

Predicting Enterprise Application Performance Measures through Time-series Forecasting

Daniel Elsner, 21st August 2017, Scientific advisor: Pouya Aleatrati Khosroshahi

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
www.matthes.in.tum.de



Motivation and Approach



Research Artifact



Research Questions



Data Architecture



Project Plan

Problem Domains in Application Performance Monitoring (APM)



Performance



Availability



Maintainability

“Companies are sitting on a **treasure trove**
– if only they knew how to use it.”

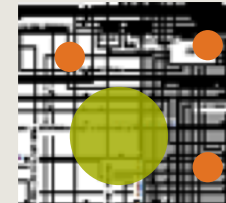
A. Samuel, Wall Street Journal, 2015

Evolution of Enterprise Architecture (EA)

Growing complexity
and **high business**
relevance of **Enterprise**
Architecture (EA)



Harness potential of
monitoring data



Detect root causes,
reduce complexity and
lead to a **higher agility** in
EA management



1

What consecutive patterns can be identified by analyzing **performance flaws** in APM data?

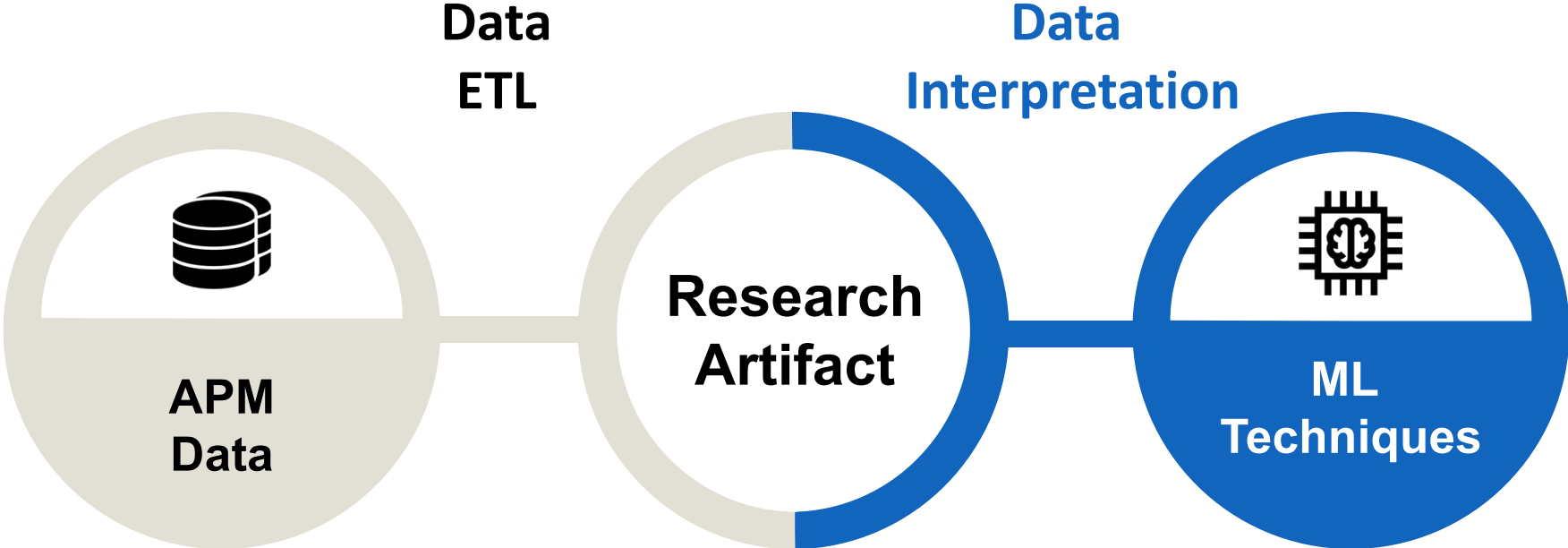
2

How can APM data be used to forecast **availability insufficiencies** in advance of occurrence?

3

What **proactive actions** can be derived to avoid the identified patterns and increase performance and availability?

