Predictive Analytics: a Shortcut to Dependable Computing

SERENE Workshop 2017

Geneva

Miroslaw Malek

September 4, 2017

Università della Svizzera italiana
Faculty of Informatics
Advanced Learning and Research Institute ALaRI

Technology is changing all the time

So are
the methods

Pan Am Airlines
transferring
5 MB hard drive in 1956

The Age of Computricity
- An ever more Complex World

Sensors

Networks

Actuators

Mobile Devices

People/Stakeholders

Data Centers/Clouds/Grids

The Age of Computricity
- An ever more Complex World

10B per year

Sensors

Networks

Actuators

Mobile Devices

People/Stakeholders

Data Centers/Clouds/Grids

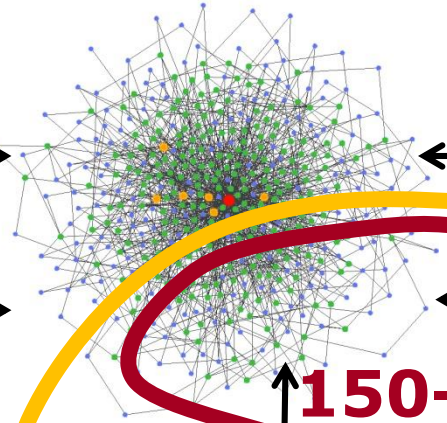
The Age of Computricity
- An ever more Complex World

Sensors

10B per year

Networks

Actuators



150-300M per year

Mobile Devices

People/Stakeholders

Data Centers/Clouds/Grids

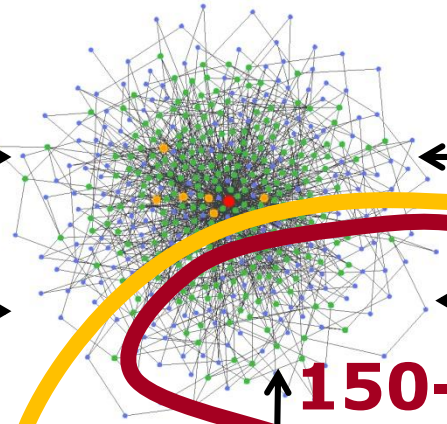
The Age of Computricity
- An ever more Complex World

Sensors

10B per year

Actuators

Networks



150-300M per year

7B

Mobile Devices

People/Stakeholders

(80M per year)

Data Centers/Clouds/Grids

Internet of Things - the Impact

- Over 20 billion IoT devices will be connected by 2020 (Gartner), some talk about over 100 B by 2025

- Economic impact of IoT technology:
  2.7 to 6.2 trillion dollars turnover by 2025
  (McKinsey & Co.)

- Many of IoT devices have to operate in environments characterized by uncertainty, insecurity (both physical and cyber), and instability

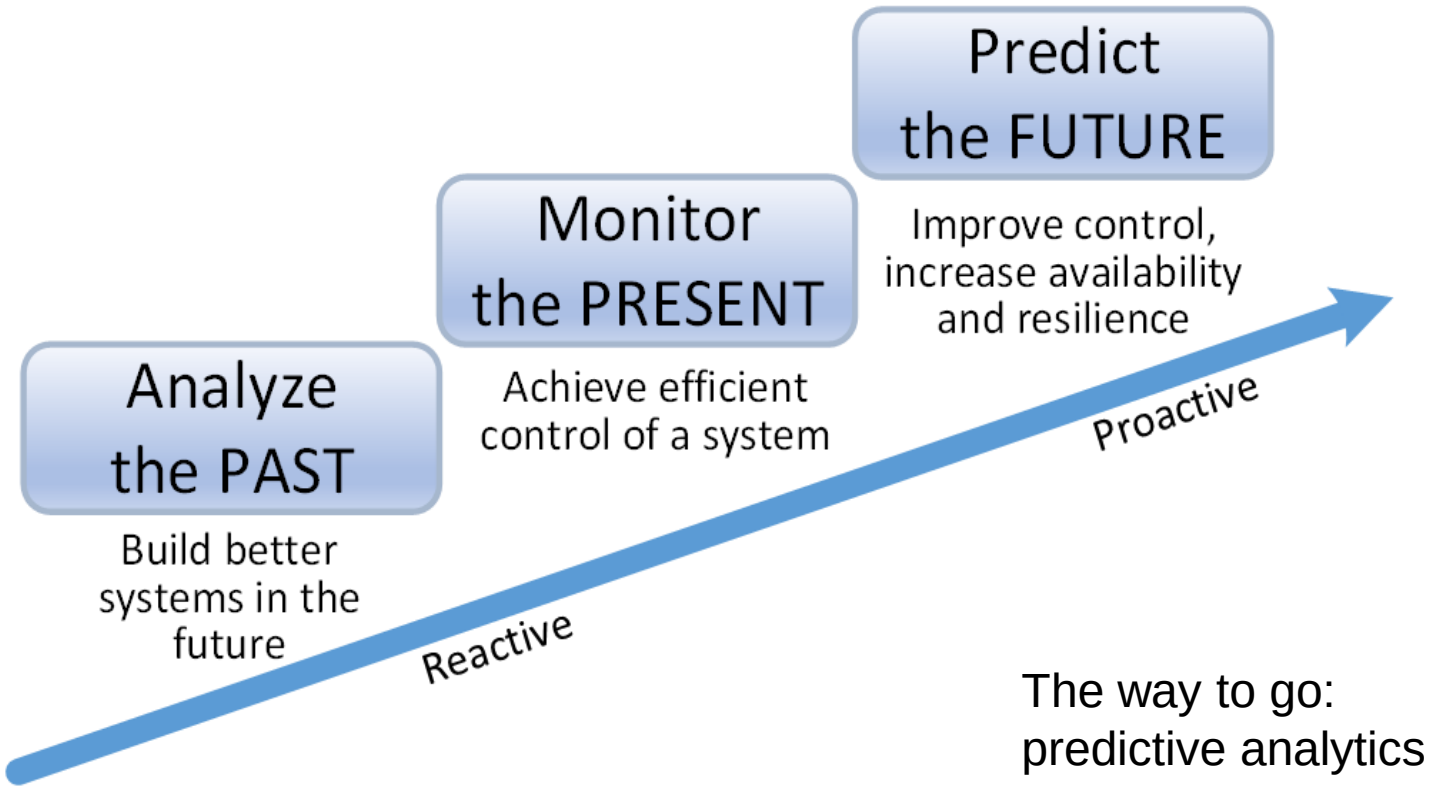
The Three Tyrants* Challenge

- **Complexity**
  - Growth in practice can hardly be stopped
  - Striving for new functional features
  - Striving for improved properties (e.g. higher performance, higher reliability)
  - Striving for having everybody and everything on the net
  - Open systems, adding environment, including people
  - BIG DATA, data analytics
- **Time**
  - Time can neither be stopped nor regained
  - Disparity between physical and logical time
- **Uncertainty**
  - Can be controlled to a limited extent, we have "to cope" with it
  - New failure modes, new environmental conditions, new attacks

***Inspired by a quote from Johann Gottfried von Herder (1744-1803): Die zwei größten Tyrannen der Erde: der Zufall und die Zeit**
"Two biggest tyrants on Earth are: the chance and the time."

Paradigm Shift

From analyzing the past and the present to predicting and constructing the future



The way to go:
predictive analytics

Change Your Mindset

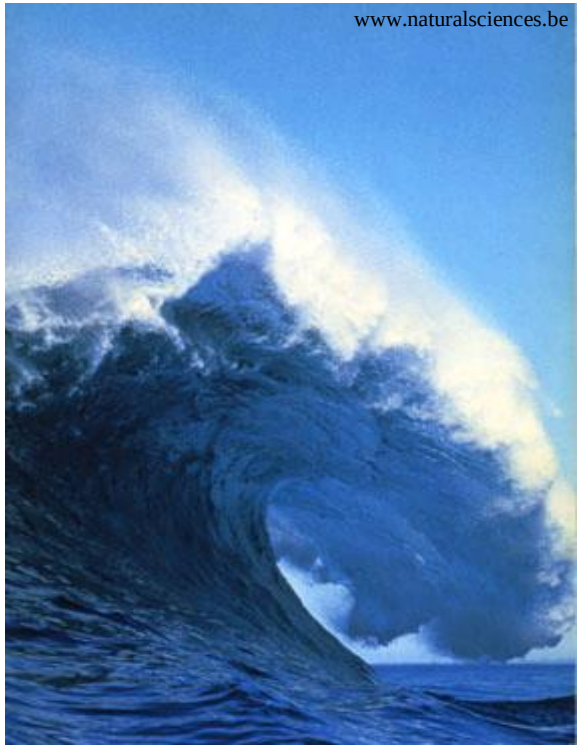
Don‘t wait for a failure, anticipate and avoid it,

or at least minimize the potential damage

Five-Minute Prediction

for computers is like …

FIVE-DAY prediction for tsunami



What Is Predictive Analytics?

- Predictive analytics applies a set of models, methods/algorithms and classifiers that use historical and current data to forecast future activity, behavior and trends.

- Predictive analytics involves creating predictive models and applying stochastic analysis, function approximation, machine learning and other methods to determine the likelihood of a particular event taking place.

- In our field, it is used in proactive fault management, failure prediction and predictive maintenance

Models and their Interpretation

The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes **observed phenomena**. The justification of such a mathematical construct is solely and precisely that it is expected to work.

