IFIP WG 10.4 Winter Meeting

Software Dependability Modeling with A Data-Driven AI Paradigm

Michael R. Lyu

Department of Computer and Engineering

The Chinese University of Hong Kong

January 20, 2022

香港中文大學 The Chinese University of Hong Kong

ARISE Automated Reliable Intelligent Software Engineering

Background

- Modern software systems are serving many aspects of our life

Search Engine | Cloud Service | Office Software | Operating Systems | …

Most of these software systems are expected to be available on a 24 × 7 basis.

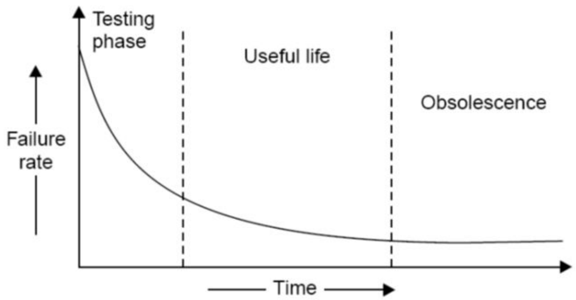
Software dependability modeling

- Dependability definition

The trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers to its users.

Software dependability modeling

- Dependability modeling



Analytical models

describe quantitatively

Dependability attributes

Software dependability modeling

- The life of a system is perceived by its users as:
  - Correct service
  - Incorrect service

Correct-incorrect service alternation quantification

enables

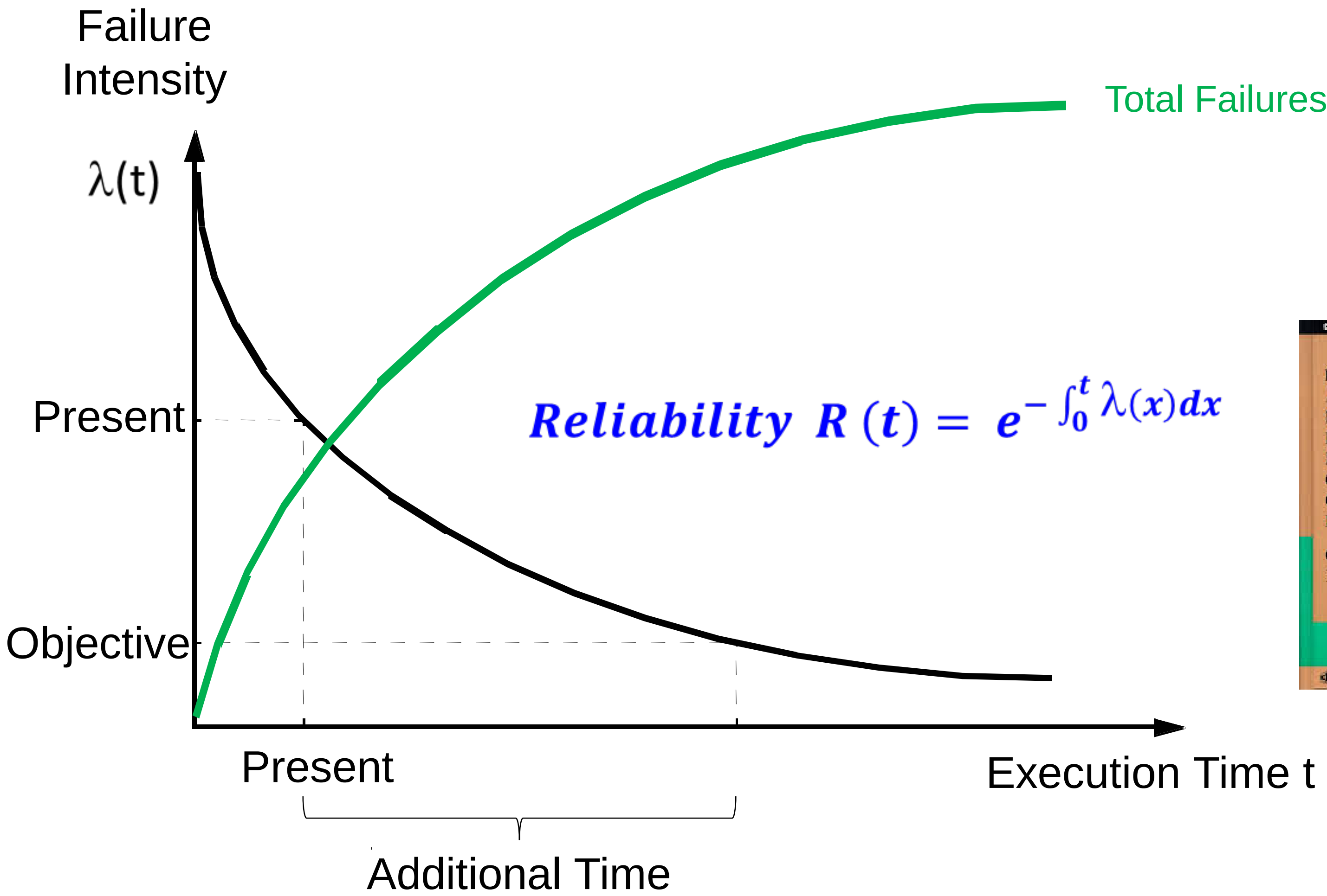
Reliability to be defined as a measure of dependability

- The basic approach
  - Model past failure data to predict future behavior

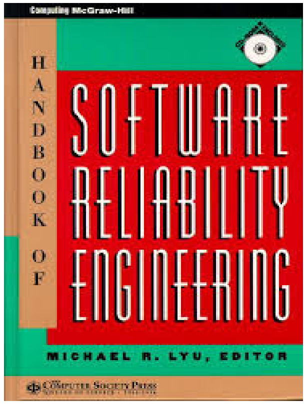
1 Failures per time period

2 Time between failures

Software Reliability Engineering



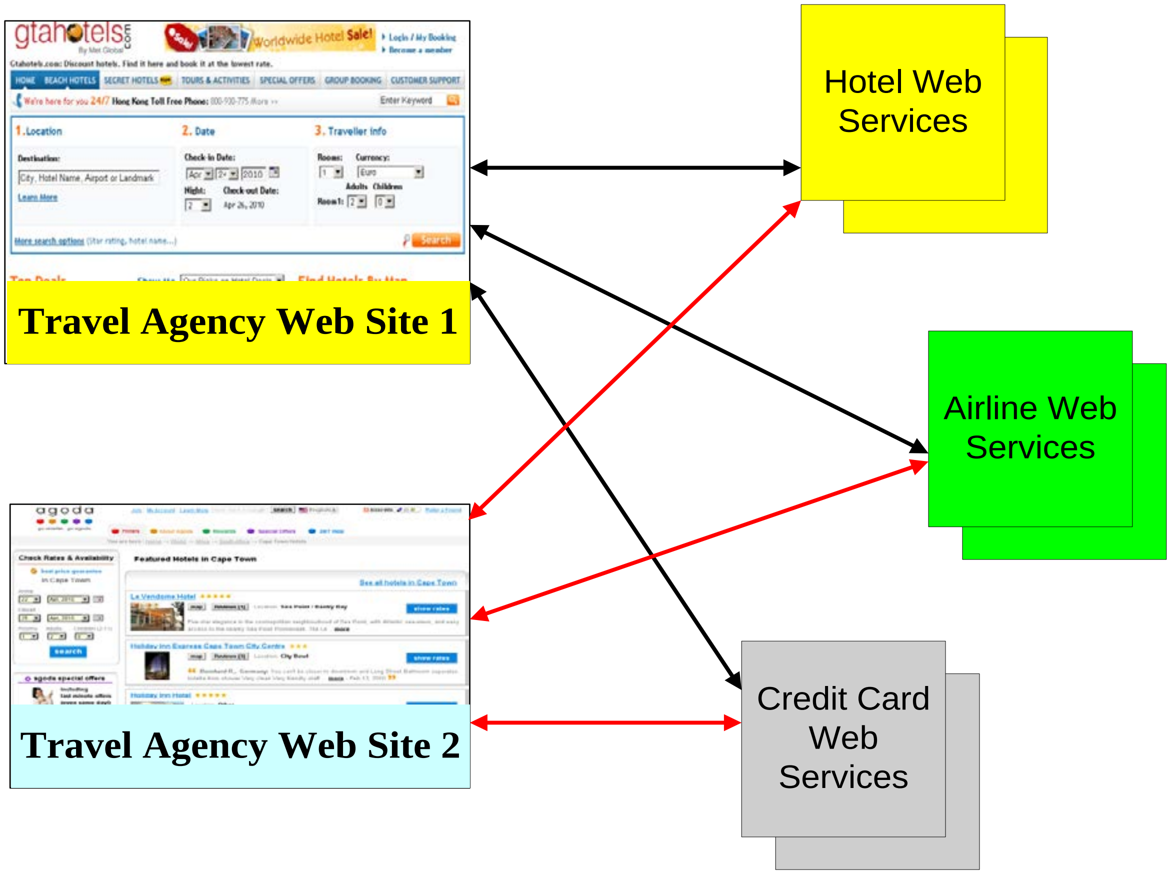
$$\textbf{Reliability}\ R(t) = e^{-\int_0^t \lambda(x)dx}$$



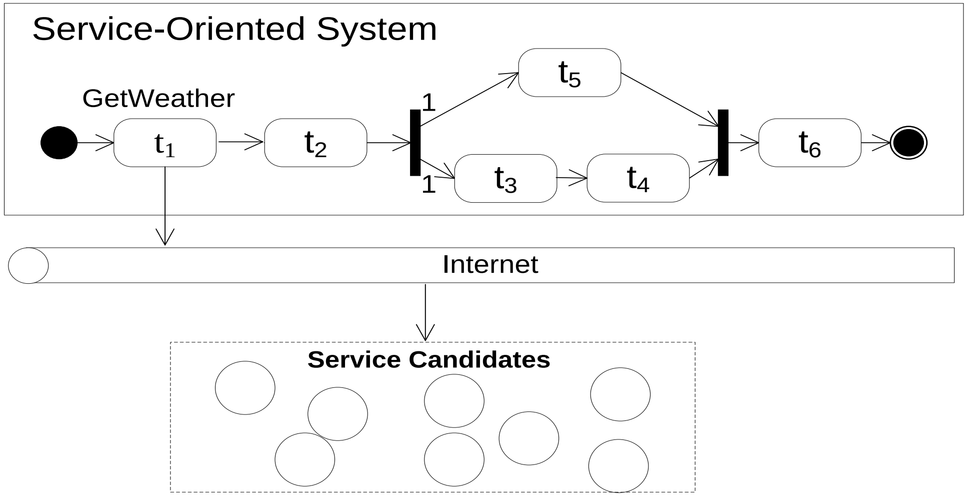
Fault Avoidance → Fault Removal → Fault Tolerance → Fault Prediction

