Making Software Tracing Applicable and Scalable: Experiences and Lessons Learned

Wahab Hamou-Lhadj, PhD, P. Eng.
Software Behaviour Analysis Research (SBA) Lab
Concordia University
Montréal, QC, Canada
http://www.ece.concordia.ca/~abdelw

Ryerson University
Toronto, ON
Nov. 3, 2014

Software (Persistent) Challenges

- Complexity
- Lack of structure and appropriate documentation
- Initial design no longer valid
- Initial designers no longer available
- New computing platforms do not make things easier

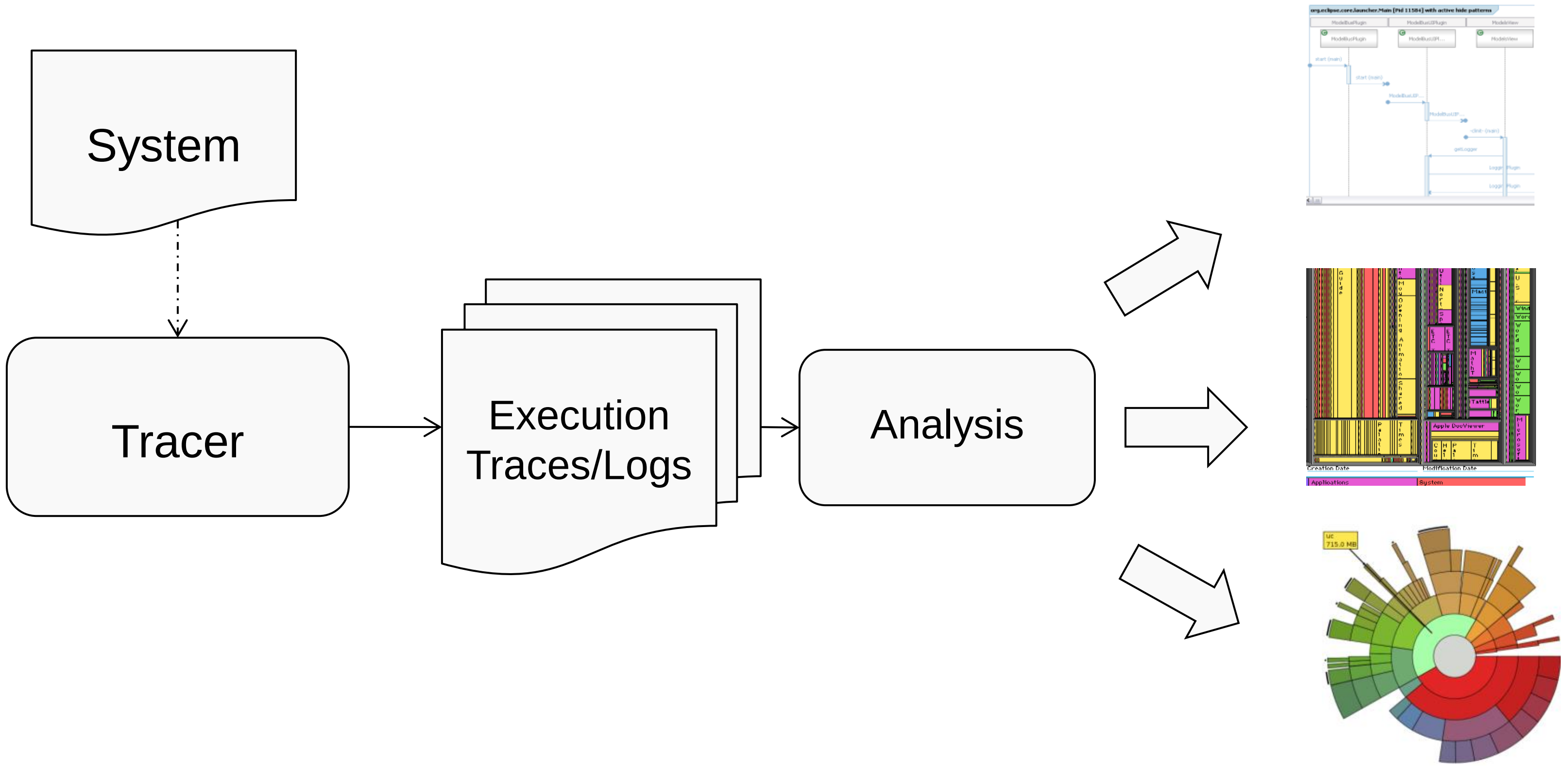
Software (Persistent) Challenges

- Complexity
- Lack of structure and appropriate documentation

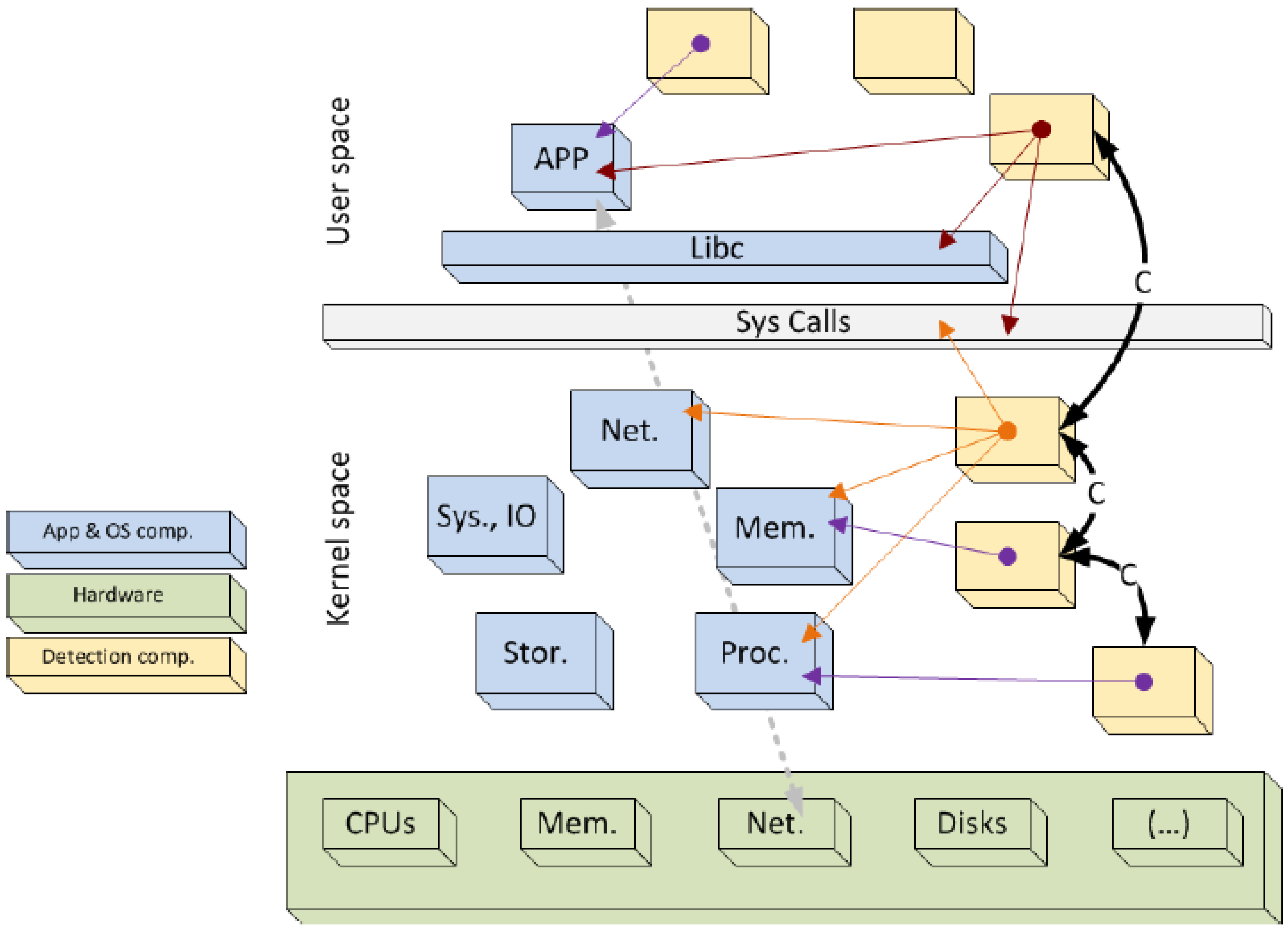
Investment in software analysis techniques and tools is critical

- New computing platforms do not make things easier

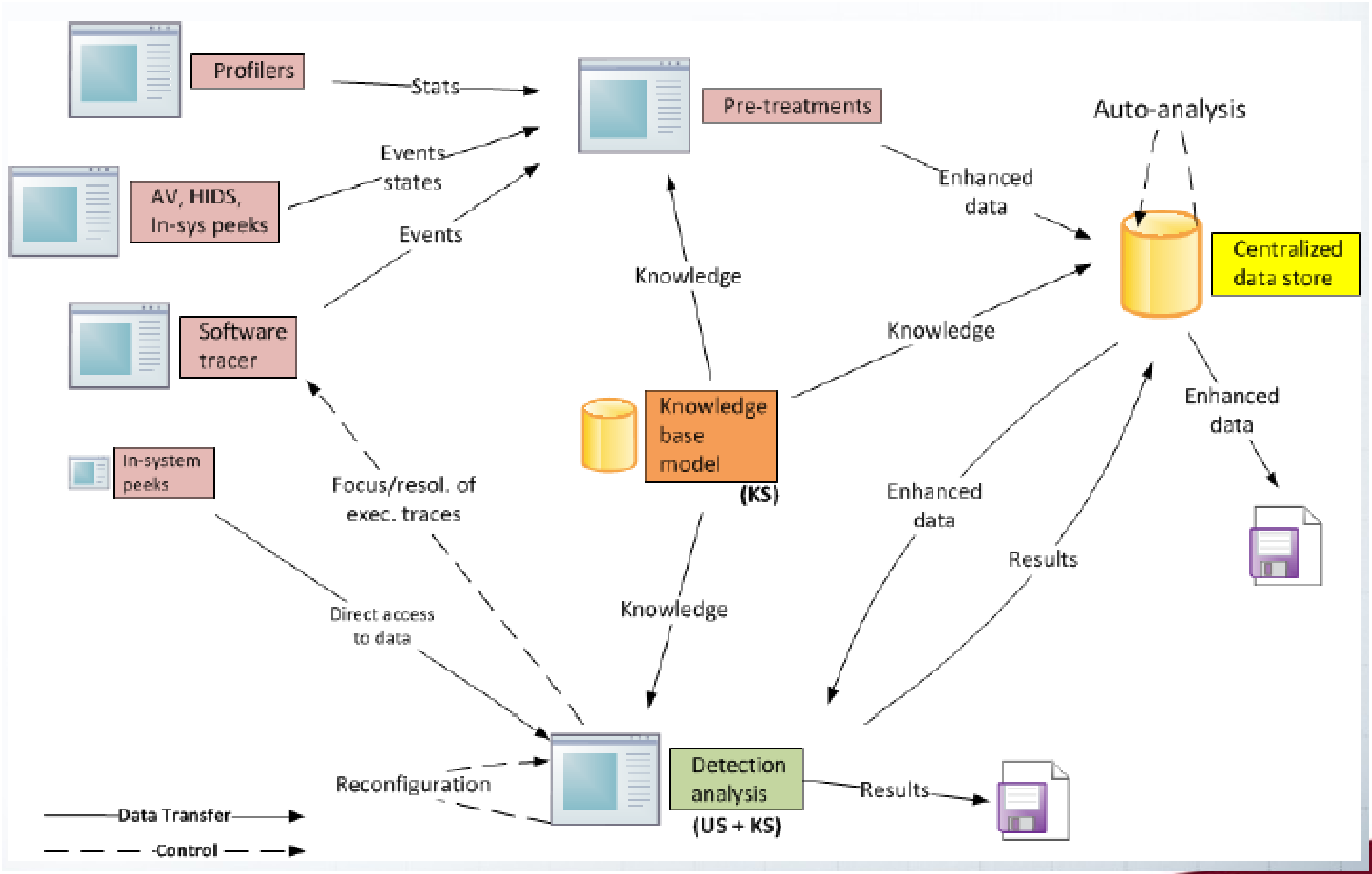
Software Behaviour Analysis: Simplified View