John von Neumann *(1903 - 1957)*

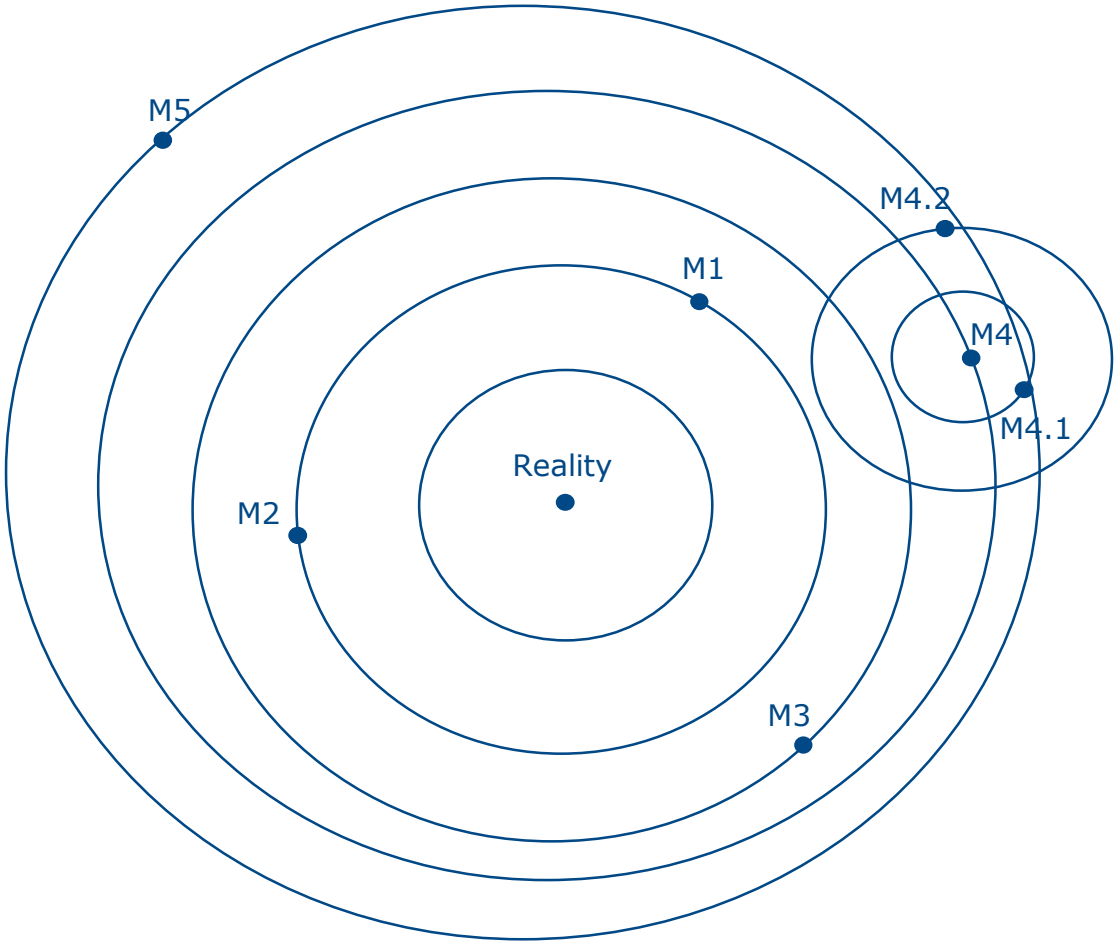
The Three Tyrants and the Models

- **Complexity**
  - many models are useless
  - most models do not scale

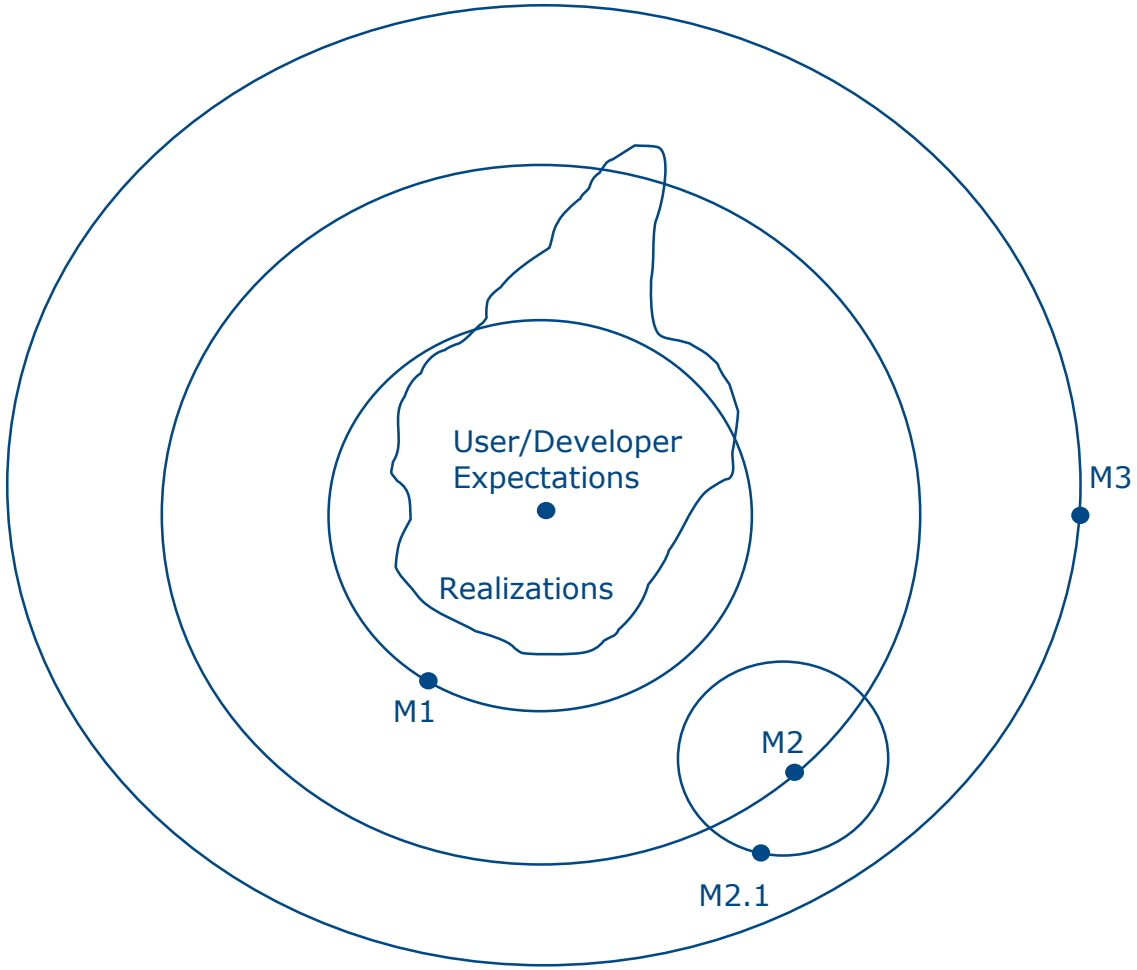
- **Time**
  - most models do not consider time

- **Uncertainty**
  - most models consider neither faults nor cyber attacks
  - some models consider faults as exceptions rather than as a part of specification

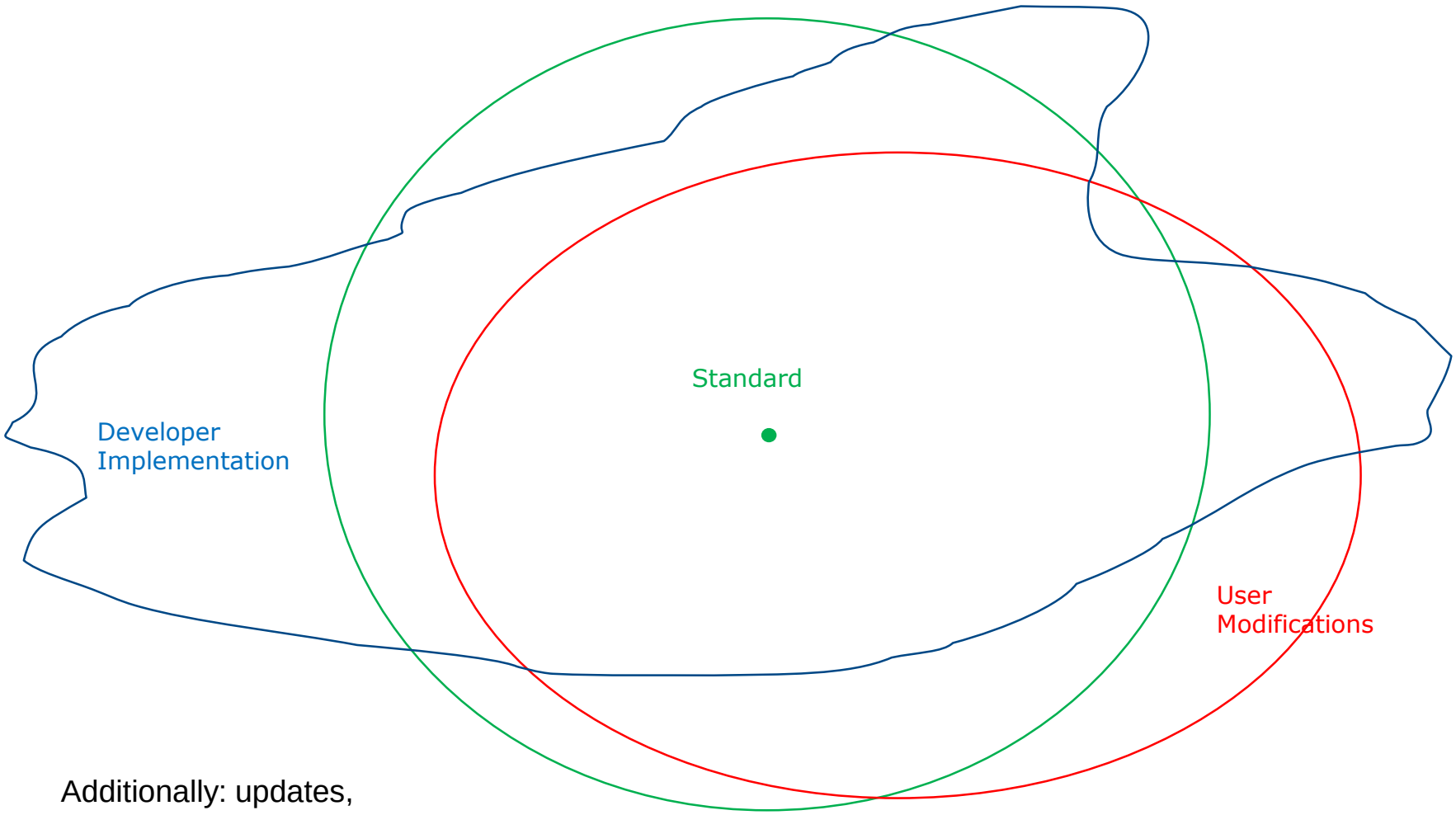
Distance from Reality



Distance from Reality in Software



Real World (even if there is a standard)



Additionally: updates, upgrades, etc.

