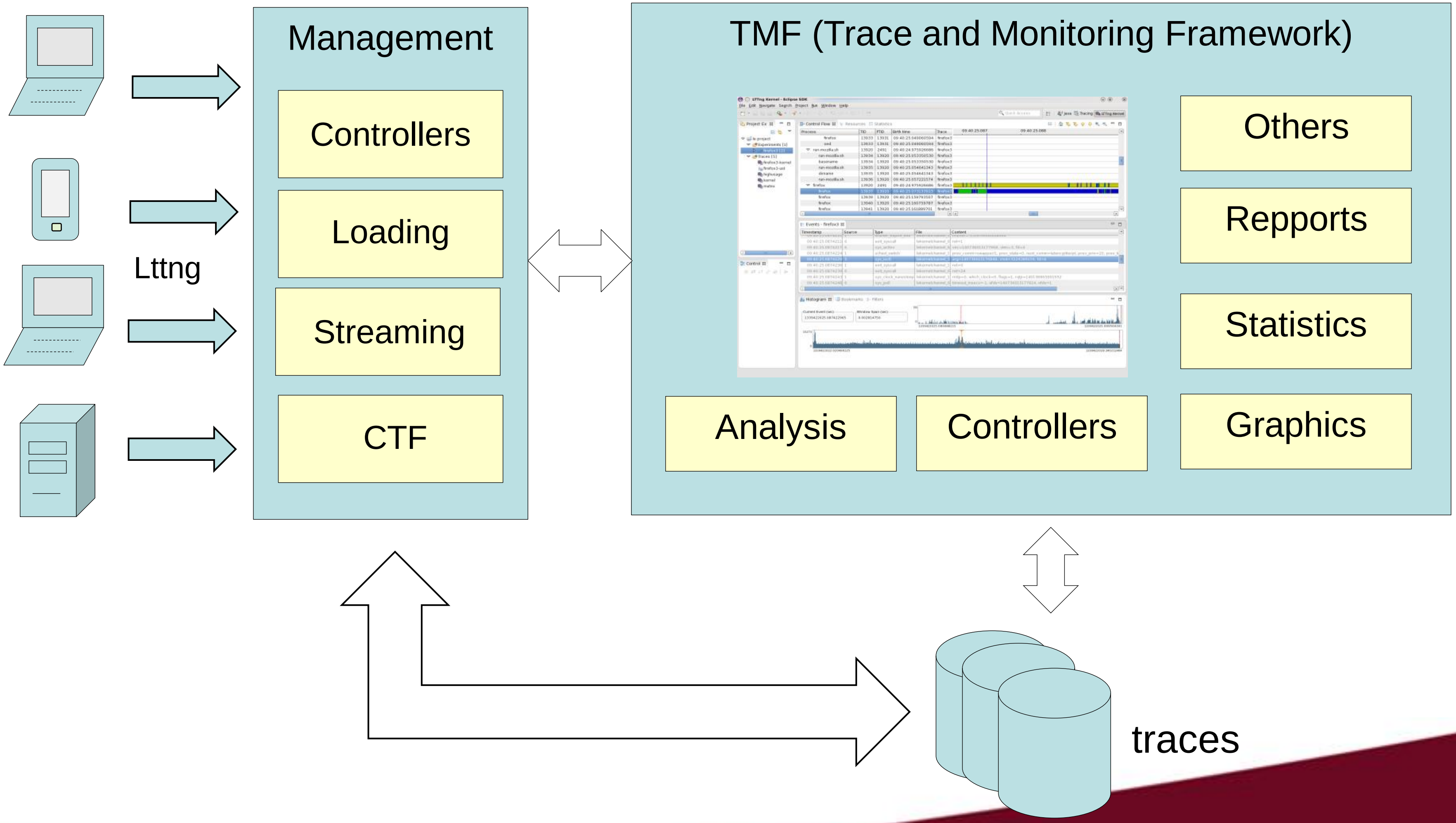
Tracing and monitoring tools for multi-core systems

To develop techniques and tools for the generation and analysis of execution traces of multi-core systems with minimum disturbance and overhead

ERICSSON

DEFENCE DÉFENSE

Project vision



Lttng

Management

Controllers

Loading

Streaming

CTF

TMF (Trace and Monitoring Framework)

Others

Repports

Statistics

Analysis

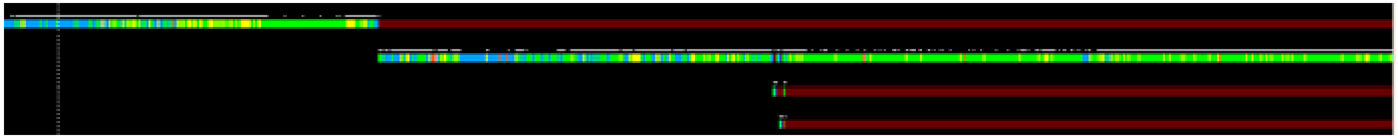
Controllers

Graphics

traces

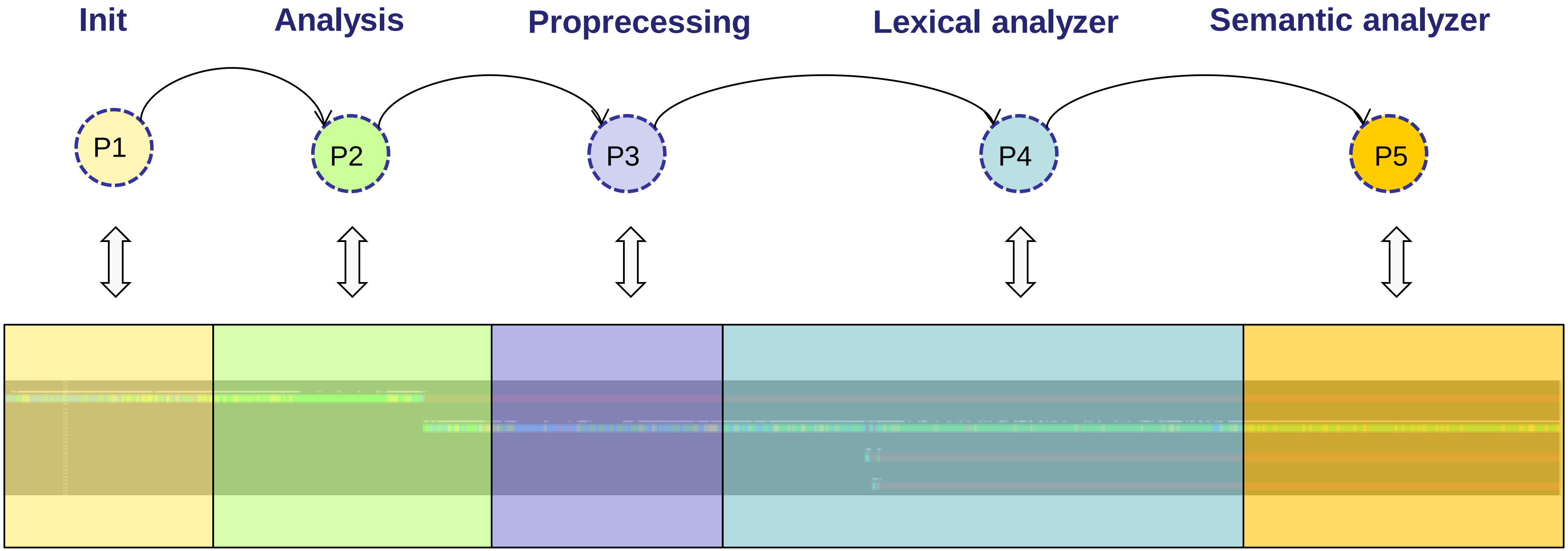
Trace analysis: example and motivation

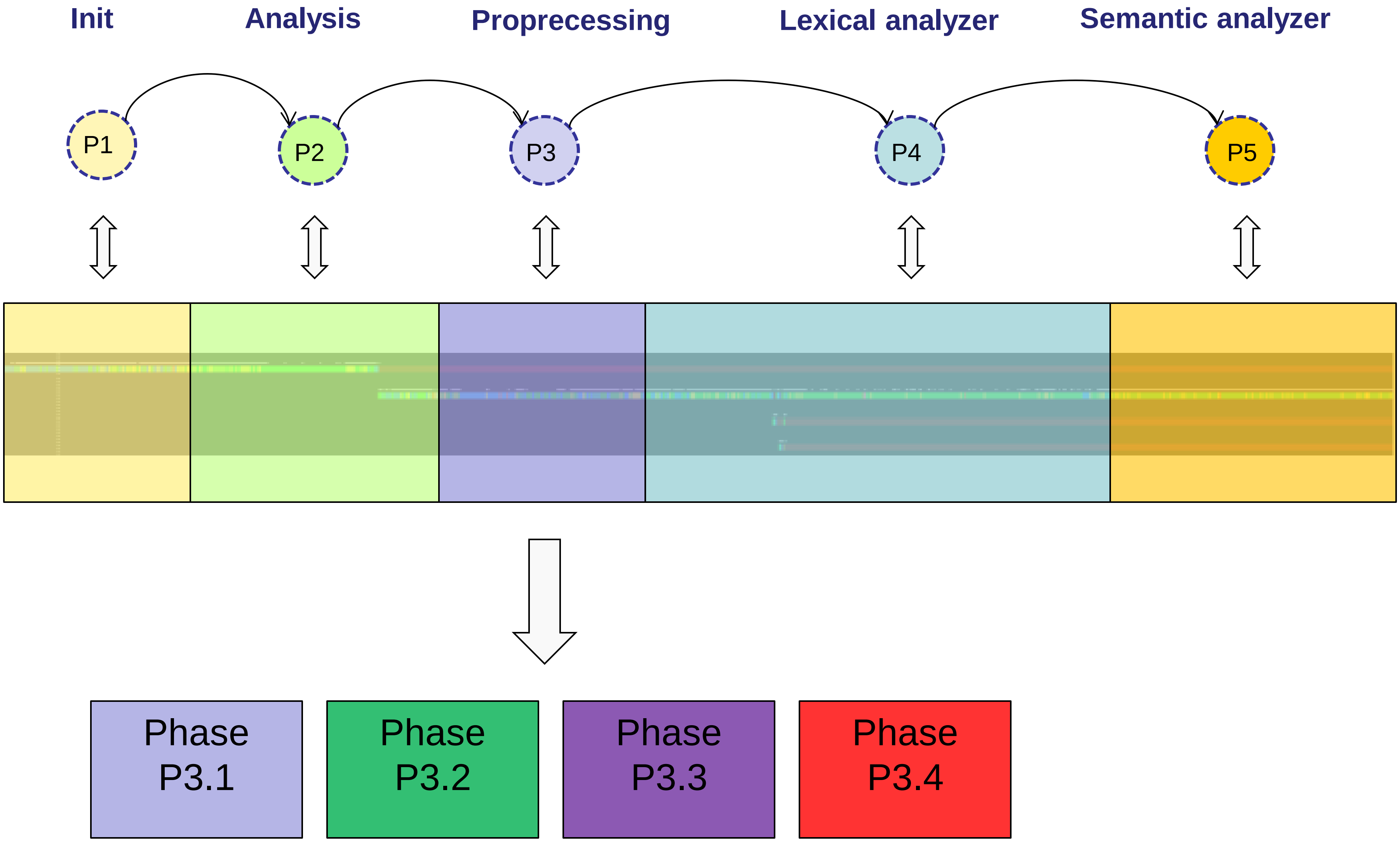
- Let’s take a look at a trace generated from running a compiler
- The trace will most likely contains the following phases: Parsing, preprocessing, lexical analyzer, semantic analyzer, etc.
- A typical trace analysis tool will show the following:



How do we know what happens where?