… a bit more complex



…very complex



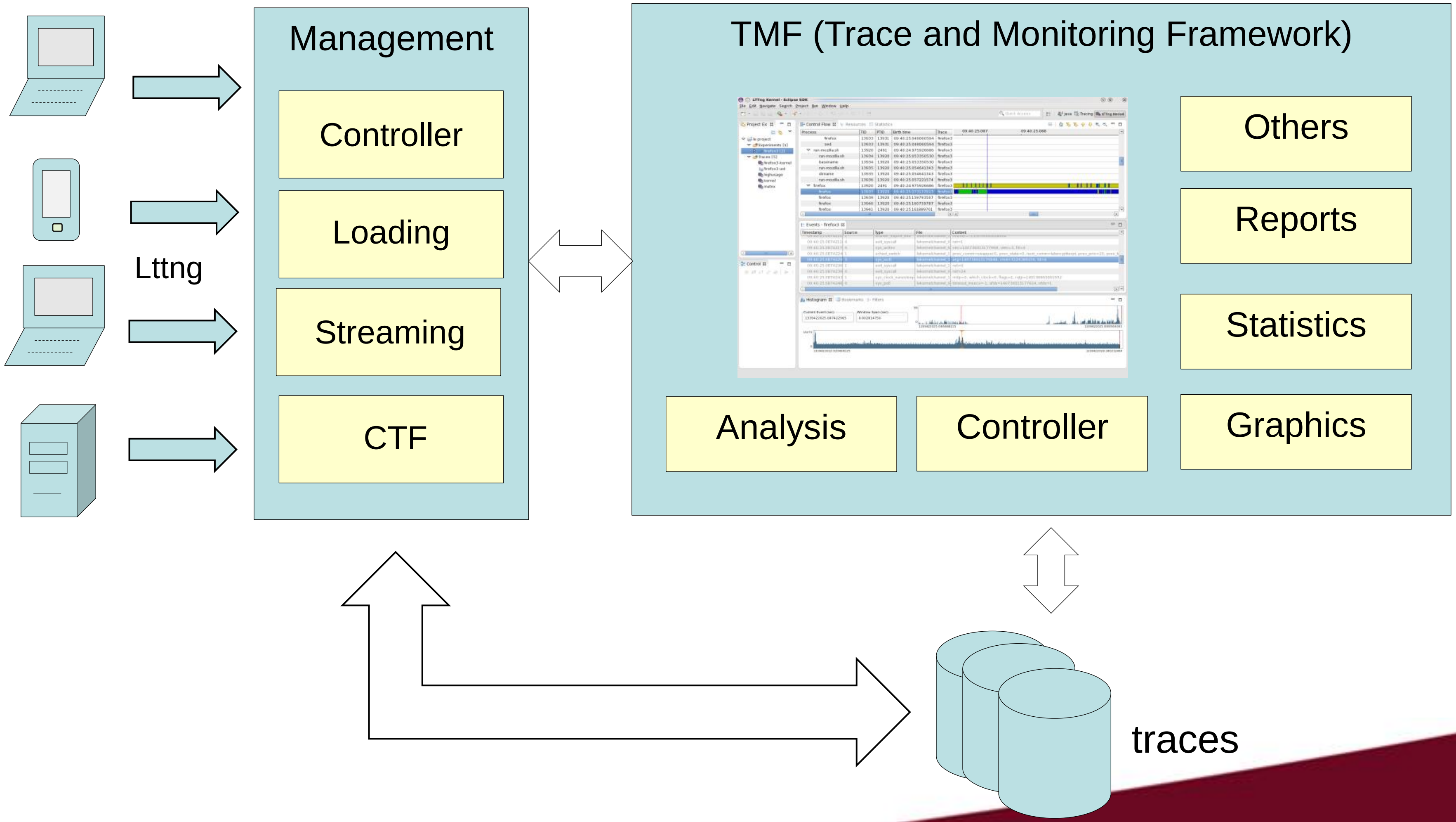
Industrial projects

- 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

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

Project vision

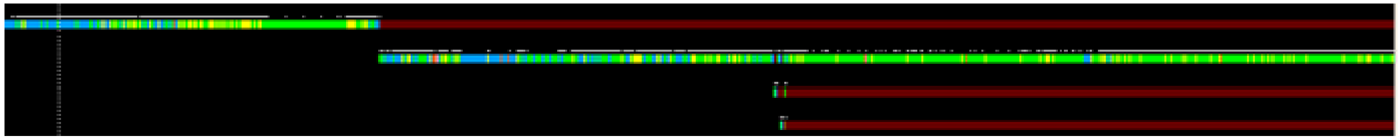


Making trace analysis scalable

- Motivating Example:

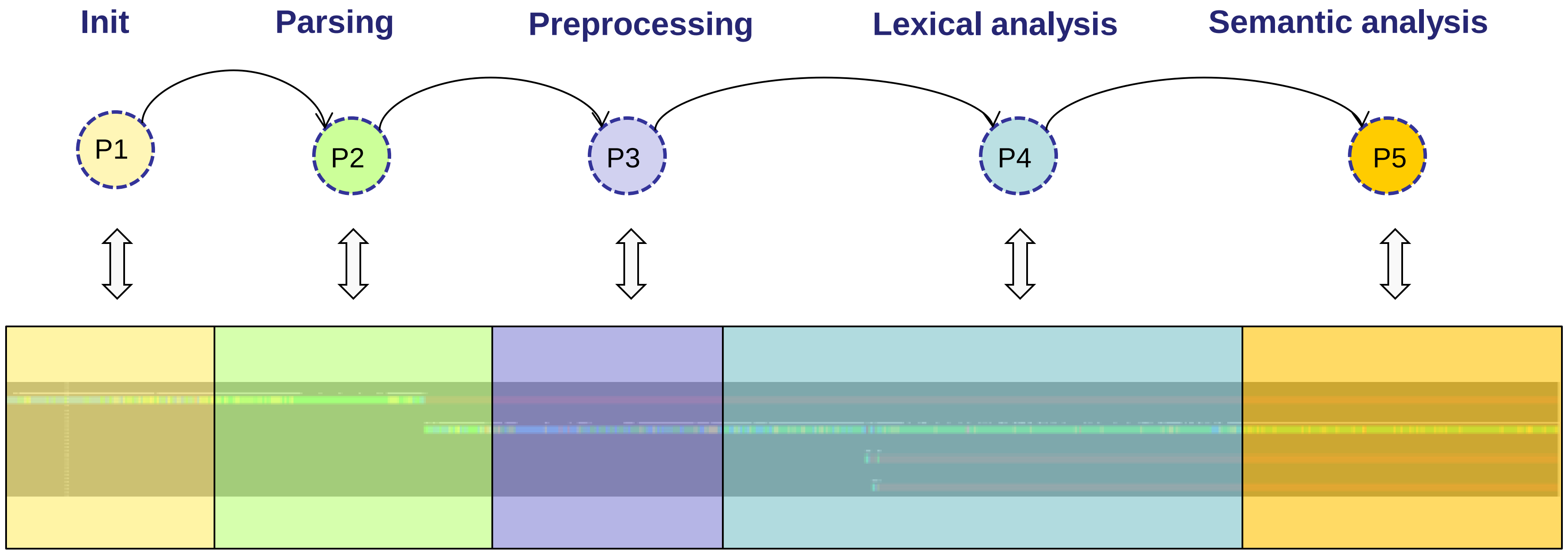
  A trace generated from a compiler: parsing, preprocessing, lexical analysis, semantic analysis, etc. may contain hundred of thousands of events.

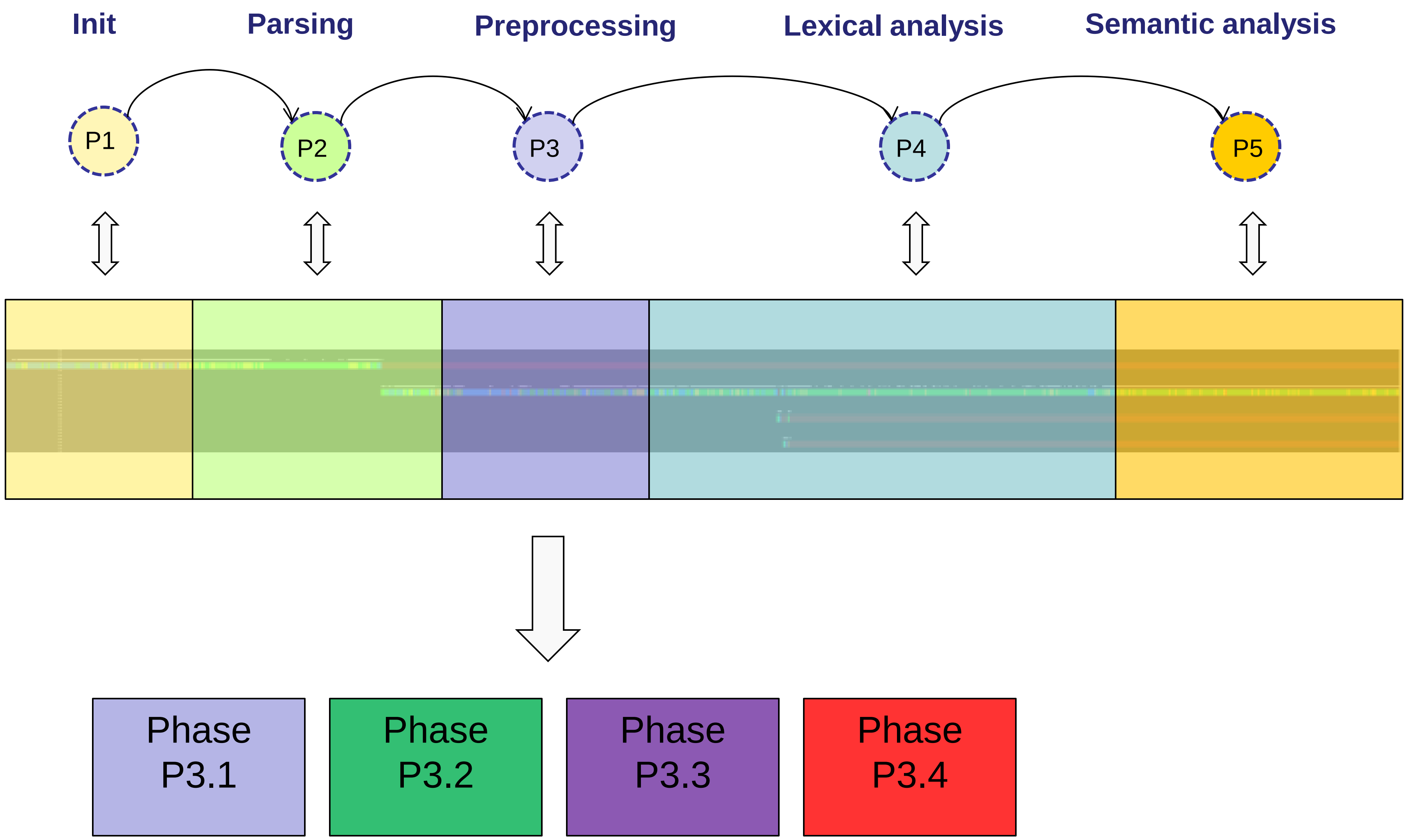
- Typical tools will show this:



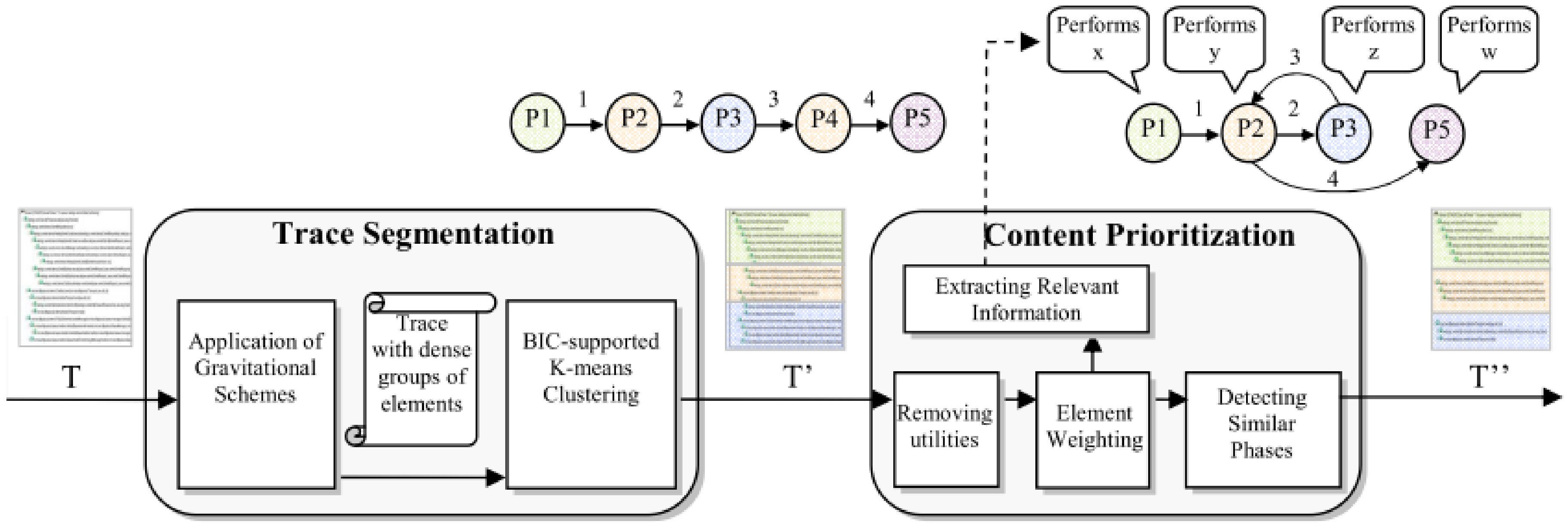
How do you know what happens where?

Automatic extraction of execution phases

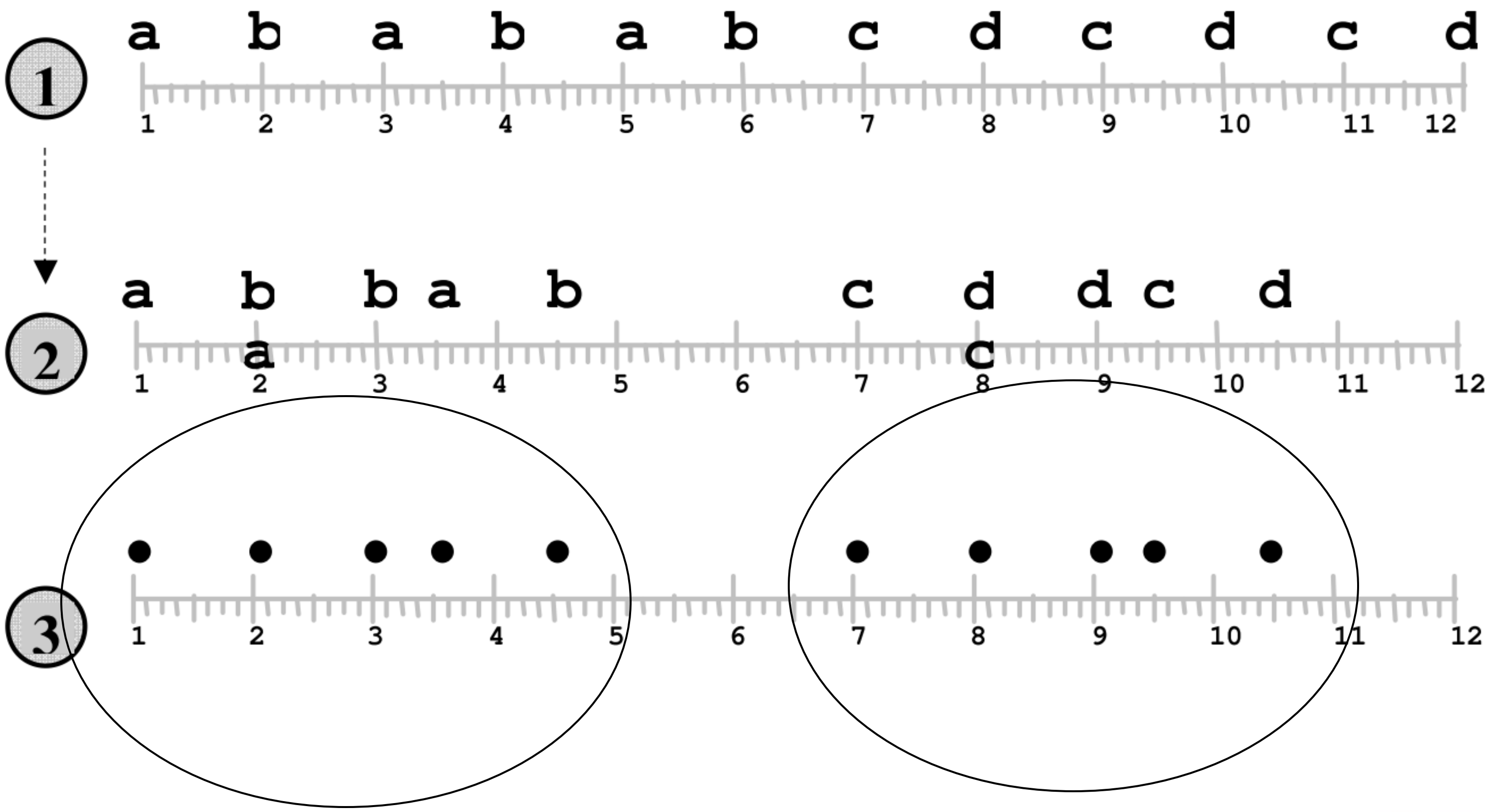



Init
Parsing
Preprocessing
Lexical analysis
Semantic analysis
P1
P2
P3
P4
P5
Phase P3.1
Phase P3.2
Phase P3.3
Phase P3.4