Service-oriented systems

Composed by distributed Web services



Reliability prediction for service-oriented systems



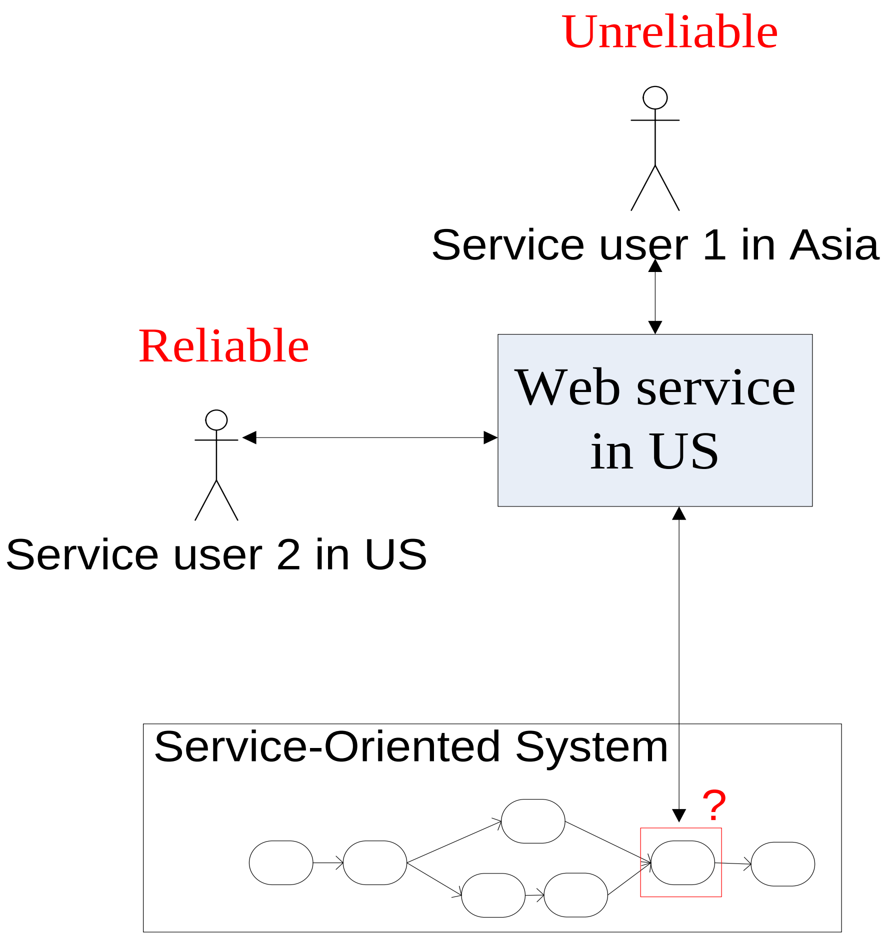
Target: determine the optimal Web service from a set of functionally equivalent candidates.

- It is difficult to model the reliability of service-oriented systems
  - Reliability of the system is highly dependent on the invoked Web services
  - Web services are provided by third-party organizations
  - The Internet environment is unpredictable

Reliability prediction of Web services

- Key idea: Using past usage data to find similar users and Web services to predict the reliability of a given service.

Reliability is extended to Quality-of-Service (QoS)



Zheng and Lyu, "Collaborative Reliability Prediction of Service-Oriented Systems".

[ICSE'10, ACM SIGSOFT Distinguished Paper Award]

Meaning of dependability modeling

An airplane typically has nine nines of reliability, i.e., 99.9999999%.

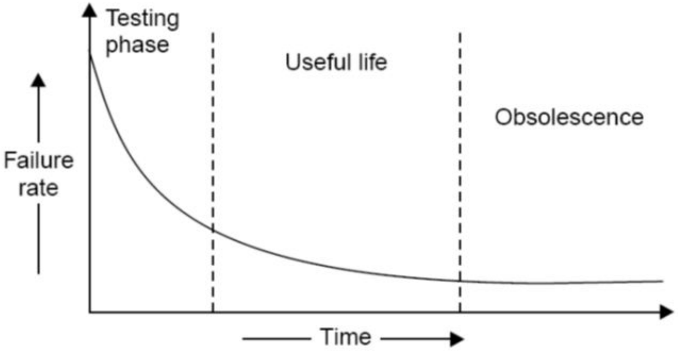
Only one crash every 3,000 weeks!

Self-driving cars have a better reliability figure than human drivers. Yet, we still cannot fully use them.

A single reliability figure is losing its practical significance…

Dependability modeling: new challenges

pose

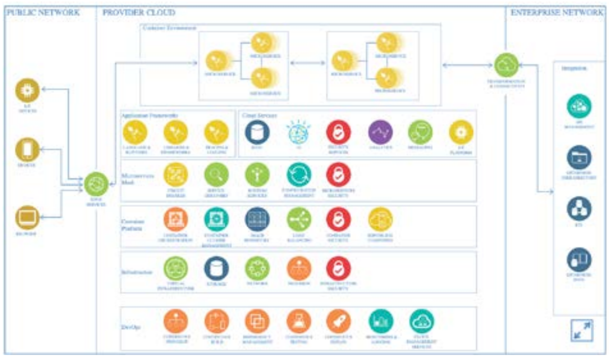


Modern software systems

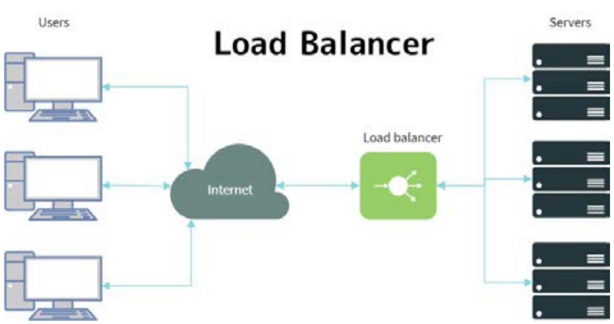
Traditional dependability modeling

Dependability modeling: new challenges

- Complex modern software system architecture



- Microservices architectures in clouds
- Complicated service dependencies



- Load balancing
- Fault tolerance
- Self-healing ability

Dependability modeling: new challenges

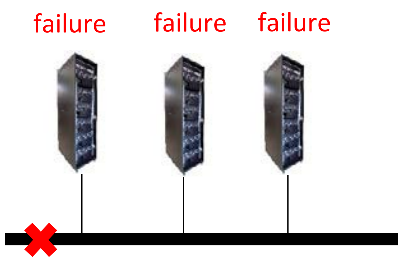
- More subtle software failures
  - Gray failures

> Different from fail-stop failures, the manifestations of gray failures are fairly subtle and thus defy fast and definitive detection.

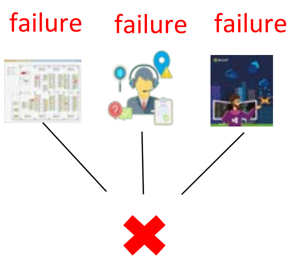
  - Transient failures

> Transient failures disappear quickly and thus are hard to detect, e.g., temporary timeout or unavailability of a service.

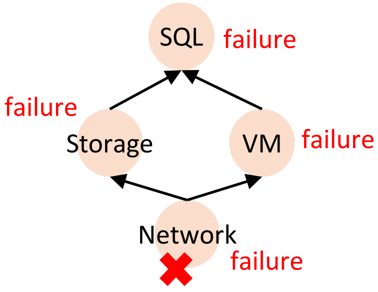
  - Failure cascading effects



Multi-location



Multi-source

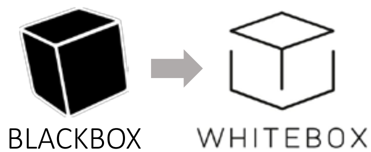


Multi-layer

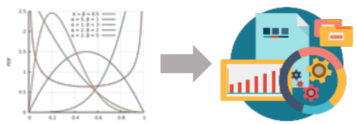
Dependability modeling evolution

Data-driven Software Dependability Modeling

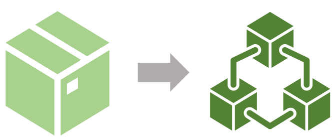
Data-driven software dependability modeling



Black-box to white-box



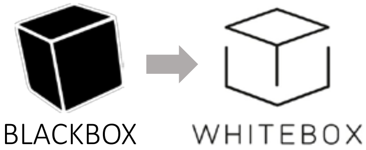
Model-centric to data-centric



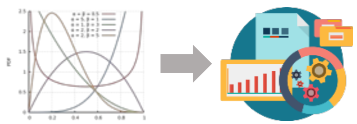
Macro-level to micro-level

Static analysis to dynamic analysis

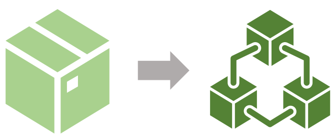
Data-driven software dependability modeling



Black-box to white-box



Model-centric to data-centric

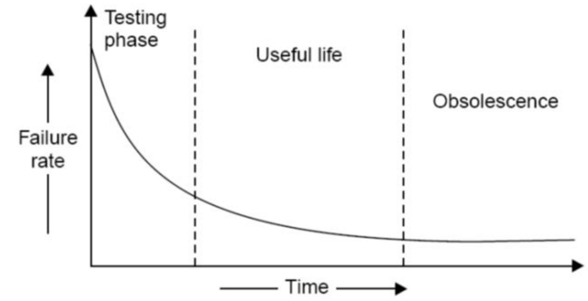


Macro-level to micro-level

Static analysis to dynamic analysis

Black-box dependability modeling

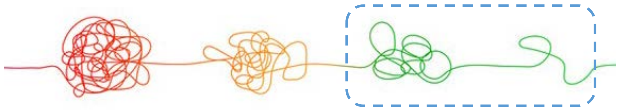
Dependability modeling



- Traditional software systems (standalone, shrink-wrapped)

