Data Analytics for Software Systems Observability: Challenges and Opportunities

Prof. Wahab Hamou-Lhadj
Concordia University
Montréal, QC, Canada
wahab.hamou-lhadj@concordia.ca

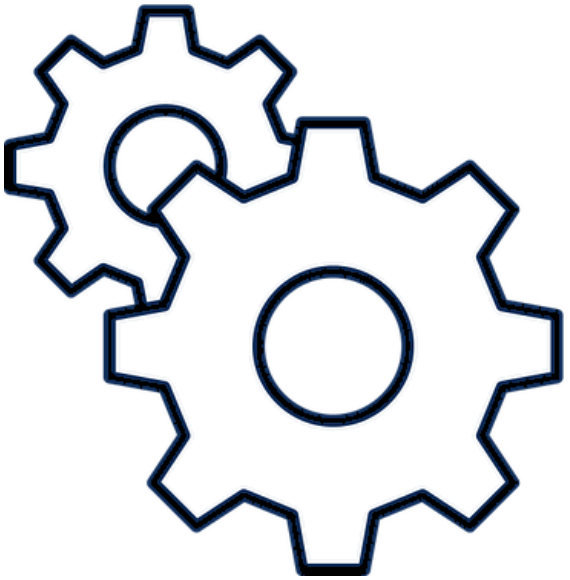
Keynote Presentation
IEEE 22nd International Conference on
Information Reuse and Integration for Data Science (IRI'21)
August 11, 2021

User vs. Operational Data

- **User data** describes information about users.
  - E.g. social media data, user preferences, geo-location data, images, etc.
  - Applications include marketing campaigns, fraud detection, image recognition, etc.

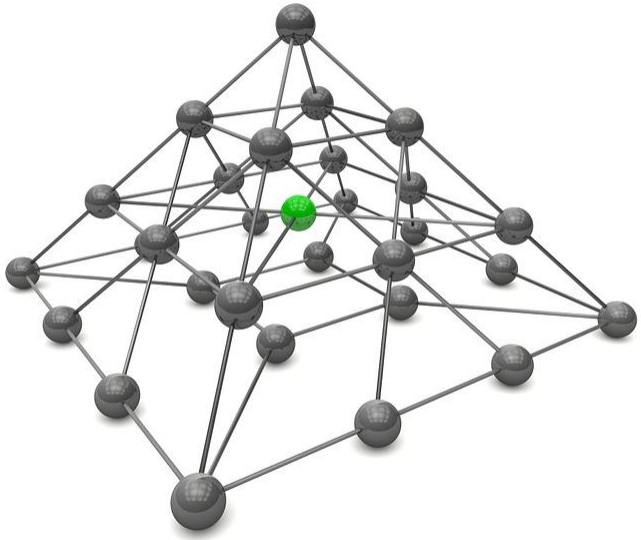
User vs. Operational Data

- **Operational (machine) data** describes information about a system (or a machine)
- It is collected automatically from devices, IT platforms, applications with no direct user intervention.
  - Useful for diagnosing service problems, ensuring reliability, detecting security threats, improving operations, and so on.



Operational Data for Software-Intensive Systems

- The proper functioning of software-intensive systems **relies heavily on operational data** to diagnose and prevent problems.
- New trends in SW dev. make this challenging:
  - Highly distributed and parallel systems
  - Micro-service architectures
  - Virtualisation and containerization
  - Device connectivity and IoT
  - Cyber physical systems
  - Intelligent and autonomous systems
  - Agile, DevOps, and continuous delivery processes

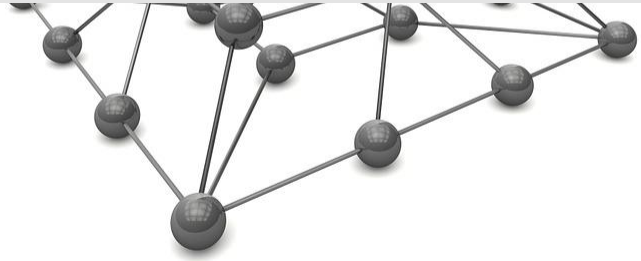


Operational Data for Software-Intensive Systems

- The proper functioning of software-intensive systems **relies heavily on operational data** to diagnose and prevent problems.

We need better runtime system analysis and fault diagnosis and prediction methods that provide full visibility of a system's internal states.

- Micro-service architectures
- Virtualisation and containerization
- Device connectivity and IoT
- Cyber physical systems
- Intelligent and autonomous systems
- Agile, DevOps, and continuous delivery processes



Software Observability

- In control theory:
  - **Observability** is “a measure of how well internal states of a system can be inferred from knowledge of its external outputs” [Wikipedia]

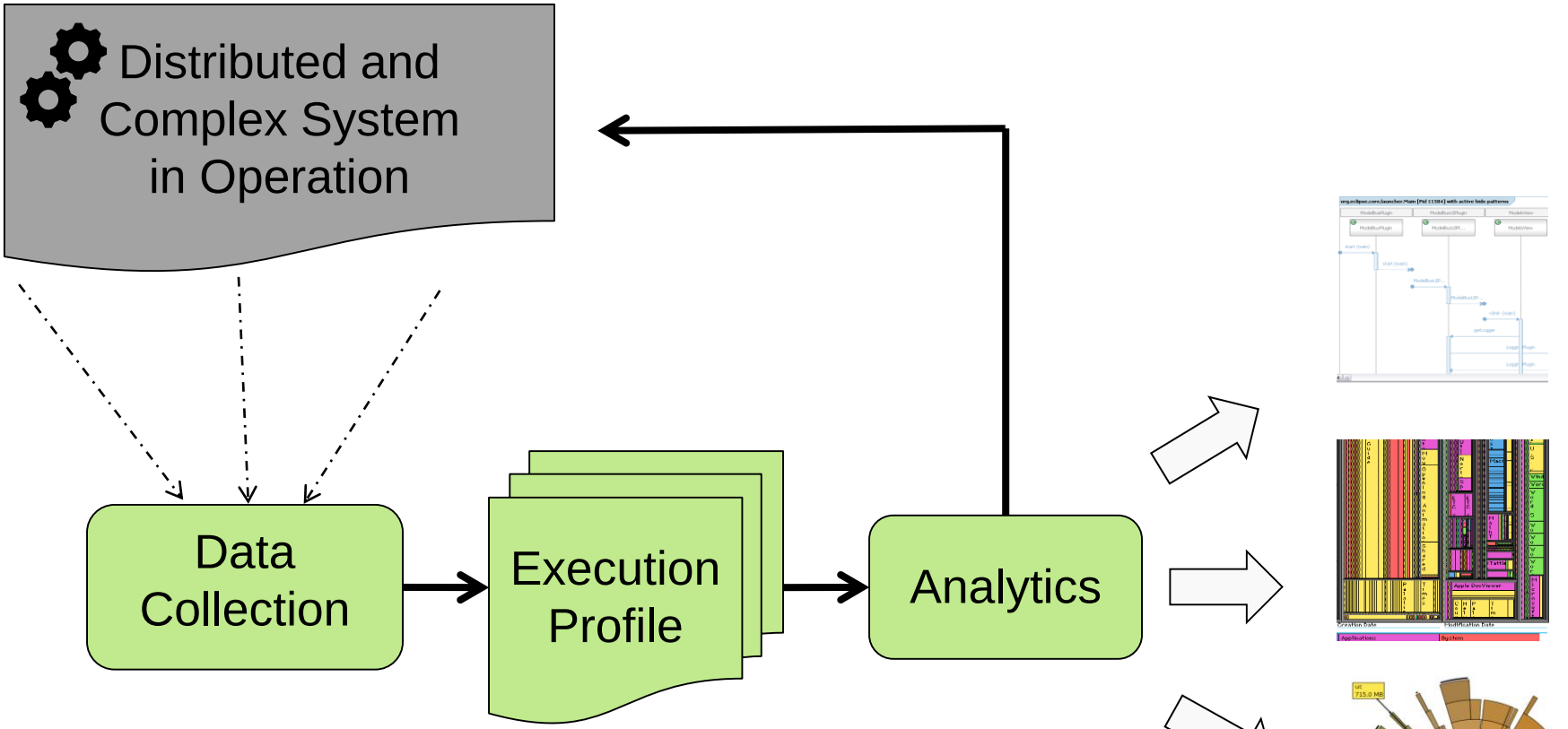
- Software Observability:
  - A set of end-to-end techniques and processes that allow us **to reason** about **what a software system is doing and why** by analyzing its external outputs.

Monitoring vs Observability

- **Monitoring:**
  - Tracks known metrics and raises alerts when thresholds are not met (e.g., 4 golden signals of Google SRE: latency, traffic, errors, and saturation)
  - Answers the question: “how is the system doing?”
  - Helps diagnose known problems

- **Observability:**
  - Answers the question: “what is the system doing and why?”
  - Enables to reason about the system by observing its outputs
  - Helps diagnose known and unknown problems