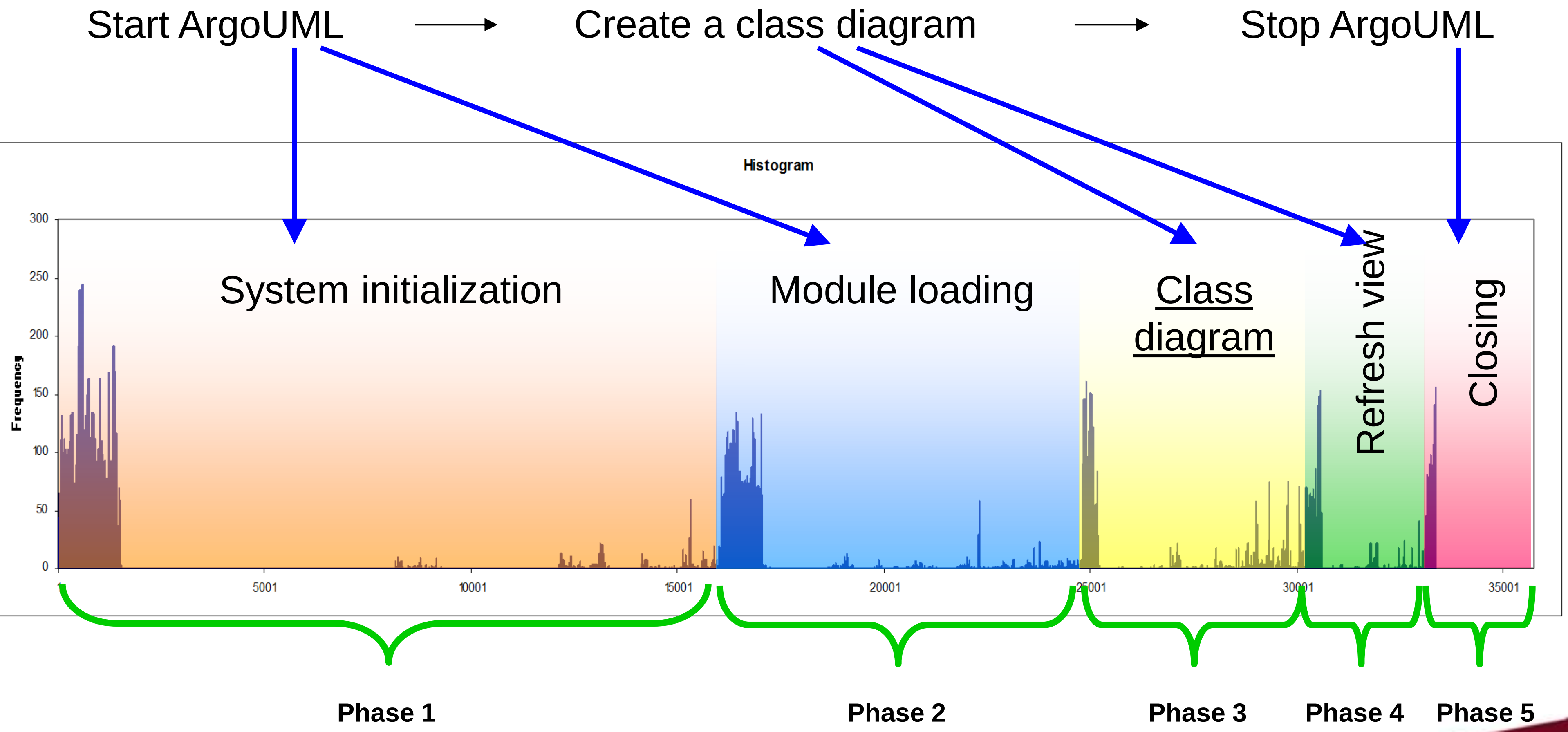
TSER: Trace Segmentation through Event Repositioning



Example: Repositioning based on similarity

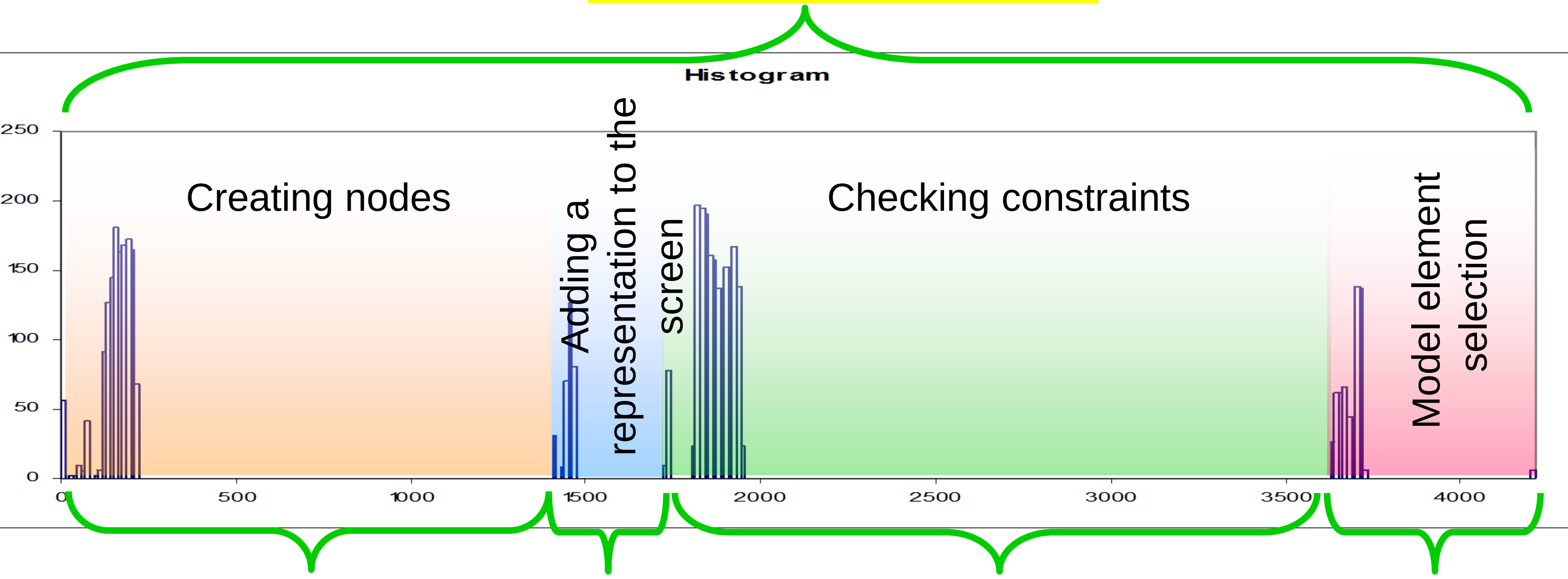


Application: ArgoUML Trace



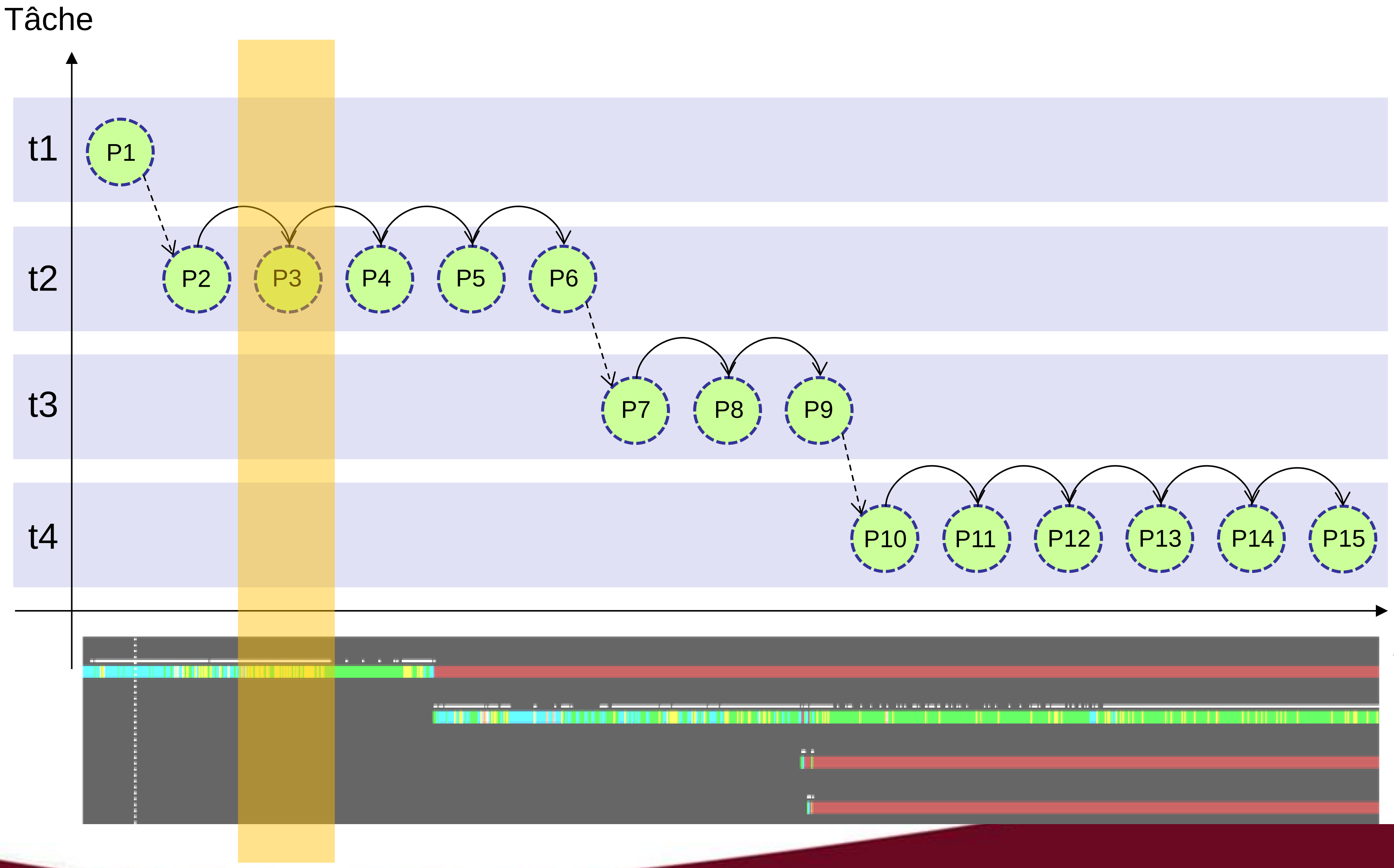
Start ArgoUML → Create a class diagram → Stop ArgoUML