Homogeneity

- Easy data collection
- Single point of failure



Low complexity

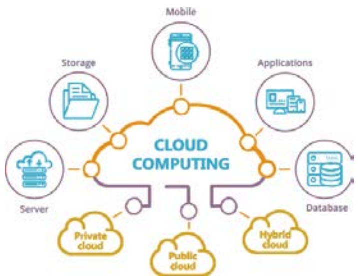
- Simple functionalities
- Easy failure patterns

Stable failure patterns

- Infrequent software updates
- The learned models remain valid

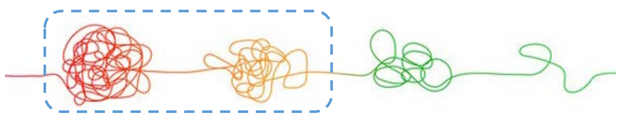
White-box dependability modeling

- Modern software systems (distributed systems, clouds)



Heterogeneous scale

- Hard to collect comprehensive failure data
- Non single point of failure

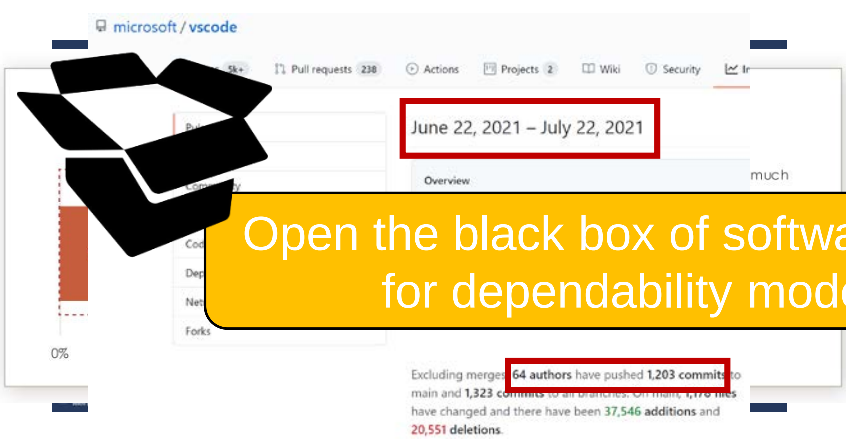


High complexity

- Component dependencies
- Complex failure patterns

Unstable failure patterns

- Frequent software updates
- Concept drifts



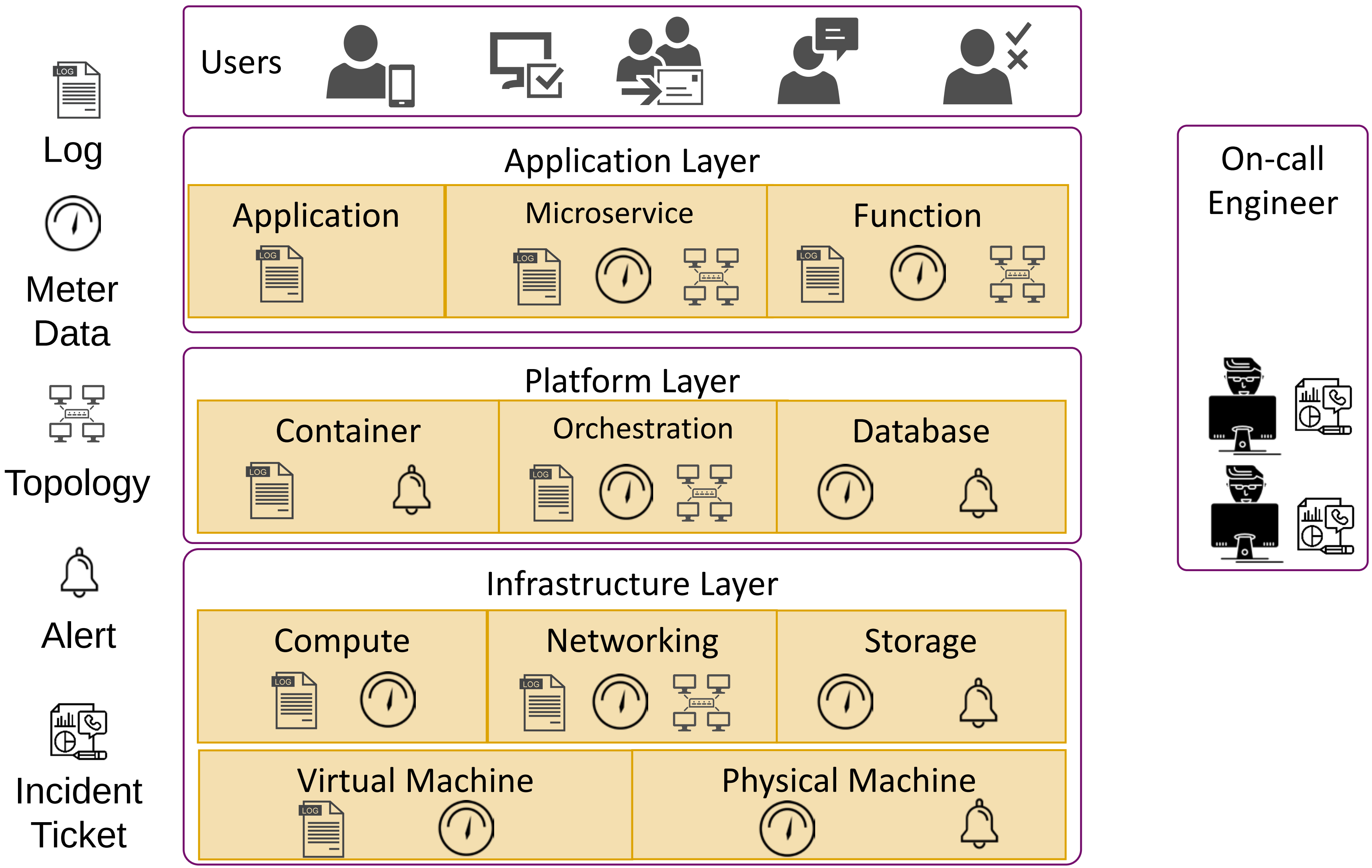
Vscode, in one month, 64 authors pushed ,546 line of code is inserted code is removed.

Open the black box of software systems for dependability modeling!

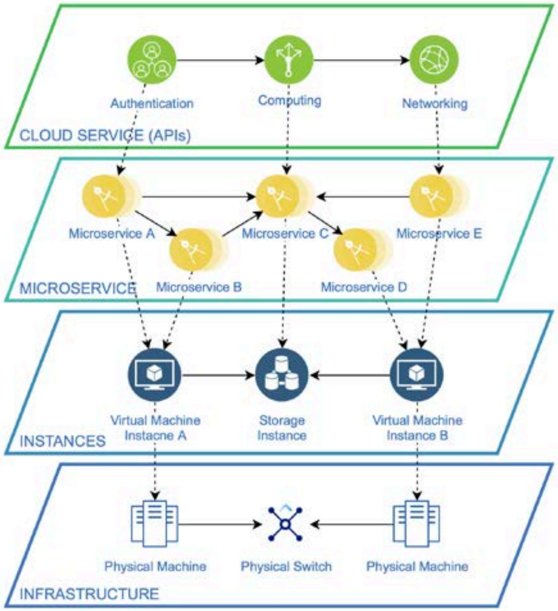
https://arstechnica.com/gadgets/2020/10/sourcegraph-devs-are-making-100x-more-code-now-than-they-did-in-2010/

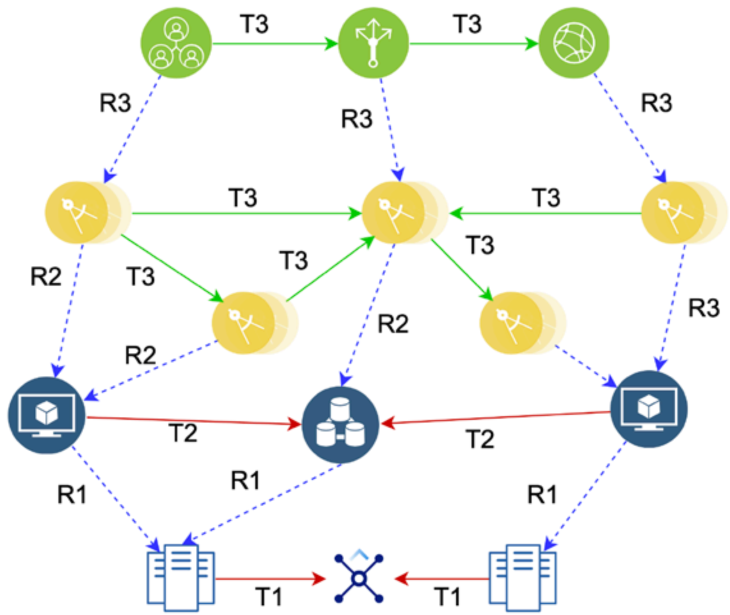
https://azure.microsoft.com/en-us/global-infrastructure/global-network/

Cloud system architecture



Knowledge graph construction for cloud system





The multi-layer architecture of a cloud system:

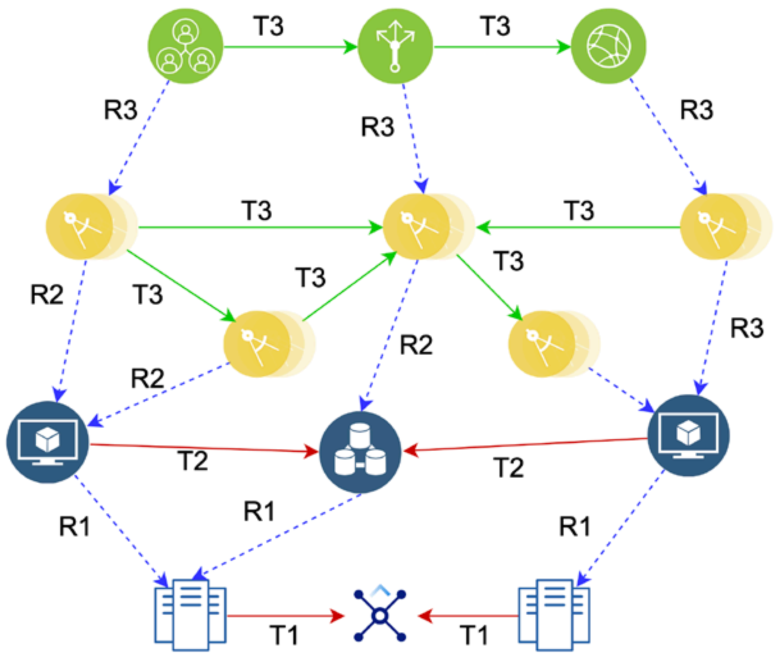
- Infrastructure layer
- Instance layer
- Microservice layer
- Cloud service layer

An excerpt of cloud knowledge graph:

- Physical machine connections
- Network communications
- Placement relationships
- Microservices dependencies

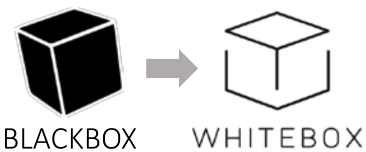
Knowledge graph for cloud system failure detection

- **Target**
  - Prompt failure detection for cloud systems
- **Method**
  - Heterogeneous graph representation learning
  - Detect abnormal state change for nodes based on their feature vectors

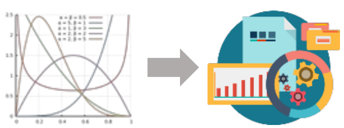


Key direction: to examine the system internal structure and capture the details of the system failure mechanism with a KG.