Automatic extraction of execution phases





Init

Analysis

Proprecessing

Lexical analyzer

Semantic analyzer

P1

P2

P3

P4

P5

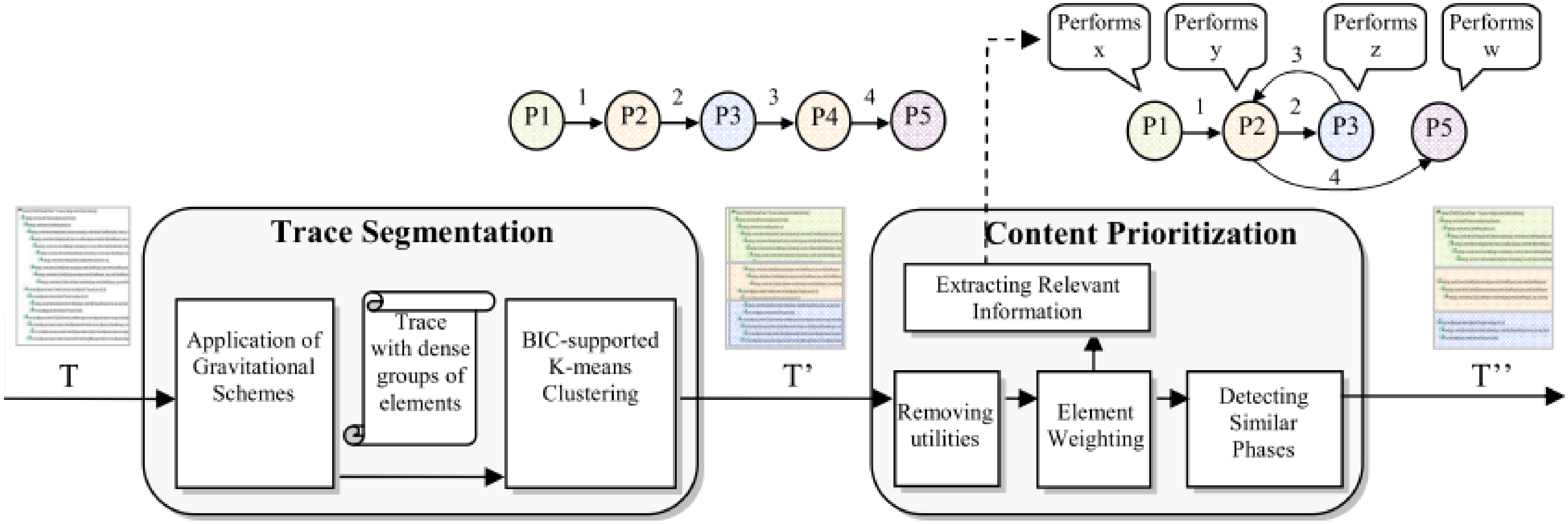
Phase P3.1

Phase P3.2

Phase P3.3

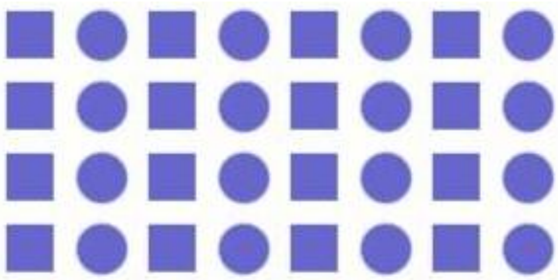
Phase P3.4

TRASER: TRAce Segmentation through Event Repositioning

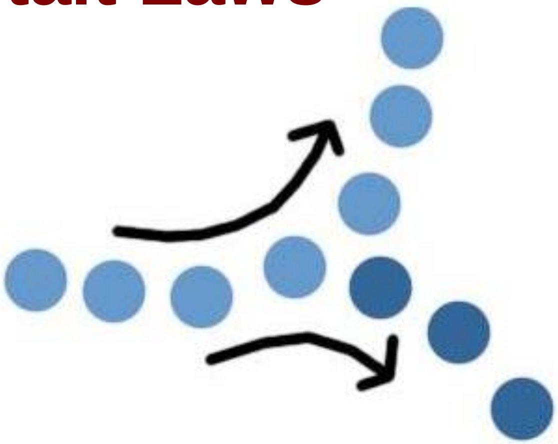


H. Pirzadeh, A. Hamou-Lhadj, "A Software Behaviour Analysis Framework Based on the Human Perception System," ICSE (NIER Track), pp. 948 - 951, 2012.
H. Pirzadeh, A. Hamou-Lhadj, M. Shah, "Exploiting Text Mining Techniques in the Analysis of Execution Traces," ICSM'12, pp. 223-232, 2012.