Phase 3: Add a class diagram

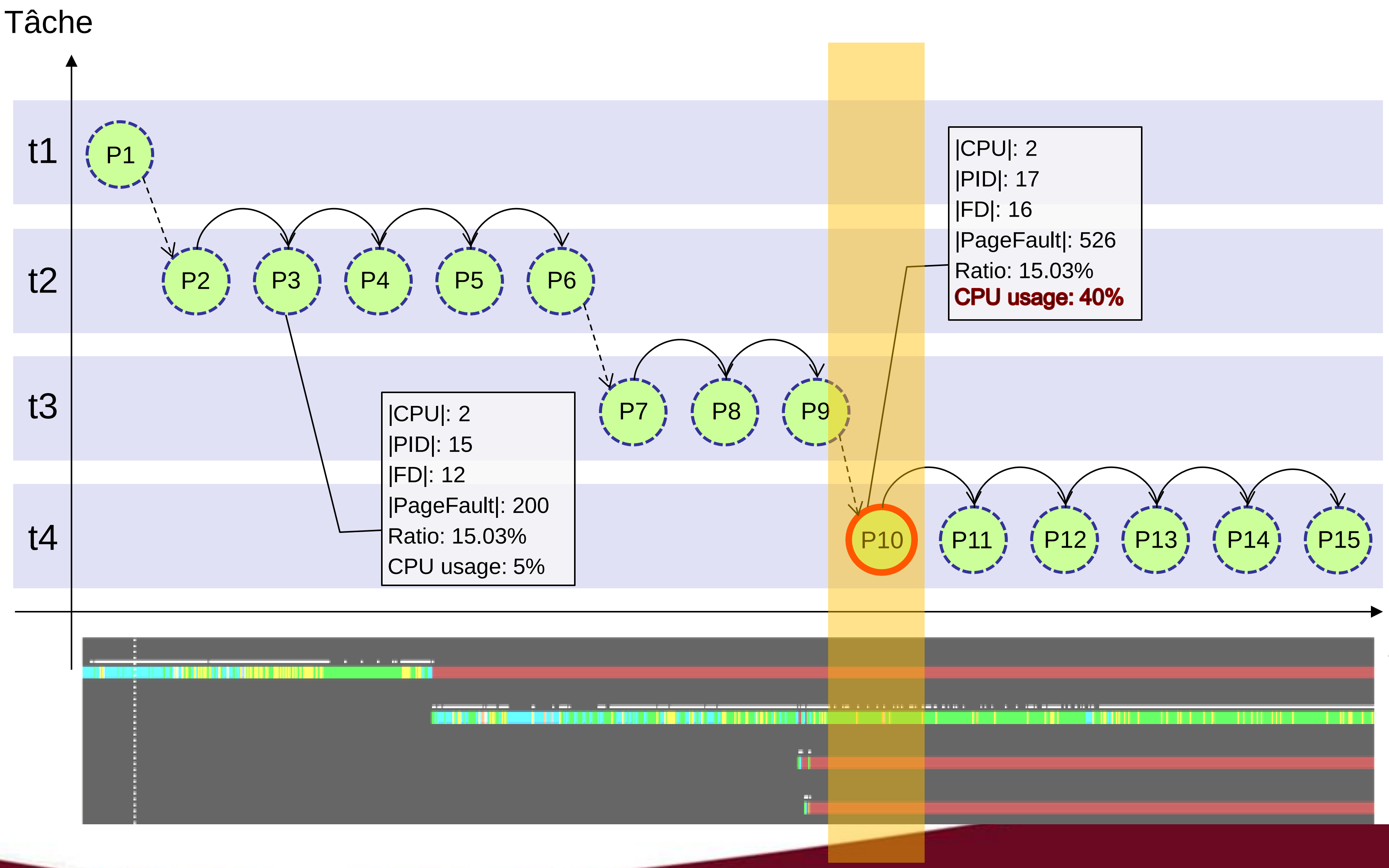


Phase 3.1 **Phase 3.2** **Phase 3.3** **Phase 3.4**

Aligning user and kernel spaces



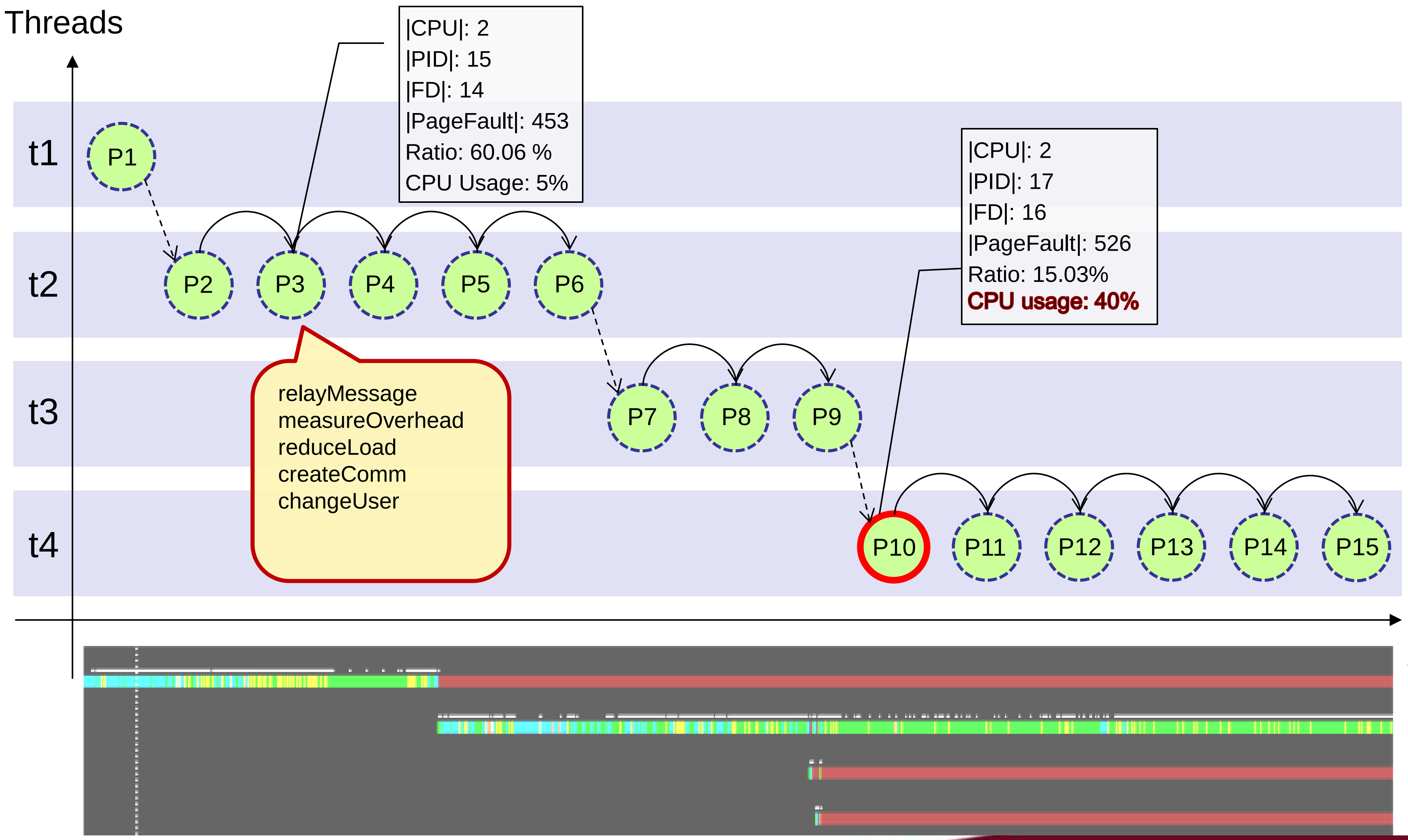
Adding state information



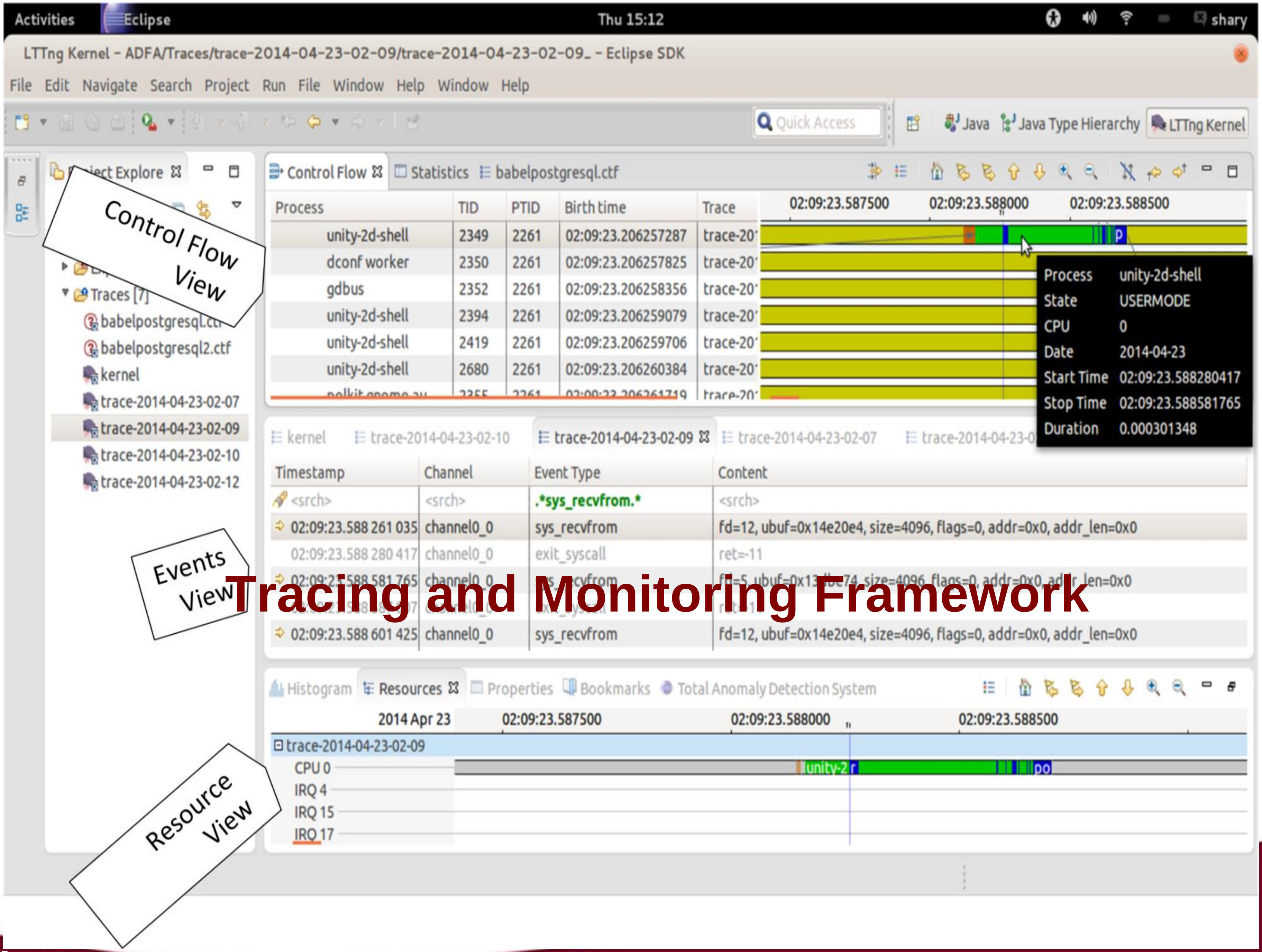
|CPU|: 2
|PID|: 15
|FD|: 12
|PageFault|: 200
Ratio: 15.03%
CPU usage: 5%

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

Identification du contenu pertinent



Tracing and Monitoring Framework



LTTng Kernel - ADFA/Traces/trace-2014-04-23-02-09/trace-2014-04-23-02-09_ - Eclipse SDK

Traces [7]
- babelpostgresql.ctf
- babelpostgresql2.ctf
- kernel
- trace-2014-04-23-02-07
- trace-2014-04-23-02-09
- trace-2014-04-23-02-10
- trace-2014-04-23-02-12

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

02:09:23.587500 02:09:23.588000 02:09:23.588500

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

kernel | trace-2014-04-23-02-10 | trace-2014-04-23-02-09 | trace-2014-04-23-02-07 | trace-2014-04-23-0

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=0x13[illegible]74, size=4096, flags=0, addr=0x0, addr_len=0x0
02:09:23.588 [illegible]7	channel0_0	exit_syscall	ret=[illegible]
02:09:23.588 601 425	channel0_0	sys_recvfrom	fd=12, ubuf=0x14e20e4, size=4096, flags=0, addr=0x0, addr_len=0x0

2014 Apr 23 02:09:23.587500 02:09:23.588000 02:09:23.588500

trace-2014-04-23-02-09
- CPU 0 unity-2 r po
- IRQ 4
- IRQ 15
- IRQ 17

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

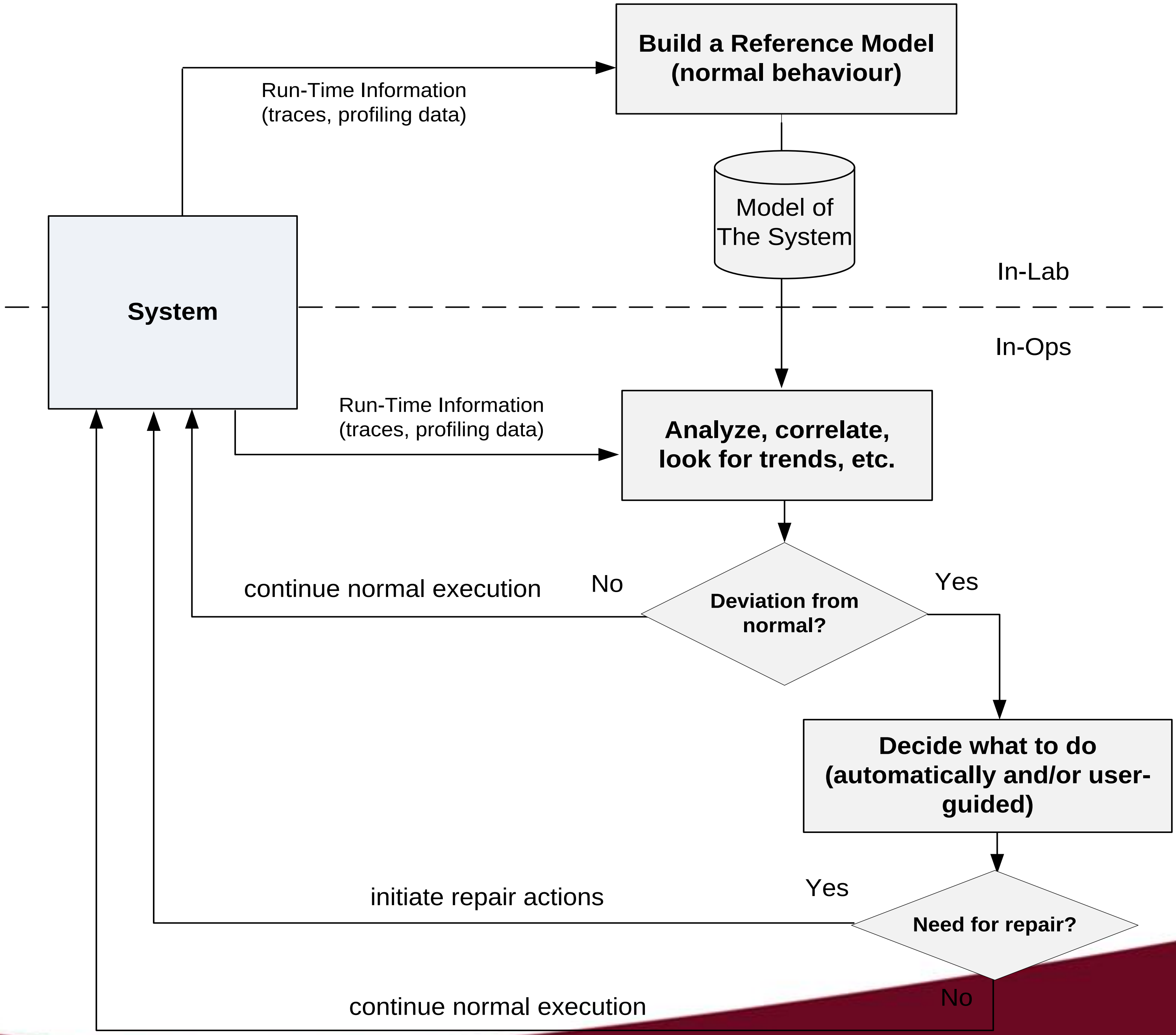
Host-based Anomaly Detection- Advanced Host-Level Surveillance