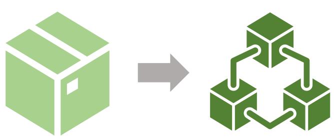
Data-driven software dependability modeling



Black-box to white-box



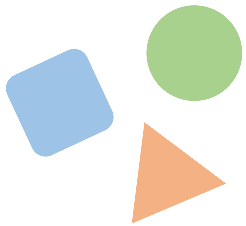
Model-centric to data-centric



Macro-level to micro-level

Static analysis to dynamic analysis

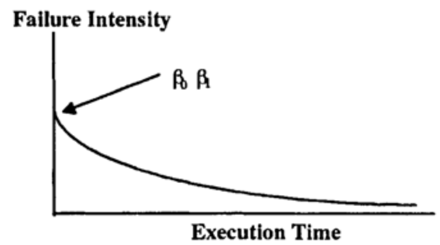
Model-centric dependability modeling



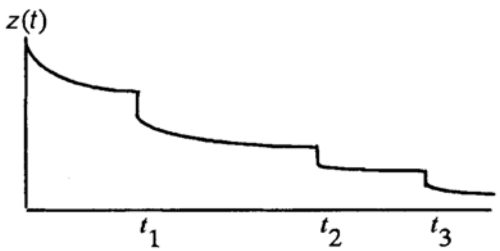
Distributional model selection

Analytical models assuming statistical distributions

- Exponential
- Weibull
- …



Failure intensity function for Musa’s basic execution model



Program hazard rate function for Weibull

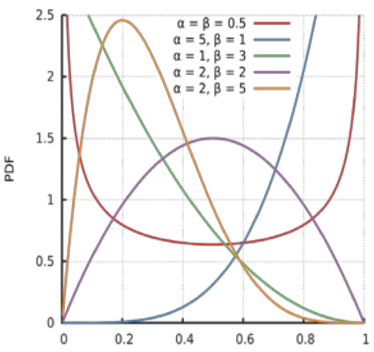
Assumptions

Assumptions behind model-centric dependability modeling

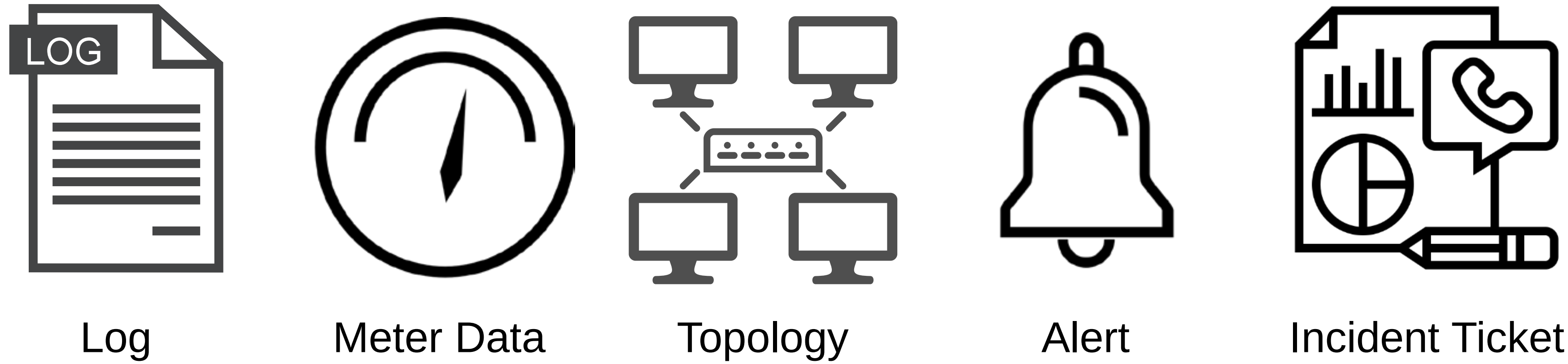
- Simple and predictable software behaviors
- Failure data follow specific distributions
- No extensive software changes

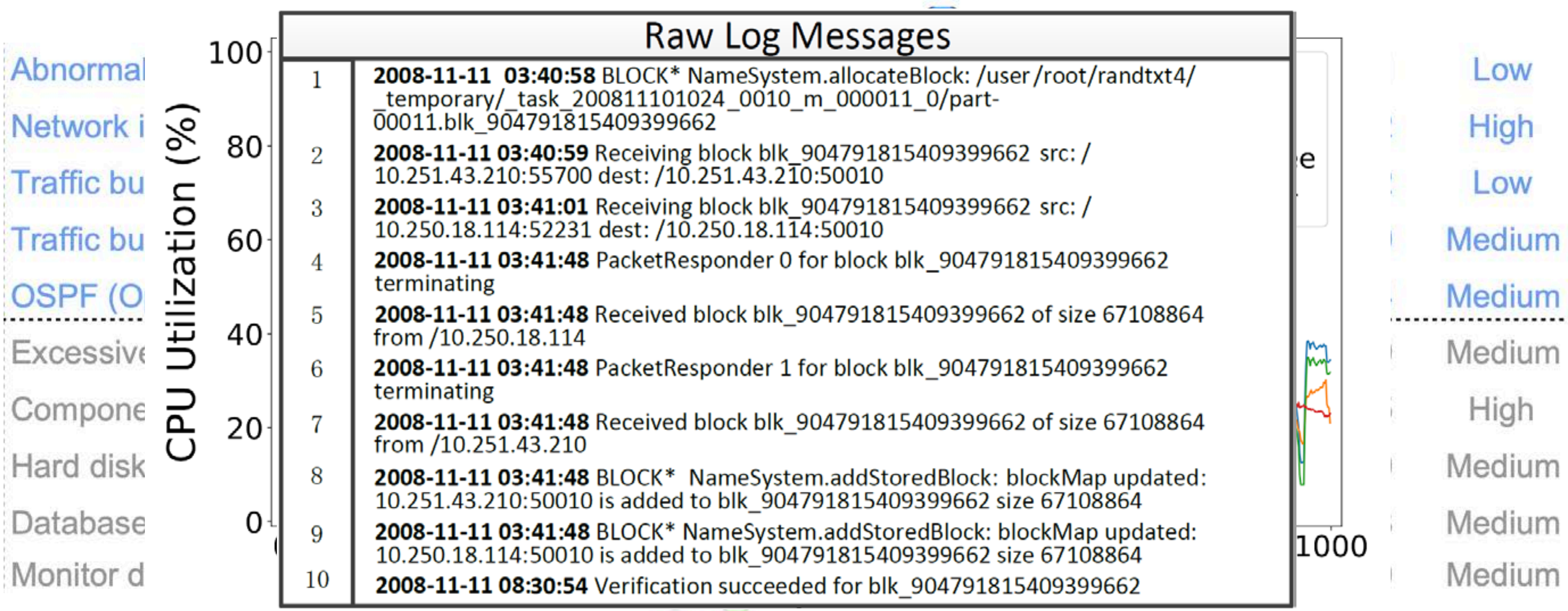
Data-centric dependability model

- Incorrect assumptions on data distribution
- Failure data alone is insufficient
  - Failures per time period/time between failures too high level
  - Other monitoring data characterize the system better



Data-centric dependability model





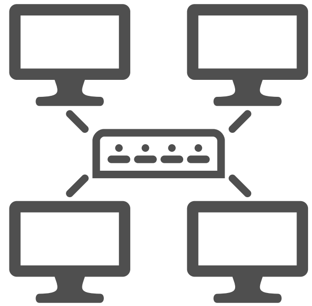
Raw Log Messages

1	**2008-11-11 03:40:58** BLOCK* NameSystem.allocateBlock: /user/root/randtxt4/_temporary/_task_200811101024_0010_m_000011_0/part-00011.blk_904791815409399662
2	**2008-11-11 03:40:59** Receiving block blk_904791815409399662 src: /10.251.43.210:55700 dest: /10.251.43.210:50010
3	**2008-11-11 03:41:01** Receiving block blk_904791815409399662 src: /10.250.18.114:52231 dest: /10.250.18.114:50010
4	**2008-11-11 03:41:48** PacketResponder 0 for block blk_904791815409399662 terminating
5	**2008-11-11 03:41:48** Received block blk_904791815409399662 of size 67108864 from /10.250.18.114
6	**2008-11-11 03:41:48** PacketResponder 1 for block blk_904791815409399662 terminating
7	**2008-11-11 03:41:48** Received block blk_904791815409399662 of size 67108864 from /10.251.43.210
8	**2008-11-11 03:41:48** BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.210:50010 is added to blk_904791815409399662 size 67108864
9	**2008-11-11 03:41:48** BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.18.114:50010 is added to blk_904791815409399662 size 67108864
10	**2008-11-11 08:30:54** Verification succeeded for blk_904791815409399662