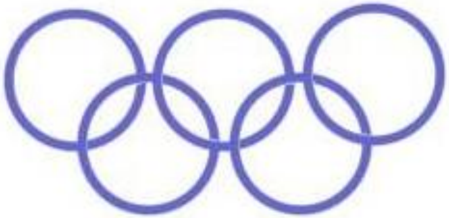
Inspiration: Gestalt Laws



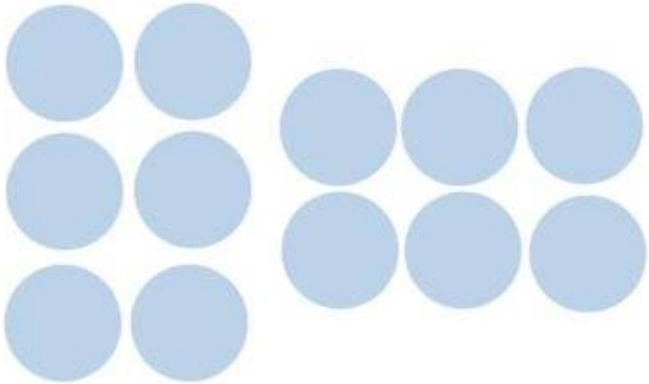
Law of Similarity



Law of Continuity

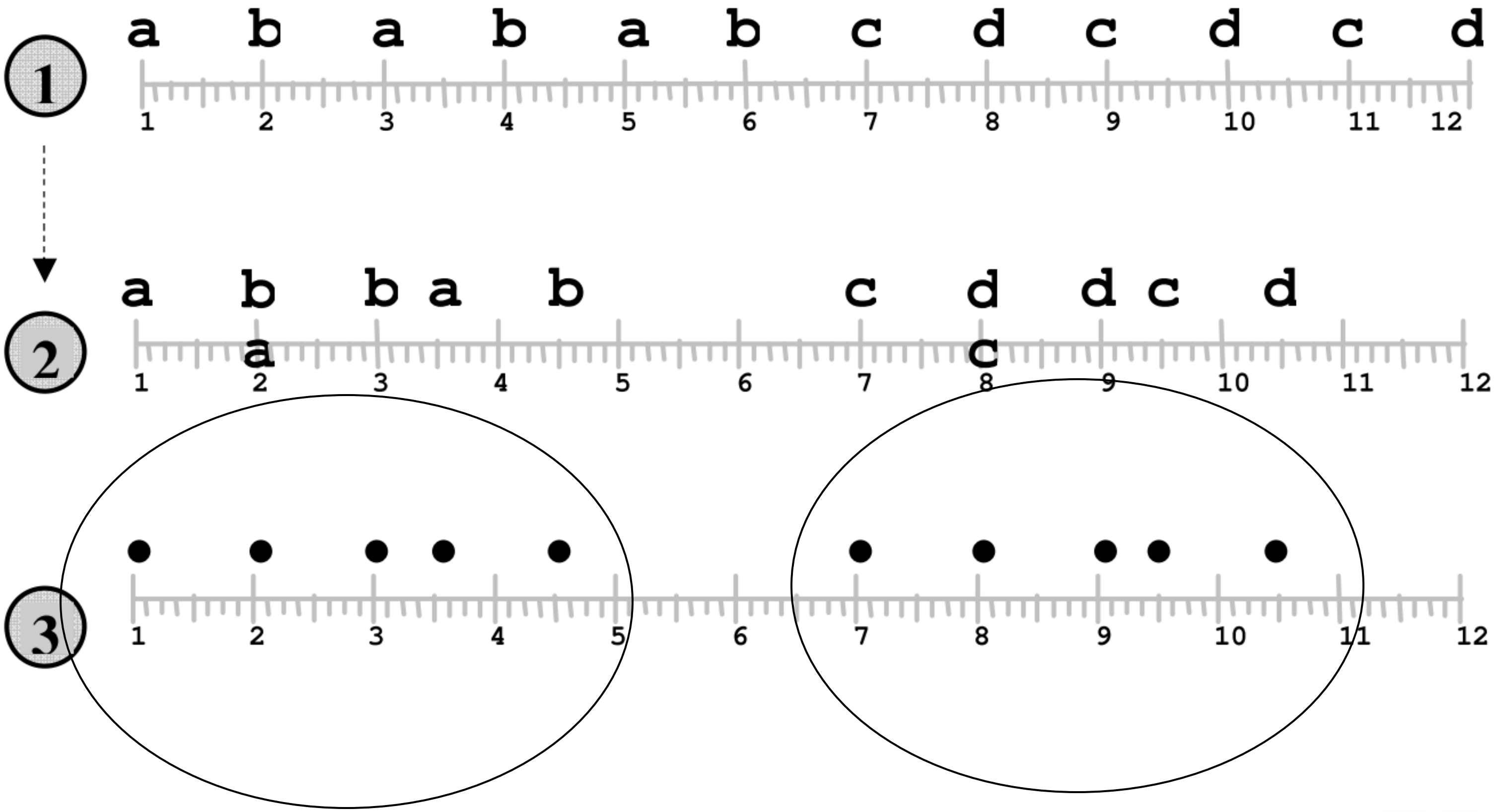


Law of Pragnanz

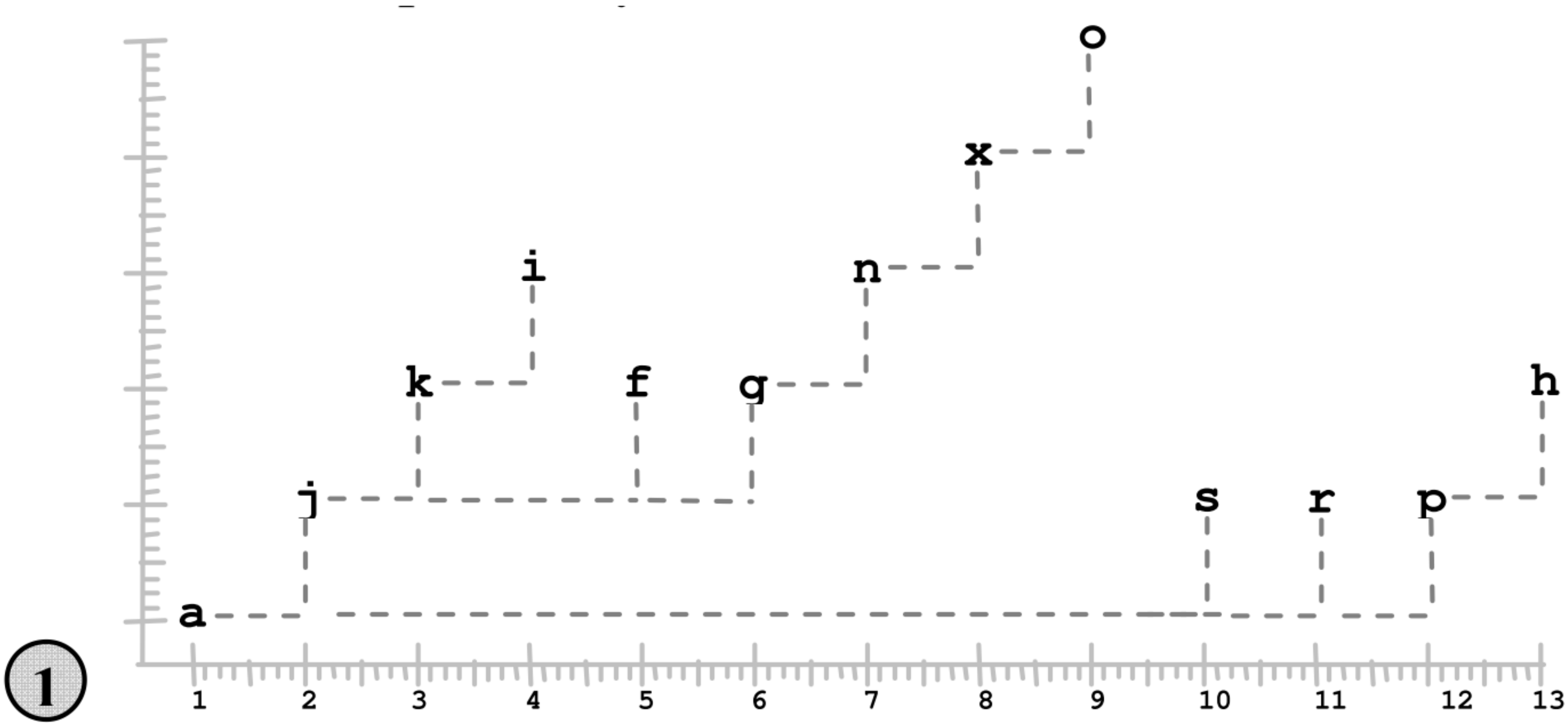


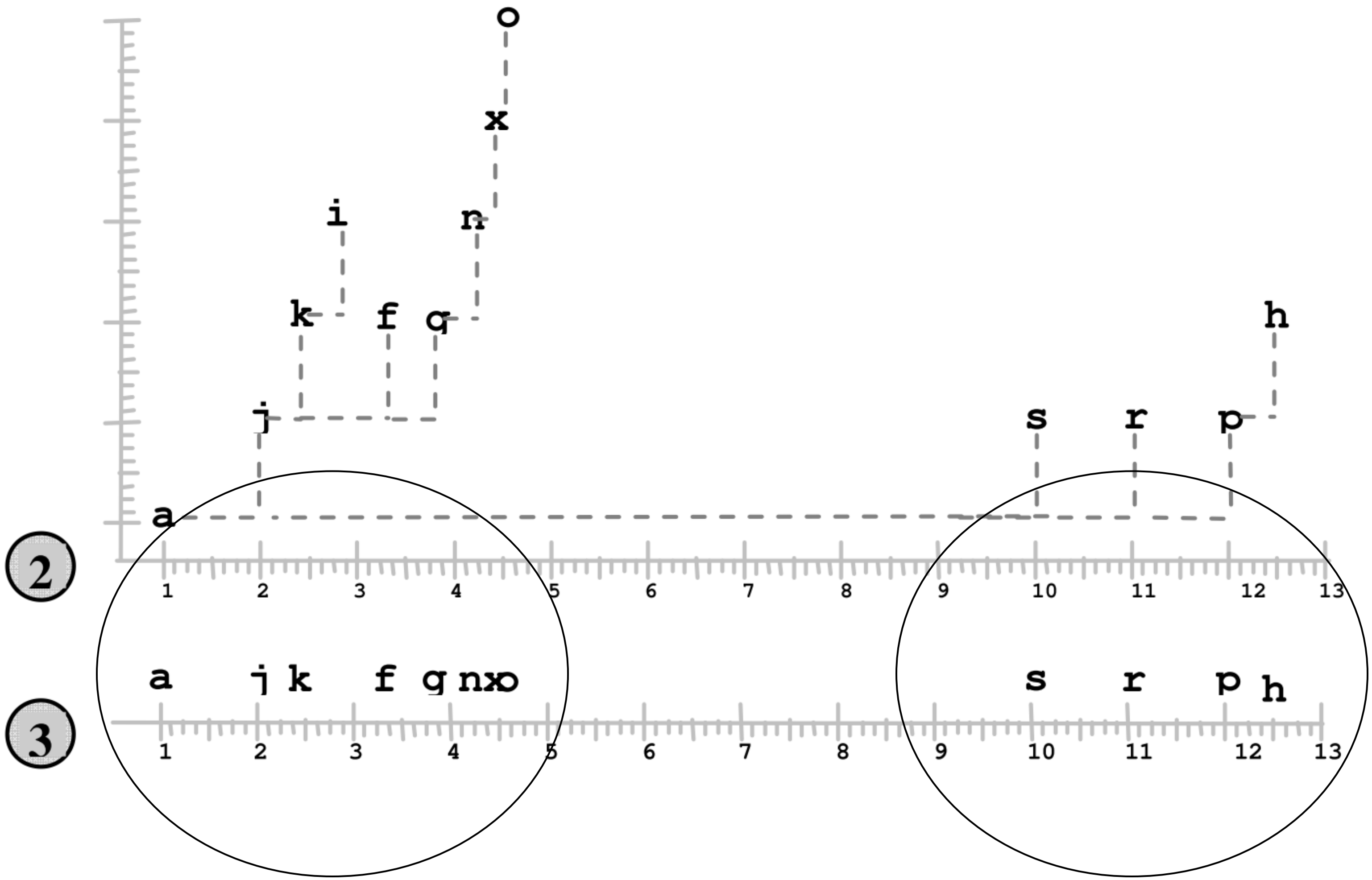
Law of Proximity

Repositioning of trace events based on similarity

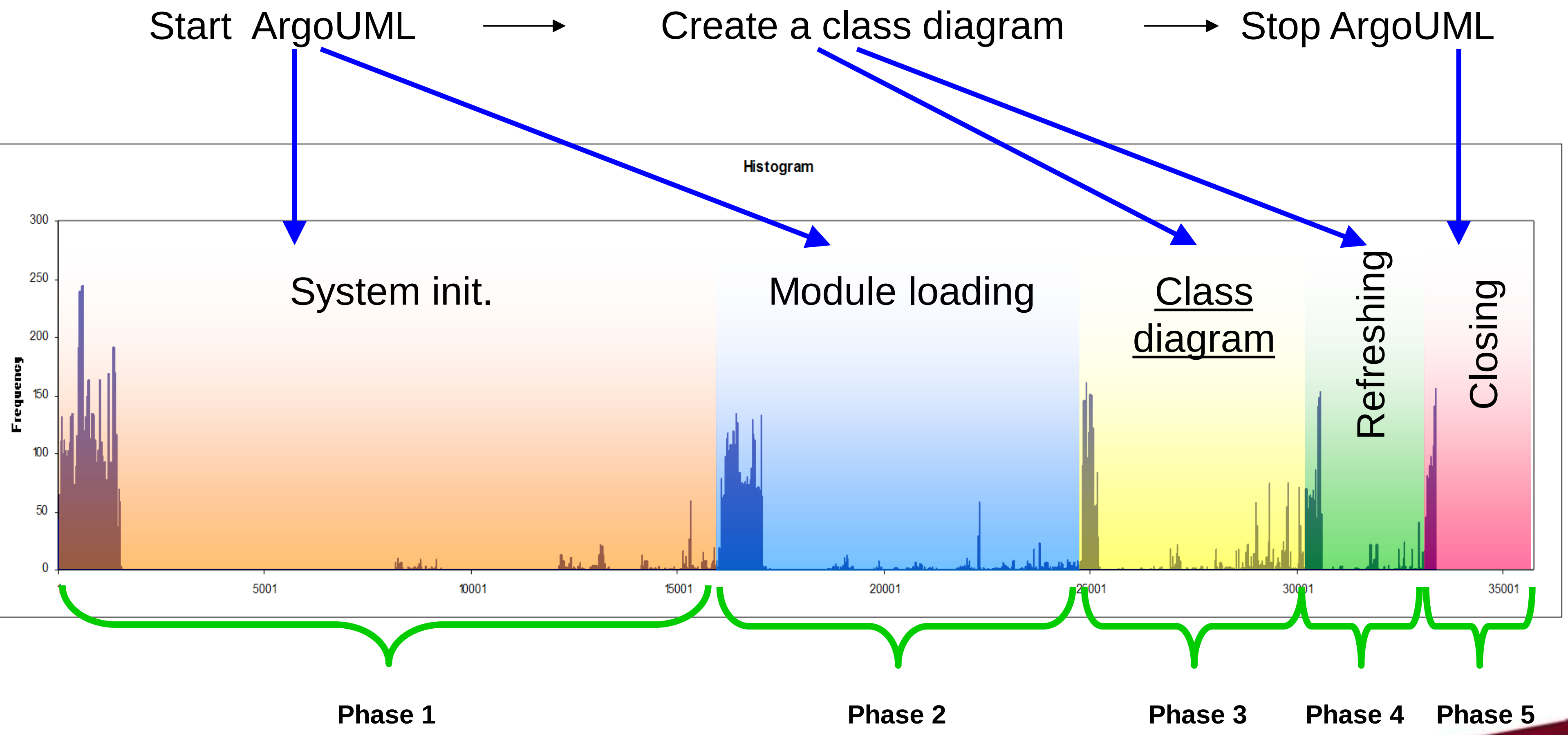


Repositioning of trace events based on continuity




o
x
i
n
k
f
q
h
j
s
r
p
a
2
1
2
3
4
5
6
7
8
9
10
11
12
13
a
j k
f
g nxo
s
r
p
h
3
1
2
3
4
5
6
7
8
9
10
11
12
13

Application: An ArgoUML trace with hundreds of thousands of calls



Start ArgoUML → Create a class diagram → Stop ArgoUML

System init.

Module loading

Class diagram

Refreshing

Closing

Phase 1

Phase 2

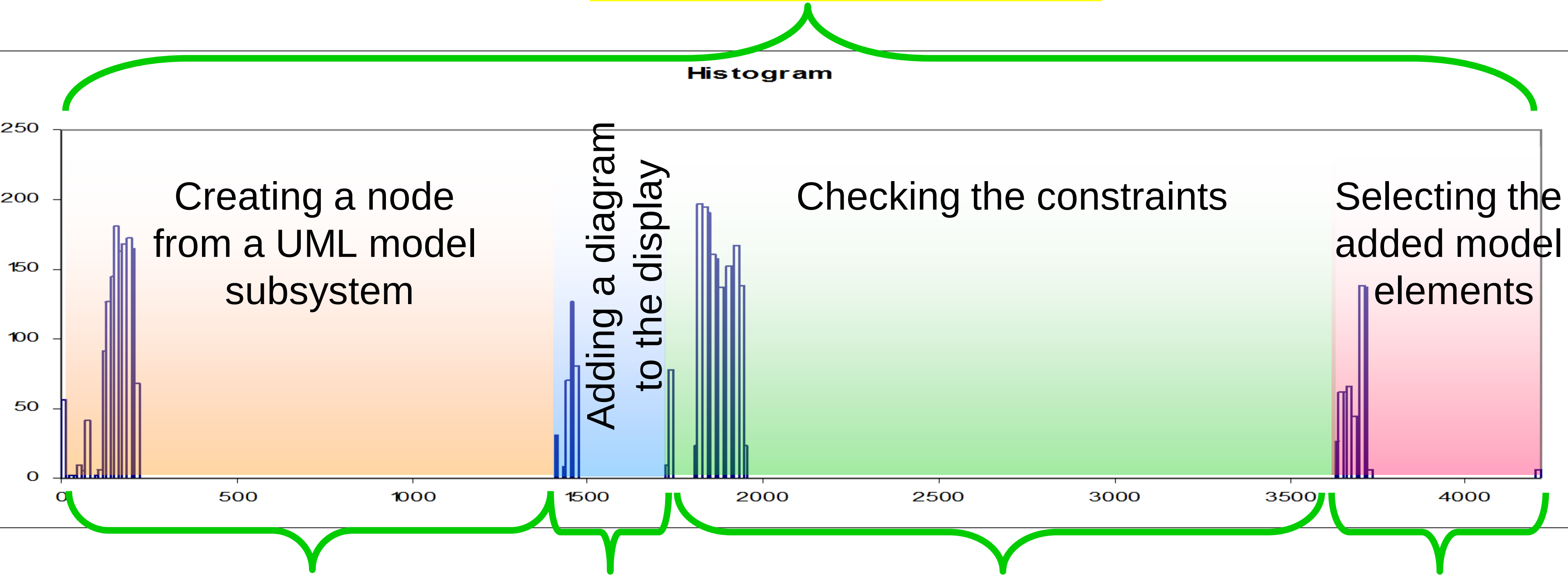
Phase 3

Phase 4

Phase 5

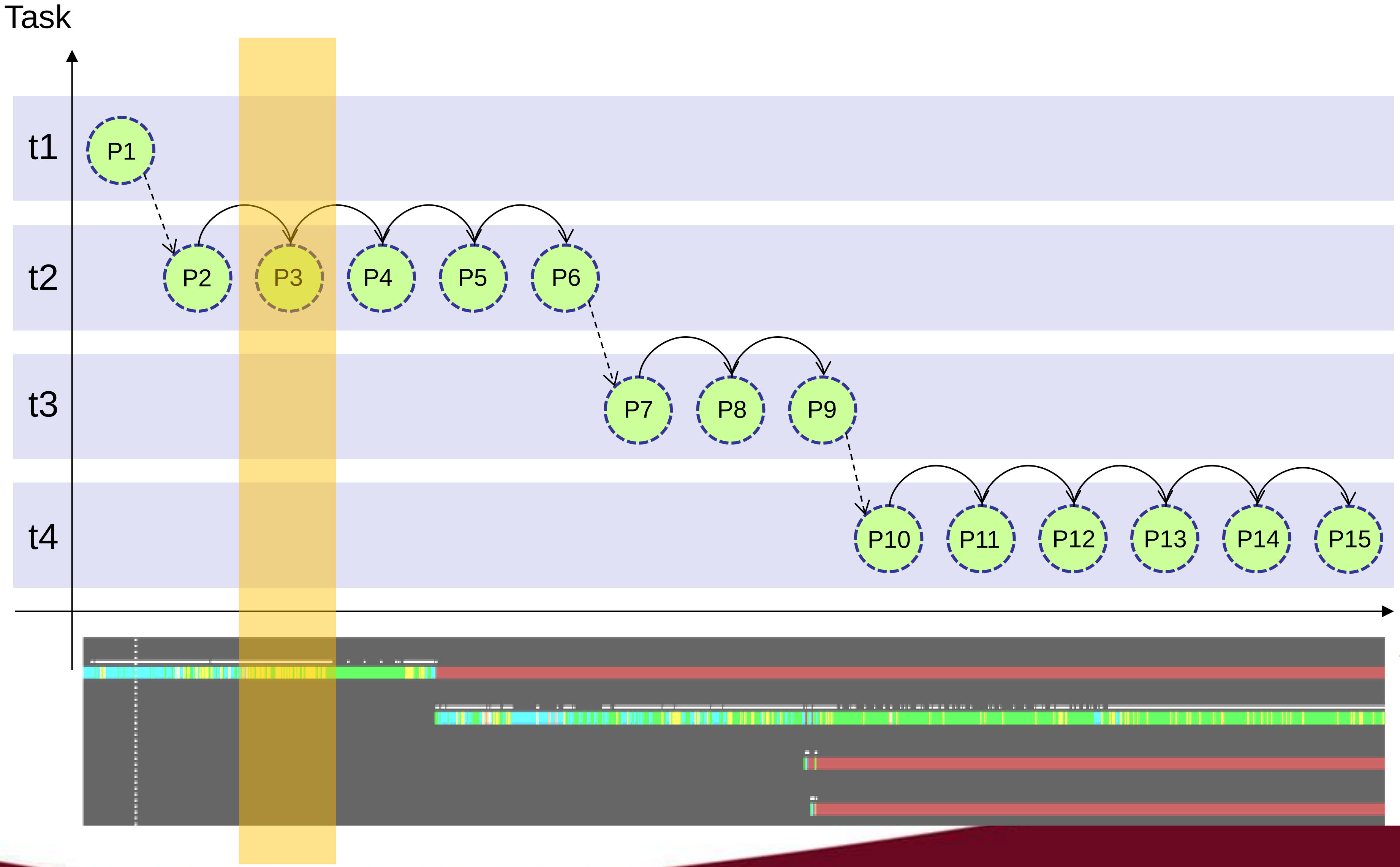
Start ArgoUML → Create a class diagram → Stop ArgoUML