Building Blocks



Offline and/or real-time analytics

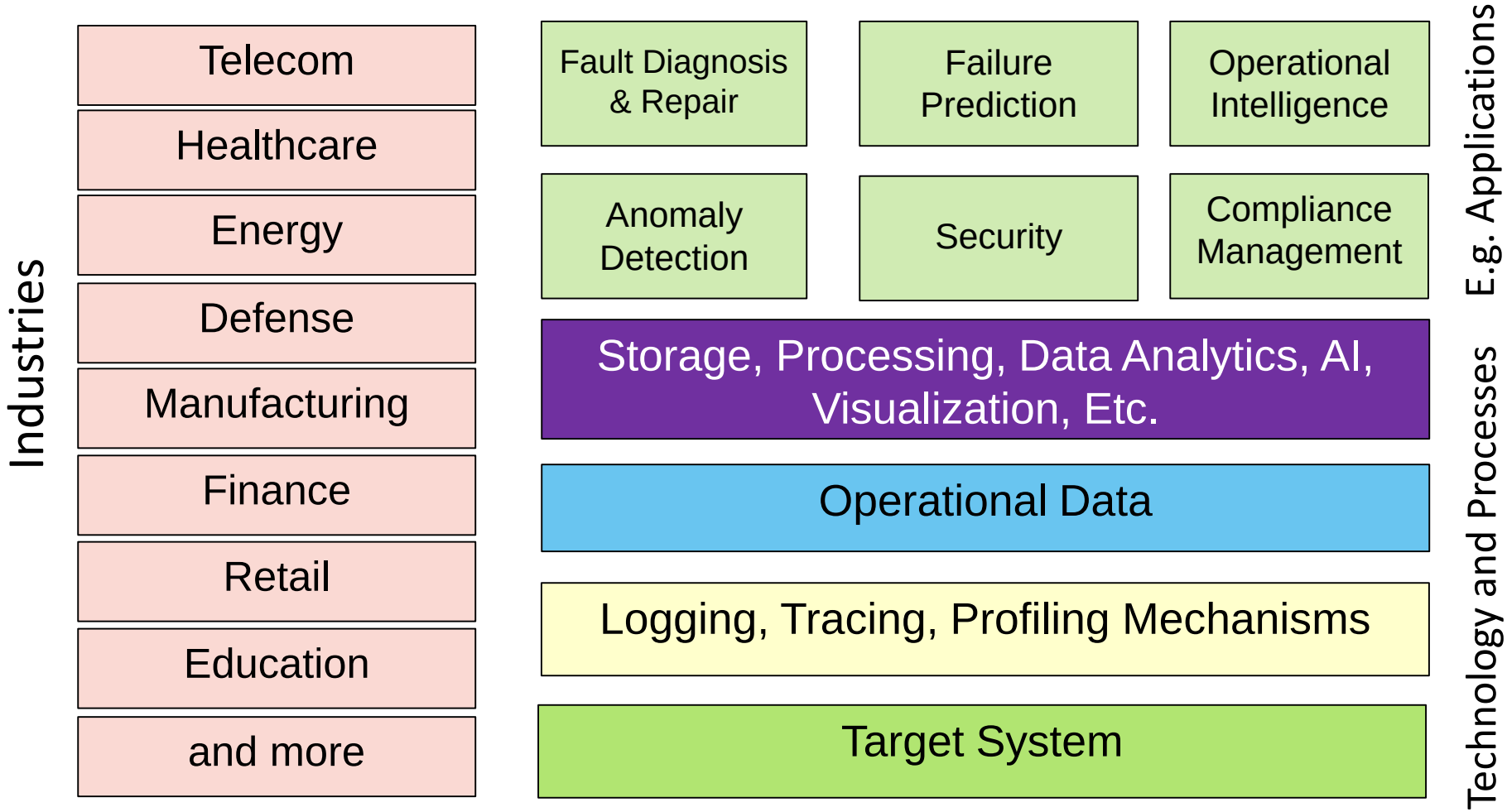
Operational Data

- **Logs:**
  - Records of events generated from logging statements inserted in the code to track system execution, errors, failures, etc.
  - Different types of logs: system logs, application logs, event logs, etc.
- **Traces:**
  - Records of events showing execution flow of a service or a (distributed) system with causal relationship
  - Require additional instrumentation mechanisms
- **Profiling Metrics:**
  - Aggregate measurements over a period of time (e.g., CPU usage, number of user requests, etc.)

```
2015-10-18 18:01:47,978 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Created MRAppMaster for application appattempt_1445144423722_0020_000001
2015-10-18 18:01:48,963 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Executing with tokens:
2015-10-18 18:01:48,963 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Kind: YARN_AM_RM_TOKEN, Service: , Ident: (appAttemptId { application_id { id: 20 clus
2015-10-18 18:01:49,228 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Using mapred newApiCommitter.
2015-10-18 18:01:50,353 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: OutputCommitter set in config null
2015-10-18 18:01:50,509 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2015-10-18 18:01:50,556 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.jobhistory.EventType for class org.apache.
2015-10-18 18:01:50,556 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.job.event.JobEventType for class or
2015-10-18 18:01:50,556 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.job.event.TaskEventType for class o
2015-10-18 18:01:50,556 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.job.event.TaskAttemptEventType for
2015-10-18 18:01:50,572 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.commit.CommitterEventType for class
2015-10-18 18:01:50,572 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.speculate.Speculator$EventType for
2015-10-18 18:01:50,572 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.rm.ContainerAllocator$EventType for
2015-10-18 18:01:50,588 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.launcher.ContainerLauncher$EventTyp
2015-10-18 18:01:50,634 INFO [main] org.apache.hadoop.mapreduce.v2.jobhistory.JobHistoryUtils: Default file system [hdfs://msra-sa-41:9000]
2015-10-18 18:01:50,666 INFO [main] org.apache.hadoop.mapreduce.v2.jobhistory.JobHistoryUtils: Default file system [hdfs://msra-sa-41:9000]
2015-10-18 18:01:50,713 INFO [main] org.apache.hadoop.mapreduce.v2.jobhistory.JobHistoryUtils: Default file system [hdfs://msra-sa-41:9000]
2015-10-18 18:01:50,728 INFO [main] org.apache.hadoop.mapreduce.jobhistory.JobHistoryEventHandler: Emitting job history data to the timeline server is not enabled
2015-10-18 18:01:50,806 INFO [main] org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class org.apache.hadoop.mapreduce.v2.app.job.event.JobFinishEvent$Type for c
2015-10-18 18:01:51,197 INFO [main] org.apache.hadoop.metrics2.impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2015-10-18 18:01:51,306 INFO [main] org.apache.hadoop.metrics2.impl.MetricsSystemImpl: Scheduled snapshot period at 10 second(s).
2015-10-18 18:01:51,306 INFO [main] org.apache.hadoop.metrics2.impl.MetricsSystemImpl: MRAppMaster metrics system started
2015-10-18 18:01:51,322 INFO [main] org.apache.hadoop.mapreduce.v2.app.job.impl.JobImpl: Adding job token for job_1445144423722_0020 to jobTokenSecretManager
2015-10-18 18:01:51,619 INFO [main] org.apache.hadoop.mapreduce.v2.app.job.impl.JobImpl: Not uberizing job_1445144423722_0020 because: not enabled; too many maps; too muc
2015-10-18 18:01:51,650 INFO [main] org.apache.hadoop.mapreduce.v2.app.job.impl.JobImpl: Input size for job job_1445144423722_0020 = 1256521728. Number of splits = 10
2015-10-18 18:01:51,650 INFO [main] org.apache.hadoop.mapreduce.v2.app.job.impl.JobImpl: Number of reduces for job job_1445144423722_0020 = 1
2015-10-18 18:01:51,650 INFO [main] org.apache.hadoop.mapreduce.v2.app.job.impl.JobImpl: job_1445144423722_0020Job Transitioned from NEW to INITED
2015-10-18 18:01:51,650 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: MRAppMaster launching normal, non-uberized, multi-container job job_1445144423722_0020
2015-10-18 18:01:51,713 INFO [main] org.apache.hadoop.ipc.CallQueueManager: Using callQueue class java.util.concurrent.LinkedBlockingQueue
2015-10-18 18:01:51,775 INFO [Socket Reader #1 for port 62260] org.apache.hadoop.ipc.Server: Starting Socket Reader #1 for port 62260
2015-10-18 18:01:51,791 INFO [main] org.apache.hadoop.yarn.factories.impl.pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.mapreduce.v2.api.MRClientProtocolPB
2015-10-18 18:01:51,791 INFO [main] org.apache.hadoop.mapreduce.v2.app.client.MRClientService: Instantiated MRClientService at MININT-FNANLI5.fareast.corp.microsoft.com/1
2015-10-18 18:01:51,806 INFO [IPC Server Responder] org.apache.hadoop.ipc.Server: IPC Server Responder: starting
2015-10-18 18:01:51,806 INFO [IPC Server listener on 62260] org.apache.hadoop.ipc.Server: IPC Server listener on 62260: starting
2015-10-18 18:01:51,885 INFO [main] org.mortbay.log: Logging to org.slf4j.impl.Log4jLoggerAdapter(org.mortbay.log) via org.mortbay.log.Slf4jLog
```

Source: https://github.com/logpai/loghub/blob/master/Hadoop/Hadoop_2k.log

Scope of Observability



Emergence of AI for IT Operations

- AIOps is the application of AI to enhance IT operations
- An important enabler for digital transformation
- Building Blocks:
  - Data collection and aggregation
  - Pattern recognition
  - Predictive analytics
  - Visualization
- Applications:
  - Fault detection and prediction
  - Root cause analysis
  - Security
  - Regulatory compliance
  - Operational intelligence

Beyond Software Systems

- Using machine data analytics to drive operational efficiency (a Splunk success story)
- Dubai airport uses machine data to increase airport capacity
- Machine data sources:
  - Flight schedules,
  - Wi-Fi network data
  - Metal detector data
  - Baggage system
  - Sensor data (doors, faucets, etc.)

Source: https://www.splunk.com/en_us/customers/success-stories/dubai-airports.html

Characteristics of Logs and Traces

- **Velocity:** the data (in some cases) must be processed in real time
- **Volume:** mountain ranges of historical data
- **Variety:** captured data can be structured or unstructured
- **Veracity:** captured data must be cleaned
- **Value:** not all captured data is useful

A Quick Look at Log Analytics Research

- A good list of recent studies is maintained on LOGPAI Github repository:
  - https://github.com/logpai/awesome-log-analysis
- **Focus:**
  - Anomaly detection, data leakage analysis, failure prediction, failure diagnosis, regulatory compliance (GDPR), log abstraction and parsing
- **Techniques:**
  - Deep learning, NLP, taint flow analysis, machine learning, statistical methods, latent error prediction, mining software repositories, etc.