Specification, Models and Reality

Specification is an ideal thing, implementation is a real thing; the confusion of the ideal with the real never goes unpunished.

Inspired by Goethe: “Love is an ideal thing, marriage is a real thing; the confusion of the ideal with the real never goes unpunished.”

What to Do? – Define a Goal and Get Closer to Empirical Modeling and Reality

After having done analysis, simulation, design and prototypic implementation of the system get closer by:

Reevaluating the model using data-driven, runtime monitoring, feature selection methods, fault injection and measurement

On Experiments

- *Today's scientists have substituted mathematics for experiments, and they wander off through equation after equation, and eventually build a structure which has no relation to reality.*

  Nikola Tesla (1856 – 1943)

- *There are three principal means of acquiring knowledge... observation of nature, reflection, and experimentation. Observation collects facts; reflection combines them; experimentation verifies the result of that combination.*

  Denis Diderot (1713 – 1784)

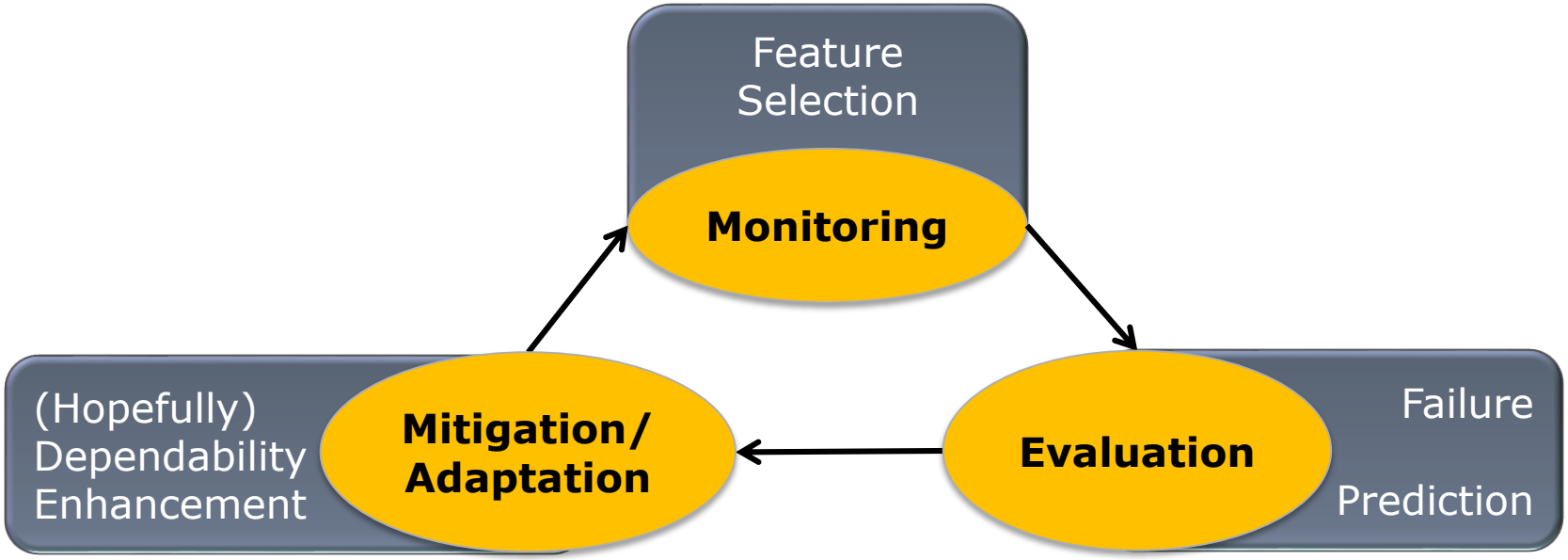
Data-Driven, Runtime Monitoring

- Although a plethora of tools exists the question remains: What to monitor and how often?
- Usually, an almost arbitrary feature (variable, parameter, event) selection
- Diverse formats, interfaces, databases, data quality and capabilities
- Flood of data (some tools or companies generate or process tens of GB‘s, TB‘s or even PB‘s per day)

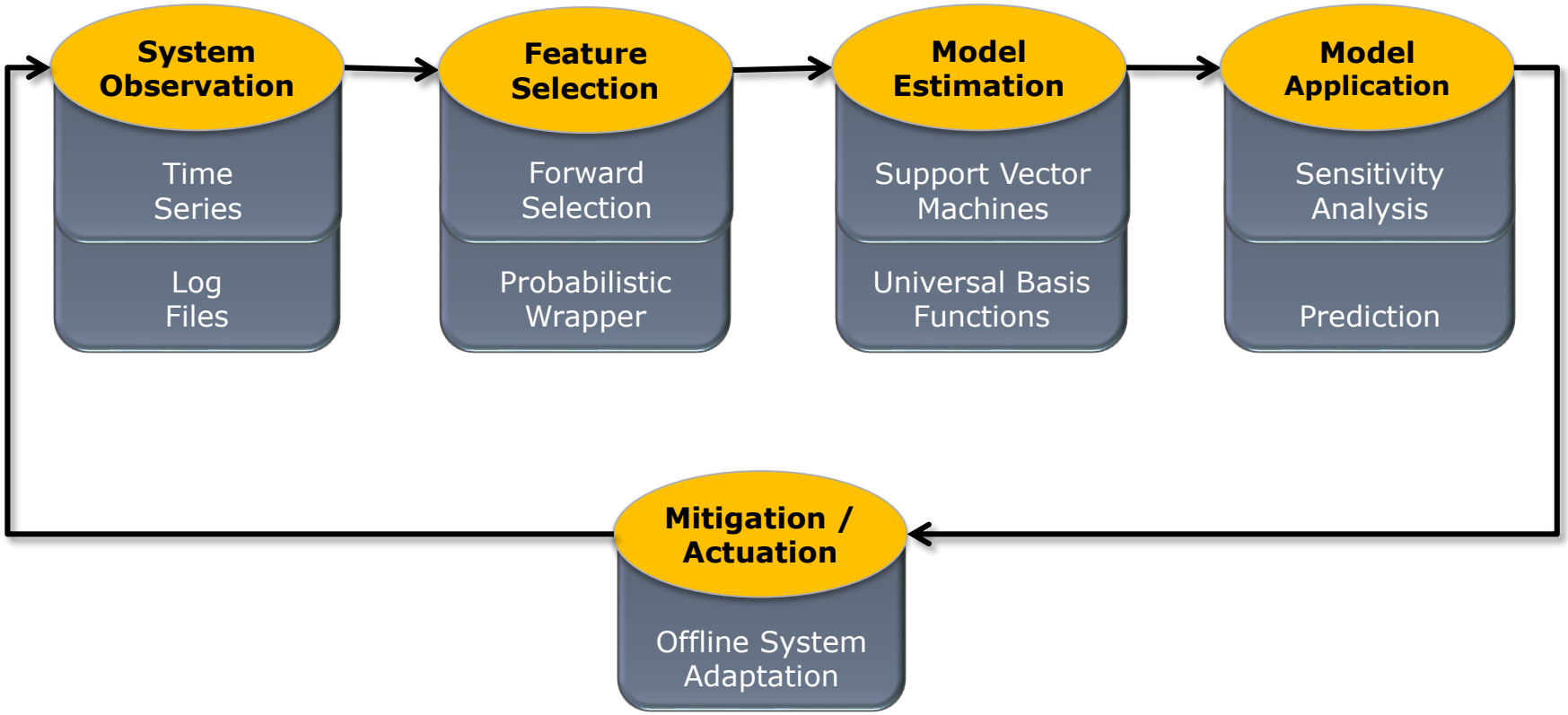
Proactive Fault Management (PFM)

- PFM is an umbrella term for techniques such as monitoring, diagnosis, prediction, recovery and preventive maintenance

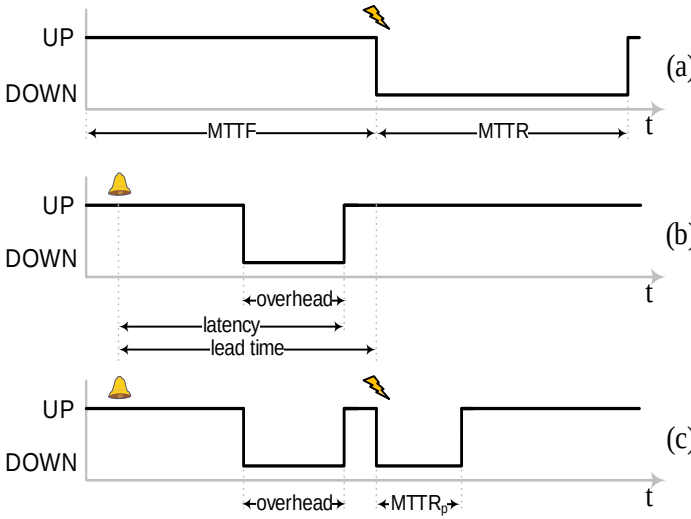
- Don‘t wait for a failure, act in advance



How to Get There (Off-Line Loop)?



Proactive Management Model

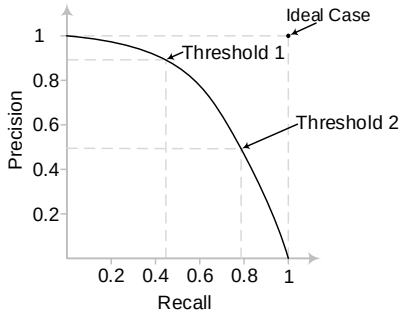


Reactive

Failure Prevention

Repair-Time Minimization

		Observation	
		Failure	***No failure***
Prediction	***Failure***	True positive	False positive
	No failure	False negative	True negative



$$Penalty = overhead$$
$$Reward_{fp} = \text{MTTR} - \text{overhead}$$
$$Reward_{rtm} = \text{MTTR} - (\text{overhead} + \text{MTTRp})$$

$$Precision = \frac{n_{tp}}{n_a} = \frac{n_{tp}}{n_{tp} + n_{fp}}$$
$$Recall = \frac{n_{tp}}{n_f} = \frac{n_{tp}}{n_{tp} + n_{fn}}$$

To Predict or Not to Predict?