Phase 3: Adding a class diagram

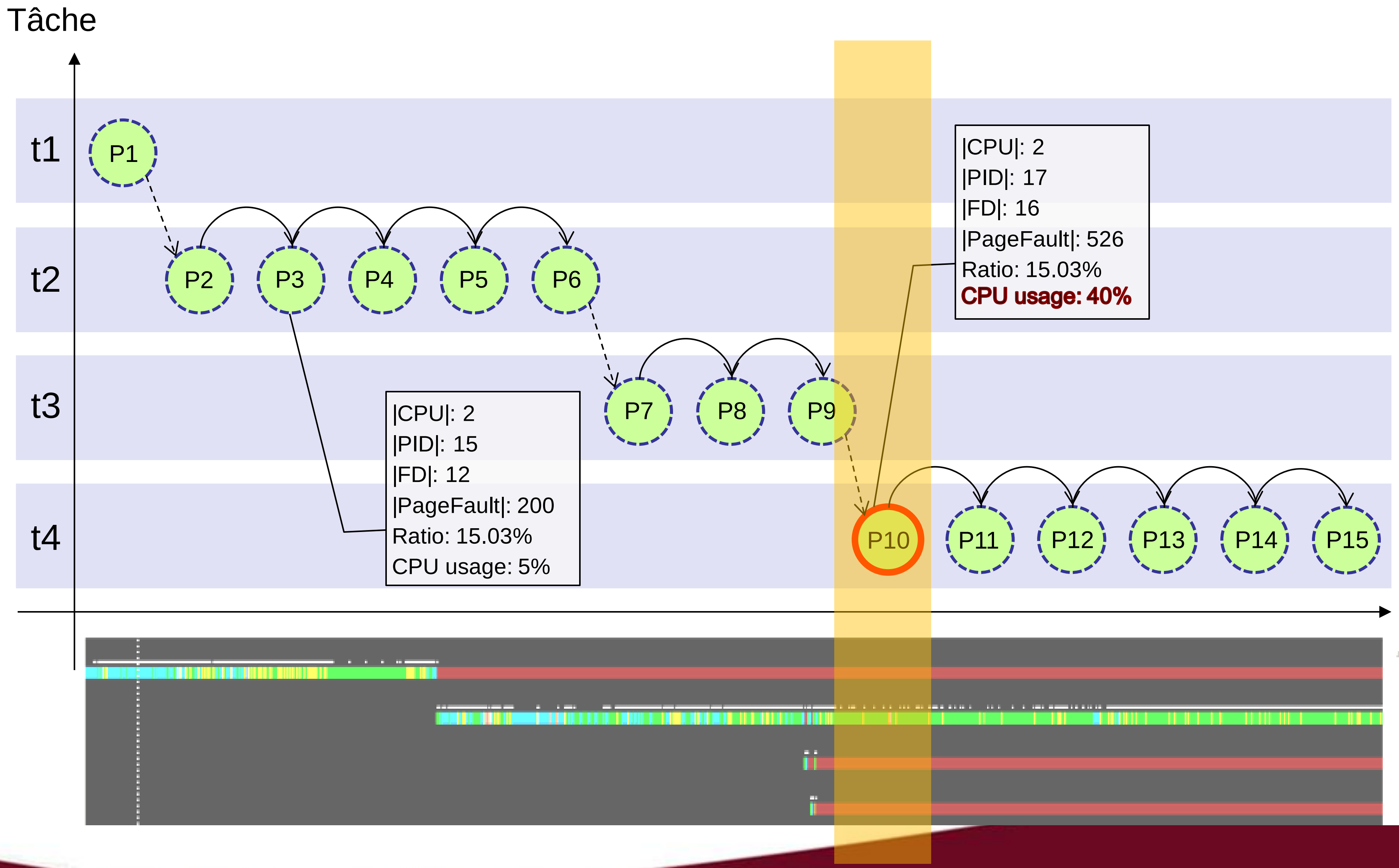


Phase 3.1 **Phase 3.2** **Phase 3.3** **Phase 3.4**

Aligning user and kernel spaces



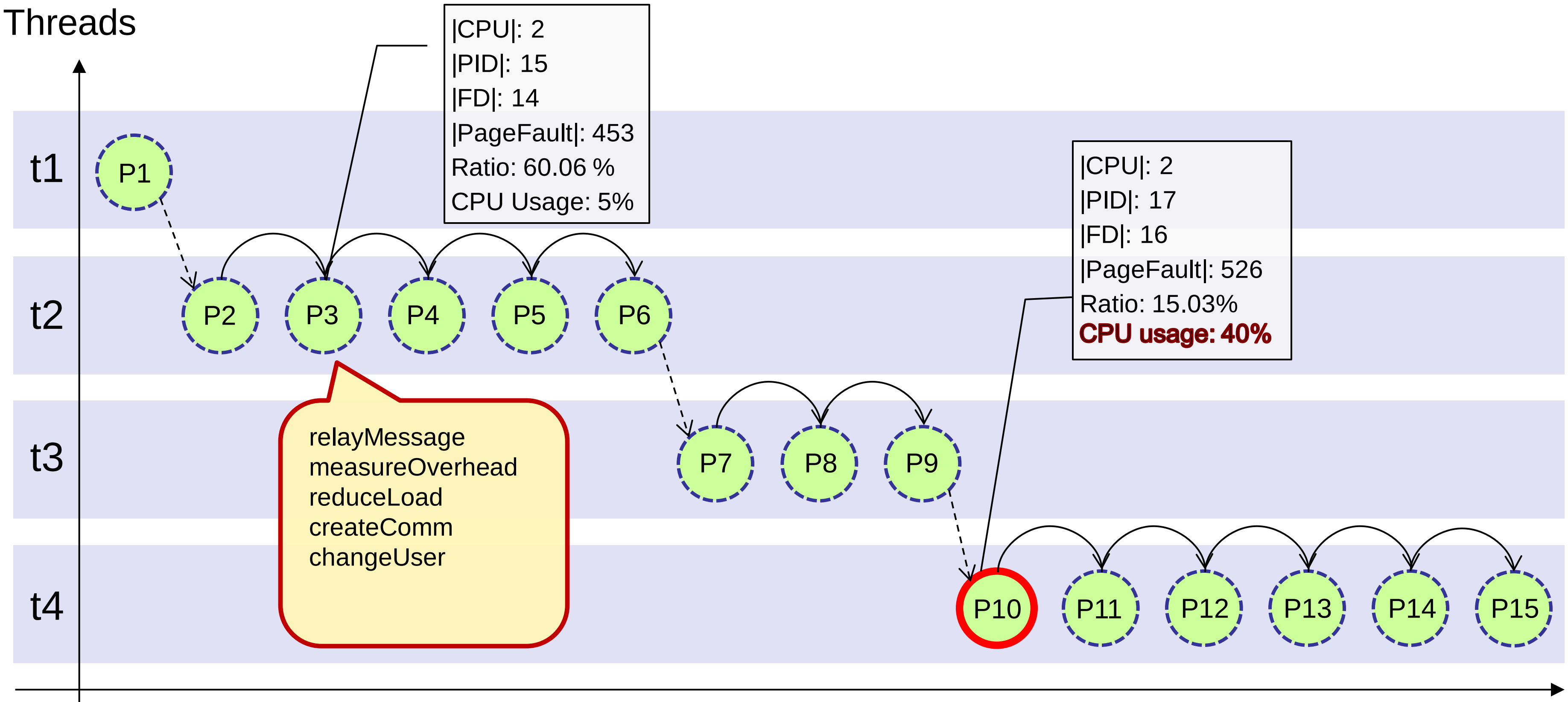
Adding state information



|CPU|: 2
|PID|: 15
|FD|: 12
|PageFault|: 200
Ratio: 15.03%
CPU usage: 5%

|CPU|: 2
|PID|: 17
|FD|: 16
|PageFault|: 526
Ratio: 15.03%
CPU usage: 40%

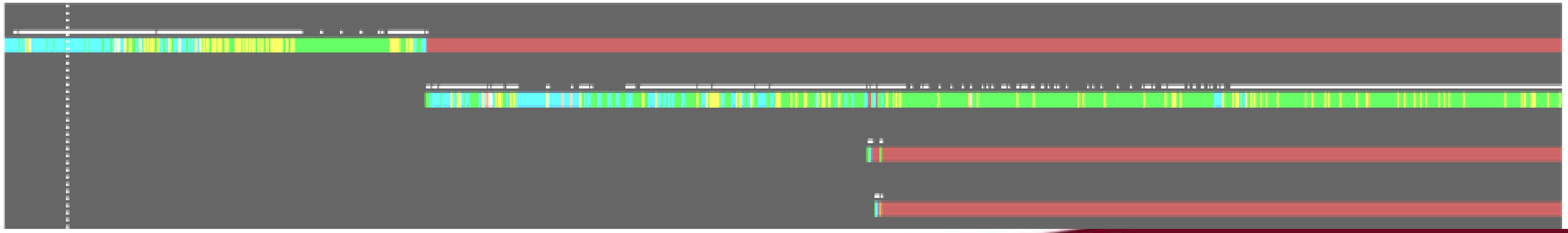
Identifying most relevant content



|CPU|: 2
|PID|: 15
|FD|: 14
|PageFault|: 453
Ratio: 60.06 %
CPU Usage: 5%

|CPU|: 2
|PID|: 17
|FD|: 16
|PageFault|: 526
Ratio: 15.03%
CPU usage: 40%

relayMessage
measureOverhead
reduceLoad
createComm
changeUser



Software tracing in industrial projects

Project 1: Tracing and monitoring tools for multi-core systems

Project 2: Host-based anomaly detection systems

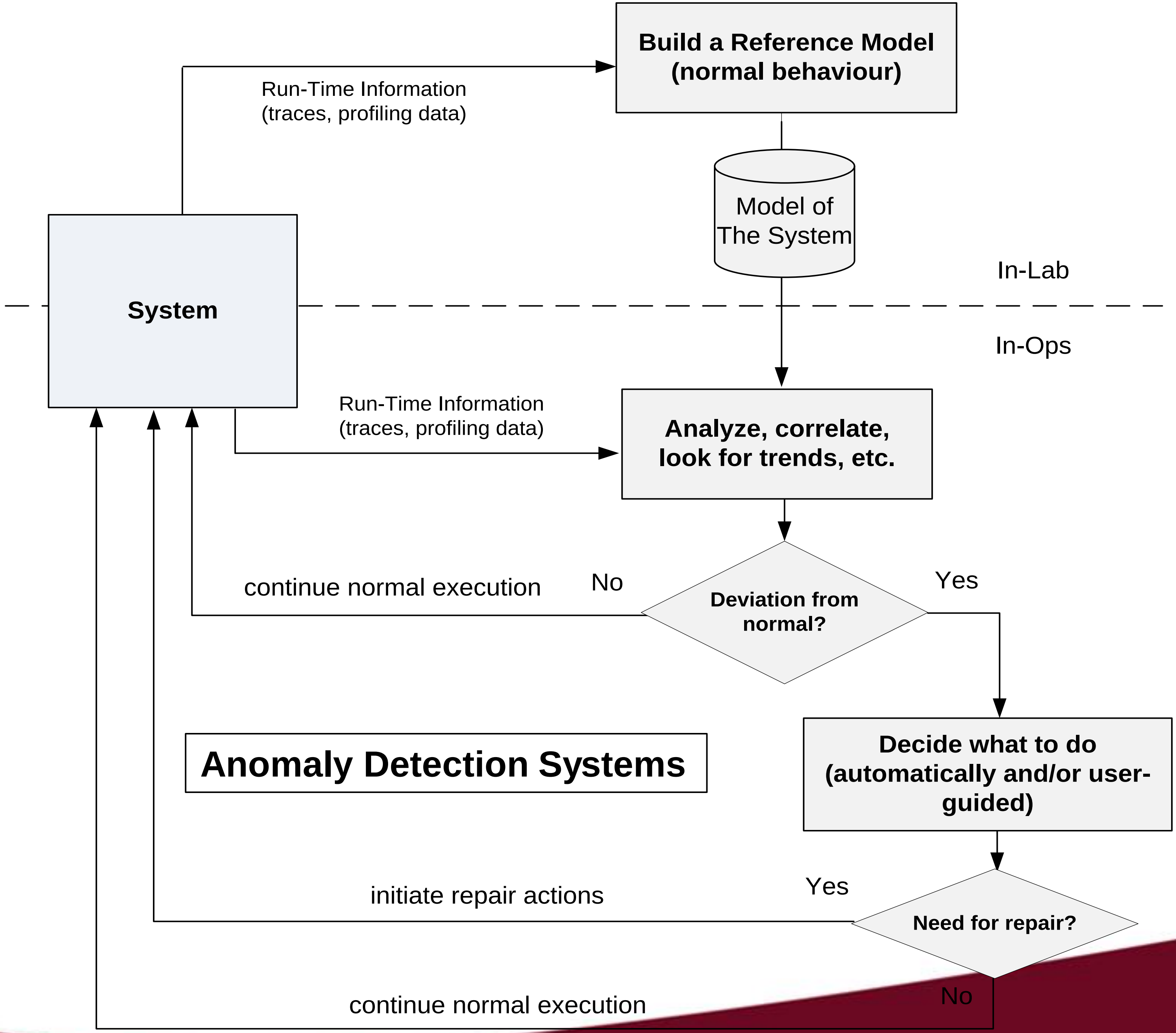
Project 3: Tracing, debugging and configuration of avionic systems