Data-centric dependability model

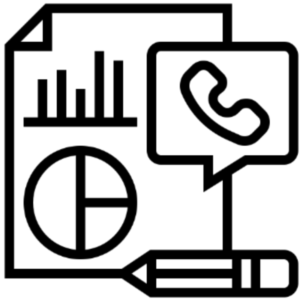
Log

Meter Data



Topology

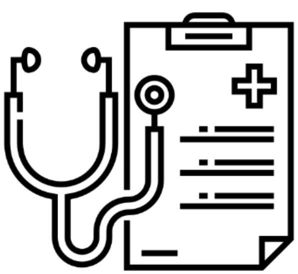
Alert



Incident Ticket

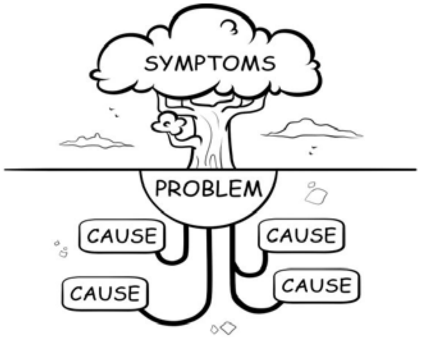
Anomaly Detection

Fault Avoidance



Failure Diagnosis

Fault Tolerance



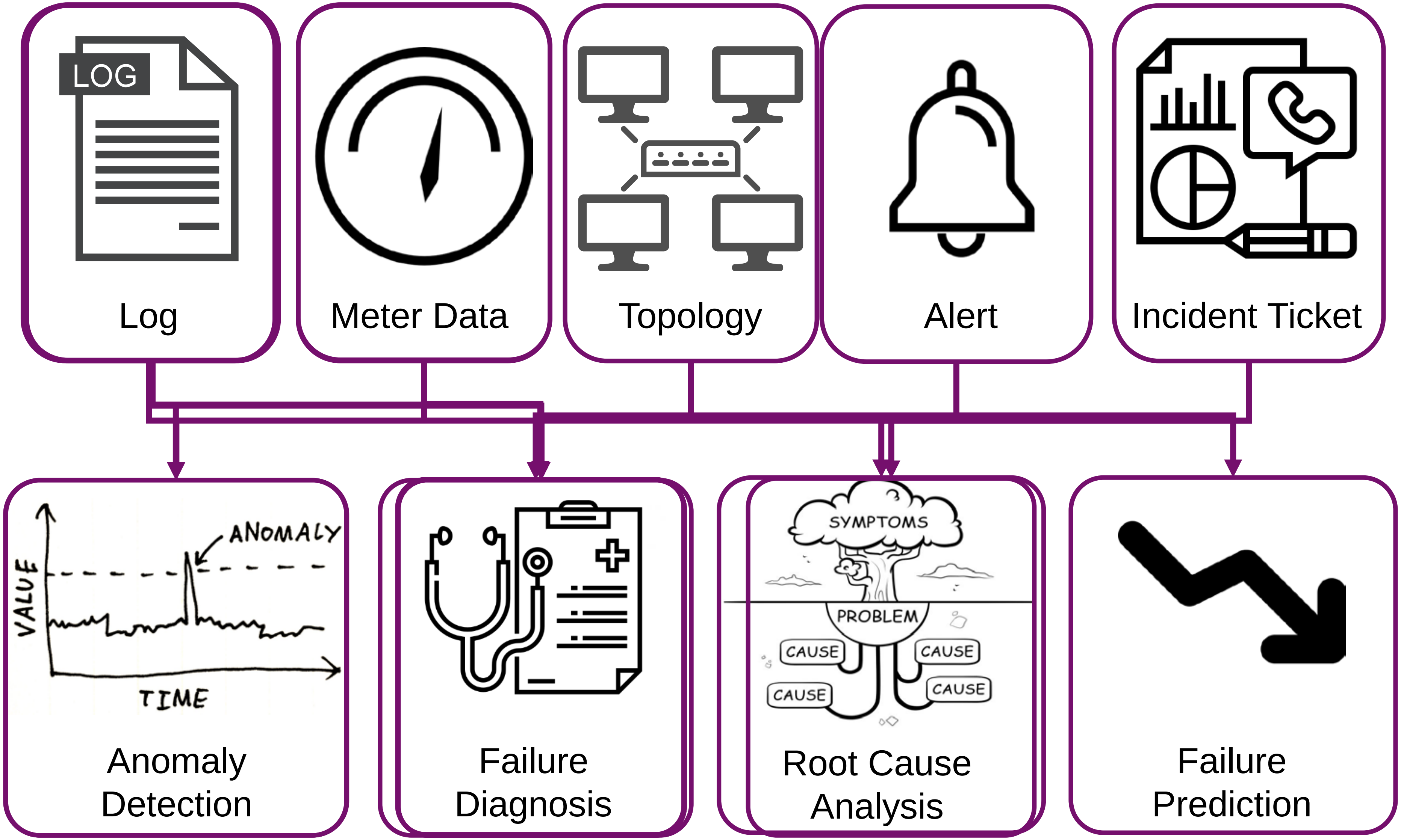
Root Cause Analysis

Fault Removal

Failure Prediction

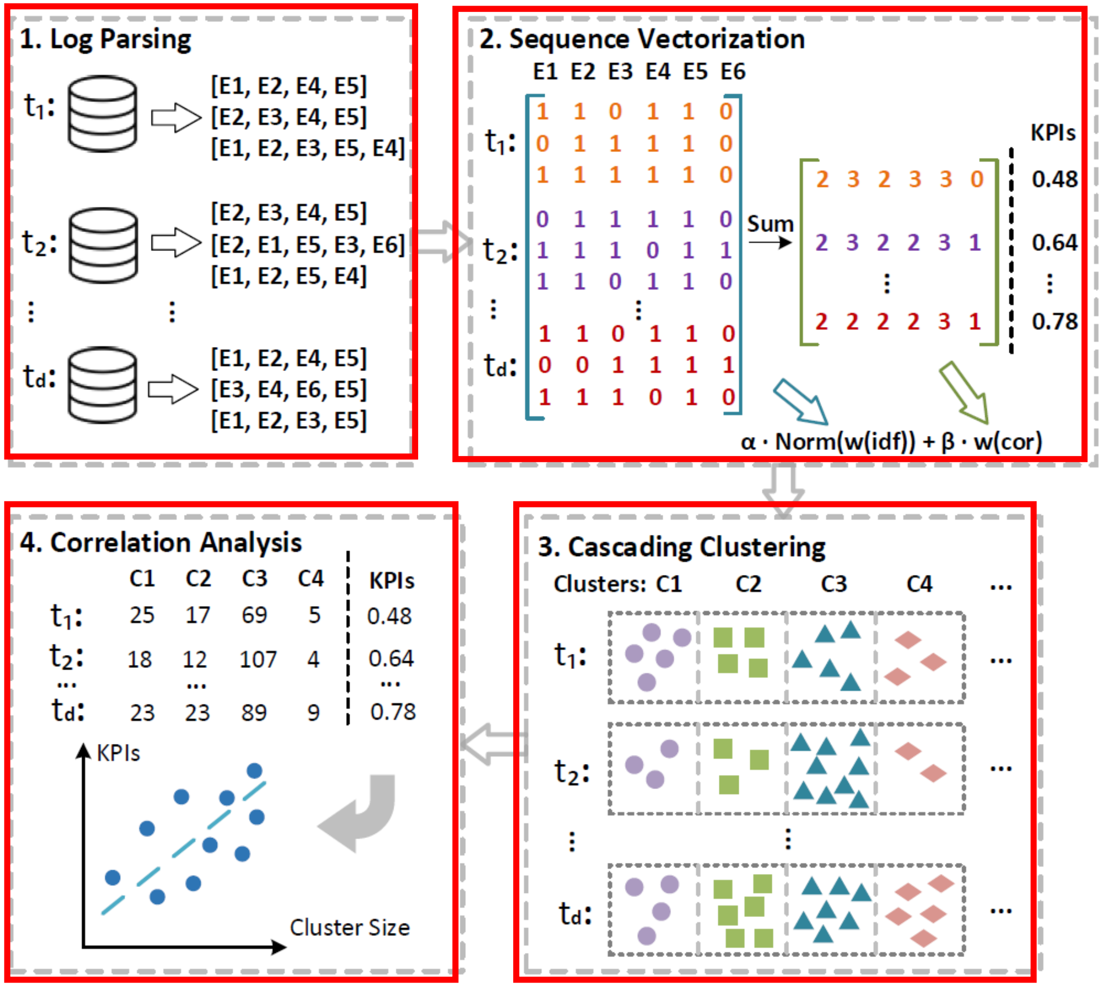
Fault Prediction

Software reliability engineering tasks



Log-based problem identification

- Impactful system problems:
  - Can lead to the degradation of service KPI.
- **Target:**
  - Identify clusters that are highly correlated with service KPI's changes.
- **Method:**
  - Model the relation between cluster sizes and KPI values



He et al., "Identifying impactful service system problems via log analysis," FSE 2018.

Log-based problem identification

- Evaluation on real Microsoft Azure data

Table 1: Summary of Service X Log Data

Data	Snapshot starts	#Log Seq (Size)	#Events	#Types
Data 1	Sept 5th 10:50	359,843 (722MB)	365	16
Data 2	Oct 5th 04:30	472,399 (996MB)	526	21
Data 3	Nov 5th 18:50	184,751 (407MB)	409	14

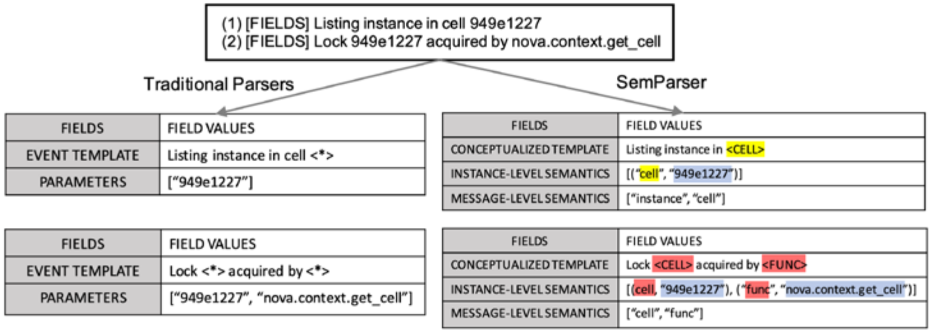
Table 2: Accuracy of Problem Detection on Service X Data