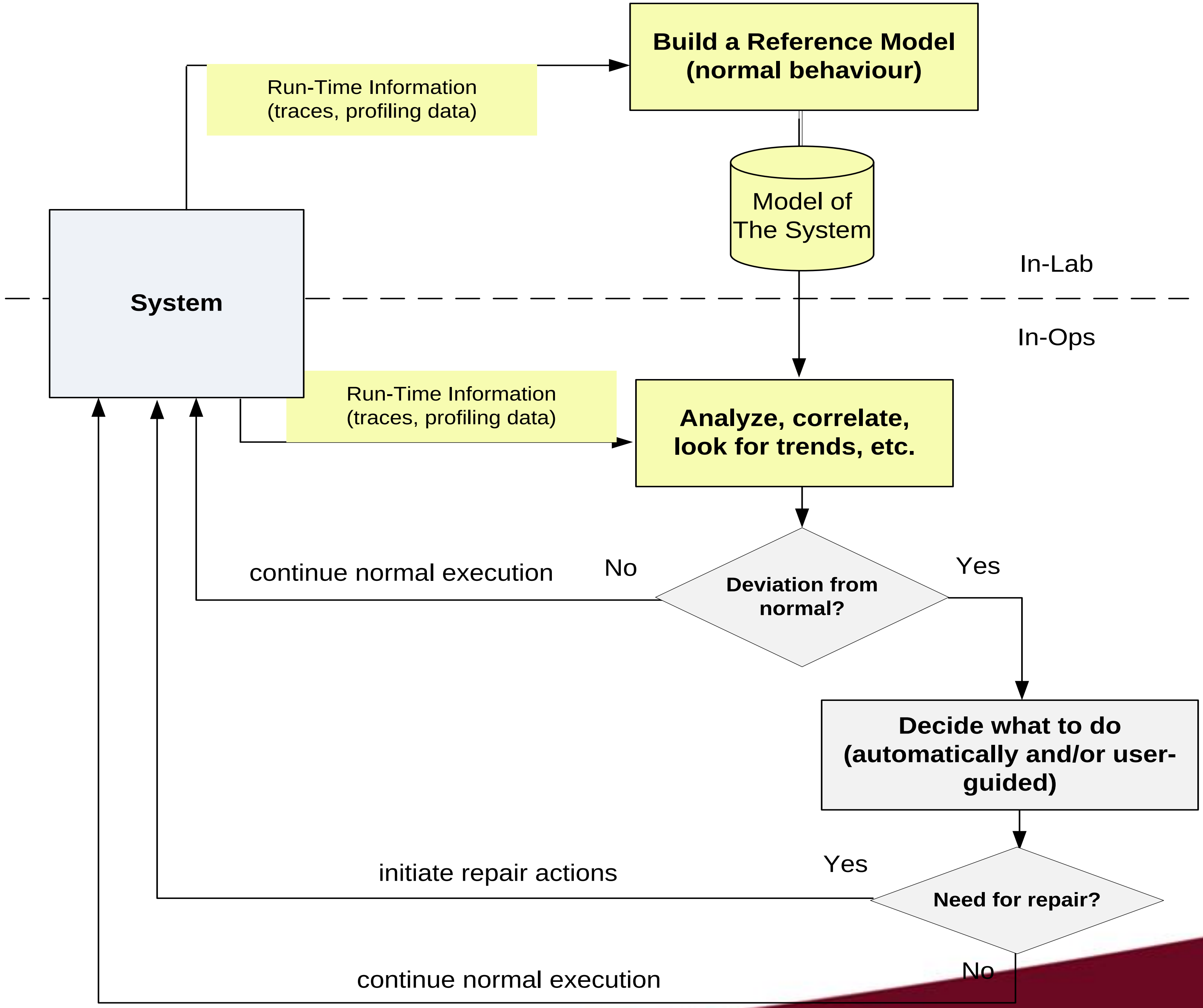
Develop modular, adaptive, and scalable Anomaly Detection Systems (ADS) at the level of system call traces; reduce false positives (alarms) and improve the true positives

NSERC CRSNG

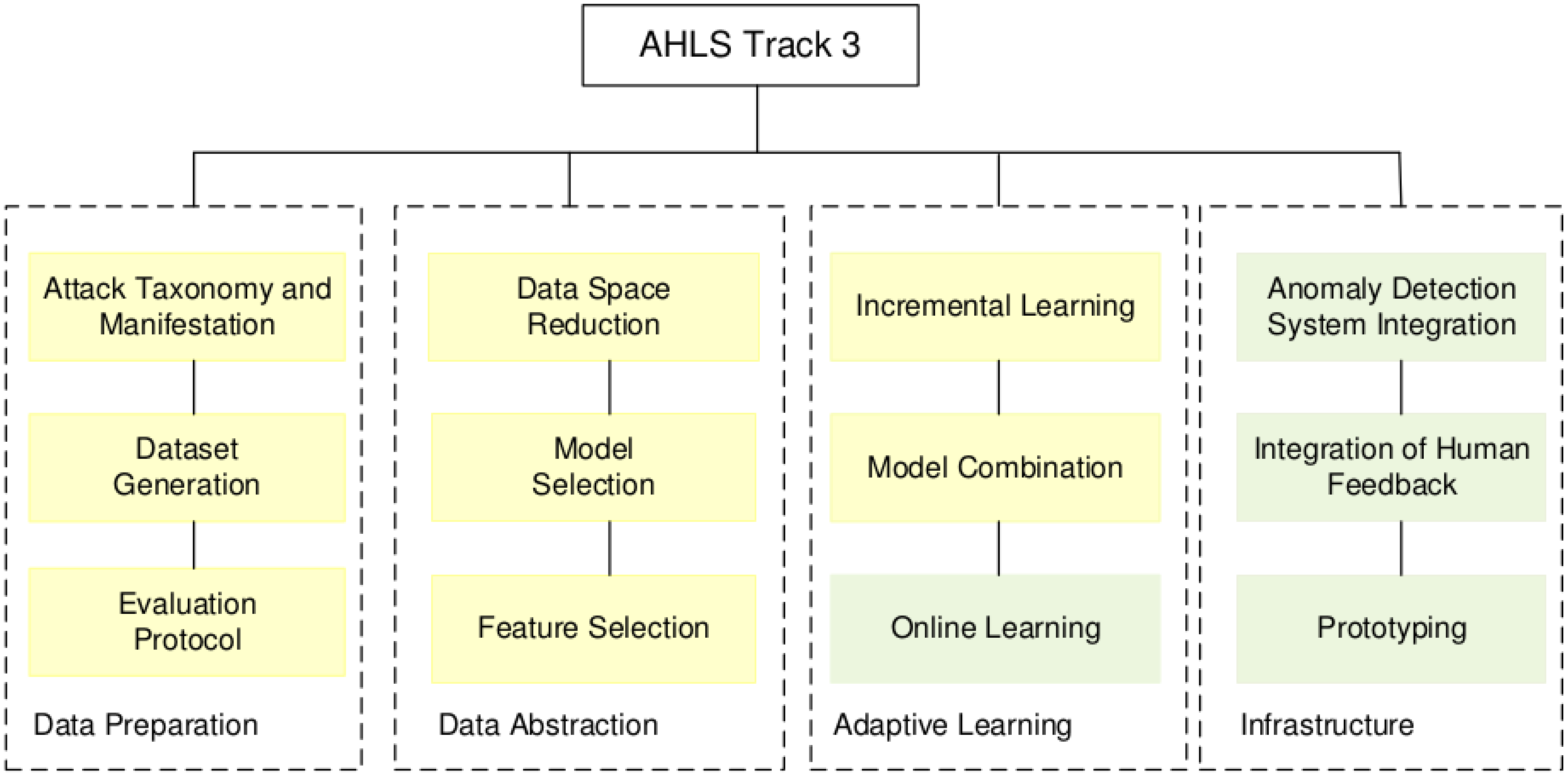
DEFENCE DÉFENSE

ERICSSON

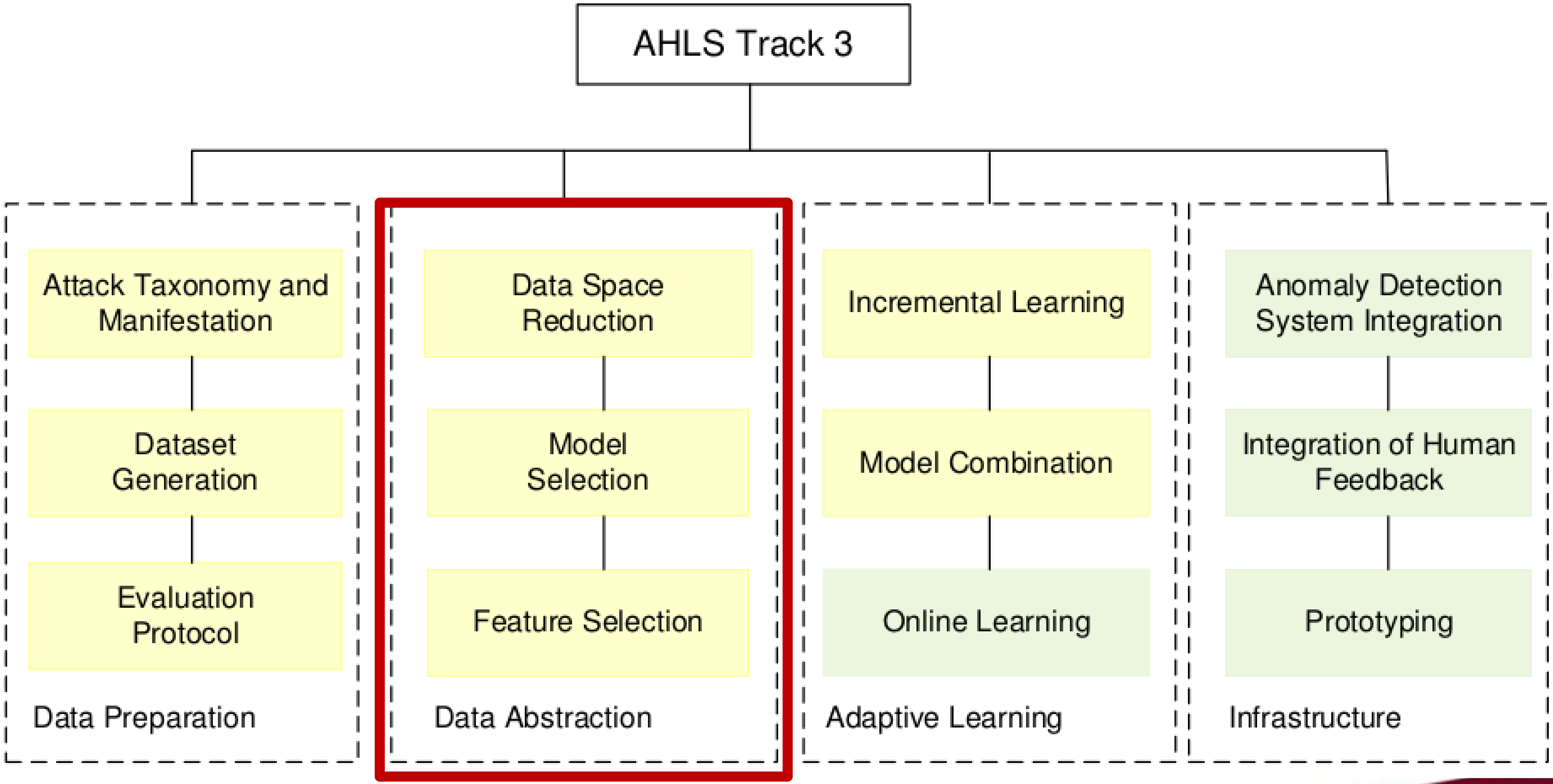

Build a Reference Model (normal behaviour)
Run-Time Information (traces, profiling data)
Model of The System
In-Lab
System
In-Ops
Run-Time Information (traces, profiling data)
Analyze, correlate, look for trends, etc.
continue normal execution
No
Deviation from normal?
Yes
Decide what to do (automatically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


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
Decide what to do (automatically and/or user-guided)
initiate repair actions
Yes
Need for repair?
No
continue normal execution

Advanced Host-Level Surveillance



Advanced Host-Level Surveillance



Kernel State Modeling (KSM)