Host-based Anomaly Detection-
Advanced Host-Level Surveillance

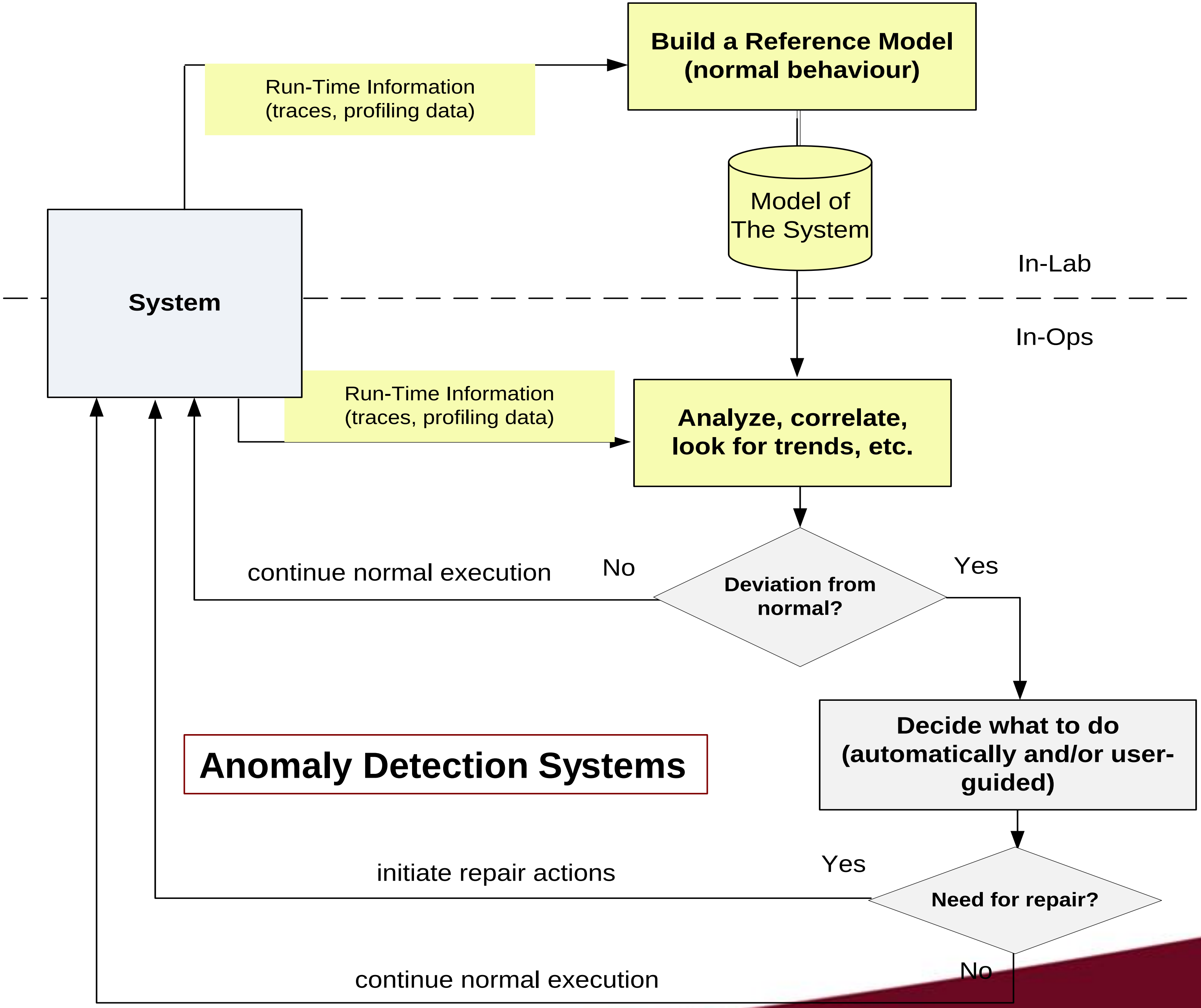
Develop modular, adaptive, and scalable Anomaly Detection Systems (ADS) at the level of system call traces; reduce false positives (alarms) and improve the true positives

DEFENCE RD DÉFENSE

ERICSSON


Build a Reference Model
(normal behaviour)
Run-Time Information
(traces, profiling data)
Model of
The System
In-Lab
System
In-Ops
Run-Time Information
(traces, profiling data)
Analyze, correlate,
look for trends, etc.
continue normal execution
No
Deviation from
normal?
Yes
Anomaly Detection Systems
Decide what to do
(automatically and/or user-guided)
initiate repair actions
Yes
Need for repair?
No
continue normal execution

Anomaly Detection Systems



Build a Reference Model (normal behaviour)

Run-Time Information (traces, profiling data)

Model of The System

System

In-Lab

In-Ops

Run-Time Information (traces, profiling data)

Analyze, correlate, look for trends, etc.

Deviation from normal?

No

continue normal execution

Yes

Decide what to do (automatically and/or user-guided)

Need for repair?

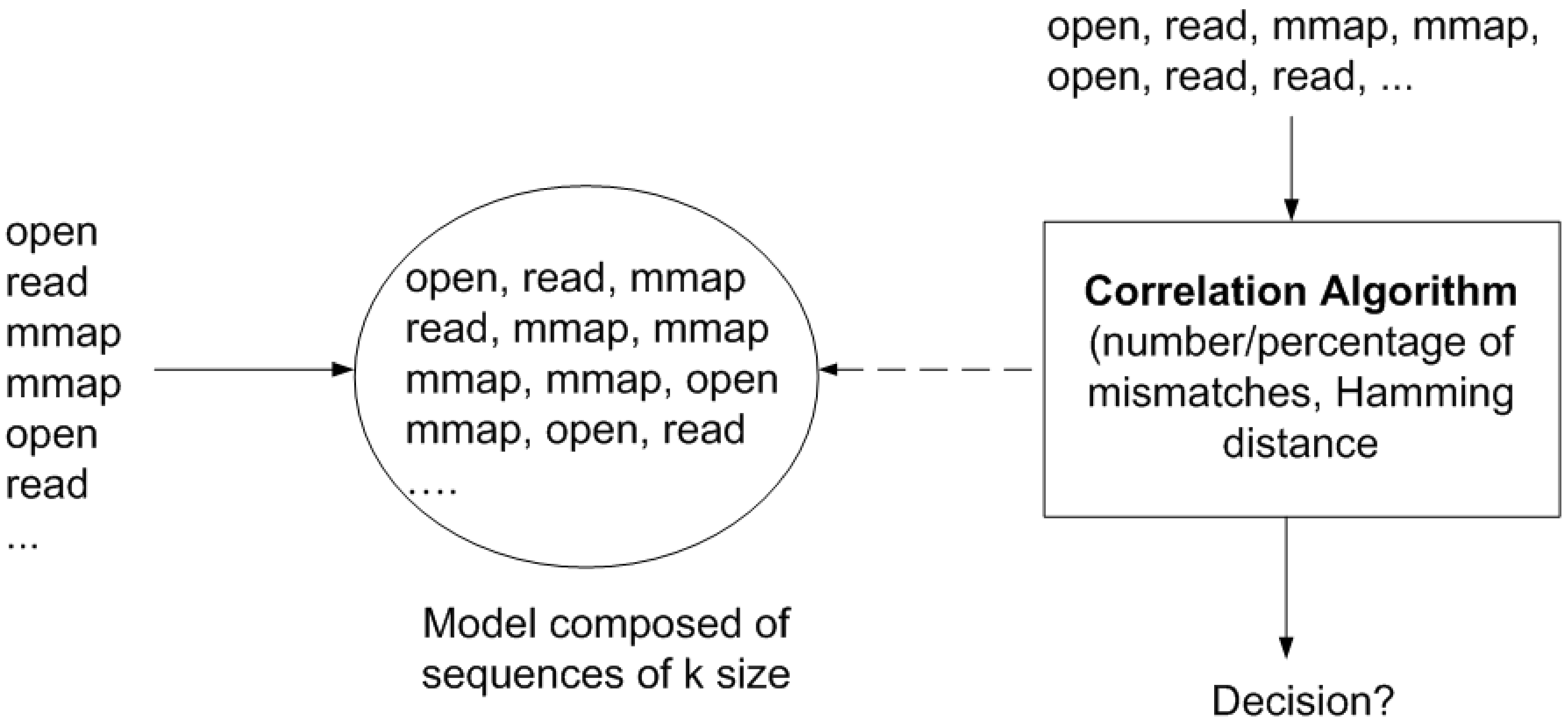
Yes

initiate repair actions

No

continue normal execution

Example: Sliding Approach (STIDE)



open, read, mmap, mmap,
open, read, read, ...

open
read
mmap
mmap
open
read
...

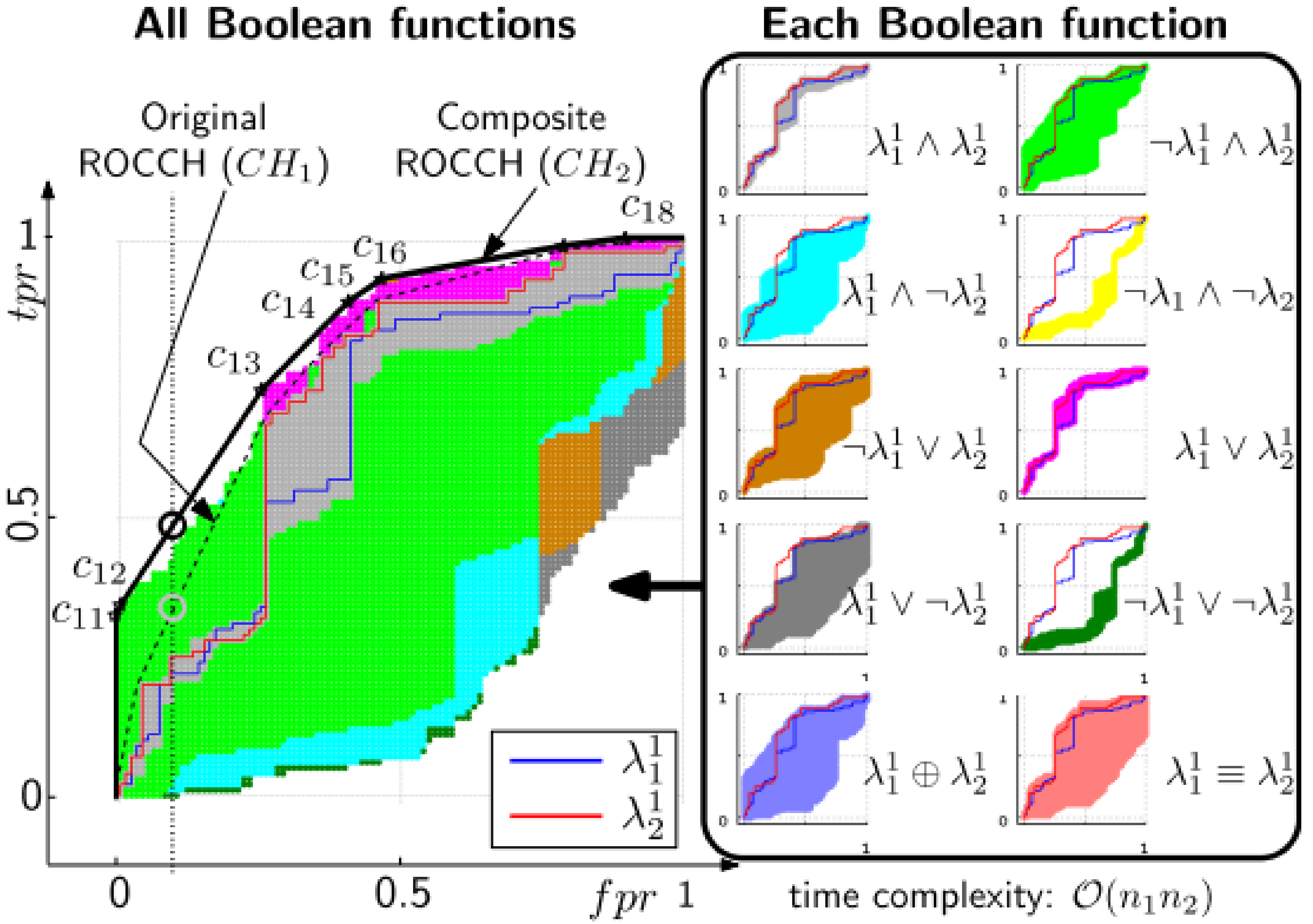
open, read, mmap
read, mmap, mmap
mmap, mmap, open
mmap, open, read
….