- Steady-state availability

$$A_P = \frac{MTTF}{MTTF+MTTR-R*(reward-\frac{1-P}{P}*penalty)}$$

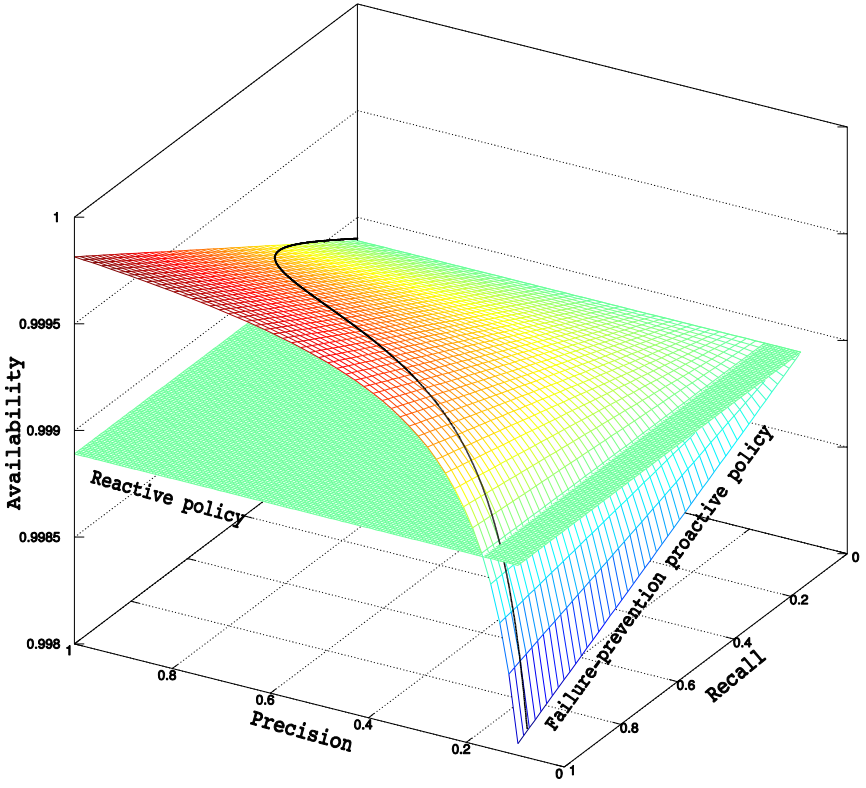
- *Breakeven point*

$$P \geq \frac{penalty}{penalty+reward}$$

- *A-score*

$$A_{score} = R * \left(reward - \frac{1-P}{P} * penalty\right)$$

		Optimization Metric		
		A-score	***F1-measure***	***High Recall***
r	***Availability***	0.998890	0.998890	0.998890
	Downtime per year [h]	9.72	9.72	9.72
fp	***Availability***	**0.999483**	0.999461	0.998506
	Downtime per year [h]	**4.52**	4.72	**13.09**
	Precision	0.5598	0.6856	0.2710
	Recall	0.7607	0.6800	0.9000
rtm	***Availability***	**0.998988**	0.998987	0.998833
	Downtime per year [h]	**8.86**	8.88	**10.22**
	Precision	0.7208	0.6856	0.2710
	Recall	0.6536	0.6800	0.9000



I. Kaitović, and M. Malek, "Optimizing Failure Prediction to Maximize Availability", IEEE ICAC 2016, Proc. of International Conference on Autonomic Computing , Würzburg, Germany, July 2016

Data-Driven Approach - Two Case Studies

We have applied data-driven approach to:

- Failure prediction in servers in telephone networks

- Early detection of malware in smartphones

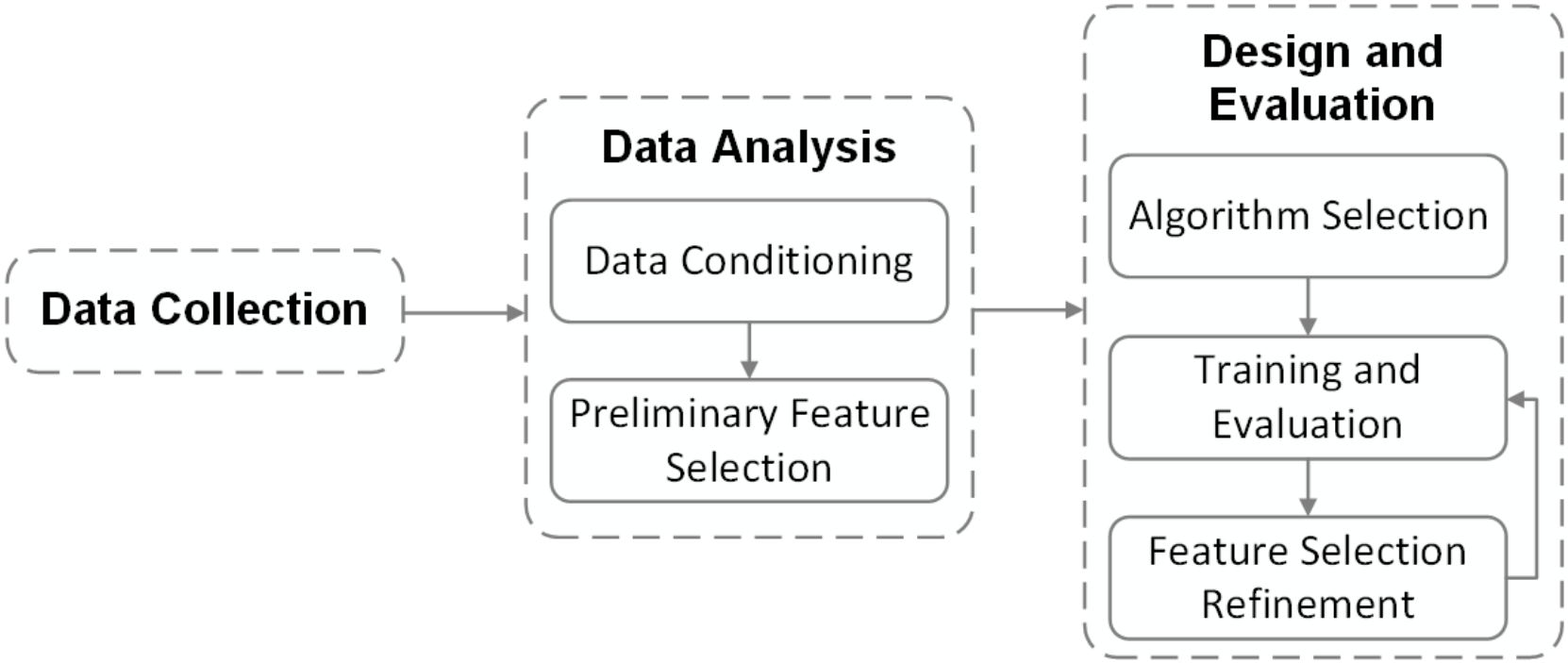
Our Failure Prediction Philosophy

- Faults, errors and failures are common events so let us treat them as part of the system behavior and learn how to cope with them

- Attractive panacea:

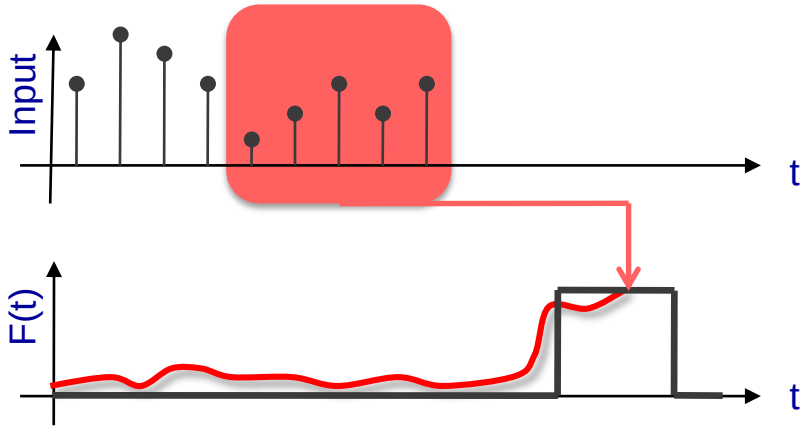
  Proactive Fault Management (PFM)