Our Past and Current Projects

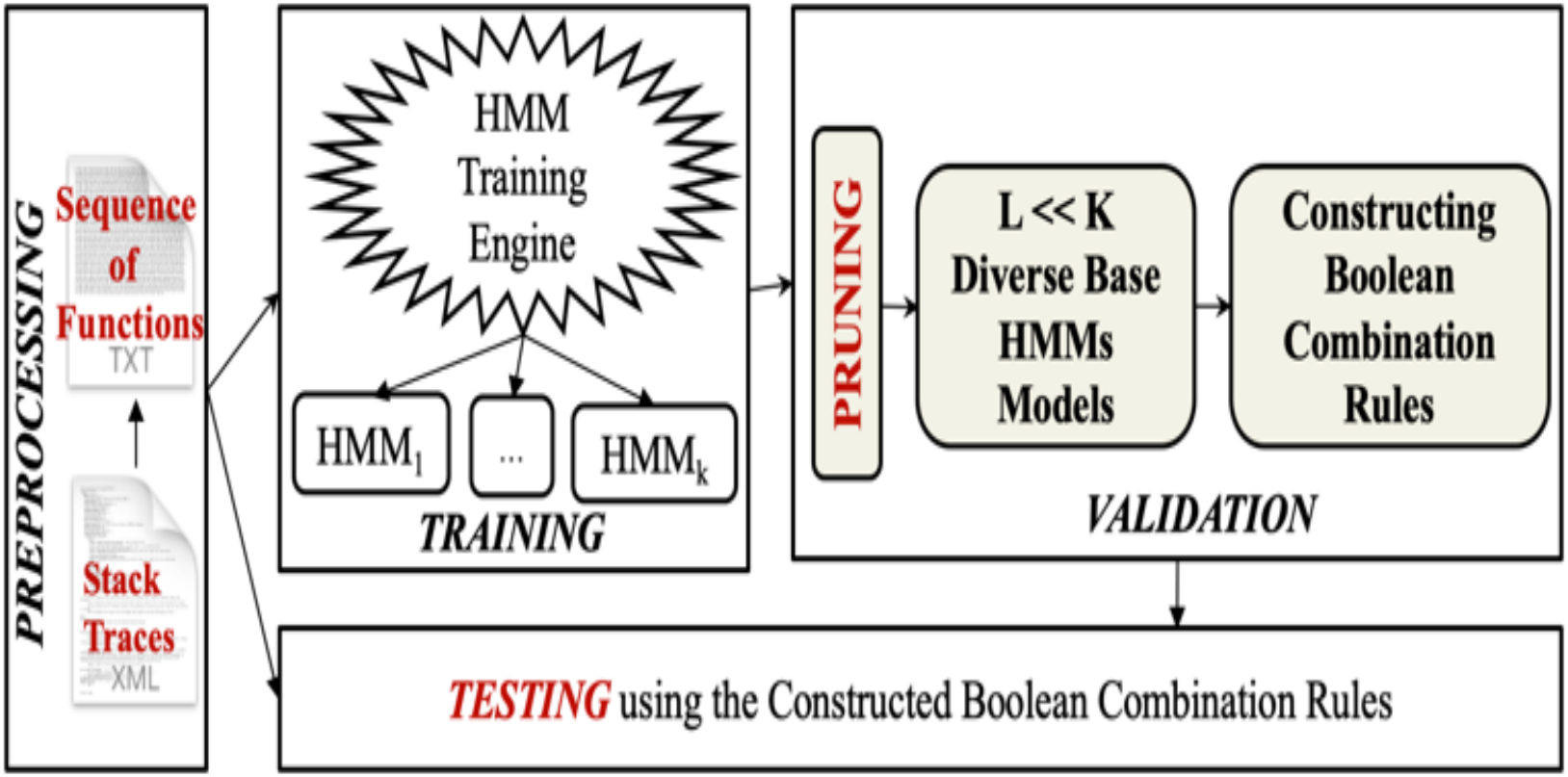
- Md Shariful Islam, "On the use of Software Tracing and Boolean Combination of Ensemble Classifiers to Support Software Reliability and Security Tasks," Ph.D. Dissertation, 2021.
- Korosh K. Sabor, "Automatic Bug Triaging Techniques Using Machine Learning and Stack Traces," Ph.D. Dissertation, 2020.
- Neda E. Koopaei, "Machine Learning and Deep Learning Based Approaches for Detecting Duplicate Bug Reports with Stack Traces," Ph.D. Dissertation, 2019.
- Fazilat Hojaji, "Techniques to Compact Model Execution Traces in Model Driven Approach," Ph.D. Dissertation, 2019.
- Heidar Pirzadeh, "Trace Abstraction Framework and Techniques," Ph.D. Dissertation, 2012.
- Luay Alawneh, "Techniques to Facilitate the Understanding of Inter-process Communication Traces," Ph.D. Dissertation, 2012.

http://www.ece.concordia.ca/~abdelw/publications.html

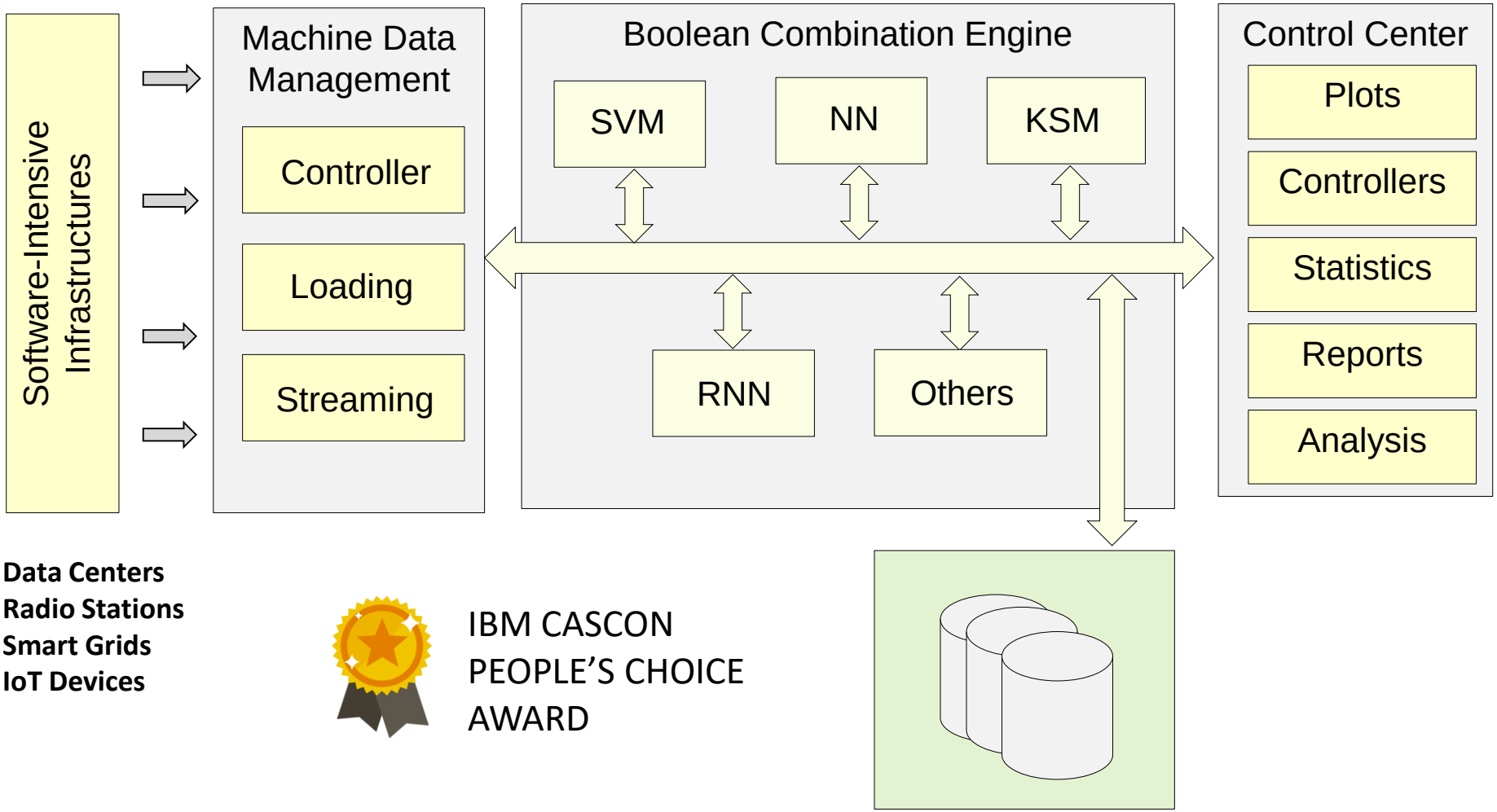
Software Tracing and Boolean Combination of Ensemble Classifiers to Support Software Reliability and Security Tasks

- PhD Thesis of Shariful Islam in collaboration with Postdoc Wael Khreich
- Contributions:
  - WPIBC: A weighted pruning ensemble of homogeneous classifiers (HMMs) applied to anomaly detection
  - EnHMM: Ensemble HMMs and stack traces to predict the reassignment of bug report fields
  - MASKED: A MapReduce solution for the Kappa-pruned ensemble-based anomaly detection system

WPIBC Using Ensemble HMMs Based on Boolean Combination



TotalADS: Total Anomaly Detection System Architecture



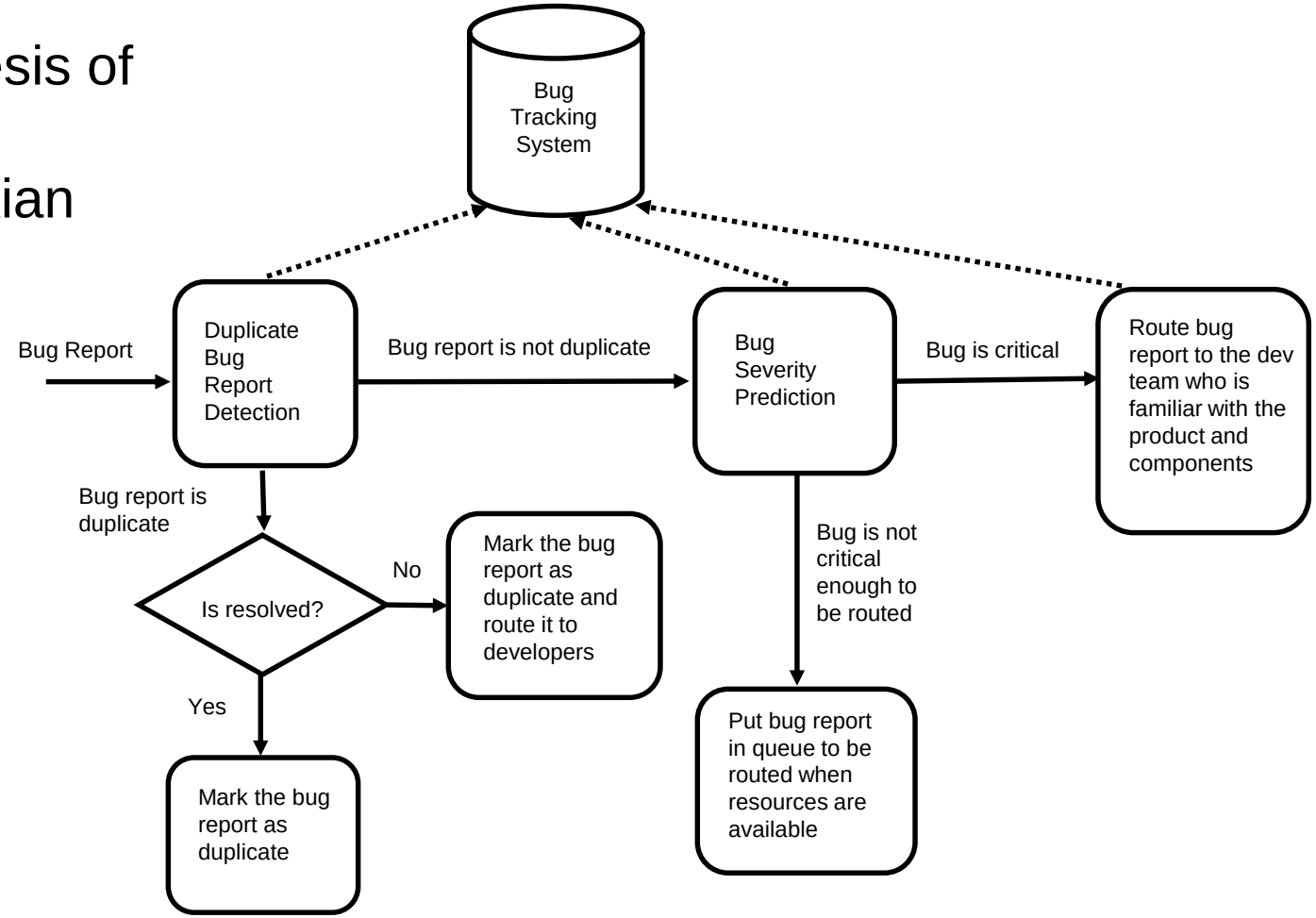
Data Centers
Radio Stations
Smart Grids
IoT Devices