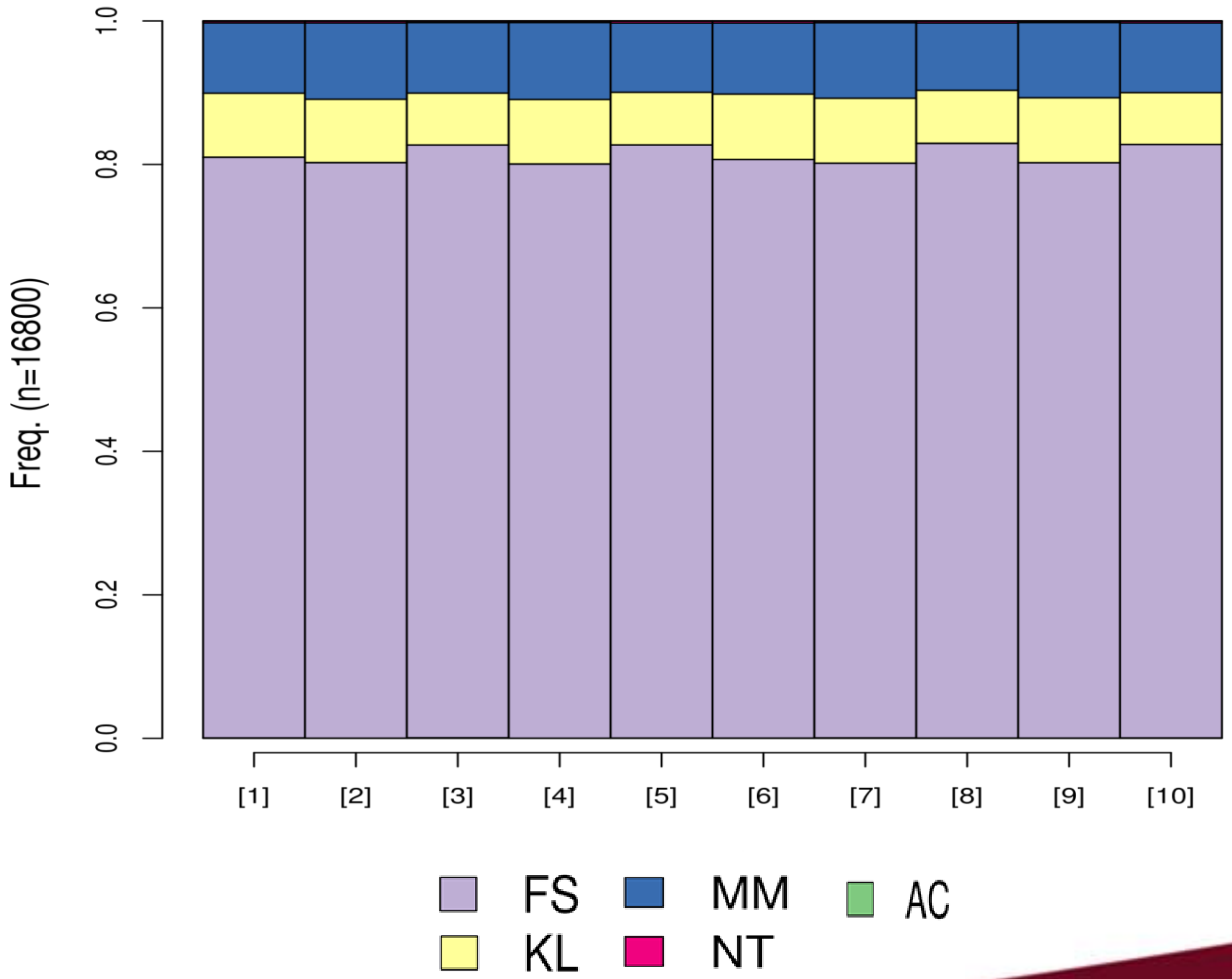
- KSM is an anomaly detection technique
  - Transforms system calls into kernel modules, called states
  - Detects anomalies at the level of interaction among kernel states
  - Reduces data space used in training and testing
  - Favors efficiency while keeping accuracy

Transforming System Calls into States of Kernel Modules

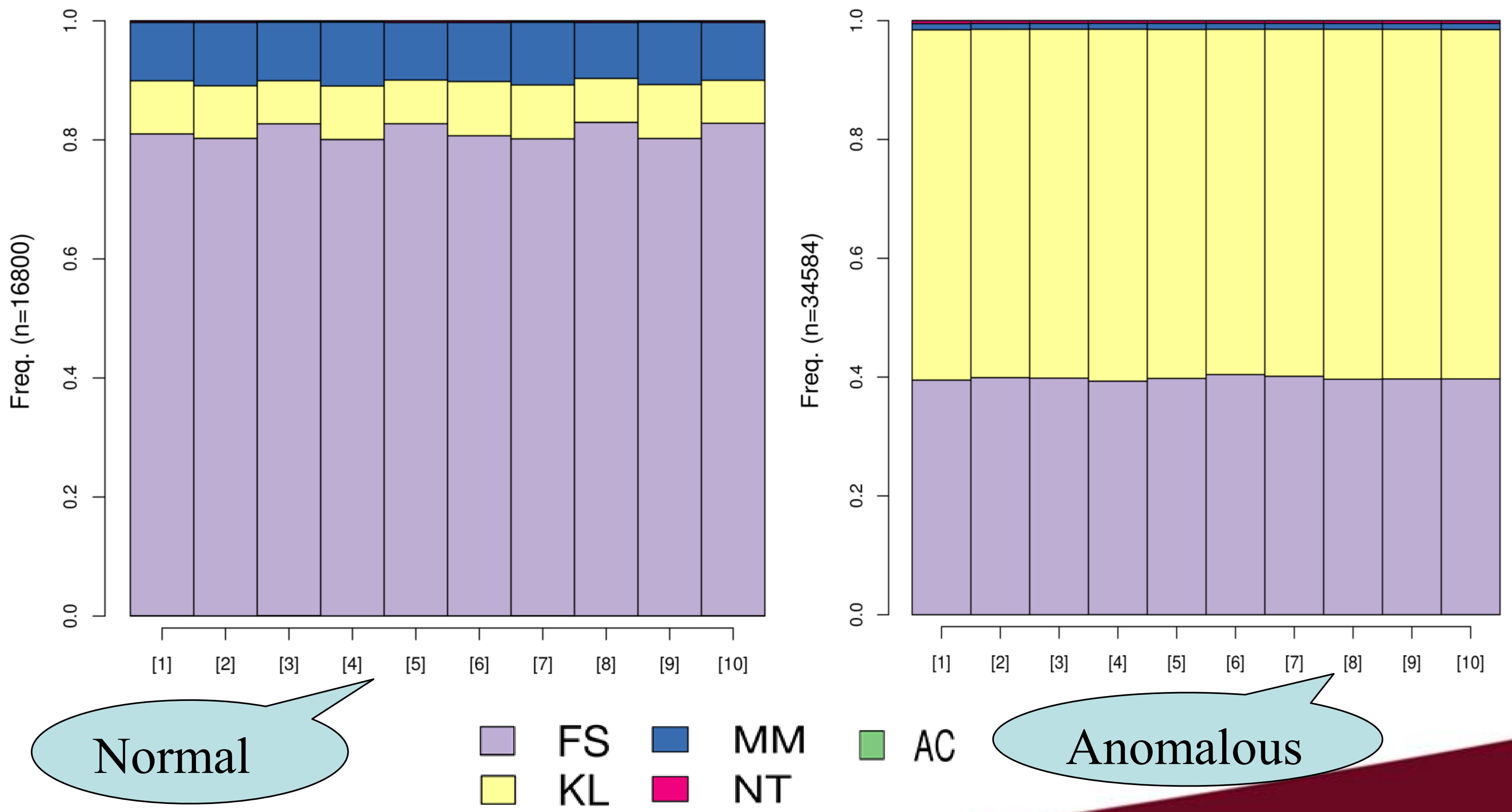
State	Module in Linux Source Code	# of System Calls
AC	Architecture	10
FS	File System	131
IPC	Inter Process Communication	7
KL	Kernel	127
MM	Memory Management	21
NT	Networking	2
SC	Security	3
UN	Unknown	37

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

KSM and Density Plots



Anomaly Detection in Firefox



Evaluation

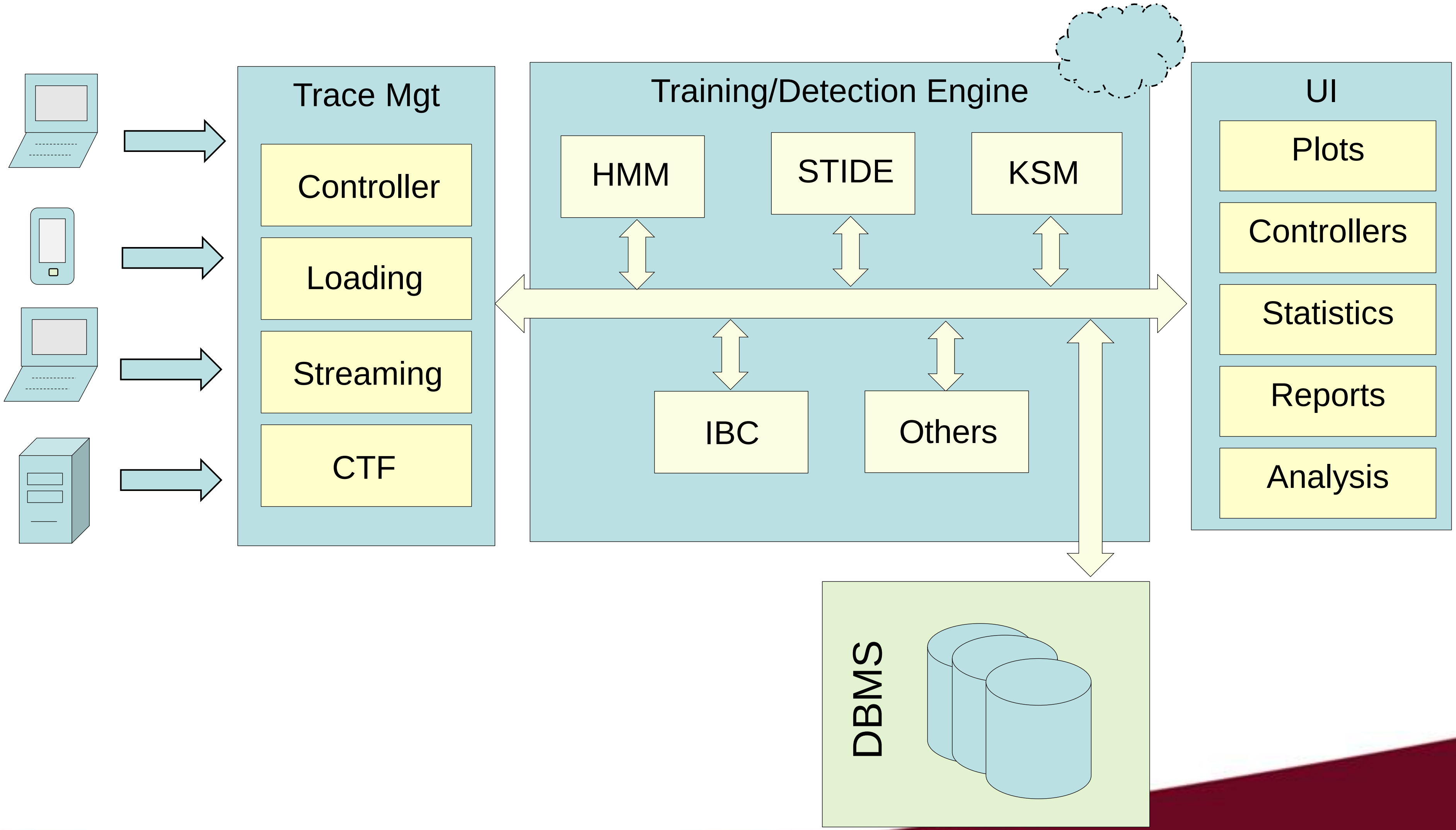
Program	# Normal Traces			#Attack Types	#Attack Traces
	Training	Validation	Testing		
Login	4	3	5	1	4
PS	10	4	10	1	15
Stide	400	200	13126	1	105
Xlock	91	30	1610	1	2
Firefox	125	75	500	5	19

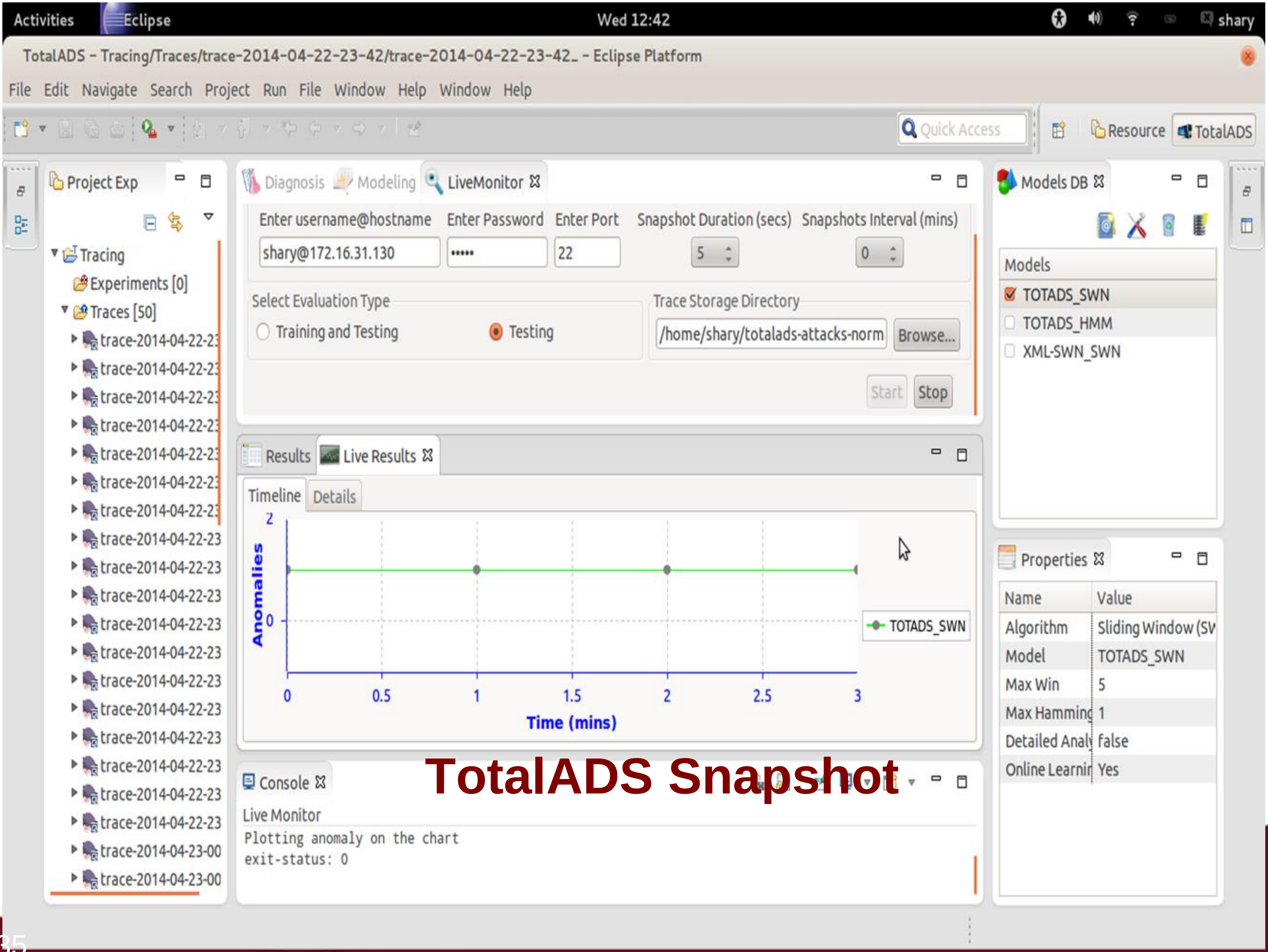
Program	Technique	TP rate	FP rate
Login	KSM (alpha=0.00)	100%	0.00%
	Stide (win=6)	100%	40.00%
	Stide (win=10)	100%	40.00%
	HMM (states=10)	100%	40.00%
PS	KSM (alpha=0.02)	100%	10.00%
	Stide (win=6)	100%	10.00%
	Stide (win=10)	100%	10.00%
	HMM (states=5)	100%	30.00%
Xlock	KSM (alpha=0.04)	100%	0.00%
	Stide (win=6)	100%	1.50%
	Stide (win=10)	100%	1.50%
	HMM (states=5)	100%	0.00%
Stide	KSM (alpha=0.06)	100%	0.25%
	Stide (win=6)	100%	4.97%
	Stide (win=10)	100%	5.25%
	HMM (states=5)	100%	0.25%
Firefox	KSM (alpha=0.08)	100%	0.60%
	Stide (win=6)	100%	44.60%
	Stide (win=10)	100%	49.20%
	HMM (states=5)	100%	1.40%