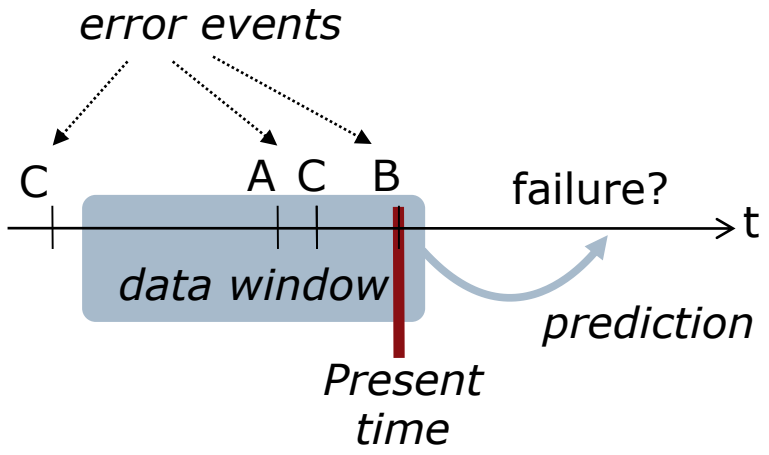
Predictor Design Phases



Two Types of Input Data

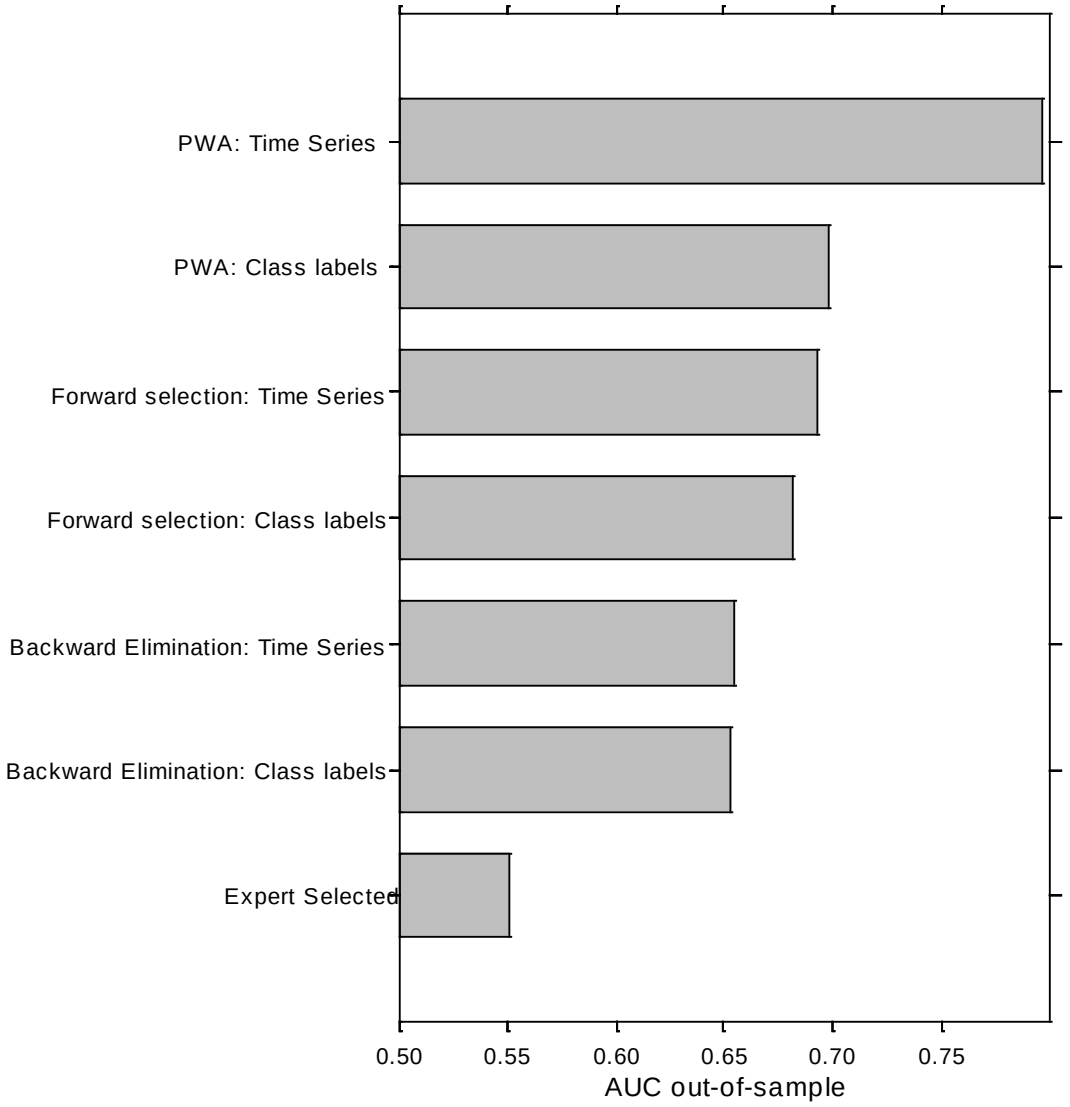
- There are two types of system measurements
  - periodic, numerical
  - event-based, categorical
- Examples for periodic data
  - system- / CPU load
  - memory usage
- Examples for event-based data:
  - interrupts
  - threshold violations
  - error events





Variable Selection

- Benchmarked four techniques
  - Forward selection
  - Backward elimination
  - Expert selected
  - PWA (Prob. Wrapper)
- Variables
  - *alloc*
  - *sema/s*
- PWA performs best on time series *and* class label data

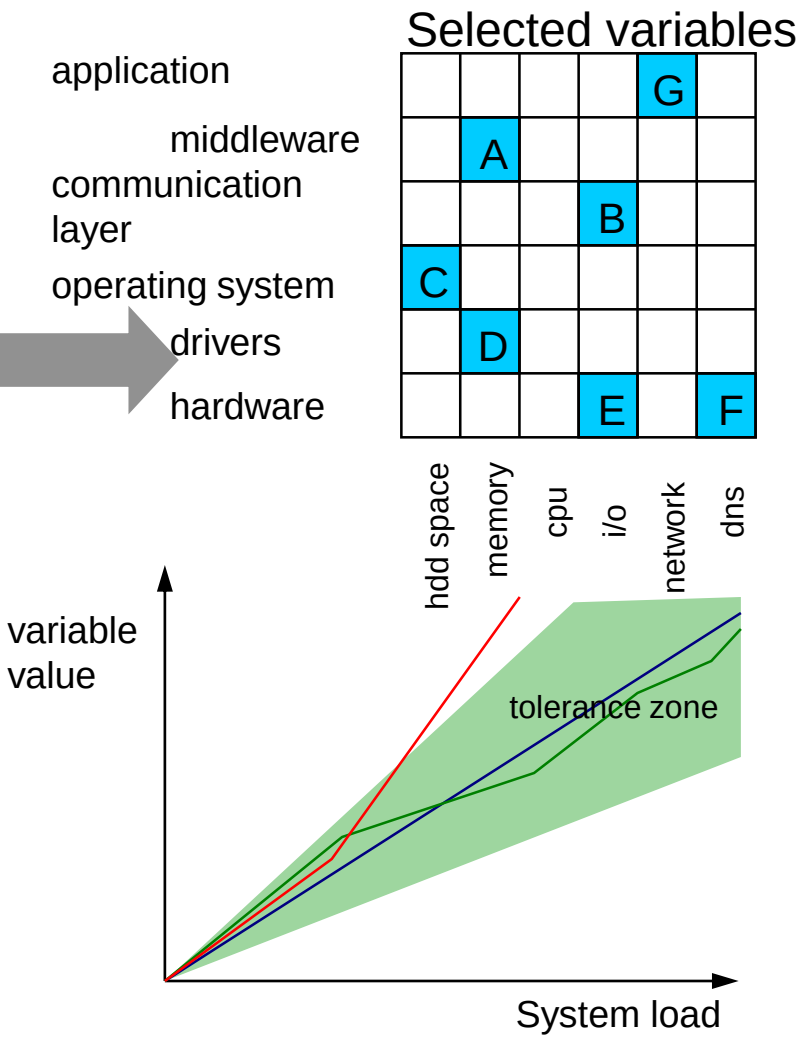


Three Models

- White Box (Nonlinearity Detection)
  - Lots of knowledge about a system is required
  - Root-cause analysis directly from the model
- Black Box (*Universal Basis Functions*)
  - Almost no assumptions about system are required
  - Root-cause analysis are limited
- Grey Box (*Hidden Markov Model*)
  - Some assumptions about system
  - Root-cause analysis more powerful

Method 1: Detection of Nonlinearities

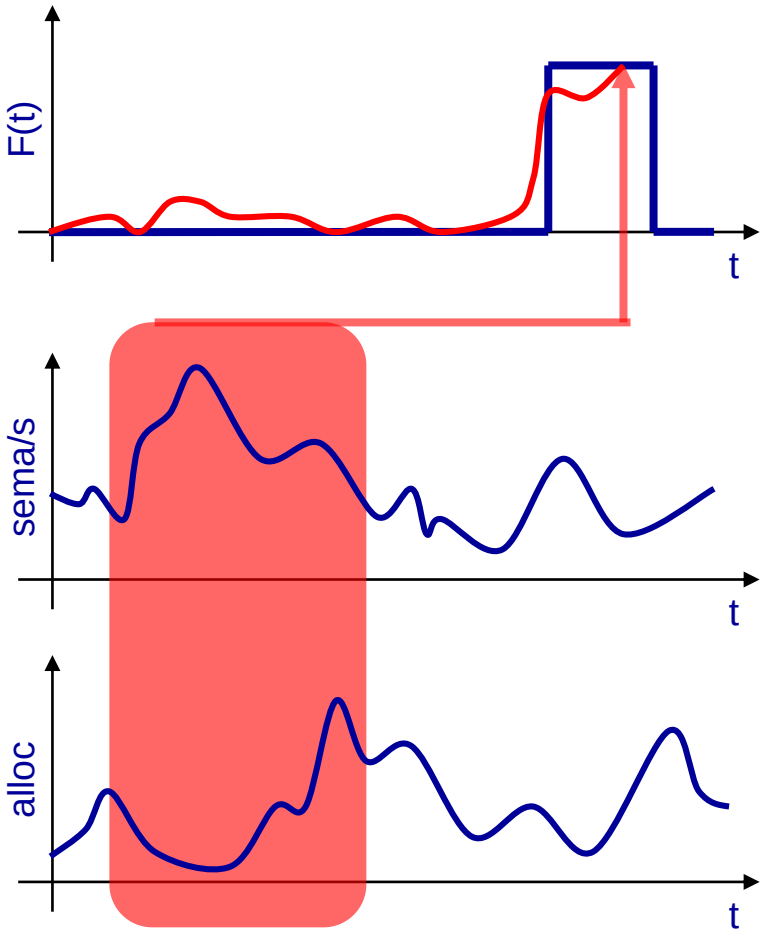
- Basic assumption:
  Well-designed systems have variables which increase linearly with the system load
- Adaptive selection of significant variables
- Monitoring the linearity of selected variables
- Failure Prediction:
  Detection of variables that move outside their tolerance zone