IBM CASCON PEOPLE'S CHOICE AWARD

S. S. Murtaza, A. Hamou-Lhadj, W. Khreich, M. Couture, "TotalADS: Automated Software Anomaly Detection System," In Proc. of the 14th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM), 2014

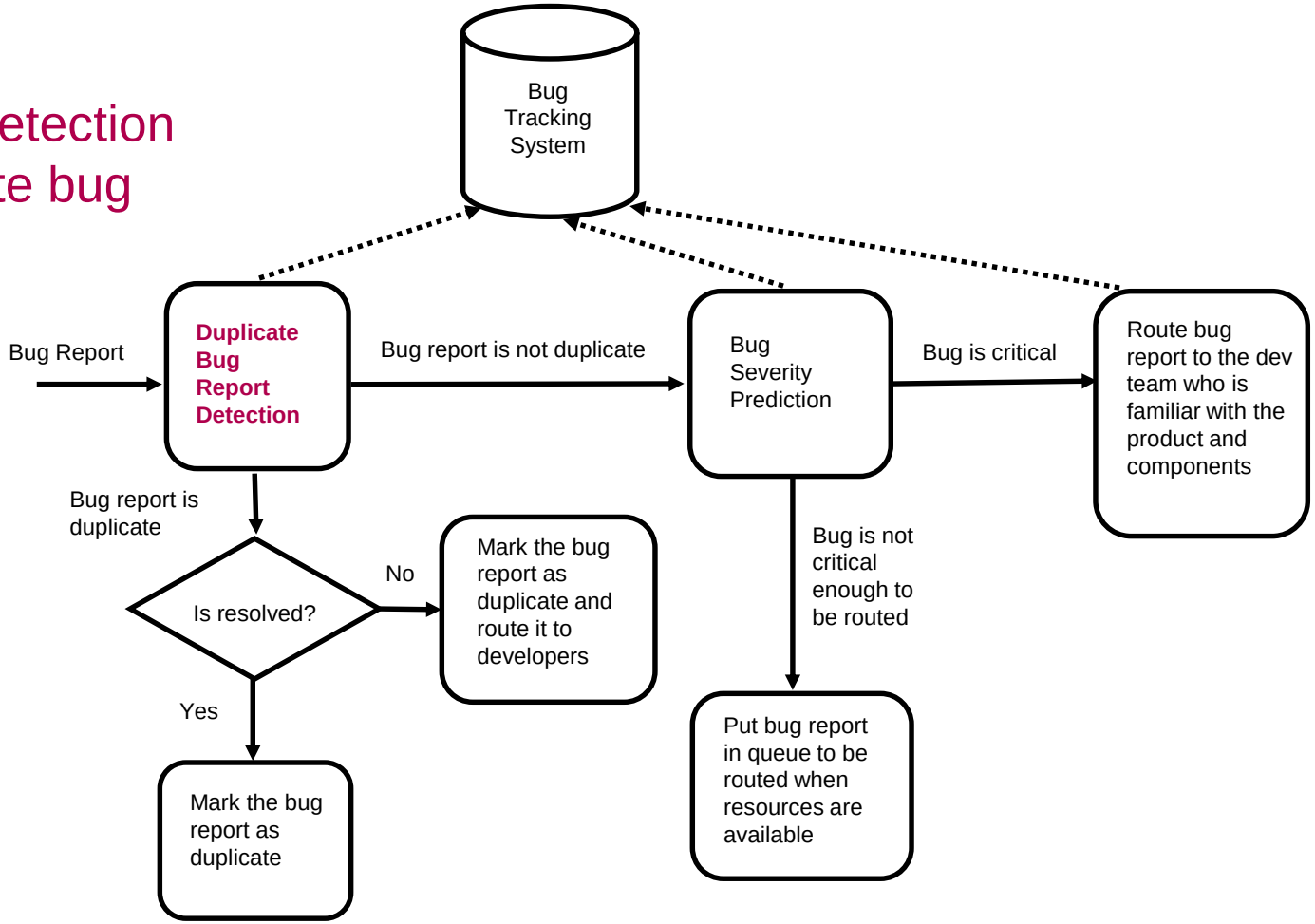
Automatic Crash/Bug Triaging Techniques Using Machine Learning and Stack Traces

- PhD Thesis of Korosh Koochekian Sabor



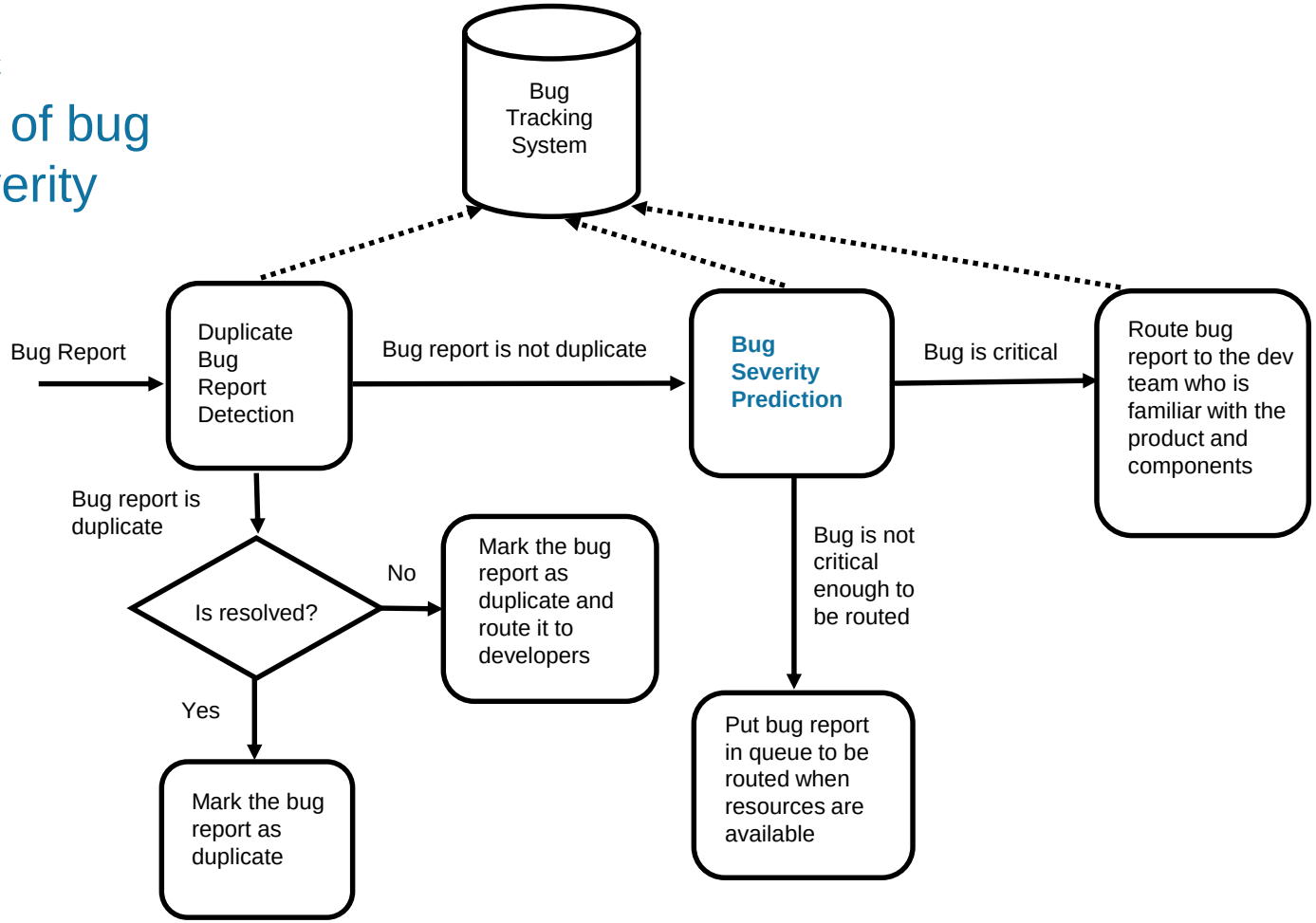
Automatic Crash Triaging Techniques Using Machine Learning and Stack Traces