Model composed of sequences of k size

Correlation Algorithm (number/percentage of mismatches, Hamming distance

Decision?

Incremental Boolean Combination of HMMs



Kernel State Modeling (KSM)

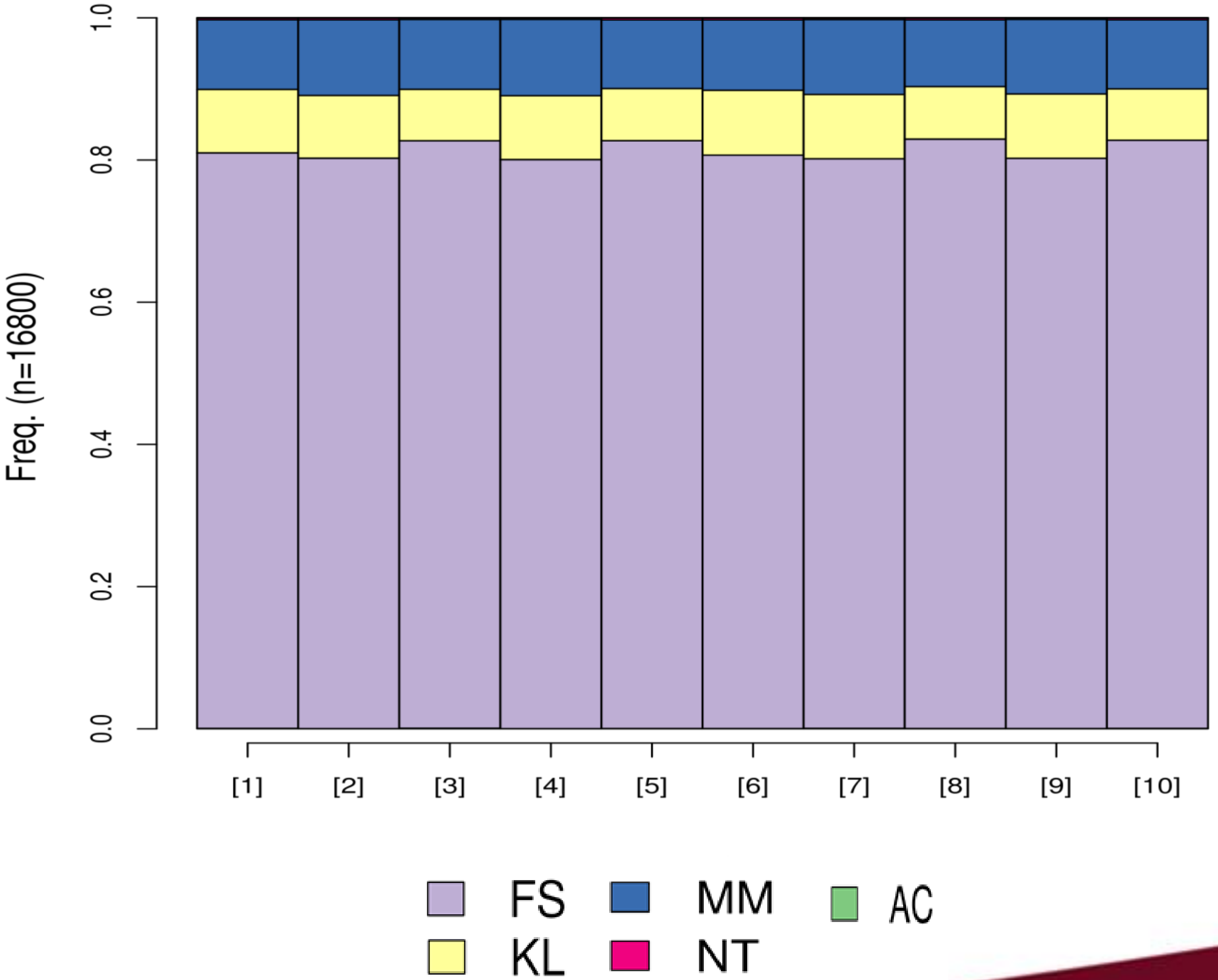
- KSM is an anomaly detection technique
  - Transforms system calls into kernel modules, called states
  - Detects anomalies at the level of interaction among kernel states
  - Reduces data space used in training and testing
  - Favors efficiency while keeping accuracy

Transforming System Calls into States of Kernel Modules

State	Module in Linux Source Code	# of System Calls
AC	Architecture	10
FS	File System	131
IPC	Inter Process Communication	7
KL	Kernel	127
MM	Memory Management	21
NT	Networking	2
SC	Security	3
UN	Unknown	37

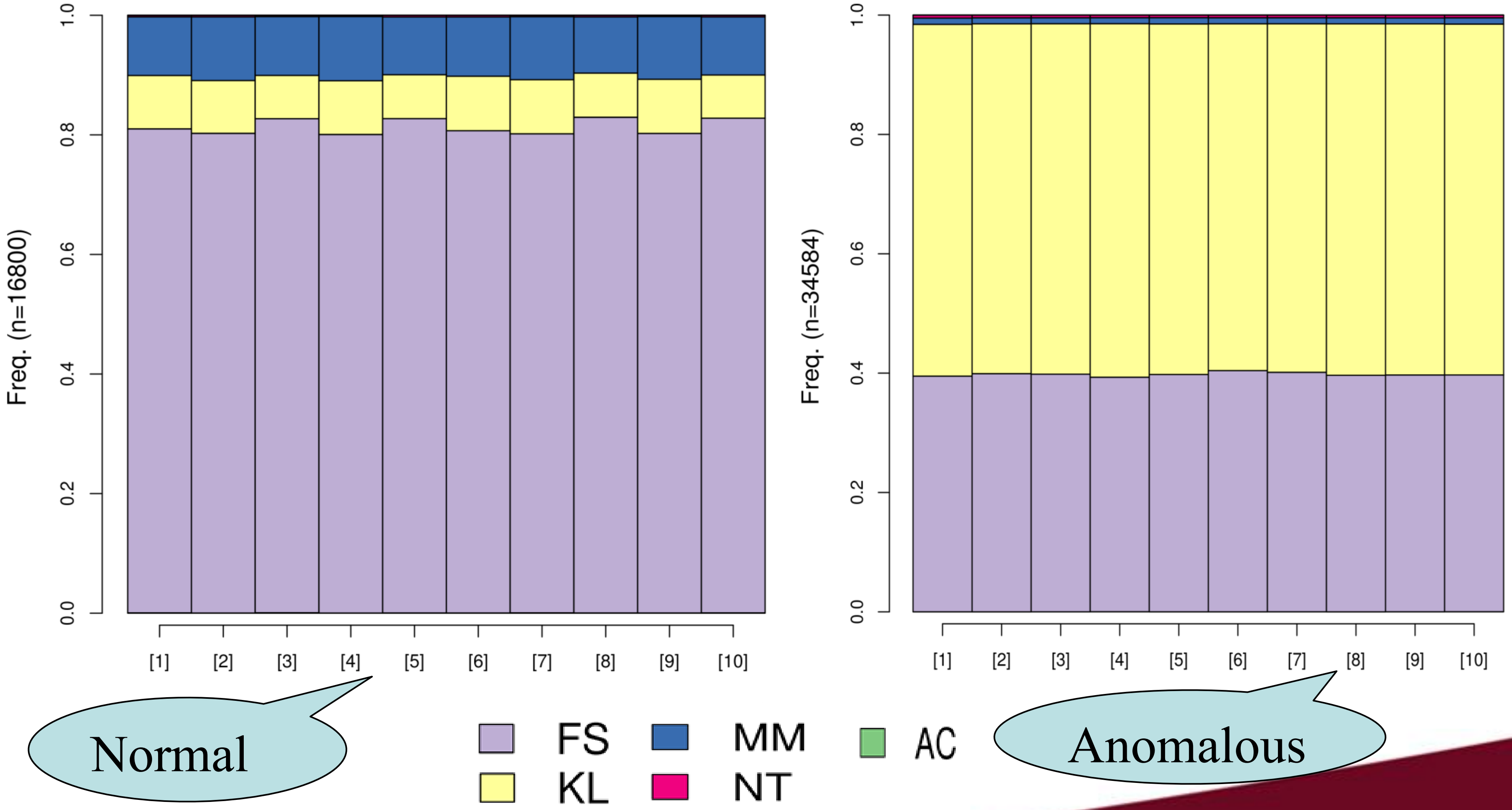
[Source]: http://syscalls.kernelgork.com