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





Enter username@hostname	Enter Password	Enter Port	Snapshot Duration (secs)	Snapshots Interval (mins)
shary@172.16.31.130	•••••	22	5	0

Select Evaluation Type: Training and Testing [✓ Testing]

Trace Storage Directory: /home/shary/totalads-attacks-norm

Traces:
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-2
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-22-23
trace-2014-04-23-00
trace-2014-04-23-00

Models:
- ✓ TOTADS_SWN
- TOTADS_HMM
- XML-SWN_SWN

Name	Value
Algorithm	Sliding Window (SV
Model	TOTADS_SWN
Max Win	5
Max Hamming	1
Detailed Analy	false
Online Learnin	Yes

```
Live Monitor
Plotting anomaly on the chart
exit-status: 0
```

TotalADS Snapshot

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 <u>trace analysis debugging, and feature location</u> modules for avionic systems

CAE

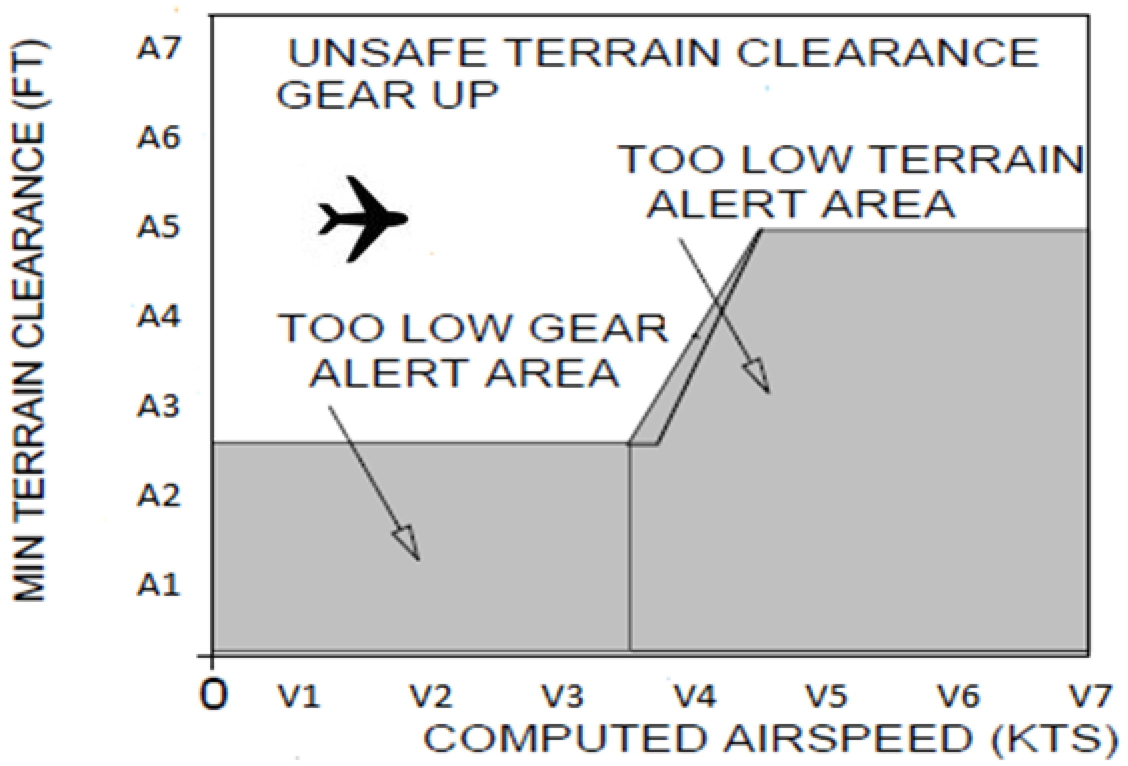
O P A L · R T

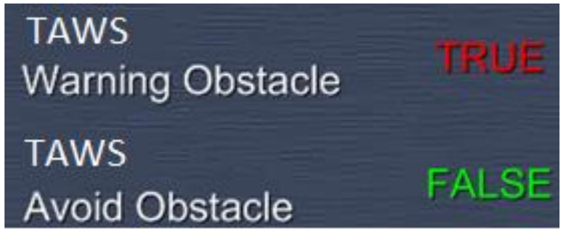
NSERC CRSNG

CRIAQ www.criaq.aero

CAE Simulators

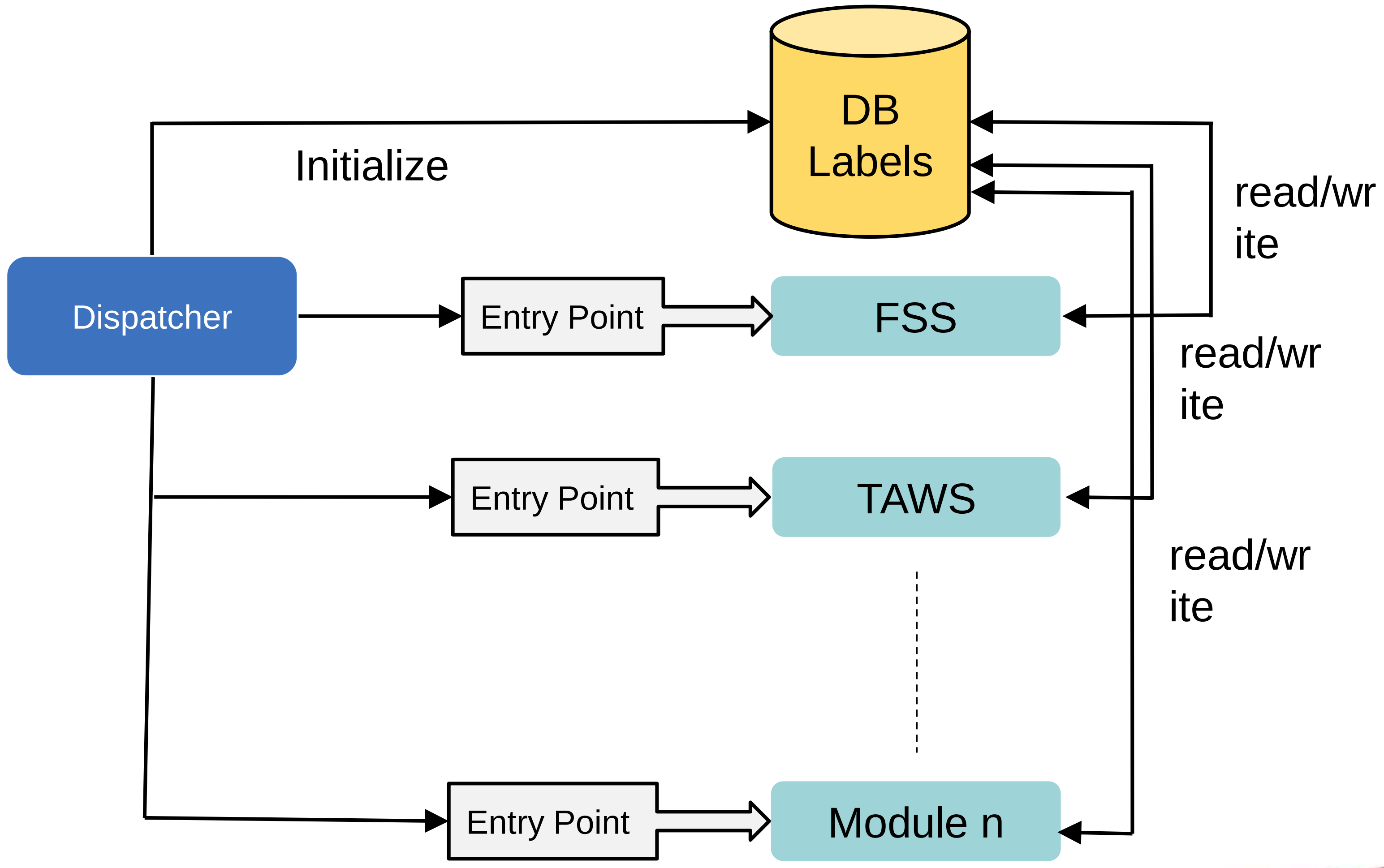
Simulation Scenario




TAWS
Warning Obstacle TRUE

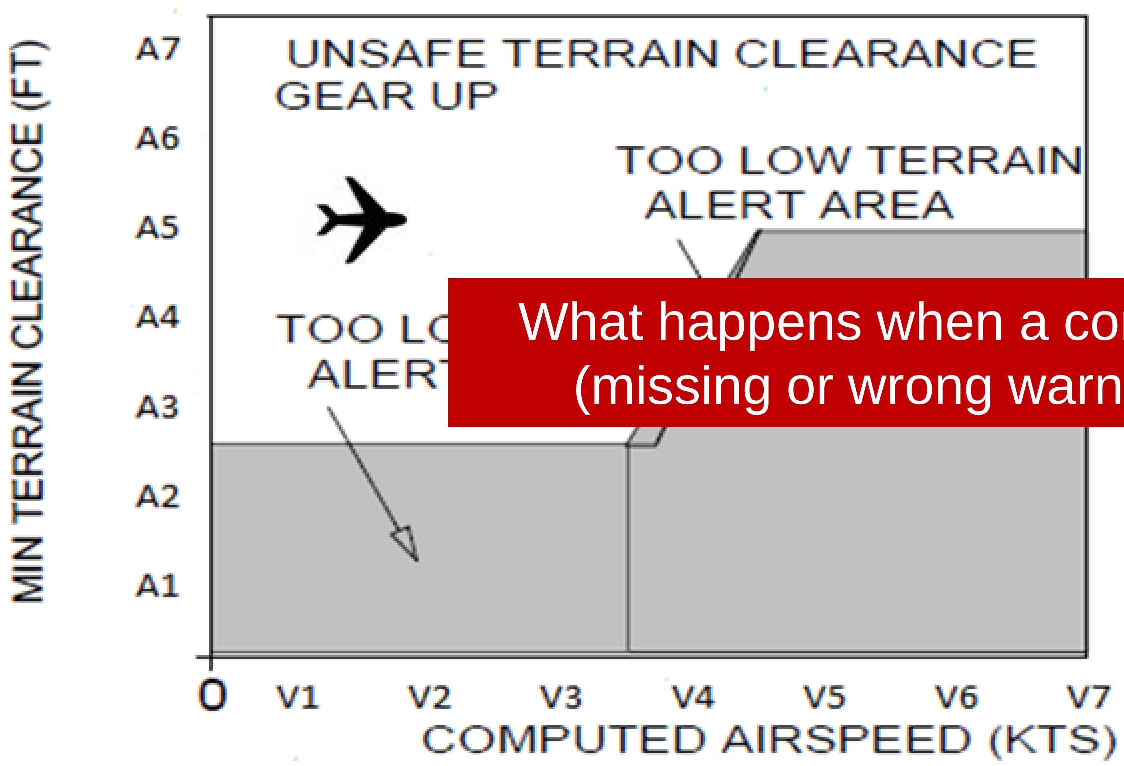
TAWS
Avoid Obstacle FALSE

CAE SW Architecture

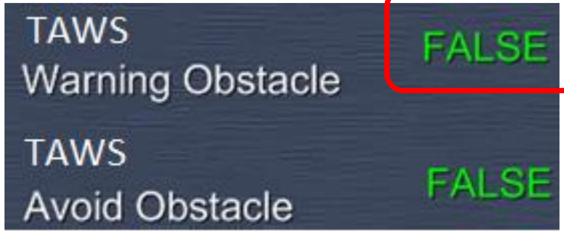


FSS: Flight Surveillance System
TAWS: Terrain Awareness and Warning System

The Problem

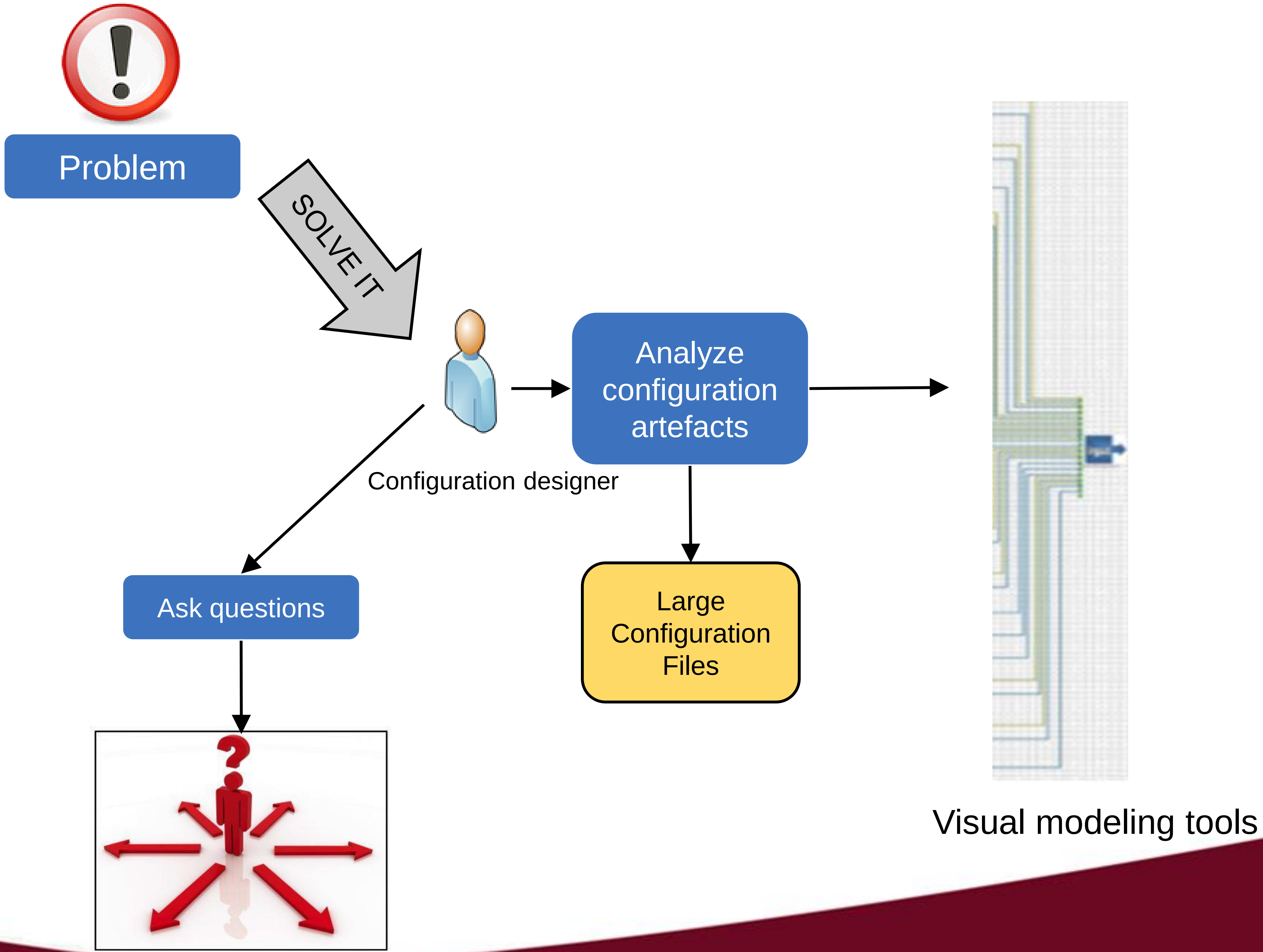


This should be TRUE

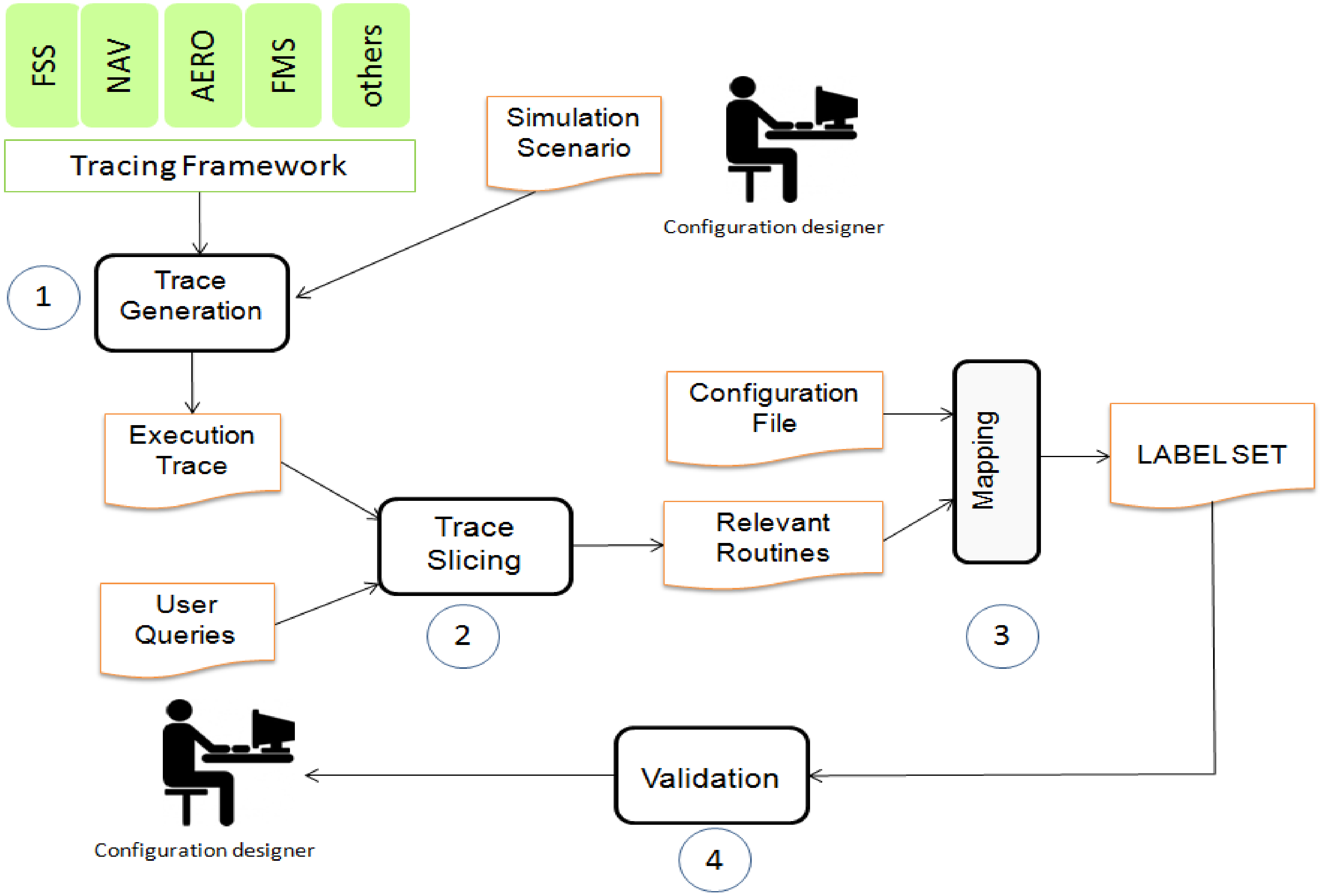


TAWS Warning Obstacle	FALSE
TAWS Avoid Obstacle	FALSE

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


Problem
SOLVE IT
Configuration designer
Analyze configuration artefacts
Large Configuration Files
Ask questions
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.

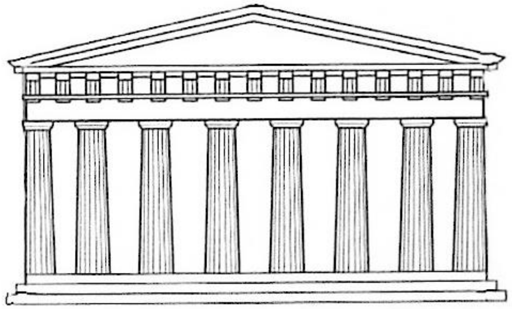
FELODE Precision and Recall