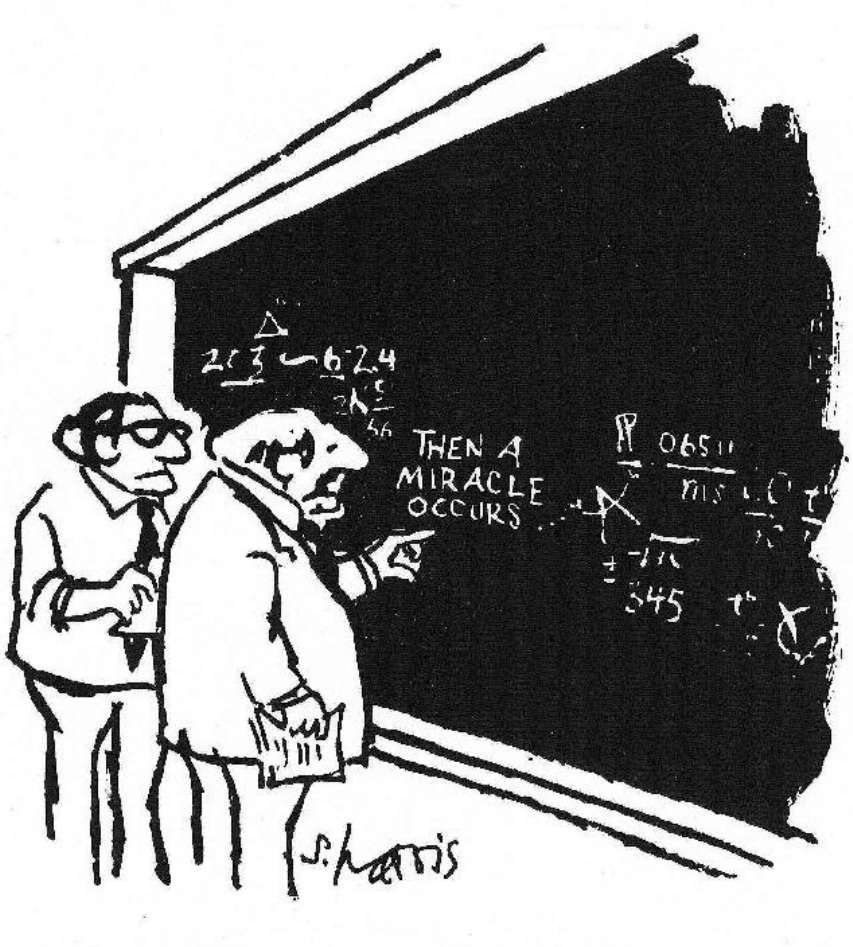
Method 2: Universal Basis Functions

- Function approximation: Failure probability as function of system variables
- Selection of indicative variables by means of variable selection
- Universal Basis Functions:
  - Linear combination of nonlinear kernel functions
  - Training with evolutionary algorithm
- Failure Prediction:

  Evaluation of kernel functions



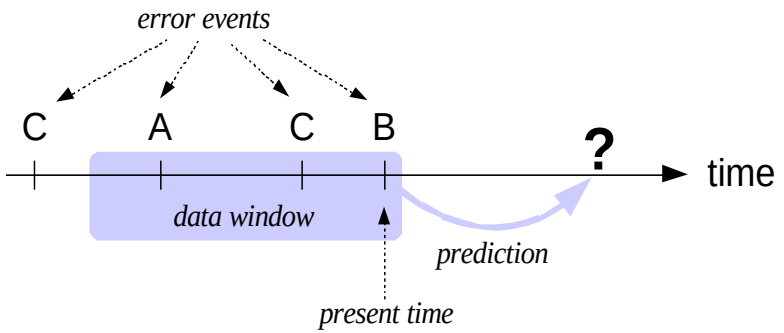
Algorithm's Details

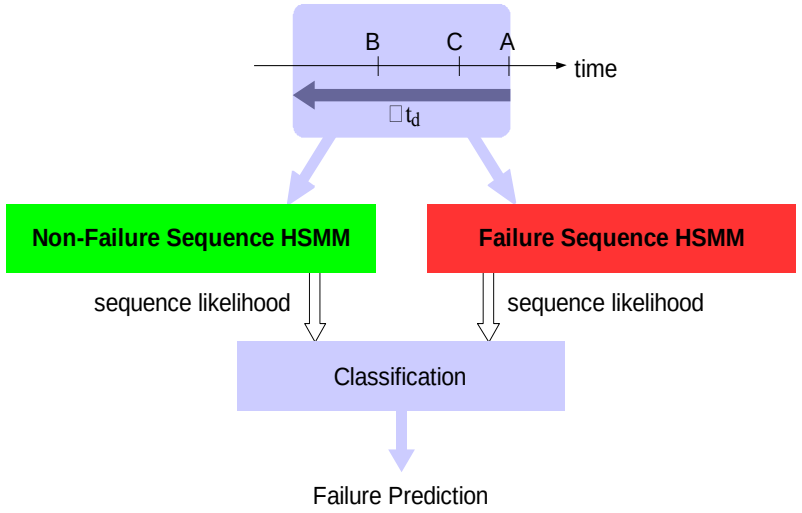


"I think you should be more explicit here in step two."

Method 3: Hidden Semi-Markov Model

- Statistical learning from previous failure occurrences
- Pattern recognition using Hidden semi-Markov models
- Two model instances:
  - Failure sequences
  - Non-failure sequences
- Failure prediction: Bayesian classification based on sequence likelihoods





Case Study 1 (with Günther Hoffmann): A Commercial Telecommunication System

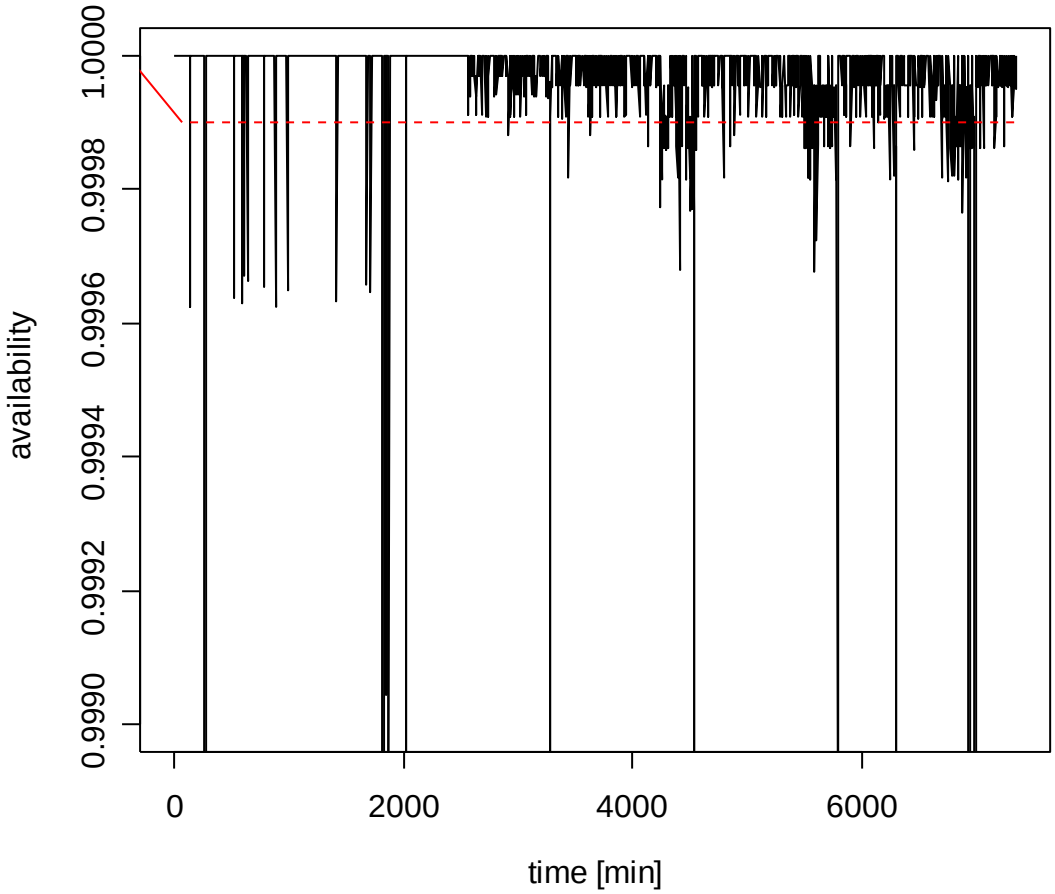
Distributed and component based system

- 1.5+ million lines of code
- 200+ components
- 1100+ variables can be measured
- handles value added services in GSM/GPRS networks (e.g. billing, SMS, pre-paid services)
- 400-10,000 service requests per minute
- Two nodes (up to eight)

→ **Measure availability and predict failures**

Plotting Availability

- Availability
  - $A_c(\Delta t) = 1 - (n_f / n_c)$
  - $A_c(\Delta t) < 0.9999$
- Interval:
  Δt = 5 minutes



Lessons Learned from the Case Study 1

- The power of predictive analytics is potentially immense (e.g., the potential to significantly improve system availability/survivability by an order of magnitude or more)
- The choice of models is not as significant as the choice of variables
- Prediction methods should be tested and applied across disciplines interchangeably
- Taming **uncertainty**, **time** and **complexity** will remain the key challenge