- DURFEX: Efficient detection of duplicate bug reports



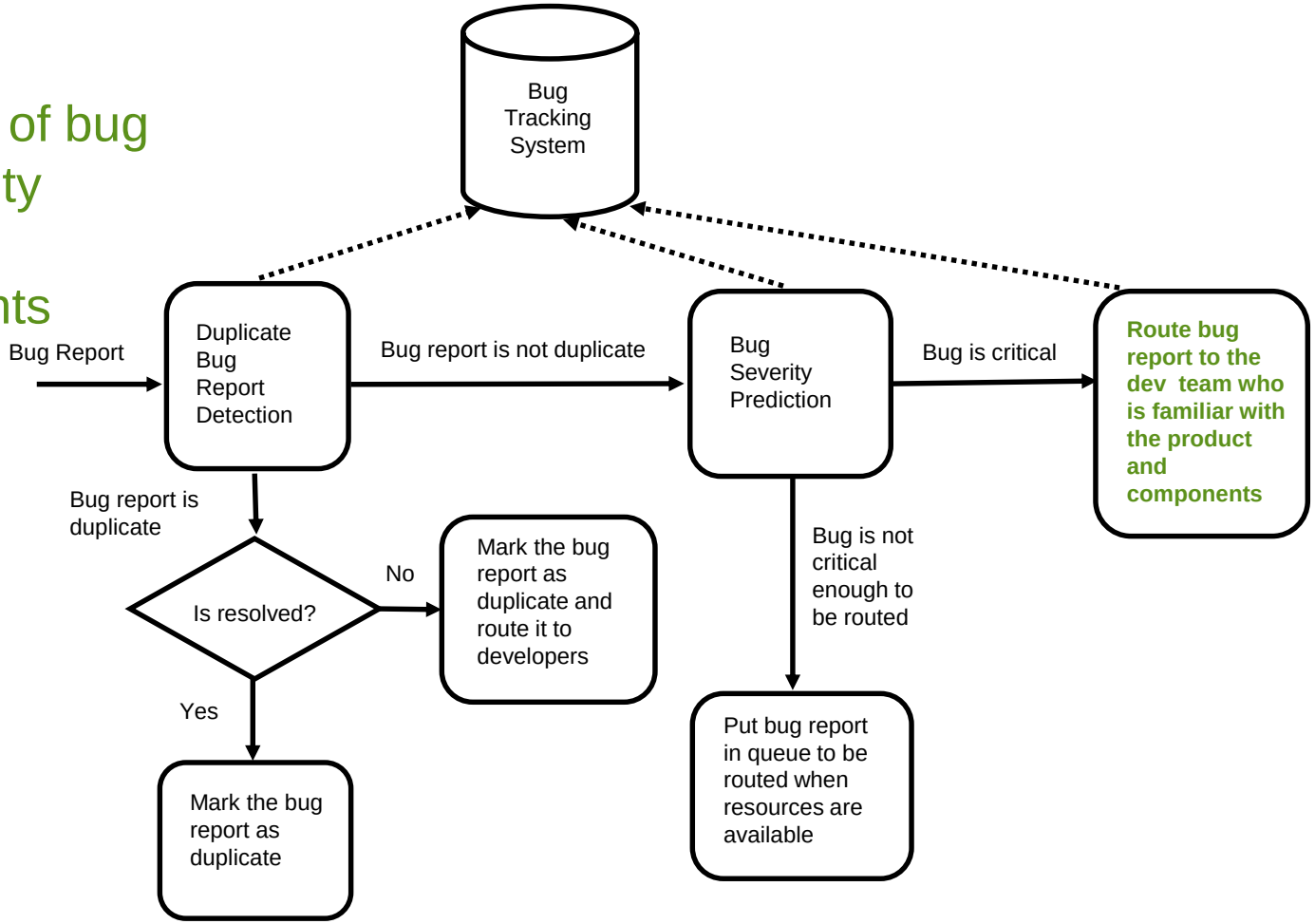
Automatic Crash Triaging Techniques Using Machine Learning and Stack Traces

- Automatic prediction of bug report severity



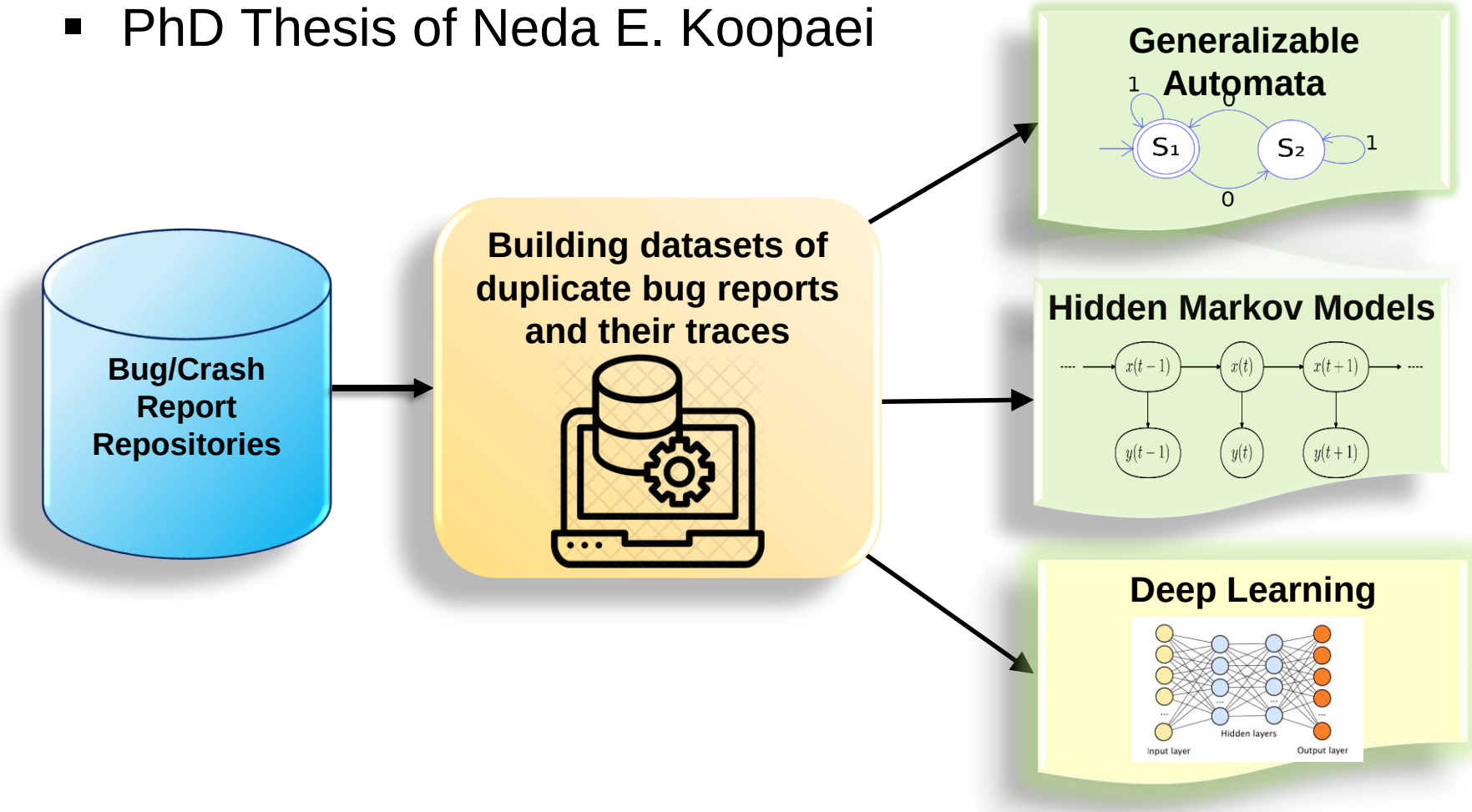
Automatic Crash Triaging Techniques Using Machine Learning and Stack Traces

- Automatic prediction of bug report faulty products components



Detection of Duplicate Bug Reports with Stack Traces and Sequential Learners

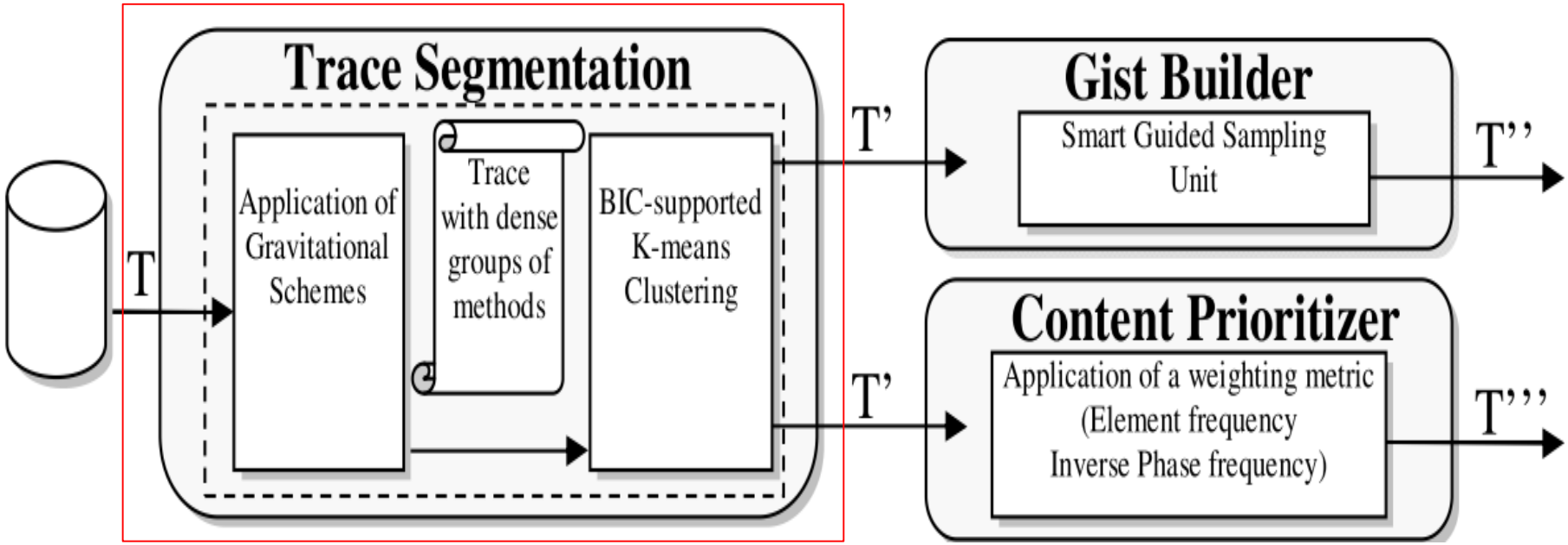
- PhD Thesis of Neda E. Koopaei



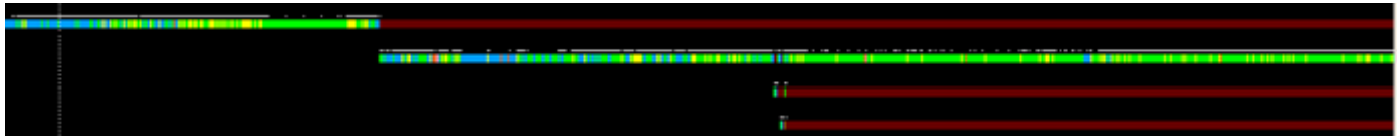
Trace Abstraction Framework and Techniques for Program Comprehension