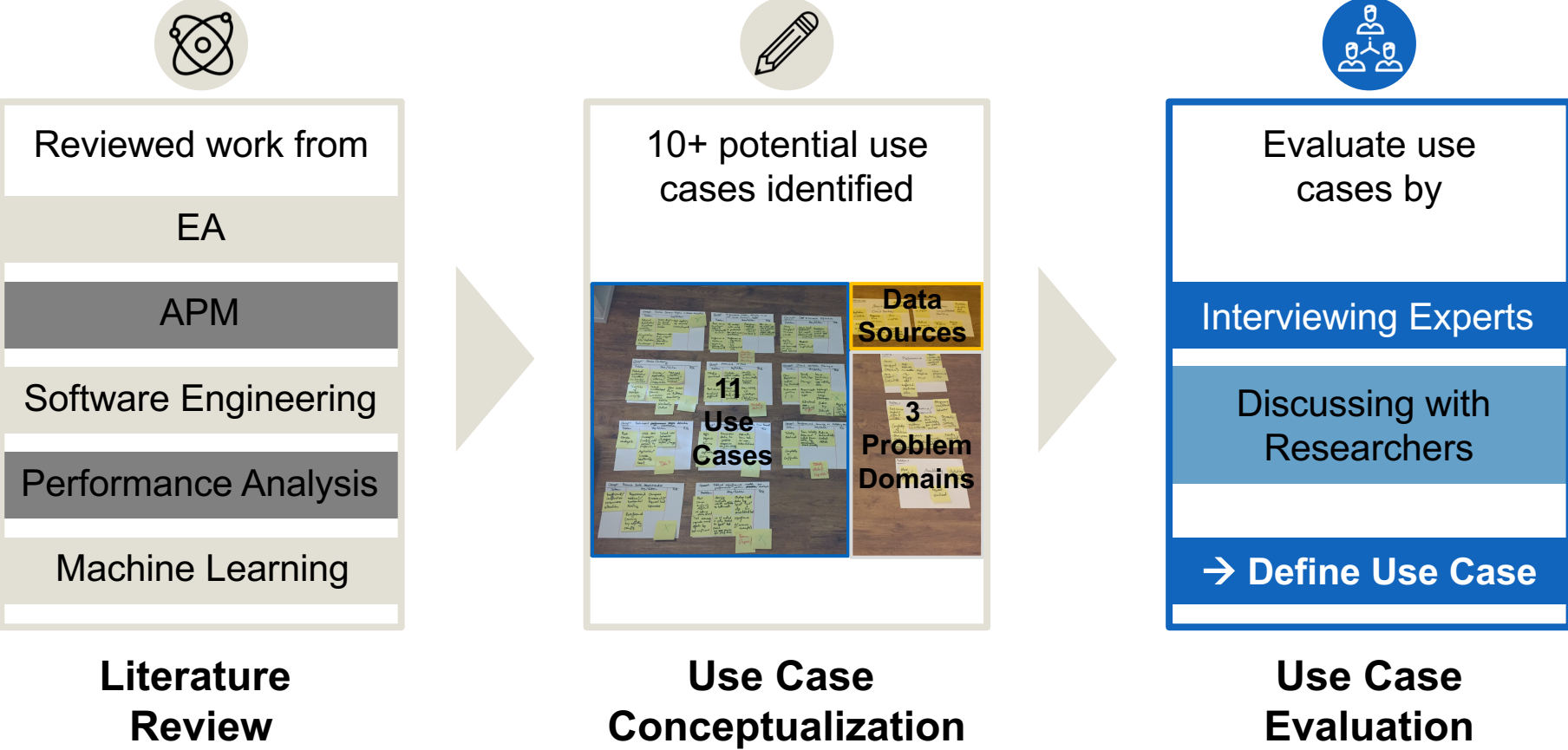




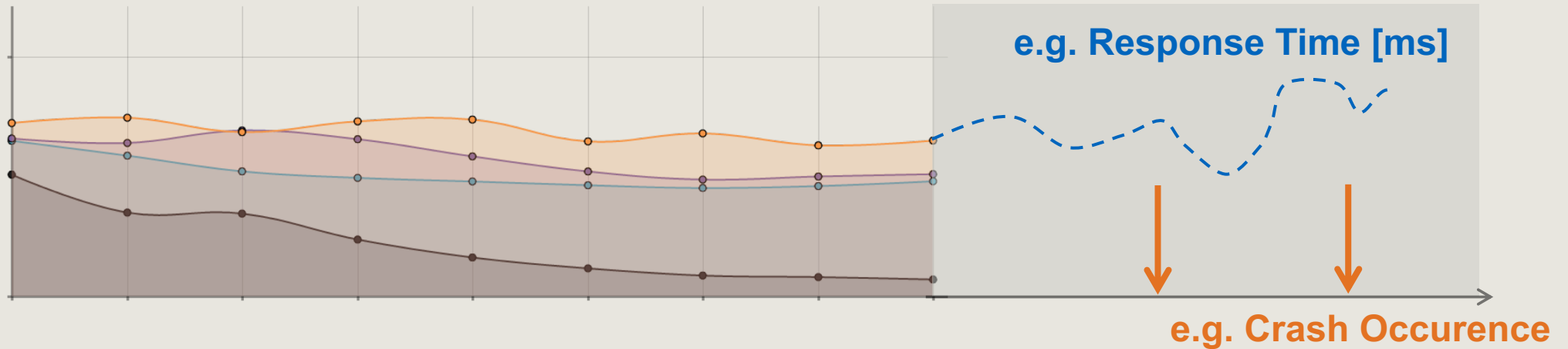
Gain insights and create value from **Application Performance Monitoring (APM)** data by applying **machine learning techniques**.