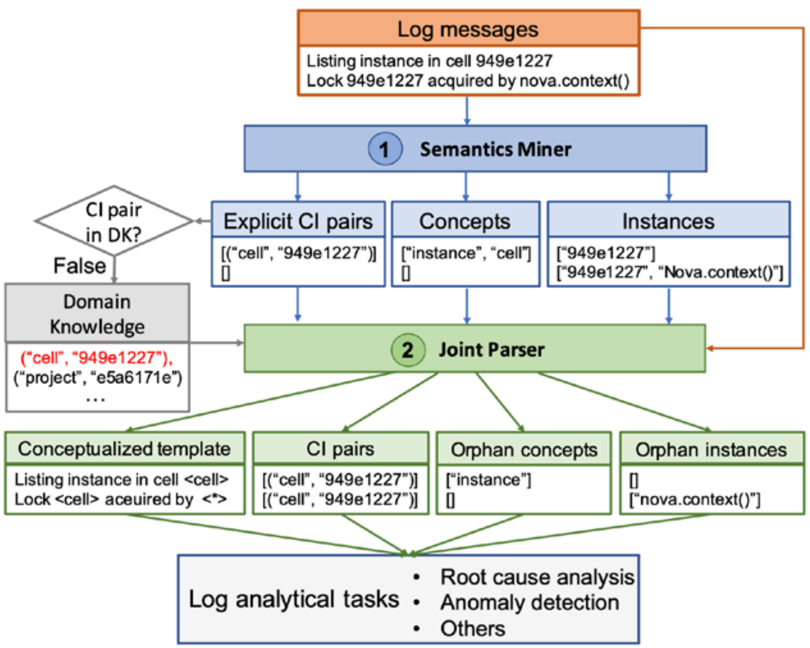
Data	Data 1			Data 2			Data 3		
Metrics	Precision	Recall	F1-measure	Precision	Recall	F1-measure	Precision	Recall	F1-measure
PCA	0.465	0.946	0.623	0.142	0.834	0.242	0.207	0.922	0.338
Invariants Mining	0.604	1	0.753	0.160	0.847	0.269	0.168	0.704	0.271
Log3C	**0.900**	0.920	**0.910**	**0.897**	0.826	**0.860**	**0.834**	0.903	**0.868**

He et al., "Identifying impactful service system problems via log analysis," FSE 2018.

Semantic log parsing for log analysis

- **Semantics in log parsing**
  - Many parameters in a log message have technical meaning, which provide extra information for log analysis
- **Target**
  - Identify meaningful tokens and the corresponding categories
- **Method**
  - Iteratively update the concept-instance knowledge base, based on which to identify new instances





Semantic log parsing for log analysis

- Experimental results

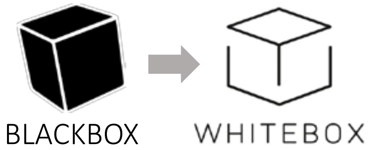
Table 4: Experiments results of mining semantics from logs.

Framework	Andriod P	Andriod R	Andriod F1	Hadoop P	Hadoop R	Hadoop F1	HDFS P	HDFS R	HDFS F1	Linux P	Linux R	Linux F1	OpenStack P	OpenStack R	OpenStack F1	Spark P	Spark R	Spark F1	Zookeeper P	Zookeeper R	Zookeeper F1
SemParser	0.951	0.935	**0.943**	0.993	0.978	**0.985**	1.000	1.000	**1.000**	0.998	0.977	**0.987**	0.999	0.998	**0.999**	1.000	0.998	**0.999**	1.000	0.989	**0.995**
- w/o F_{char}	0.981	0.909	**0.943**	0.988	0.953	0.970	1.000	0.998	0.999	0.995	0.957	0.976	0.995	0.989	0.992	1.000	0.998	**0.999**	0.993	0.987	0.990
- w/o F_{local}	0.979	0.858	0.915	0.993	0.880	0.933	1.000	0.999	0.999	0.992	0.947	0.969	0.994	0.989	0.992	1.000	0.937	0.967	0.997	0.940	0.968
- w/o $LSTM$	0.979	0.858	0.915	0.993	0.879	0.932	1.000	0.999	0.999	0.995	0.909	0.951	1.000	0.963	0.981	0.985	0.998	0.992	0.966	0.953	0.959
- w/o F_{contx}	0.977	0.060	0.113	0.984	0.253	0.403	0.999	0.289	0.449	0.999	0.242	0.389	1.000	0.256	0.407	1.000	0.268	0.423	0.842	0.197	0.319

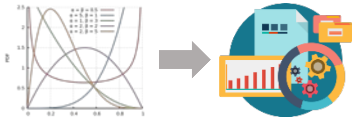
Table 5: Experimental results in anomaly detection task.

Baseline	LSTM Precision	LSTM Recall	LSTM F1	Atten-biLSTM Precision	Atten-biLSTM Recall	Atten-biLSTM F1	CNN Precision	CNN Recall	CNN F1	Transformer Precision	Transformer Recall	Transformer F1
LenMa	0.717	**0.938**	0.813	0.714	**0.924**	0.806	0.793	0.815	0.804	0.685	0.896	0.776
AEL	0.738	0.934	0.824	0.791	0.877	0.832	0.747	0.924	0.826	0.503	**0.962**	0.660
Drain	0.824	0.867	0.845	0.810	0.886	0.846	0.737	**0.943**	0.827	0.693	0.919	0.790
IPLoM	0.863	0.833	0.848	0.808	0.877	0.841	0.834	0.834	0.834	0.929	0.683	0.787
SemParser	**0.971**	0.927	**0.948**	**0.952**	0.913	**0.932**	**0.907**	0.899	**0.903**	**0.938**	0.904	**0.921**
Δ%			+11.80%			+10.17%			+8.27%			+16.58%

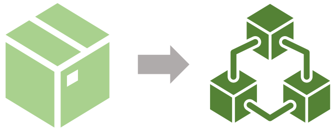
Data-driven software dependability modeling



Black-box to white-box



Model-centric to data-centric

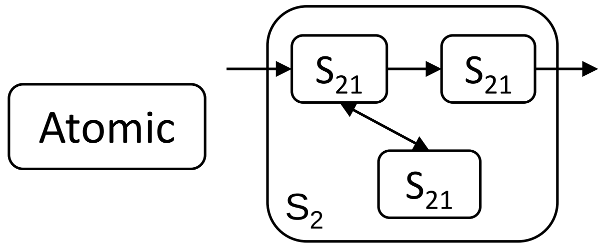


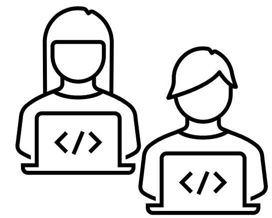
Macro-level to micro-level

Static analysis to dynamic analysis

Macro-level dependability modeling

- Macro perspective is insufficient



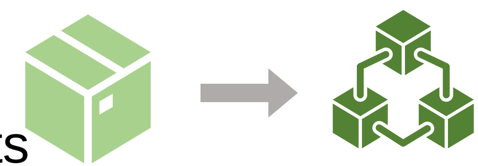


  - A single failure number for behavior prediction
  - Too course-grained for dependability modeling

  - Atomic systems
  - Oversimplified interactions for systems with components

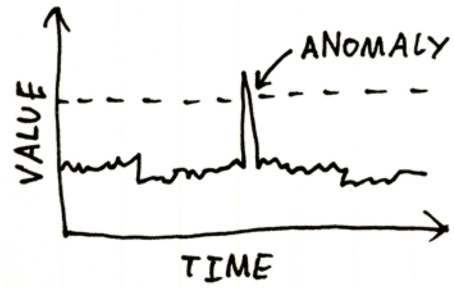
  - Manual failure data collection

- Micro perspective
  - Profile software reliability from multiple aspects
  - Dynamic interactions in microservices
  - Automated data collection process

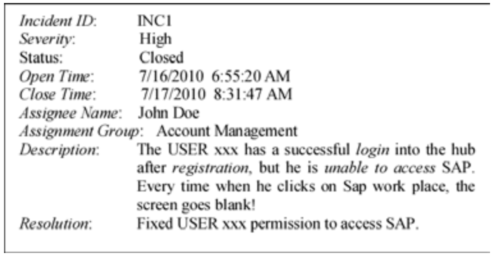


Micro-level dependability modeling

- Profile software reliability from multiple facets



Anomaly



Incident ID:	INC1
Severity:	High
Status:	Closed
Open Time:	7/16/2010 6:55:20 AM
Close Time:	7/17/2010 8:31:47 AM
Assignee Name:	John Doe
Assignment Group:	Account Management
Description:	The USER xxx has a successful *login* into the hub after *registration*, but he is *unable to access* SAP. Every time when he clicks on Sap work place, the screen goes blank!
Resolution:	Fixed USER xxx permission to access SAP.

Incident

Failure report

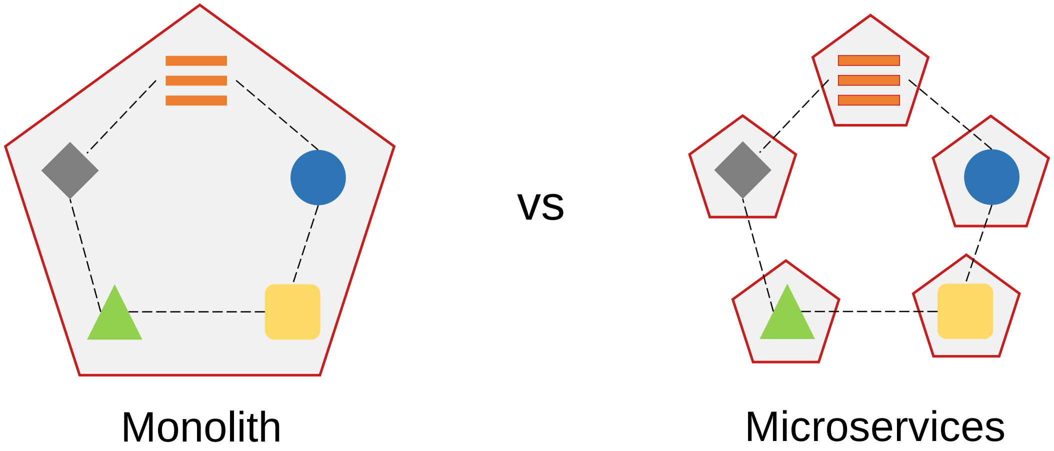
- Abnormal runtime status
- Early signal for failures
- Fine-grained information