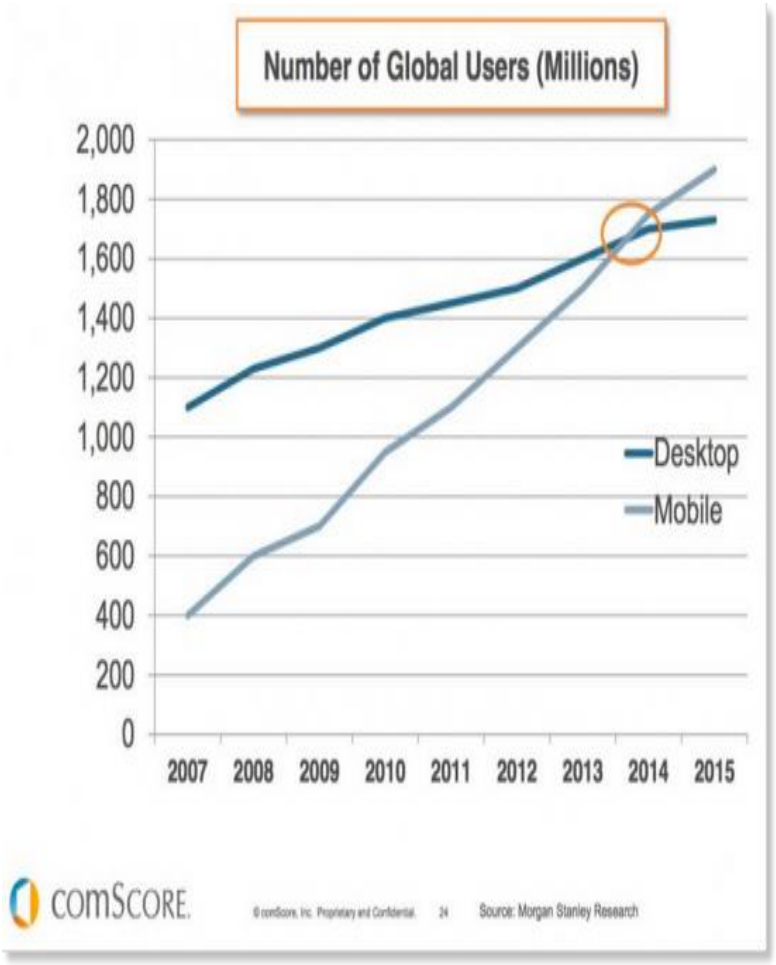
Case Study II: Early detection of malware (with Jelena Milesovic and Alberto Ferrante)

- Early detection of malware by observing CPU and memory features

- Dynamic detection on the phone

Motivation/ Why Mobile?

- Mobile devices contain more private (sensitive) data than PCs/tablets ever will
- Mobile apps represent more than a half of Internet use today [1]
- Over five billion Android devices have been sold since 2009[2]



[1] Lookout blog https://blog.lookout.com/blog/2016/09/29/chamber-of-commerce-mobile-security/
[2] Statistics and facts about Android https://www.statista.com/topics/876/android/

Requirements for Mobile Malware Detection

- Mobile malware is similar to PC malware
  - high detection performance methods are needed
  - usually requires detection methods of high complexity

- The environment is constrained:
  - by battery, memory, CPU
  - complex detection solutions are impractical

Two Approaches to Malware Detection

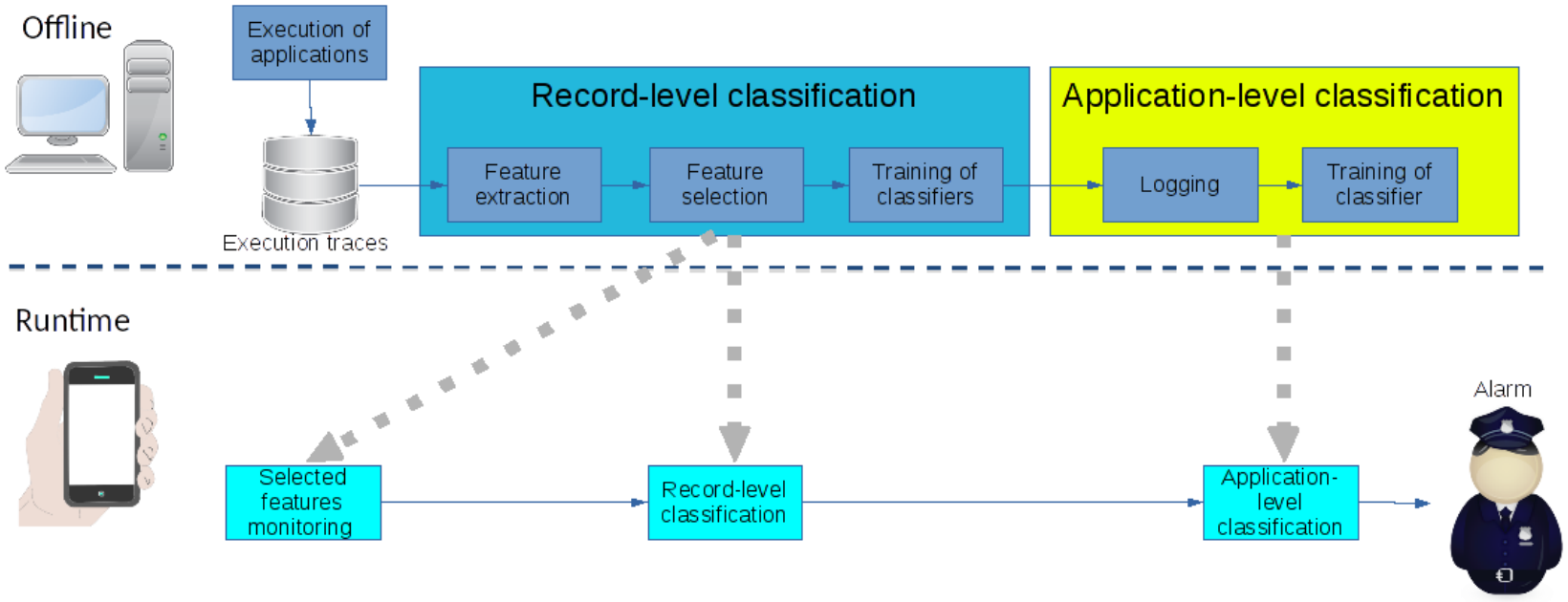
Static Detection

- (Mostly) *offline* investigation
- Analysis of *static features*

Dynamic Detection

- Investigation of the apps *during their execution*
- Analysis of *dynamic features*

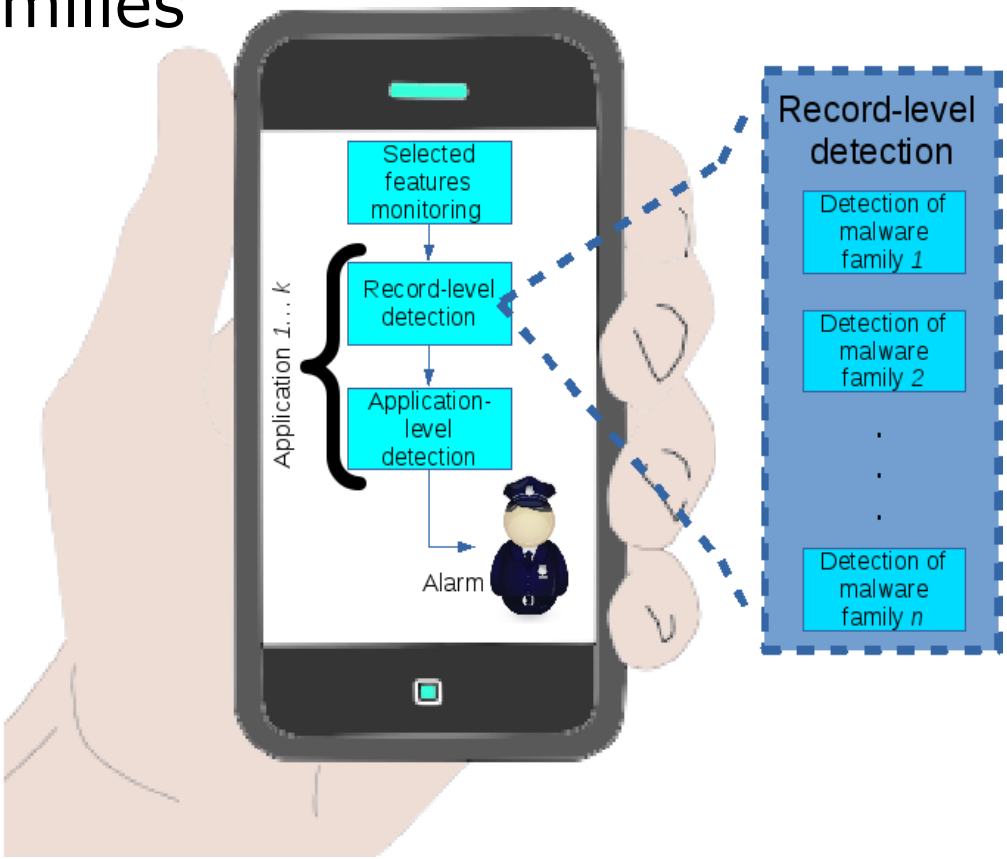
Proposed Architecture



Milosevic, J., Malek, M., and Ferrante A. A Friend or a Foe? Detecting Malware using Memory and CPU Features, SECRYPT 2016, Lisbon, Portugal, pp. 73-84, July 2016