KSM and Density Plots



Density
Plot

Anomaly Detection in Firefox



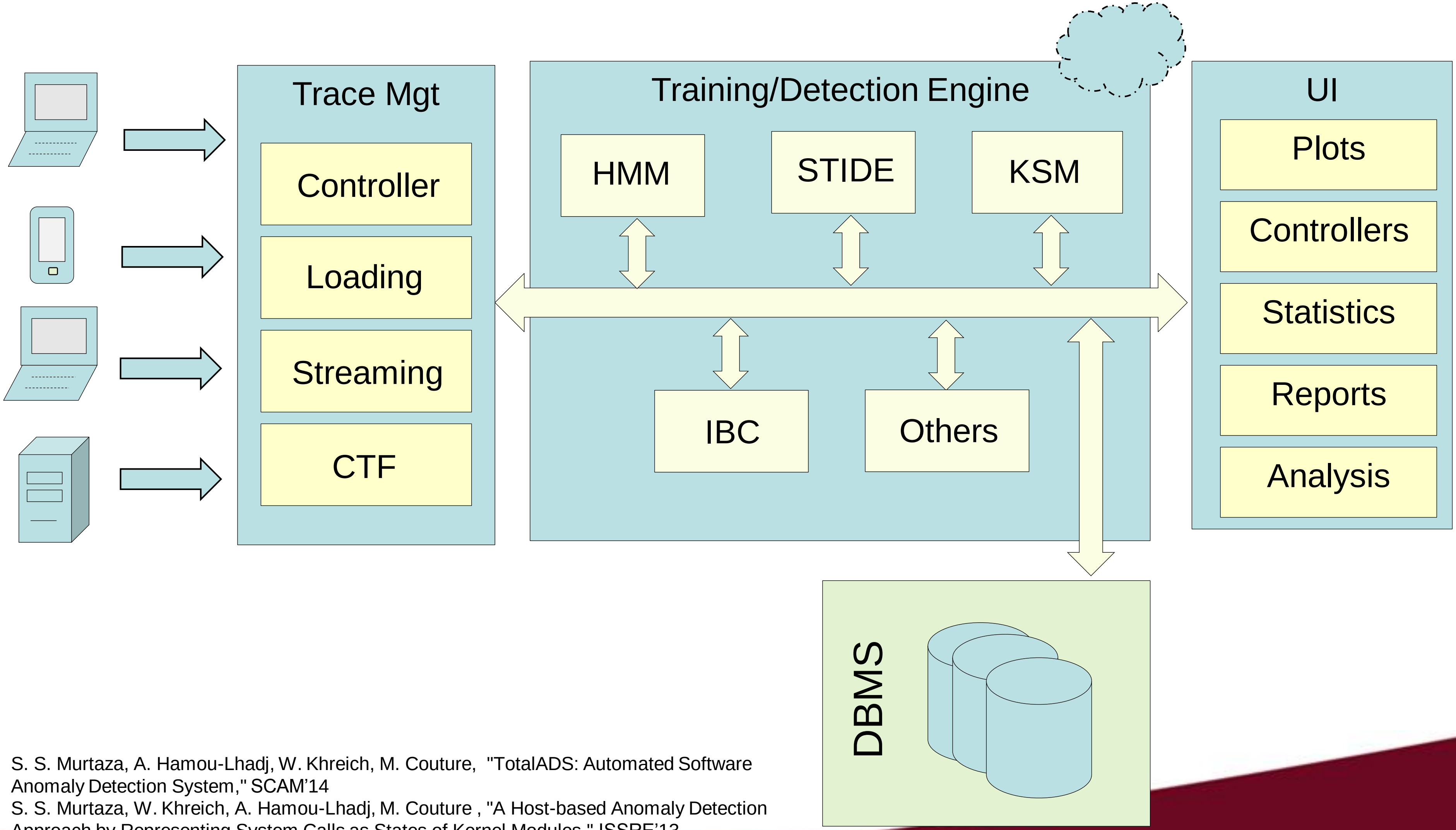
Normal

Anomalous

Case Study: Execution Time

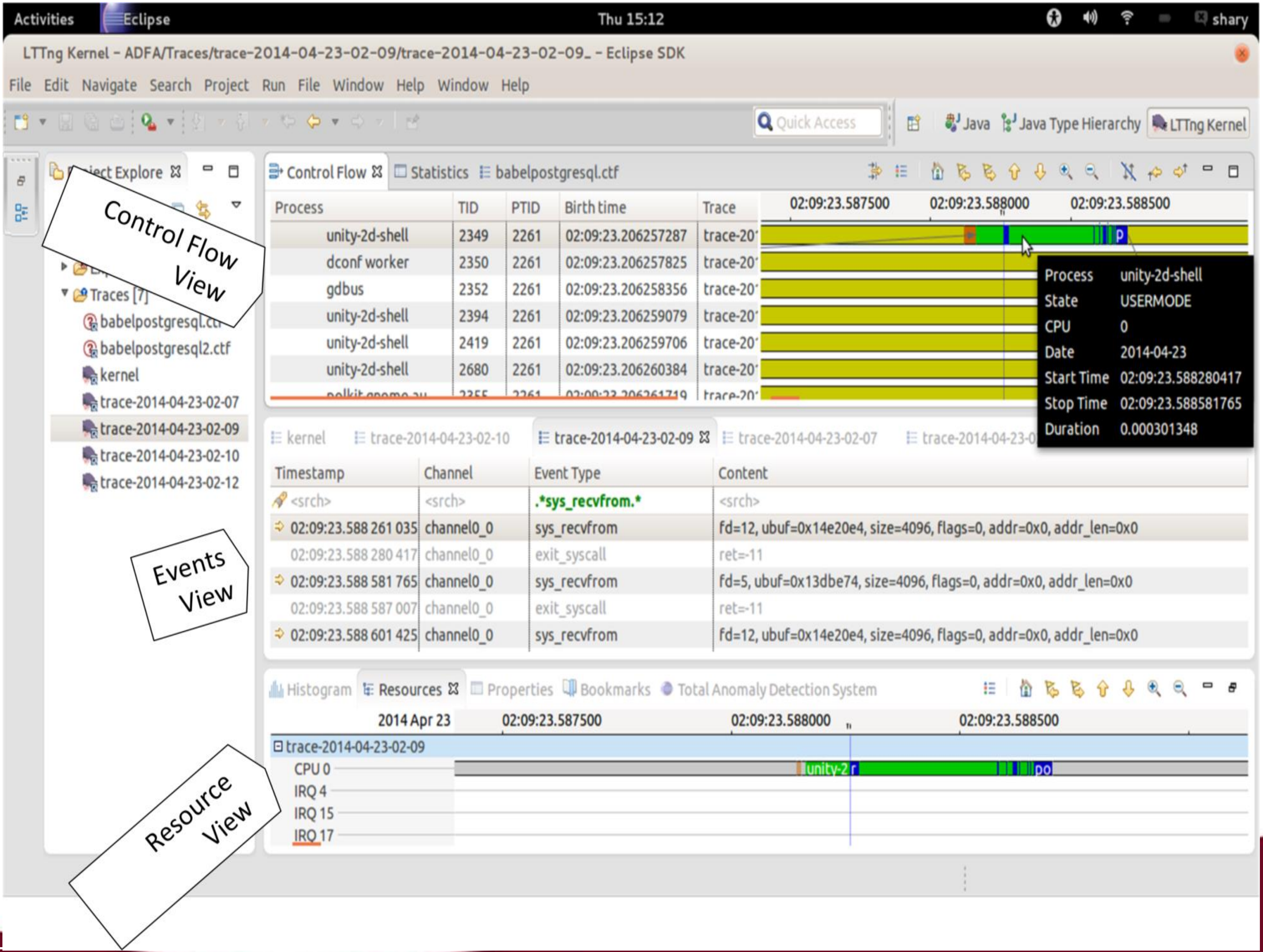
	Size of All Traces	KSM	Stide	HMM
Login	26.2KB	4.46 sec	0.03 sec	56.43 min
PS	29.6KB	5.14 sec	0.11 sec	46.24 min
Xlock	47.4MB	1.51 min	12.3 min	13.37 hr
Stide	36.2MB	5.85 min	8.53 min	2.3 day
Firefox	270.6MB	9.35 min	4.17 hr	4.03 day

TotalADS: Integrated Environment for ADS



S. S. Murtaza, A. Hamou-Lhadj, W. Khreich, M. Couture, "TotalADS: Automated Software Anomaly Detection System," SCAM'14

S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, M. Couture , "A Host-based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules," ISSRE'13



Control Flow View

Events View

Resource View

Process	TID	PTID	Birth time	Trace
unity-2d-shell	2349	2261	02:09:23.206257287	trace-20'
dconf worker	2350	2261	02:09:23.206257825	trace-20'
gdbus	2352	2261	02:09:23.206258356	trace-20'
unity-2d-shell	2394	2261	02:09:23.206259079	trace-20'
unity-2d-shell	2419	2261	02:09:23.206259706	trace-20'
unity-2d-shell	2680	2261	02:09:23.206260384	trace-20'
polkit-gnome-au	2355	2261	02:09:23.206261719	trace-20'

Process	unity-2d-shell
State	USERMODE
CPU	0
Date	2014-04-23
Start Time	02:09:23.588280417
Stop Time	02:09:23.588581765
Duration	0.000301348

Timestamp	Channel	Event Type	Content
<srch>	<srch>	.*sys_recvfrom.*	<srch>
02:09:23.588 261 035	channel0_0	sys_recvfrom	fd=12, ubuf=0x14e20e4, size=4096, flags=0, addr=0x0, addr_len=0x0
02:09:23.588 280 417	channel0_0	exit_syscall	ret=-11
02:09:23.588 581 765	channel0_0	sys_recvfrom	fd=5, ubuf=0x13dbe74, size=4096, flags=0, addr=0x0, addr_len=0x0
02:09:23.588 587 007	channel0_0	exit_syscall	ret=-11
02:09:23.588 601 425	channel0_0	sys_recvfrom	fd=12, ubuf=0x14e20e4, size=4096, flags=0, addr=0x0, addr_len=0x0

Software tracing in industrial projects

Project 1: Tracing and monitoring tools for multi-core systems

Project 2: Host-based anomaly detection systems

Project 3: Tracing, debugging and configuration of avionic systems

Tracing, debugging and configuration of avionic systems

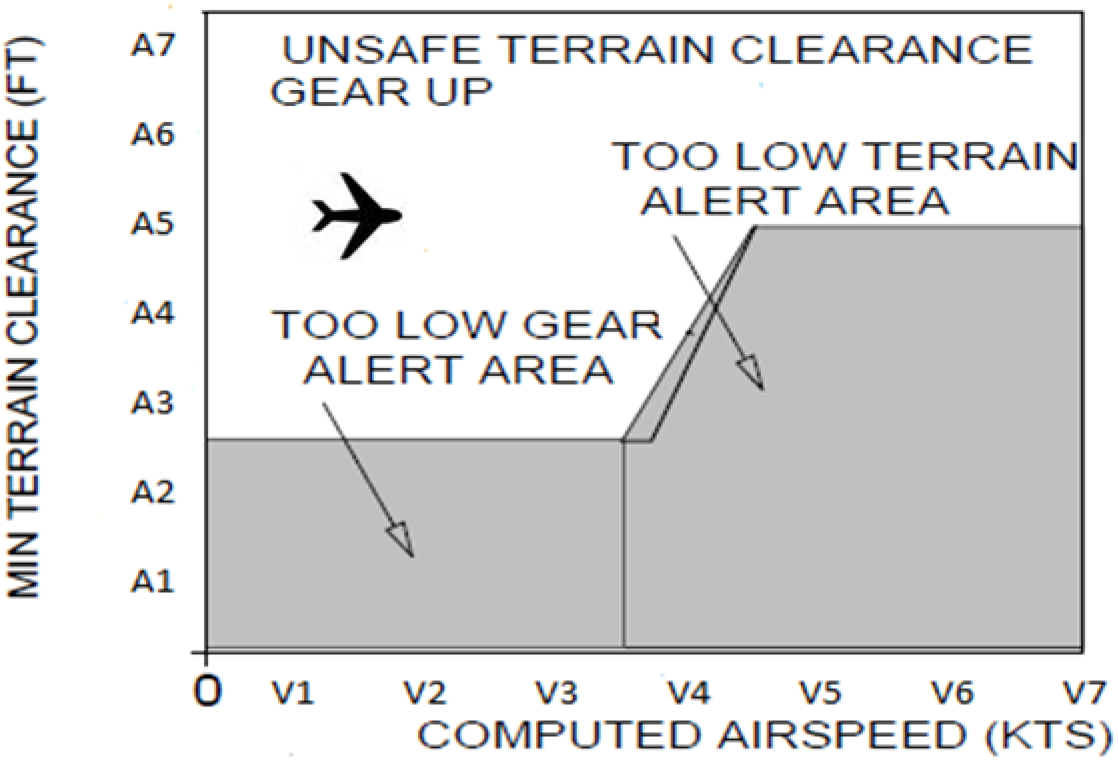
Build efficient algorithms for low overhead, low disturbance tracing of real-time embedded multi-core systems and simulators. Develop special purpose trace analysis debugging, and feature location modules for avionic systems