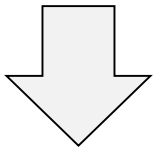
- PhD Thesis of Heidar (Amir) Pirzadeh
- Goal: To reduce the size of traces while keeping as much of their content as possible.
- Contributions:
  - Trace segmentation using clustering and Gestalt principles
  - Stratified sampling of execution traces
  - Identification of relevant events of a trace using trace segmentation and NLP
  - End-to-end trace abstraction framework

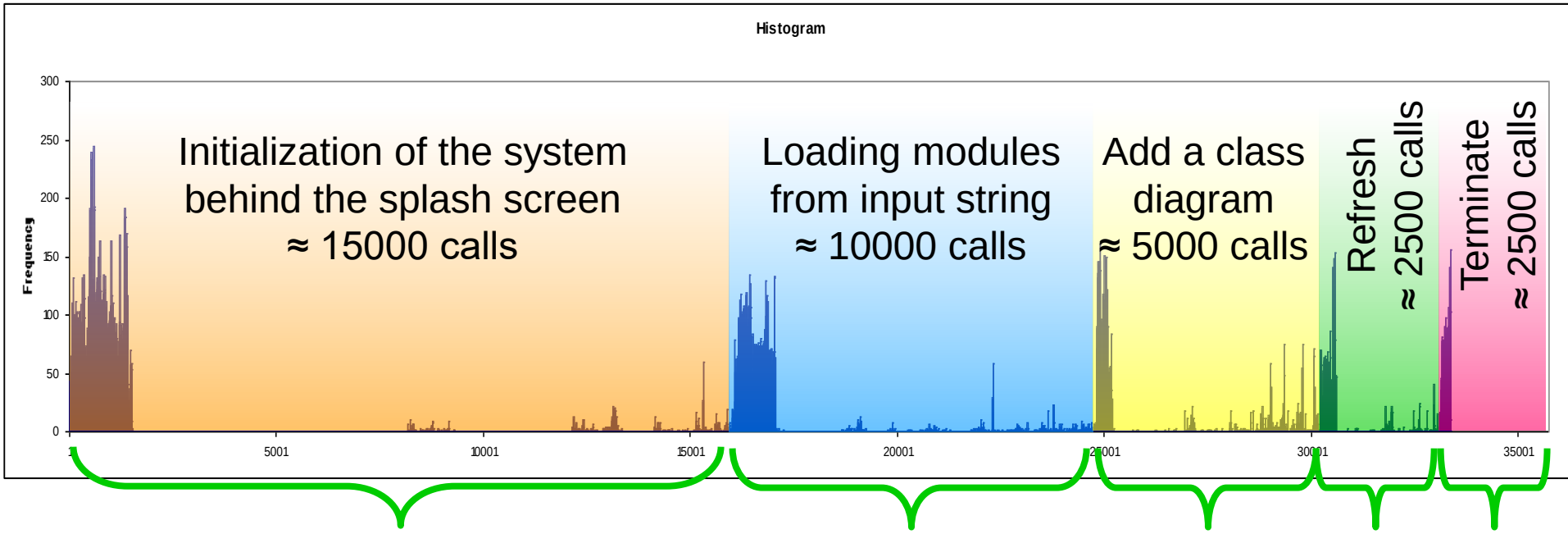
Trace Abstraction Framework and Techniques for Program Comprehension



Trace Abstraction Framework and Techniques for Program Comprehension







Techniques to Facilitate the Understanding of Inter-process Communication Traces

- PhD thesis of Luay Alawneh
- Contributions:
  - MTF: A scalable exchange format for MPI Traces
  - A techniques for extracting communication Patterns from large MPI traces
  - MPI Trace segmentation using execution phase detection

About the Practice of Logging

Operational-Log Analysis for Big Data Systems
Challenges and Solutions

Andriy Miranskyy, Ryerson University

Abdelwahab Hamou-Lhadj, Concordia University, Montreal

Enzo Cialini, IBM

Alf Larsson, Ericsson

Where Do Developers Log?
An Empirical Study on Logging Practices in Industry

Qiang Fu[1], Jieming Zhu[2], Wenlu Hu[3], Jian-Guang Lou[1], Rui Ding[1], Qingwei Lin[1], Dongmei Zhang[1], Tao Xie[4]

[1]Microsoft Research Asia, Beijing, China
{qifu,jlou,juding,qlin,dongmeiz}@microsoft.com

[2]The Chinese University of Hong Kong, HK, China
jmzhu@cuhk.edu.hk

[3]Carnegie Mellon University, PA, USA
wenlu@cmu.edu

[4]University of Illinois at Urbana-Champaign, IL, USA
taoxie@illinois.edu

Characterizing Logging Practices in Open-Source Softwa

Ding Yuan‡†, Soyeon Park†, and Yuanyuan Zhou†
†*University of California, San Diego,* ‡*University of Illinois at Urbana-Champai*
{*diyuan,soyeon,yyzhou*}*@cs.ucsd.edu*

Characterizing logging practices in Java-based open source software projects – a replication study in Apache Software Foundation

Boyuan Chen[1] · Zhen Ming (Jack) Jiang[1]

The Game of Twenty Questions: Do You Know Where to Log?

Xu Zhao, University of Toronto
Kirk Rodrigues, University of Toronto
Yu Luo, University of Toronto
Michael Stumm, University of Toronto
Ding Yuan, University of Toronto
Yuanyuan Zhou, University of California San Diego

Examining the Stability of Logging Statements

Suhas Kabinna[1], Weiyi Shang[2], Cor-Paul Bezemer[1],Ahmed E. Hassan[1]
Software Analysis and Intelligence Lab (SAIL),Queen's University,Kingston, Ontario [1]
Department of Computer Science and Software Engineering Concordia University,Montreal, QC, Canada[2],
Email:{kabinna, bezemer, ahmed}@cs.queensu.ca[1]
shang@encs.concordia.ca[2]

Key Findings

- Software logging **is pervasive** (e.g., around 1 logging statement in every 30 LOCs).
- The average **change rate** of logging code is almost **two times** compared to the entire code.
- Logging code is **modified very often** with one third of the modifications are **after-thoughts.**
- Developers often **have to adjust the verbosity level** of log messages.
- Developers do not seem **to be aware of the cost** of logging.

What about tracing?

- There are **no known guidelines** on when, how, and where to trace.
  - Tracing is like doing a detective's job!
- Tracing incurs **overhead** and requires external instrumentation tools.
  - It is often done after the fact depending on the problem.
- Continuous tracing is not possible because of the **huge amount** of data generated
  - Sampling is commonly used to reduce the size but causes other problems.

Challenges

- **Standards and Best Practices:**
  - Lack of guidelines and best practices for logging, tracing, and profiling
  - Lack of standards for representing logs, traces, and metrics (not the OpenTelemetry initiative)
- **Data Characteristics**
  - Mainly unstructured data
  - Size is a problem
  - Not all data is useful
  - High velocity

Challenges

- **Analytics and Tools:**
  - Mainly descriptive analytics
  - Predictive analytics not fully explored
  - Mainly offline analysis techniques
  - Lack of usable end-to-end observability tools

- **Cost and Management Aspects**
  - Cost vs. benefits not well understood
  - No clear alignment of observability with other initiatives
  - Roles and responsibilities are not well defined

Challenges

- **Analytics and Tools:**
  - Mainly descriptive analytics

There is a need for systematic and engineering approaches to software observability that promote best practices throughout the entire software development lifecycle

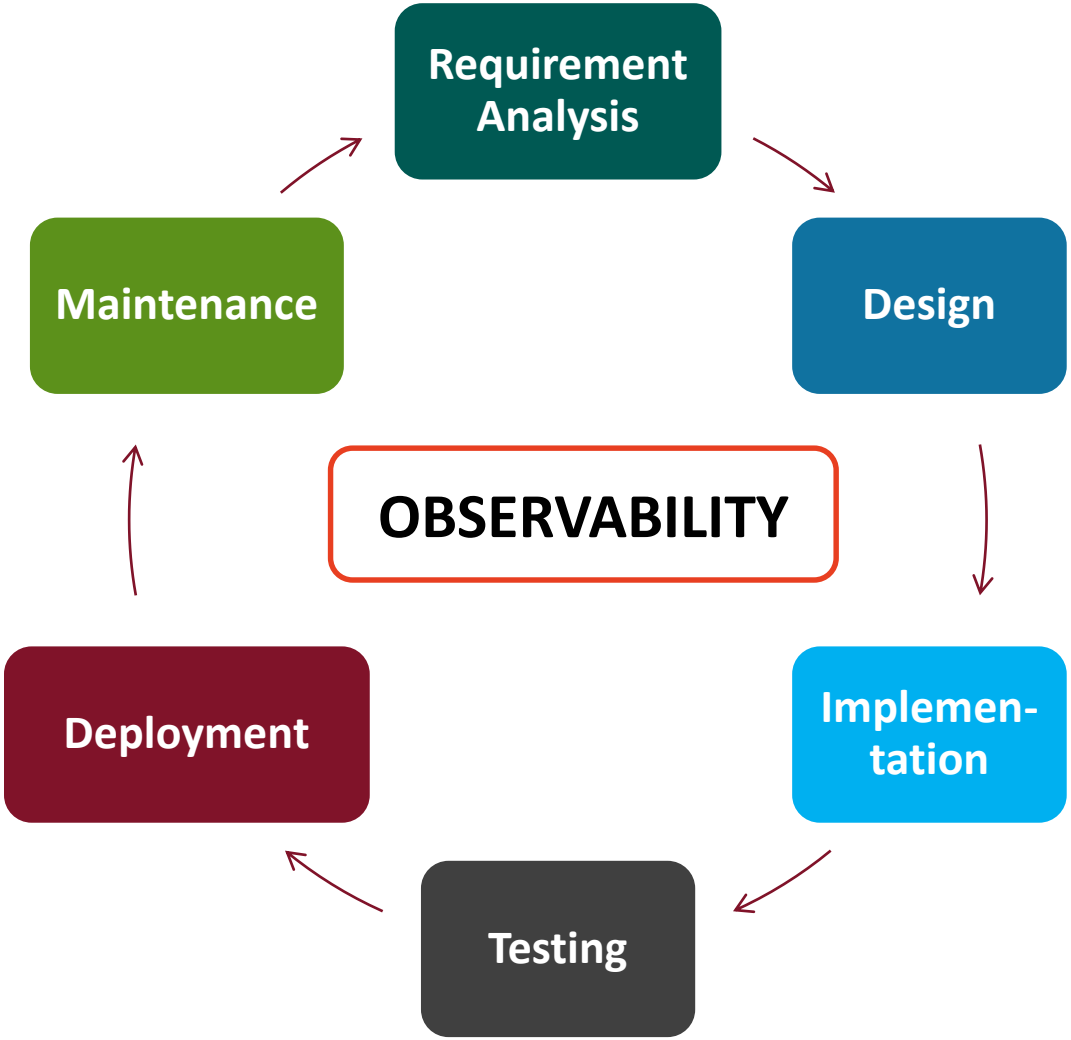
- **Cost and Management Aspects**
  - Cost vs. benefits not well understood
  - No clear alignment of observability with other initiatives
  - Roles and responsibilities are not well defined

Observability By Design

- Bringing observability **to early stages** of the software development lifecycle.
- Defining a set of **observability patterns, best practices, and reusable solutions** to be used as guiding principles for developers.
- A **systematic approach** to tracing, logging and profiling of software systems that considers different phases of the software process.