- Rendered by monitors
- Detailed failure description
- Contain human knowledge

- Failure statistics over a period
- Derive system vulnerability

Micro-level dependability modeling

- Dynamic interactions in microservices
  - Loosely-coupled and collaborating microservices
  - Dependencies are hard to capture
  - Interactions are always changing



The criticality of service dependencies

AWS Post-Event Summaries

AWS Post-Event Summaries

The following is a list of post-event summaries from major service events that impacted AWS service availability:

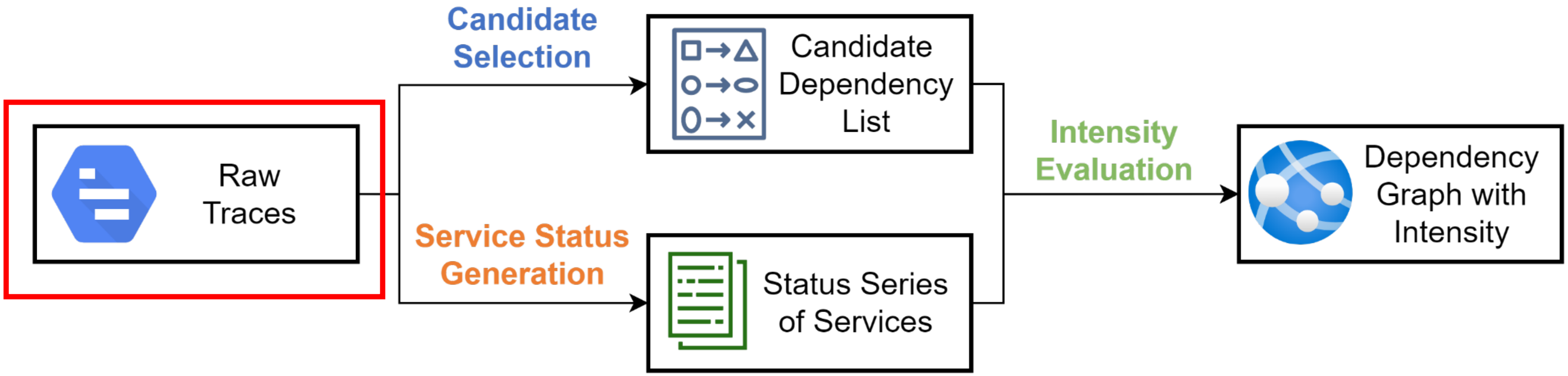
- Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region, November, 25th 2020
- Summary of the Amazon EC2 and Amazon EBS Service Event in the Tokyo (AP-NORTHEAST-1) Region, August 23, 2019
- Summary of the Amazon EC2 DNS Resolution Issues in the Asia Pacific (Seoul) Region (AP-NORTHEAST-2), November 24, 2018.
- Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region, February 28, 2017.
- Summary of the AWS Service Event in the Sydney Region, June 8, 2016.
- Summary of the Amazon DynamoDB Service Disruption and Related Impacts in the US-East Region, September 20, 2015.
- Summary of the Amazon EC2, Amazon EBS, and Amazon RDS Service Event in the EU West Region, August 7, 2014.
- Summary of the Amazon SimpleDB Service Disruption, June 13, 2014.
- Summary of the December 17th event in the South America Region (SA-EAST-1), December 20, 2013.
- Summary of the December 24, 2012 Amazon ELB Service Event in the US-East Region, December 24, 2012.
- Summary of the October 22, 2012 AWS Service Event in the US-East Region, October 22, 2012.
- Summary of the AWS Service Event in the US East Region, July 2, 2012.
- Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region, April 29, 2011.

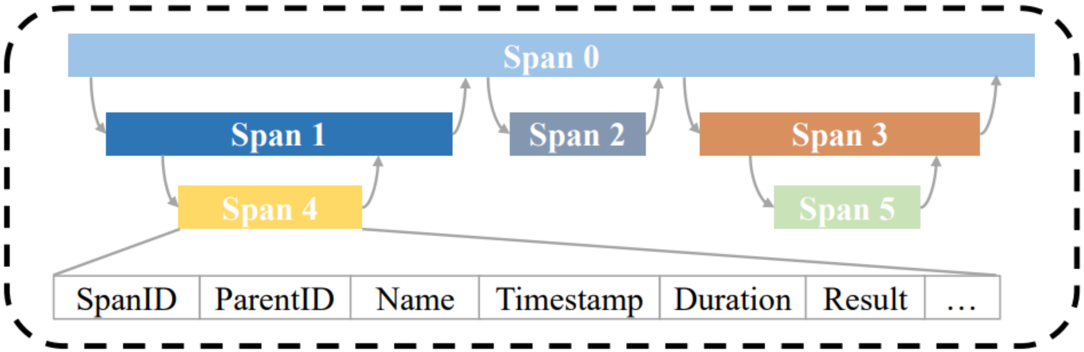
Cascading failure

5 out of 13 AWS outages are related to service dependency!

https://aws.amazon.com/premiumsupport/technology/pes/

Prediction of aggregated intensity of dependency



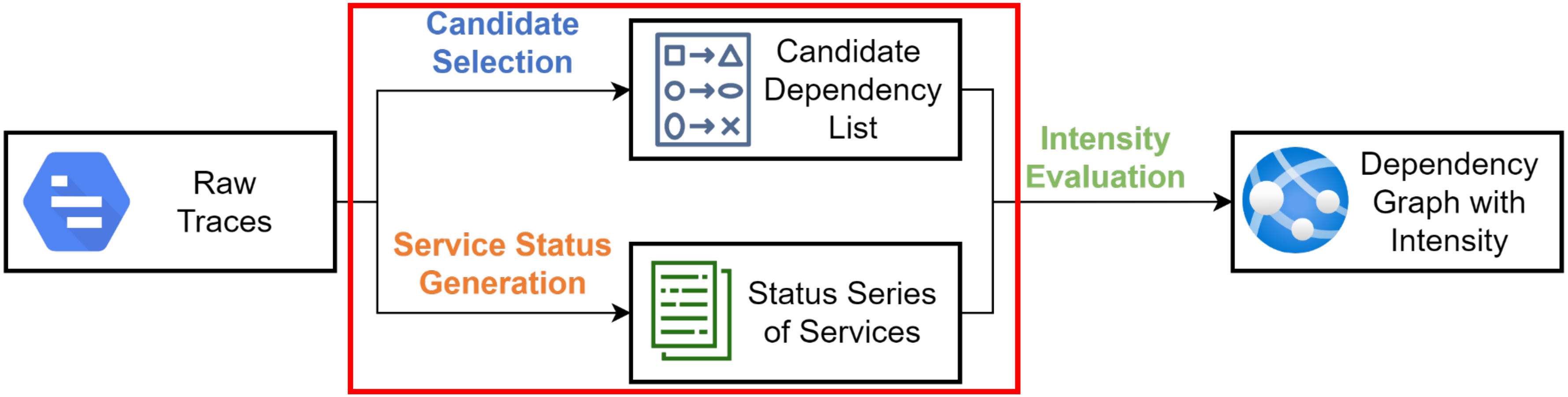


A trace log with 6 spans.

Span ID	e22f30bdbfd09134
Parent Span ID	b42a04bf18997d5d
Name	ts-preserve-service
Timestamp (μs)	1618589098705000
Duration (μs)	1126
Result	SUCCESS
Trace ID	c0d17d481f47bdd9
Additional Logs	……

Yang *et al.*, "AID: Efficient Prediction of Aggregated Intensity of Dependency in Large-scale Cloud Systems", ASE 2021

Prediction of aggregated intensity of dependency

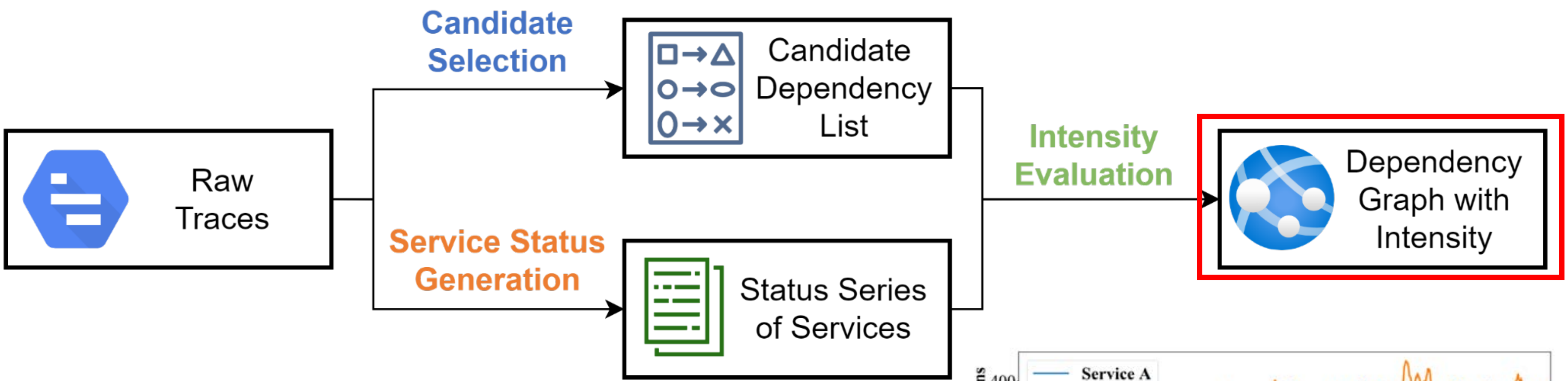


Method

- Select the candidate invocation pairs $(caller, callee)$
- Three aspects of indicators of service status
  - Number of Invocations
  - Durations of Invocations
  - Error of Invocations