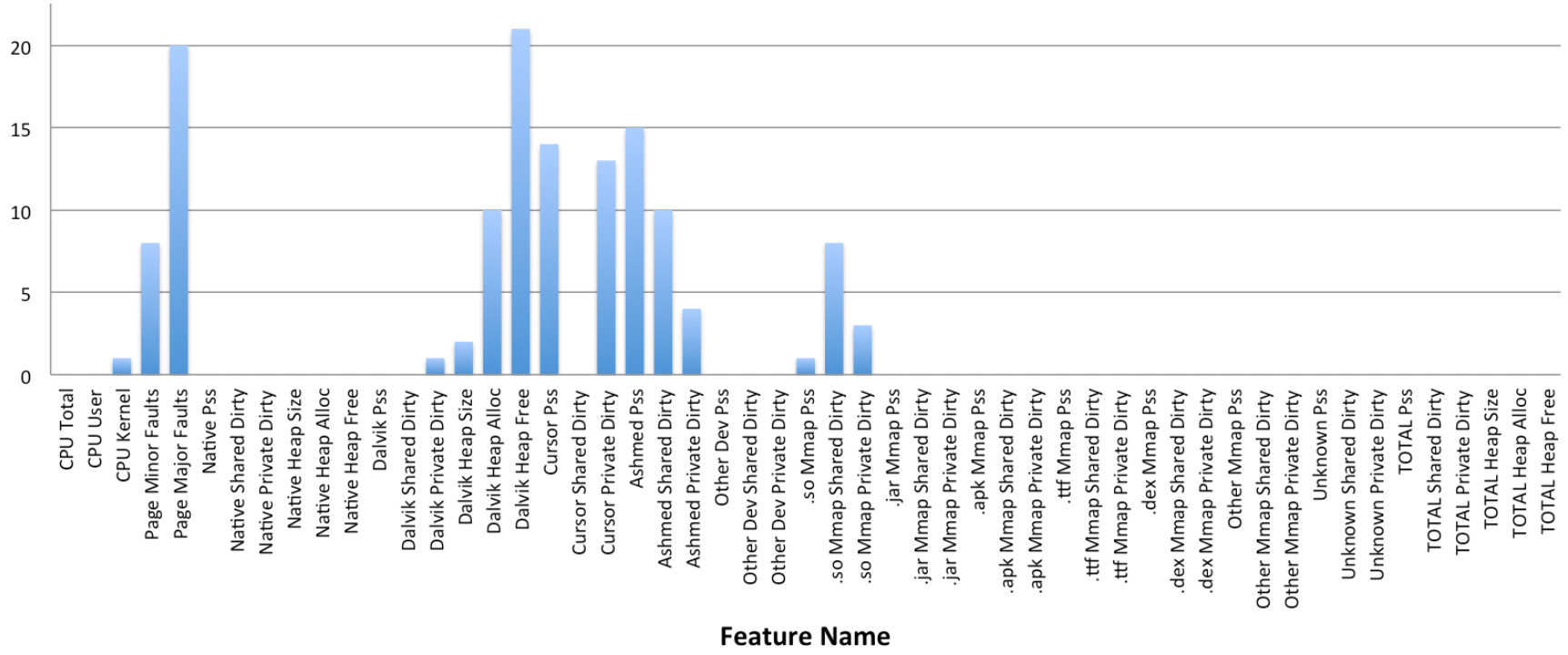
Discrimination between Diverse Malicious Behavior

- Discriminate, at runtime, between different malware families

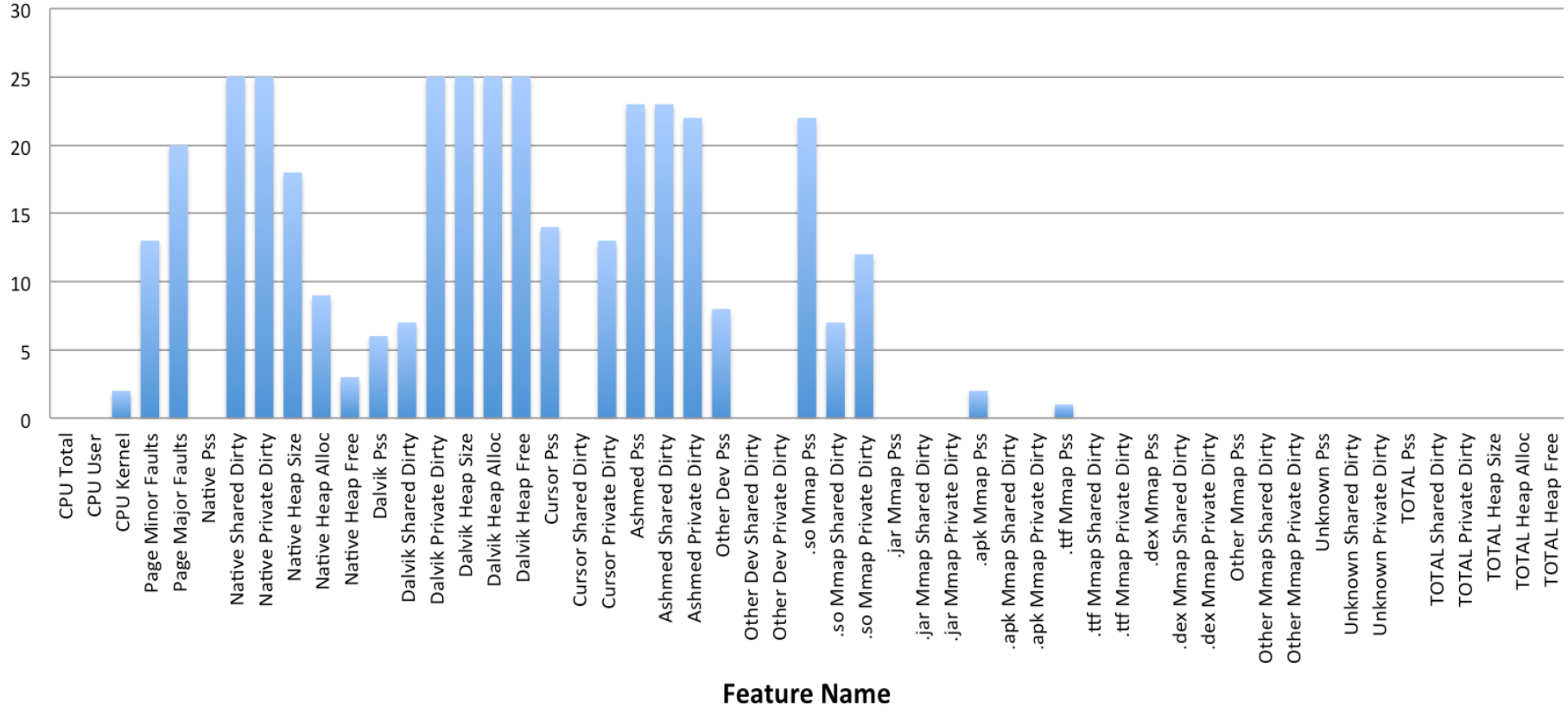


Milosevic J., Ferrante A., Malek M., *Trojan Families Identification Using Dynamic Features and Low Complexity Classifiers*, The 24th EICAR Annual Conference on Trustworthiness in IT Security Products EICAR-2016, Nuremberg, Germany

Frequency of occurrence of features among top 5



Frequency of occurrence of features among top 15



Experimental Setup: Dataset

Benign (950 applications)

- Downloaded from Google Play Store
- Belonging to different categories (education, entertainment, games, lifestyle, etc.)
- Analyzed with Virus Total to confirm that malware is not present

Malicious (1120 applications)

- Different range of behavior covered (installation methods, activation mechanisms, malicious payloads)
- Samples taken from Malware Genome Project and Drebin dataset

Execution Environment

- Run each application for ten minutes on Android Emulator
- Trigger different events of apps using Monkey runner
- Record applications behavior
  - Usage of memory, CPU, network resources and system calls
- Monitor system parameters every two seconds
- Reinitialize operating system (Android 4.0)

Extracted Features Related to

- Memory (48 features)
  - Virtual, native, Dalvik, Cursor, Android shared, memory-mapped native code, memory-mapped fonts, memory-mapped Dalvik code
- CPU usage (5 features)
  - Total, user, kernel
- Network statistics
  - Transport and Internet layer (number of packets, packets size, network load, etc.)
- System calls
  - System calls and statistics on system calls

Feature Selection

- In order to identify the most indicative features, we have used:
  - Principal Component Analysis
  - Correlation Attribute Evaluation
  - Correlation Feature Subset Evaluation
  - Gain Ratio Attribute Evaluator
  - Information Gain Attribute Evaluator
  - OneR Feature Evaluator

- The Weka tool implementation of the methods used [Hall et al]

Hall et al, The WEKA Data Mining Software: An Update, ACM SIGKDD Explorations Newsletter 2009

Results

- Identification of the most indicative features for efficient and effective malware detection
- Record-level detection
- Application-level detection
- Discrimination between diverse malicious behavior
- Detection of malicious sub-traces

Milosevic et al, *What Does the Memory Say? Towards The Most Indicative Features for Efficient Malware Detection*, The 13th Annual IEEE Consumer Communications and Networking Conference, Las Vegas, USA, January 2016

Milosevic et al, *A Friend or a Foe? Detecting Malware Using Memory and CPU Features*, The 13th International Conference on Security and Cryptography, Lisbon, Portugal, July 2016

Detection Performance of Record-level Classification

		Model		
		Initial	Max. F-measure	Optimized
Naive Bayes	Precision	0.79	0.84	0.84
	Recall	0.76	0.83	0.83
	F-measure	0.77	0.83	0.83
	No. of features	53	7	7
Logistic Regression	Precision	0.84	0.86	0.84
	Recall	0.84	0.86	0.84
	F-measure	0.83	0.86	0.84
	No. of features	53	38	7
J48 Decision Tree	Precision	–	–	0.83
	Recall	–		0.83
	F-measure	–	–	0.82
	No. of features	–	–	6

Performance of classifiers when different number of features are considered

The *F-measure* of the system is defined as the weighted harmonic mean of its precision and recall: $F = 2PR/(P + R)$

Application-level Detection Performance

Features	Metric	Window Length, Threshold, Checks	Malware Detected %	False Positives %	F-measure
Initial (53)	Highest F-measure	3, 80, 11	92.1	24.5	0.84
	Best malware detection	5, 60, 9	89.9	23.4	0.84
	Lowest false positives	15, 95, 5	65.2	10.6	0.74
Optimized (7)	**Highest F-measure**	**5, 85, 15**	**85.5**	**17.2**	**0.85**
	Best malware detection	10, 60, 7	89.9	24.7	0.83
	Lowest false positives	20, 9, 5	59.5	10.7	0.70

Malware detected, on the average, after 85 execution records (2mins and 45s)

Main Points

- We propose an approach to identification of behavioral signatures for different Trojan families and their most appropriate detectors

- Choice of features is the key.

  By observing only a limited number of features per Trojan family (from 3 to 13 features) and by using a detection algorithms of low complexity in the number of features (Naive Bayes, Logistic Regression or Support Vector Machines), execution records belonging to Trojans can be identified with a precision of up to 99.8%

- The proposed method is suitable for efficient and effective run-time usage on resource-constrained devices

Conclusions

- Data-driven, runtime monitoring and continuous model adjustment must become a standard practice and will play increasingly important role due to system dynamics and complexity. It will also result in better designs and more agile and robust systems.

- The focus on specific properties, separation of concerns, accurate experiments and measurement, fault injection, proactive control and management and getting closer to reality will result in effective methods to improve system performance and dependability.

- Taming complexity, uncertainty and time will remain a key challenge.

- Next frontier: constructing the future