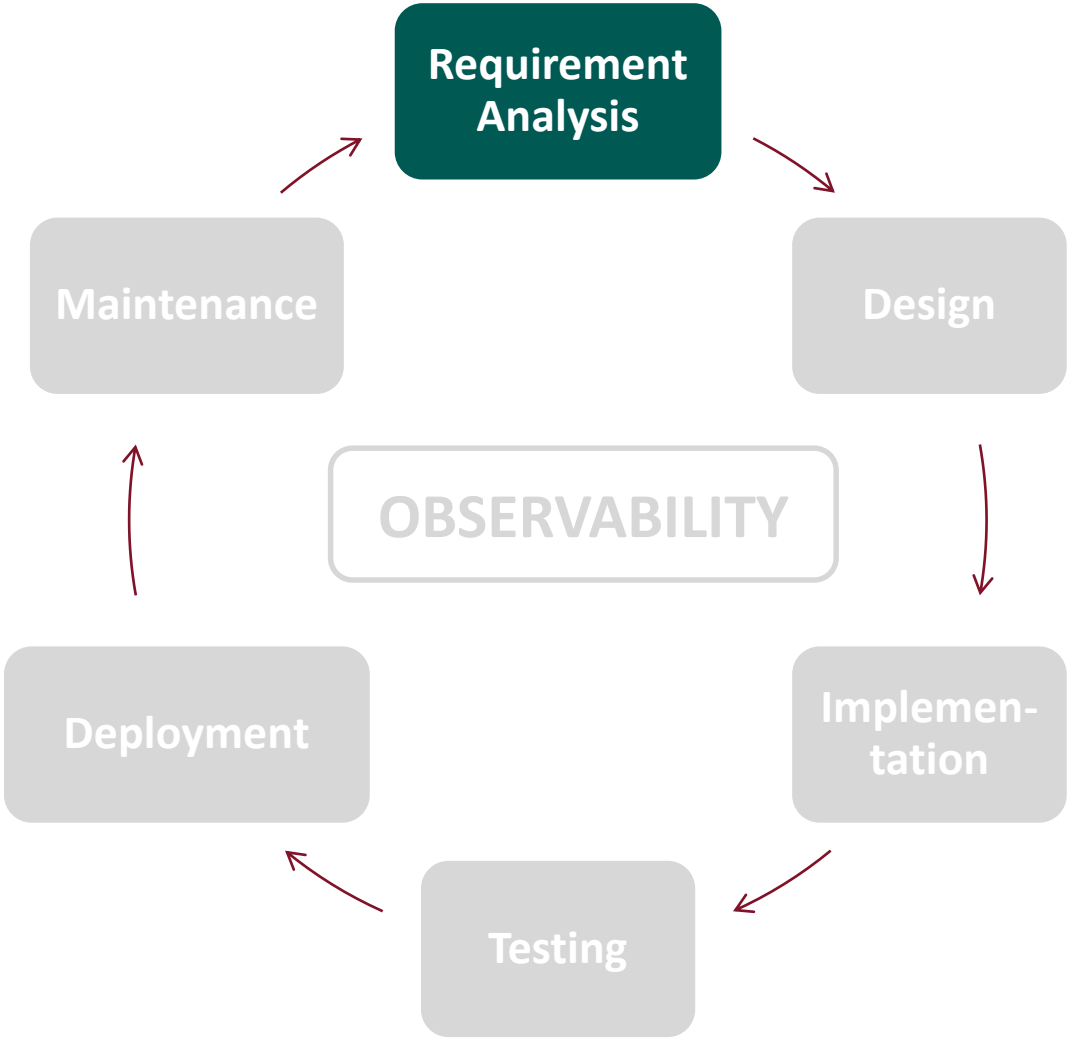
Observability By Design and SDLC

- Bringing observability **to early stages** of the software development lifecycle
- **Cost of observability** can be assessed during project planning



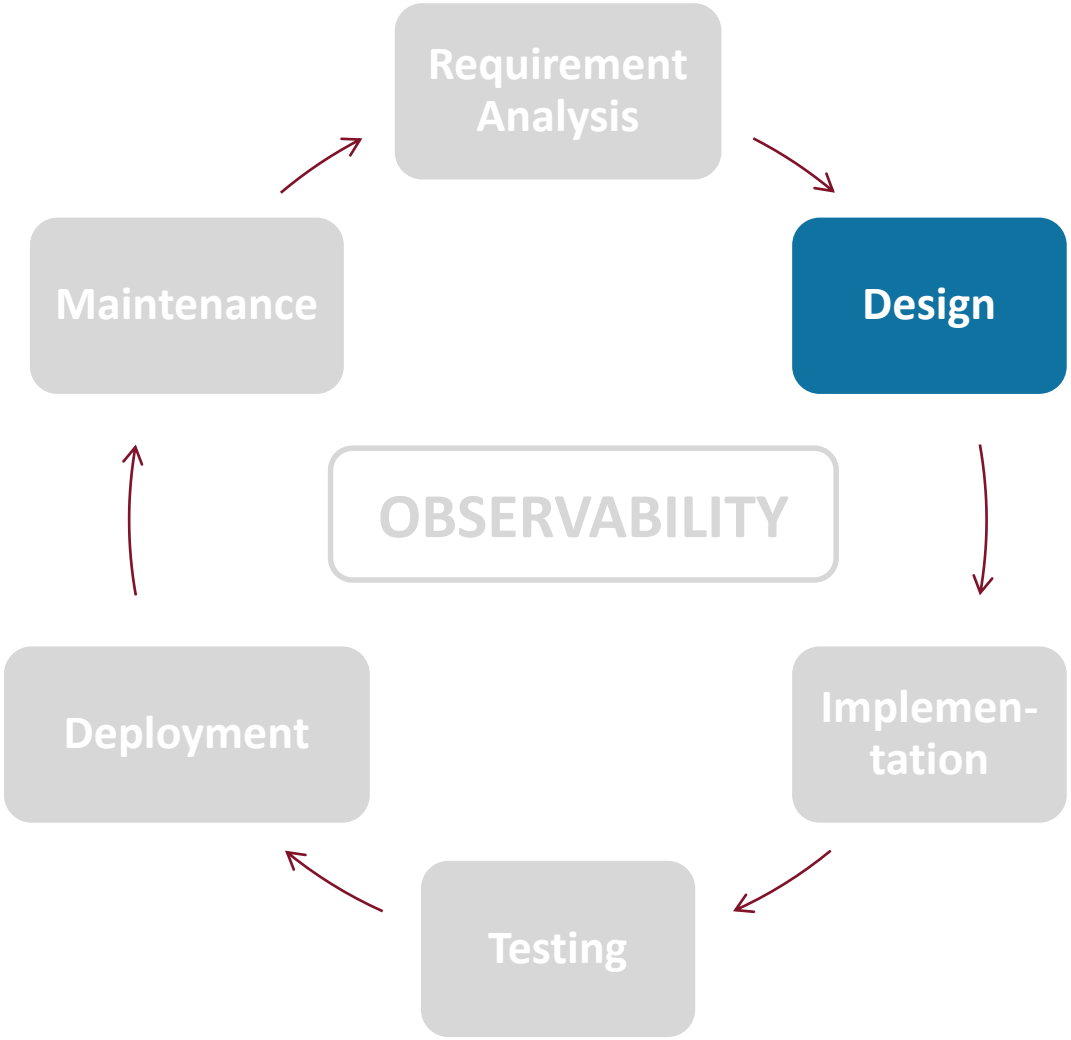
Observability By Design and SDLC

- Observability as a non-functional requirement
- What aspects of system functional requirements should be observable and how?