Approach



Time-series Forecasting



From historical sequential APM data create forecasts for **performance measures** and **incident probabilities**.

1

Forecasting of relevant performance measures

2

Incident prediction

3

Optional
Automatic ticket severity evaluation driven by ML



1

How accurately can a time-series forecasting model predict APM measures?

2

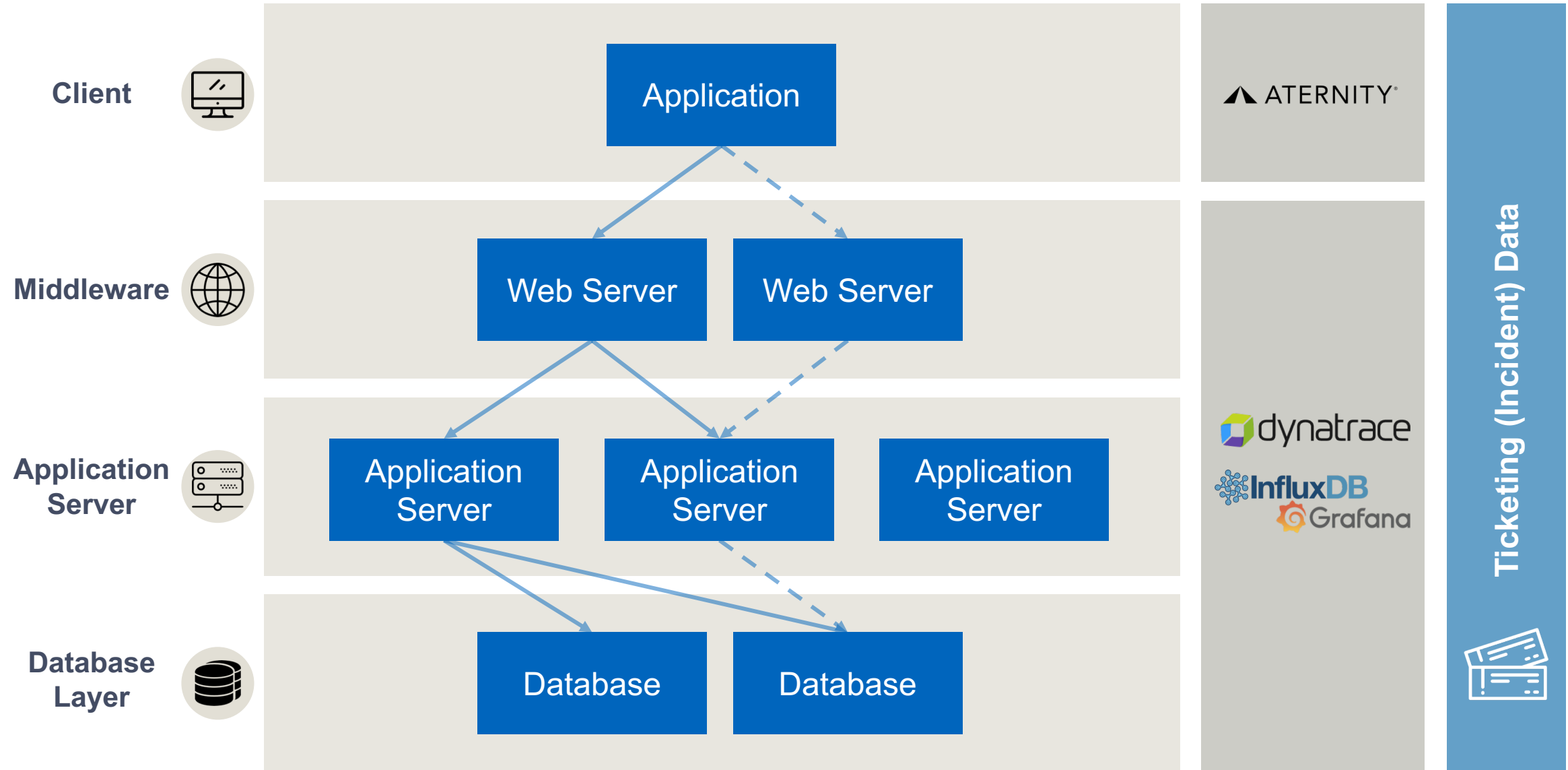
How well can a time-series forecasting model predict availability lacks (i.e. application crashes) in enterprise applications/services?