Scenarios	N1	N2	N3	Precision (N2/N1)	Recall (N2/N3)
S1	2	1	2	50%	50%
S2	6	3	3	50%	100%
S3	6	3	3	50%	100%
S4	8	3	3	38%	100%
S5	7	4	4	57%	100%

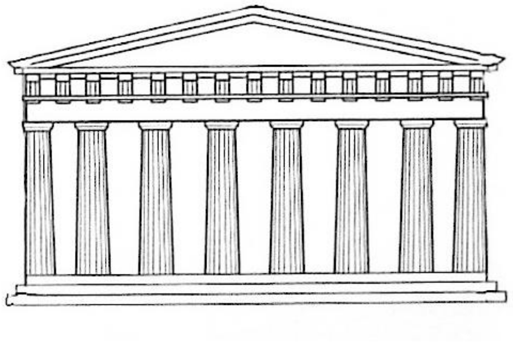
N1: Number of labels detected using FELODE
N2: Number of valid labels using FELODE
N3: Number of valid labels relevant to each scenario (provided by the users)

Future Directions

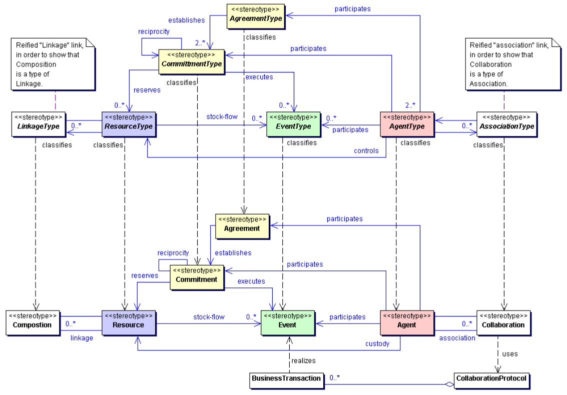
Foundations
(pattern matching,
machine learning, etc.)



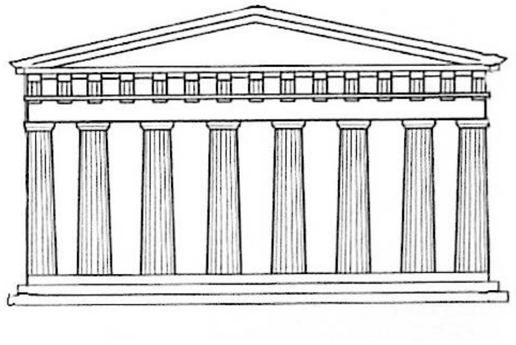
Foundations (pattern matching, machine learning, etc.)



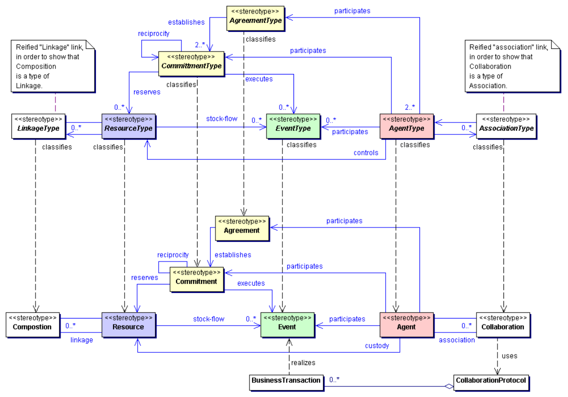
Model Tracing



Foundations (pattern matching, machine learning, etc.)

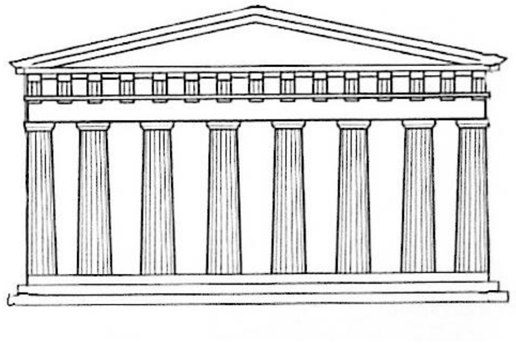


Model Tracing

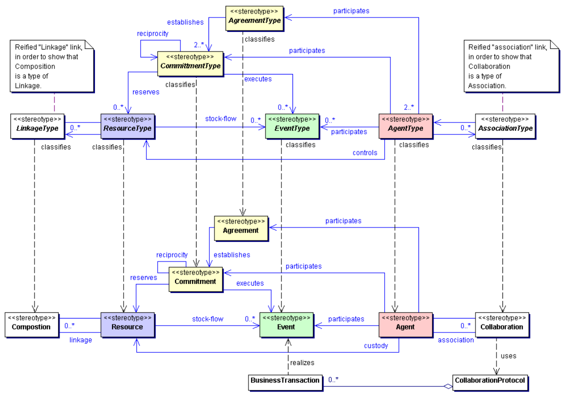


Embedded Systems

Foundations (pattern matching, machine learning, etc.)



Model Tracing

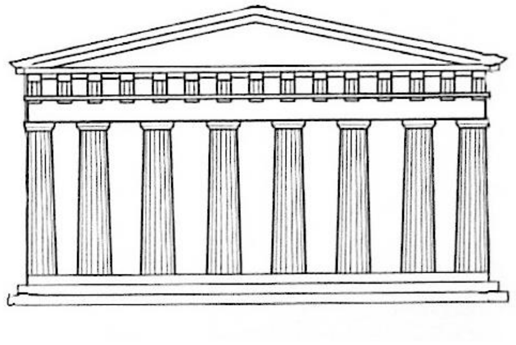


<<stereotype>> AgreementType
<<stereotype>> CommittmentType
<<stereotype>> LinkageType
<<stereotype>> ResourceType
<<stereotype>> EventType
<<stereotype>> AgentType
<<stereotype>> AssociationType
<<stereotype>> Agreement
<<stereotype>> Commitment
<<stereotype>> Compostion
<<stereotype>> Resource
<<stereotype>> Event
<<stereotype>> Agent
<<stereotype>> Collaboration
BusinessTransaction
CollaborationProtocol
Reified "Linkage" link, in order to show that Composition is a type of Linkage.
Reified "association" link, in order to show that Collaboration is a type of Association.
establishes
reciprocity
participates
classifies
executes
reserves
stock-flow
controls
linkage
custody
association
realizes
uses
0..*
2..*

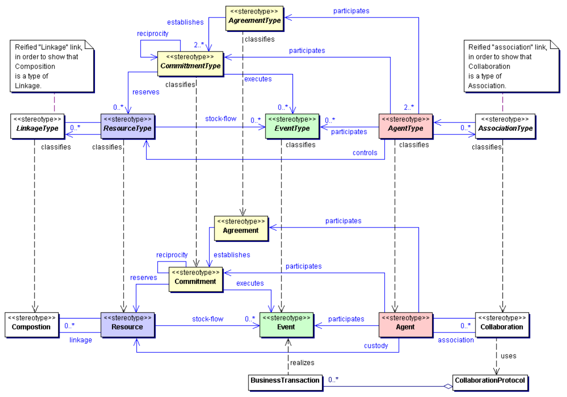
Embedded Systems

Tracing the Cloud

Foundations (pattern matching, machine learning, etc.)



Model Tracing



Embedded Systems

Tracing the Cloud

Trace Analytics

Thank you