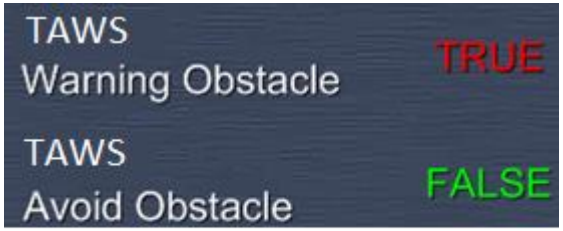
CAE

OPAL·RT

CAE Simulators

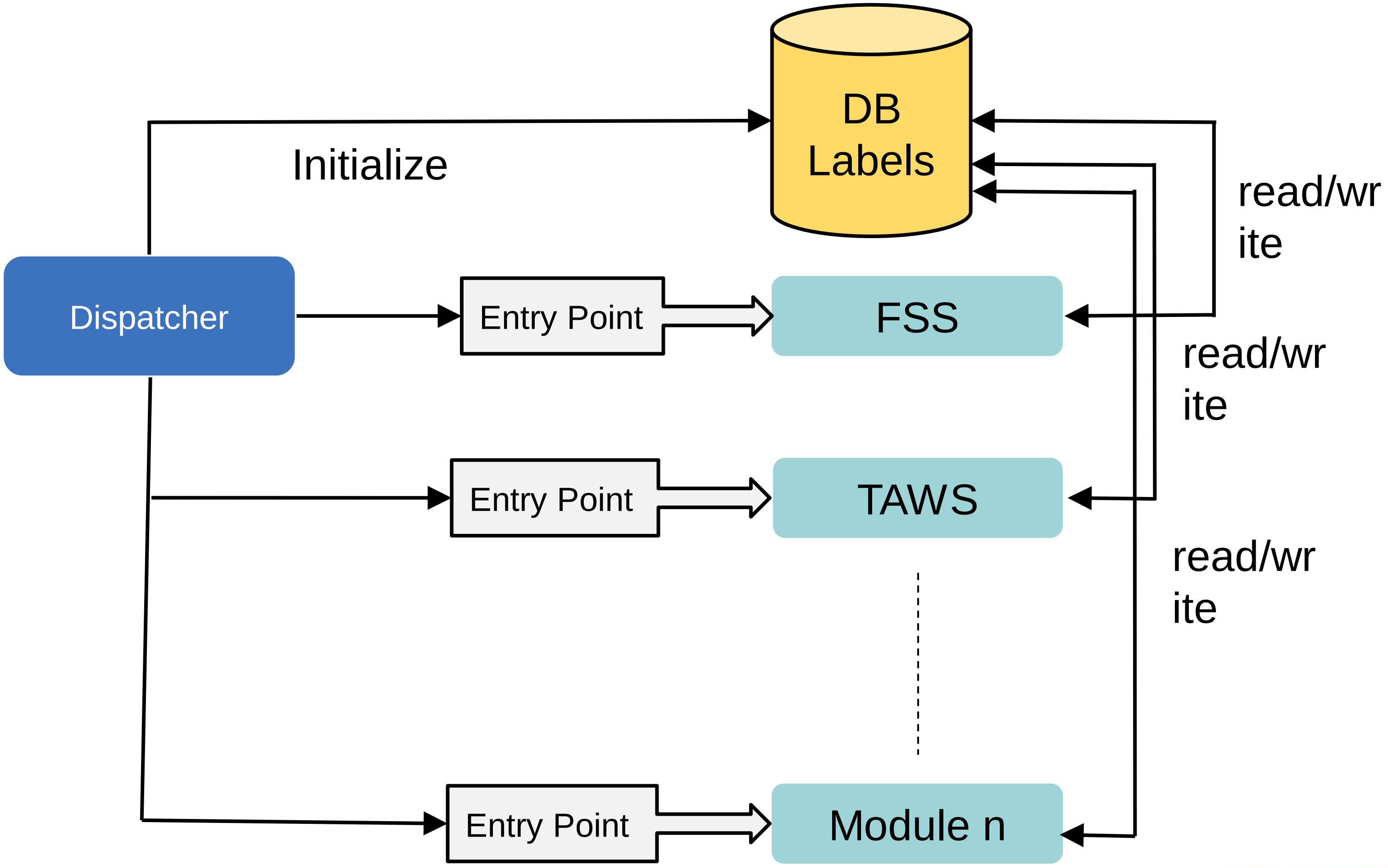
Simulation Scenario




TAWS Warning Obstacle TRUE

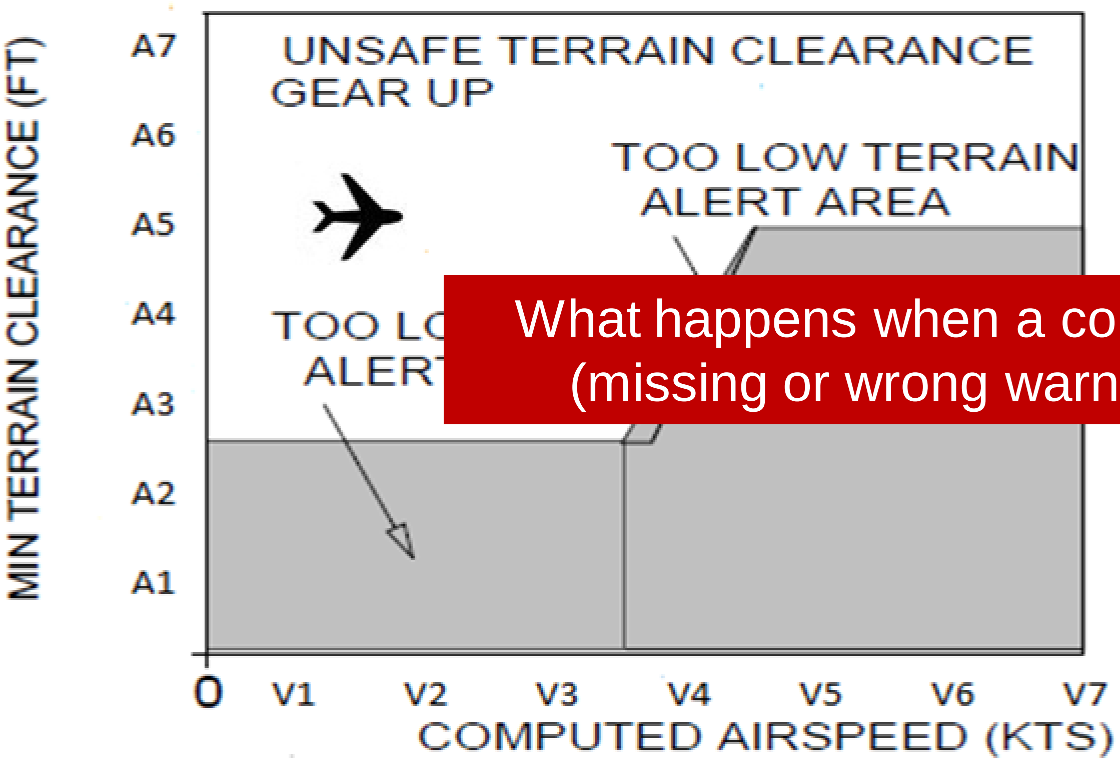
TAWS Avoid Obstacle FALSE

CAE SW Architecture



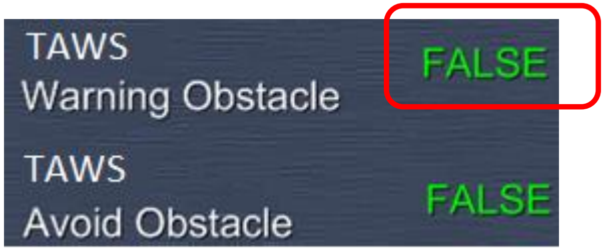
FSS: Flight Surveillance System
TAWS: Terrain Awareness and Warning System

The Problem

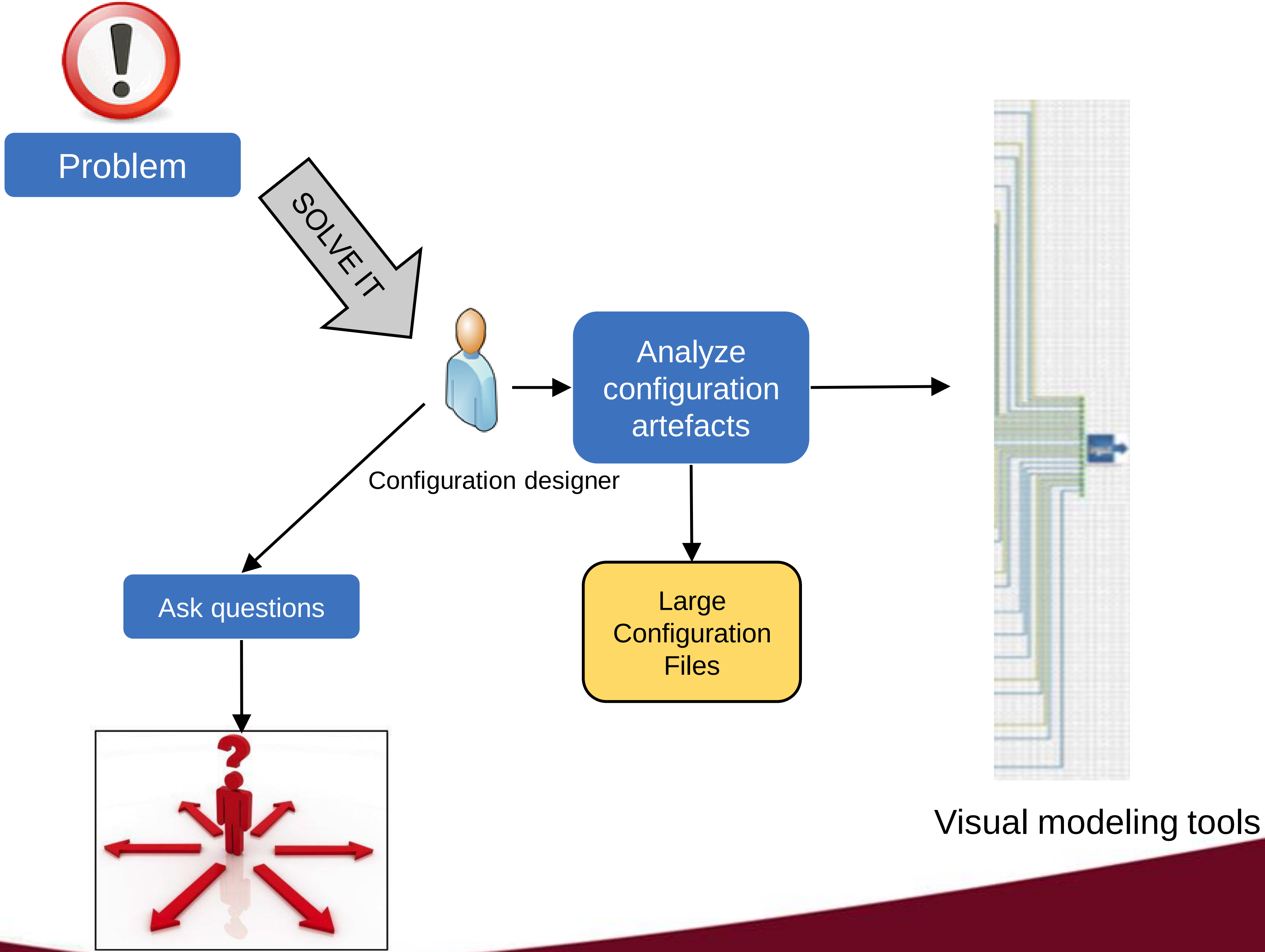


What happens when a configuration error (missing or wrong warnings) occurs?

This should be TRUE


TAWS Warning Obstacle FALSE

TAWS Avoid Obstacle FALSE



Problem

SOLVE IT

Configuration designer

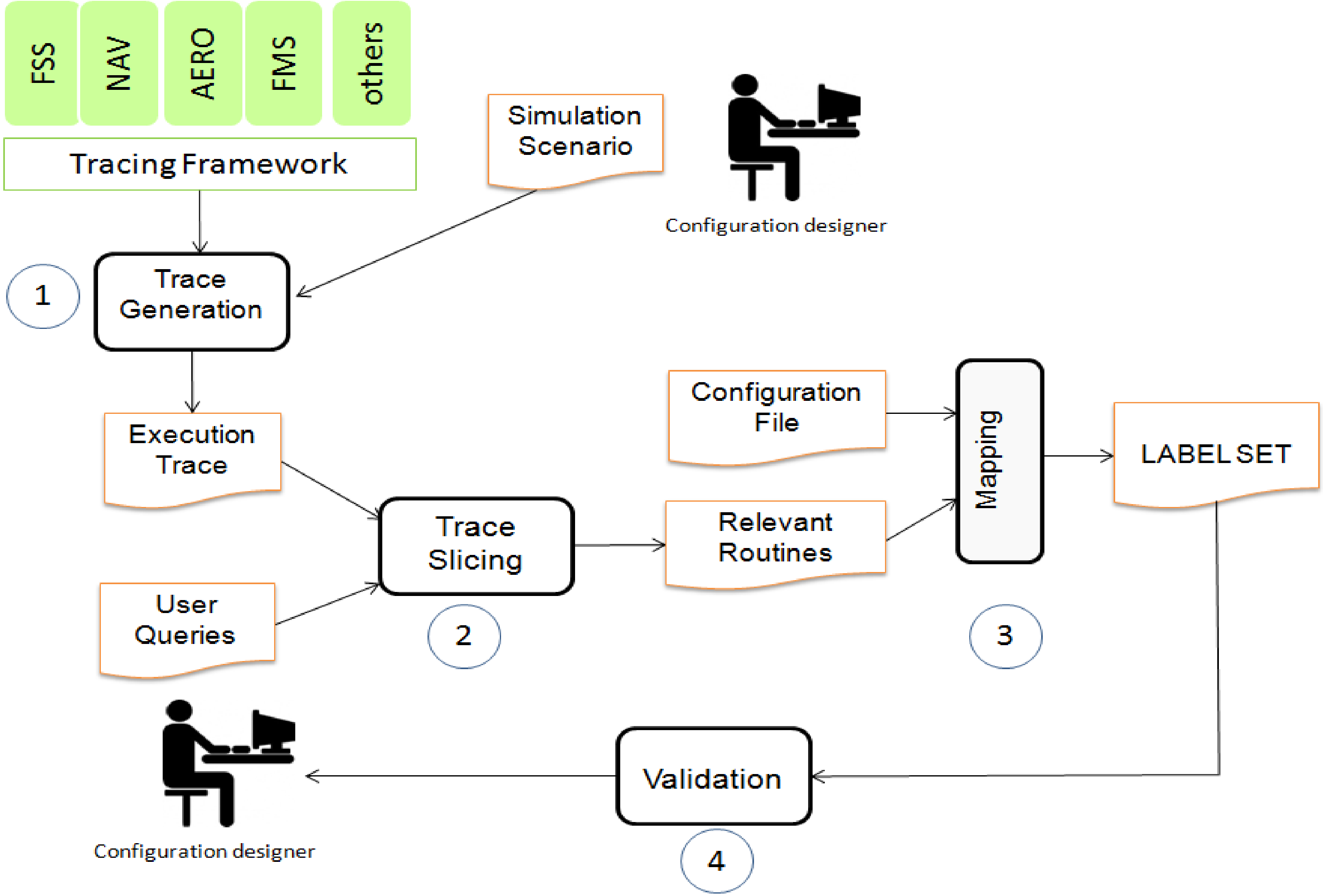
Analyze configuration artefacts

Ask questions

Large Configuration Files

Visual modeling tools

FELODE (Feature Location for Debugging)



Case Study: Selected Scenarios

Scenario	Subsystem	Scenario
S1	TAWS Mode1	Aircraft is descending at high speed while flying at low altitude.
S2	TAWS Mode4A	The aircraft is close to the ground and is prepared for landing, but the gears are still up.
S3	TAWS Mode4B	Aircraft is in landing mode but the flaps are in a flight position.
S4	TCAS	Simulate the presence of an intruder with the intention to locate its altitude.
S5	TCAS	Simulate the presence of an intruder with the intention to locate its speed.

FELODE Precision and Recall

Scenarios	Precision (N2/N1)	Recall (N2/N3)
S1	50%	50%
S2	50%	100%
S3	50%	100%
S4	38%	100%
S5	57%	100%

N1: Number of labels detected using FELODE
N2: Number of valid labels using FELODE
N3: Number of valid labels relevant to each scenario (provided by the users)

Observations

- Tracing techniques can help solve industrial problems
- A little knowledge can go a long way
- The tool is a big part of an industrial solution
- From knowledge transfer to knowledge transition
- More research is needed in: Trace modeling, model-driven tracing, tracing small devices, trace analytics