3

To what extent can we evaluate automatic ticket severity classification by analyzing APM data?



Data Architecture – Layers and Data Sources





Dimensions

User	Device	Application
Activity	Location	Time

Measures

Crash rate	Response Time	Hang Time
Resource Util.	Network I/O	



Dimensions

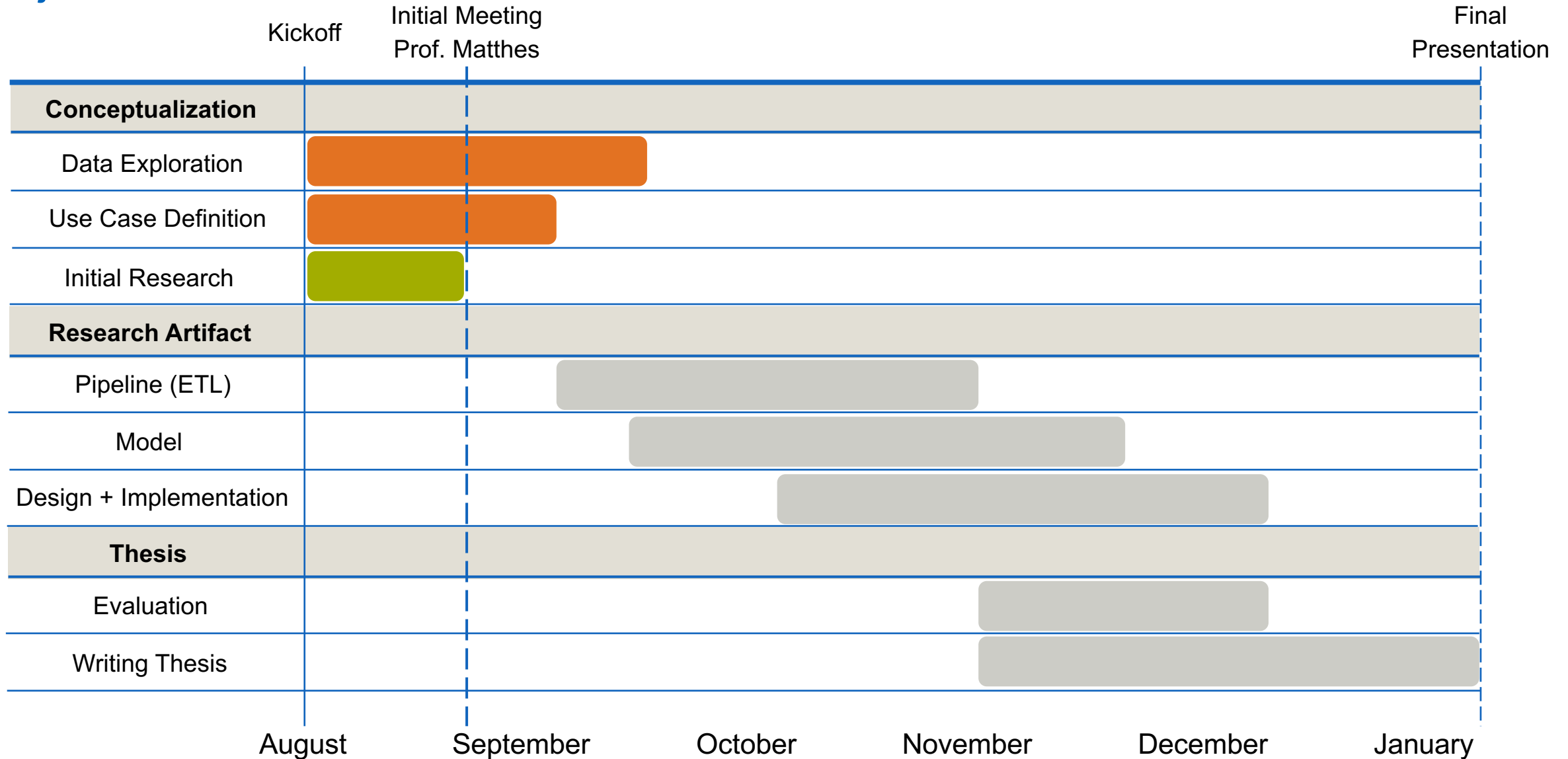
Application	App. Server	Webserver
Database	Event	Time
Method	Endpoint	

Measures

Crash Rate	Response Time	Request Load
Resource Util.	Network I/O	Method Count



Project Plan





Cand. M. Sc.