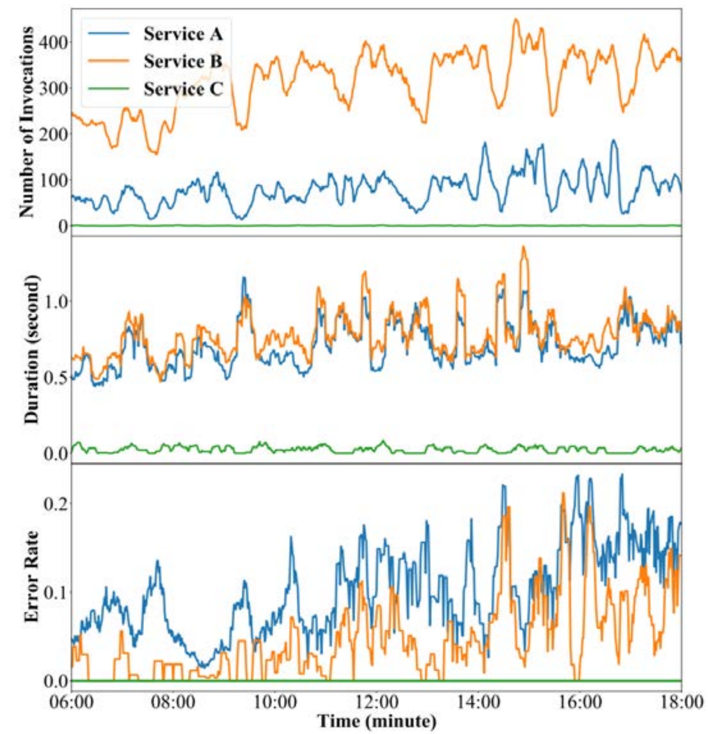
Yang *et al.*, “AID: Efficient Prediction of Aggregated Intensity of Dependency in Large-scale Cloud Systems”, ASE 2021

Prediction of aggregated intensity of dependency



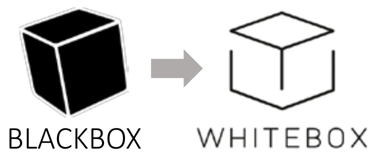
Method

- Select the candidate invocation pairs $(caller, callee)$
- Three aspects of indicators of service status
  - Number of Invocations
  - Durations of Invocations
  - Error of Invocations

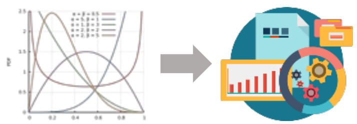


Yang *et al.*, “AID: Efficient Prediction of Aggregated Intensity of Dependency in Large-scale Cloud Systems”, ASE 2021

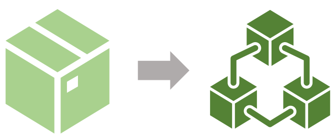
Data-driven software dependability modeling



Black-box to white-box



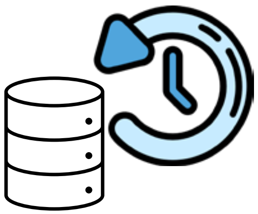
Model-centric to data-centric



Macro-level to micro-level

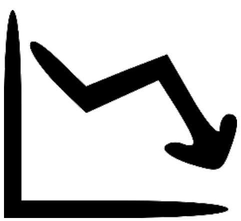
Static analysis to dynamic analysis

Static analysis-based dependability modeling



Highly rely on historical failure data

- The software is assumed to be mature enough
- Model selection
- Model training



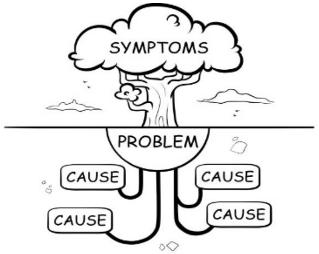
Poor performance if the software changes considerably

- New capabilities exercised
- Different testing methodology/environment employed

Dynamic analysis-based dependability modeling

Anomaly Detection

Failure Diagnosis

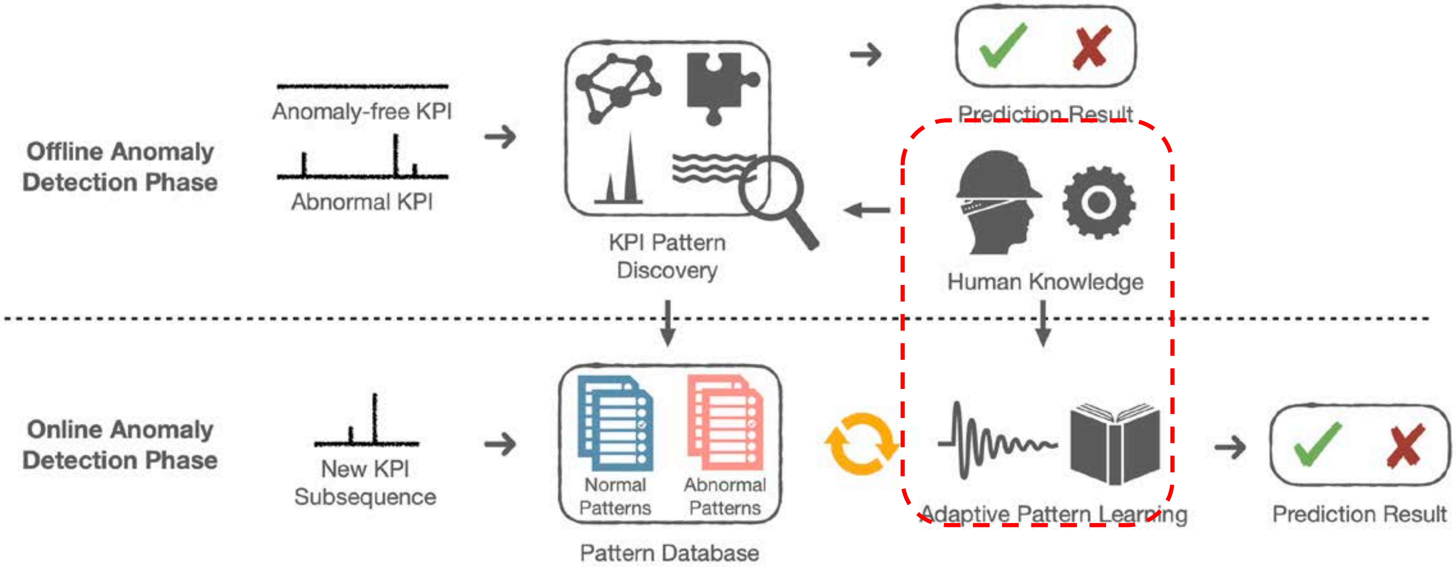


Root Cause Analysis

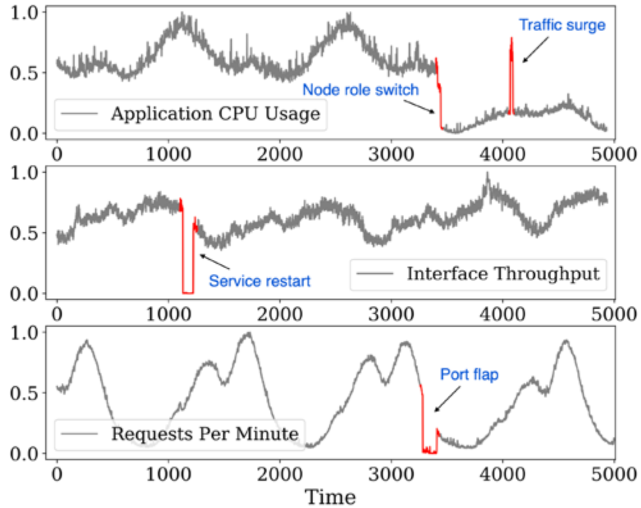
Failure Prediction

- Consider both historical and on-going data
- Models have online learning capabilities
  - Online machine learning
  - Zero-shot learning
  - Transfer learning
  - …

Adaptive KPI anomaly detection



- **Target**
  - KPI anomaly detection with online adaptability
- **Method**
  - Identify abnormal KPI patterns based on historical occurrences
  - Add new patterns based on the similarity to known patterns
  - Human knowledge can be incorporated

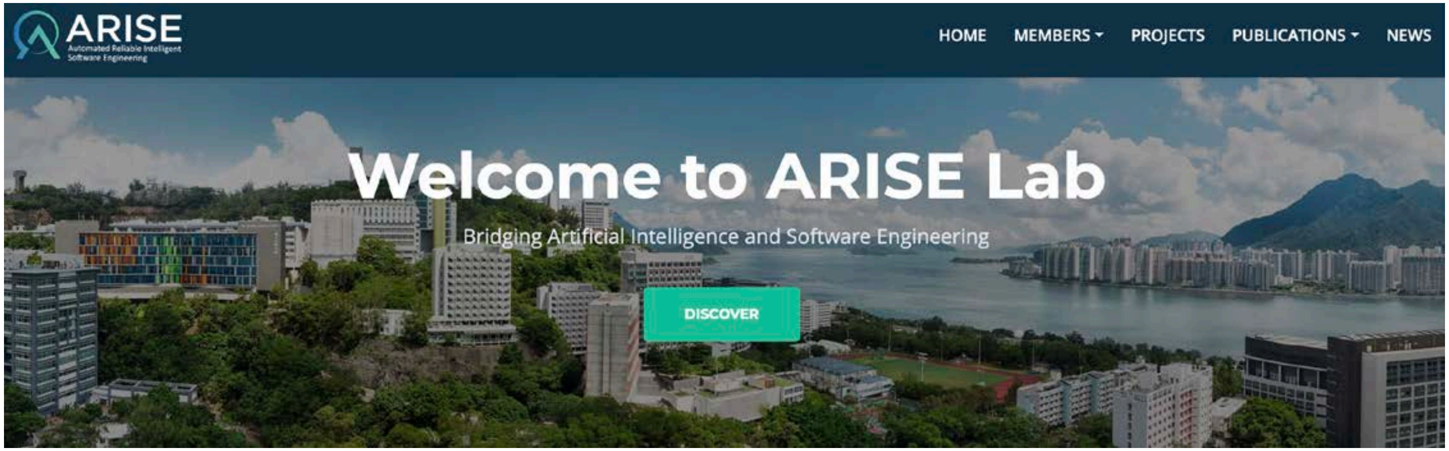


Chen *et al.*, "Adaptive Performance Anomaly Detection for Online Service Systems via Pattern Sketching", ICSE 2022

Summary and Conclusion

- Traditional dependability modeling is losing its practical significance and relevance
- Modern software systems make dependability modeling more challenging
- Data-driven dependability modeling with AI
  - From black-box to white-box
  - From model-centric to data-centric
  - From macro-level to micro-level
  - From static analysis to dynamic analysis

In the evolution of dependability modeling, the paradigm shift to a data-driven approach is an inevitable modeling effort, and AI techniques such as machine learning are called for.



Thank you!

ARISE
Automated Reliable Intelligent
Software Engineering