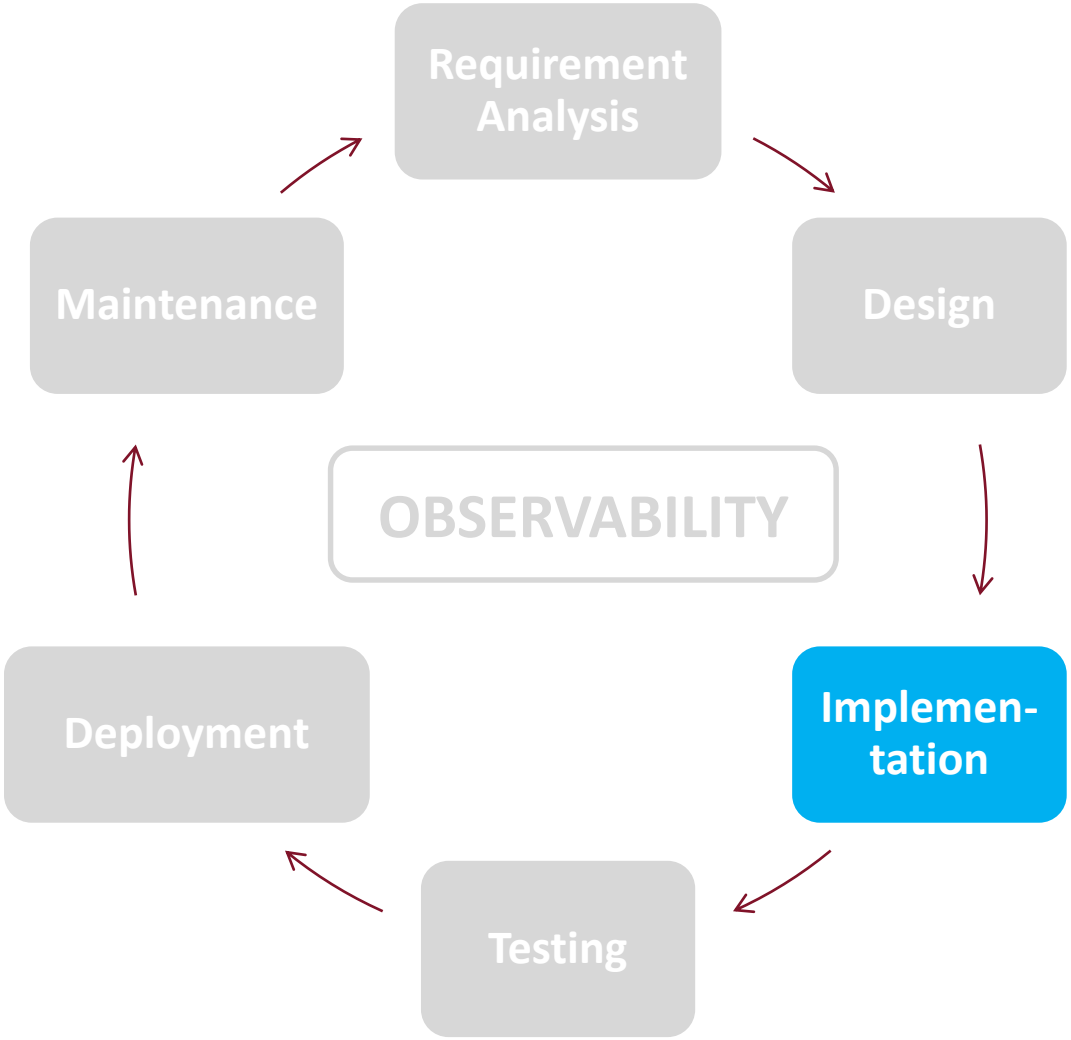
Observability By Design and SDLC

- Support of observability at the architectural level
- Detailed design for observability
- Observability patterns and best practices



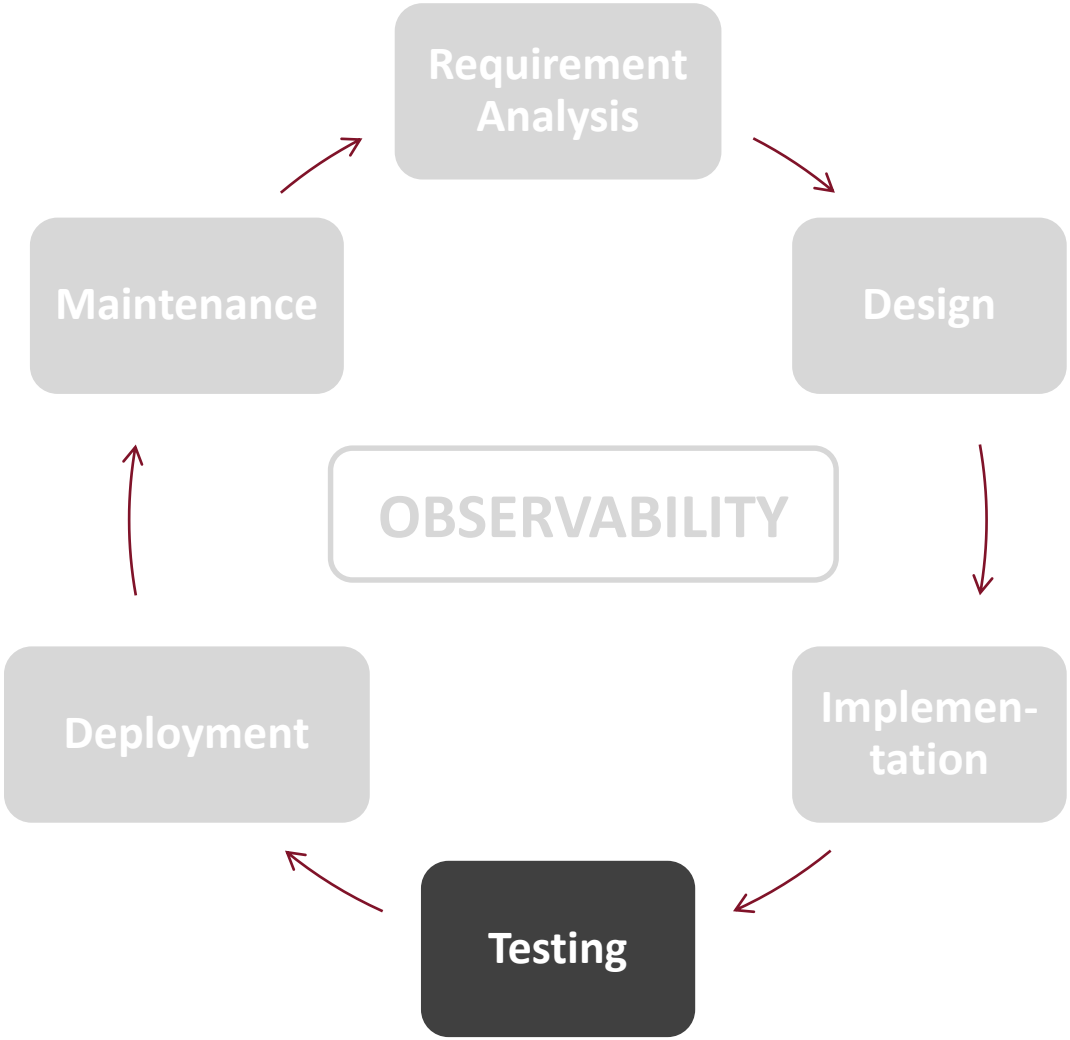
Observability By Design and SDLC

- What, where, and how to log and/or trace?
- Use of libraries and frameworks
- Patterns and best practices



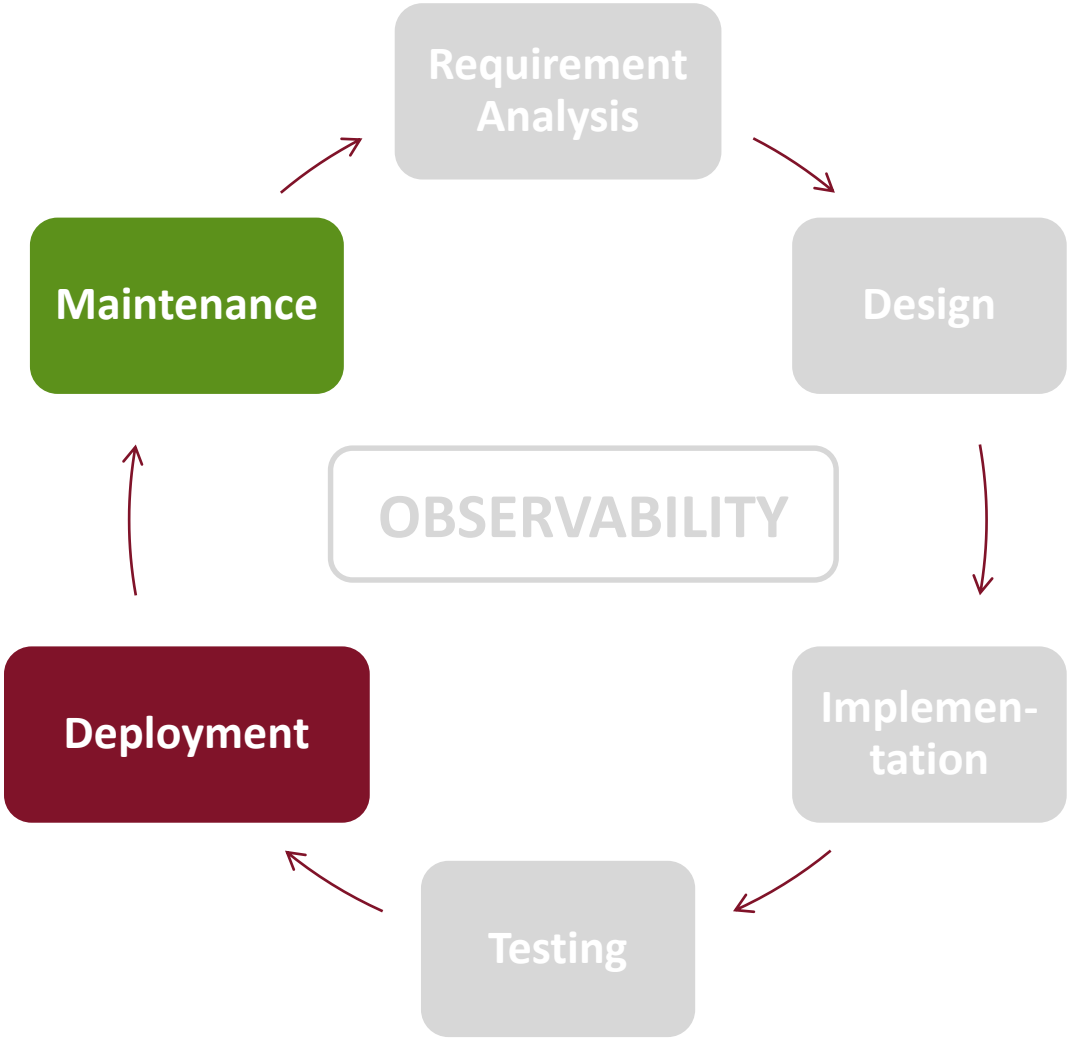
Observability By Design and SDLC

- Testing and inspection strategies for logging/tracing code



Observability By Design and SDLC

- Deployment, configuration, and maintenance aspects of observability code such as updates, performance analysis, testing, persistence, etc.



A Governance Framework for Observability By Design

Goals and objectives, Strategic alignment, KPIs,

Governance

People	Process	Technology
Training Roles & responsibilities (observability specialists)	Process maps Process compliance	AI Big data Tools & platforms

Continuous Improvement	Best Practices	Maturity Level Assessment

Conclusion

- Complex systems require sound mechanisms to ensure that they operate as intended and to detect/predict problems.
  - I presented SW system observability as one such mechanism.
  - Observability relies on processing and analyzing operational data

- The current practice is ad hoc and to take full advantage of operational data, we need to move towards systematic approaches for observability.
  - Observability By Design with its governing framework is one possible solution

Contact Information

Wahab Hamou-Lhadj, PhD, ing.

Professor

Dept. of Electrical and Computer Engineering

Gina Cody School of Engineering and Computer Science

Concordia University

wahab.hamou-lhadj@concordia.ca

http://www.ece.concordia.ca/~abdelw