Daniel Elsner

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for Business
Information Systems

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289.17135

Daniel.elsner@tum.de
www.matthes.in.tum.de



Handwritten notes on a wooden surface, organized into several conceptual cards. Each card follows a similar structure: Concept, Problem, Idea/Solution, and RCA. The notes are written in black ink on white paper, with various yellow sticky notes attached to them.

Concept: Service Query - Right → service degradation
 Problem: Polluted architecture, complex services, complexity of CA/application landscape.
 Idea/Solution: Analyze origin, class, capability to search for similar services; Recommend that suggests equally alternative services.
 RCA: ✓

Concept: Performance problem detection and root cause analysis model
 Problem: ML model with only X predictors for root cause analysis.
 Idea/Solution: Predictors: root cause, status, amount, RCA?; Root Cause: ML model with only X predictors; Performance indicators: CPU, I/O, Virtualization rate.
 RCA: ✓

Concept: Load scenario regression
 Problem: Server throughput → more clicks; Agility, warnings → low load on new load removal.
 Idea/Solution: Server load prediction for more clicks (IOT, mobile); Regression model to predict load and perf.; Product econ. implications?
 RCA: ✓

Concept: Service Clustering
 Problem: Polluted architecture, diversity of services → agile.
 Idea/Solution: Service Clusters / Categories; Detect architectural flaws → micro-services; Similarity matrix.
 RCA: ✗

Concept: Automated IT-Data
 Problem: Ticketing overhead.
 Idea/Solution: Create tickets, create with perf. scenarios; Ticket similarity; Root cause analysis difficult.
 RCA: Ticketing data? ✗

Concept: Client Application Manager
 Problem: User Experience Index (as ClientCall); Performance problems (...).
 Idea/Solution: Small Tail / App Manager; Draw conclusions from app activity data; Learnings about usage (Histograms) by Network I/O.
 RCA: Application Data → Usage Histogram ✗

Concept: Path-relevant performance issues detection
 Problem: Root cause analysis.
 Idea/Solution: Link env changes (config, prod version) to performance; Detect user behavior changes after change; Application/Service versioning too!
 RCA: Data? ✗

Concept: Timeseries-based Response Time Forecast
 Problem: High response times; Application/Service crash.
 Idea/Solution: Timeseries data to predict response time/availability; Abstracts from tech. or econ. interrelations.
 RCA: ✗

Concept: Reinforcement learning on ticketing events
 Problem: Ticketing overhead; Complexity in Configuration.
 Idea/Solution: From ticketing data and ticket transaction re-evaluation, event diversity; Reinforcement learning on ticketing events.
 RCA: Ticketing data? Lap data? ✗

Concept: Resource Scale Recommendations
 Problem: Inefficient resource allocation.
 Idea/Solution: Recommend vertical/horizontal scaling; Compare resource utilization Request load scenarios; Reinforcement learning by adjusting configs.
 RCA: ✗

Concept: Method significance model for performance problem detection analysis
 Problem: Root cause analysis difficult → reduce data amount; Test scenario repeat manual efforts by test engineers.
 Idea/Solution: Classify methods which contribute to bottlenecks; Cluster input data by "good" and "bad" with computational load; Significance = f(resource consumption).
 RCA: Focus Paper! ✗

Availability Client Monitoring
 Application: Callback; Response time → AWS; Activity → open Mail.
 Error/Lead → avoid all; Client; Usage → 99.9%; Device → HP laptop → R6300M → 2.5 GHz.

Performance
 Request → /api/; Dynamic Server/Hosted Monitoring; Possible → 2-3x File creation.
 Method → main O; Same Response → 10ms; Response Unit → 10% CPU → 20% RAM.

Problems
 Server throughput (100/min); More clients (100, mobile); High resource usage; CPU Memory still; High response times; User experience index; Affects: common at low capacity, operational costs.

Problems
 Root cause analysis difficult → data; Complexity of infrastructure; Error-prone config; Manual effort; Maintenance/Management; Monitoring data, availability, response time; Heterogeneous user/device behavior; Diversity of services → agility; Ticketing overhead; Ticketing overhead; Ticketing overhead; Ticketing overhead.

Problems
 More clients (PI, mobile); Availability; Application service crash; Request overload.