CASE STUDY

CONTINUOUS PERFORMANCE TESTING WITH MACHINE LEARNING

itera
MAKE A DIFFERENCE

TaaS: Performance testing

Description:

Some huge Icelandic bank, had a request for checking their application performance during their migration from archaic technology to .NET services. Application is to be integrated with a lot of third party system and consist of three layers. The test team was responsible not only for searching bugs and creating performance testing framework, but also for teaching client for performance testing skills

Challenges:

- No documented knowledge about the system
- No special environment and no any test data
- No knowledge and expectation about performance testing
- No accesses
- No monitoring system

TaaS: Performance testing

Platforms / Technologies:

Java, Selenium + browsermobproxy + harstorage, Jmeter, R + Shiny

TaaS: Performance testing

Approach:

- Implementation of structured performance testing life cycle
- Integration of performance testing in continuous integration
- Automation of repetitive performance testing pipeline (smoke, load, scalability, stability, error detection)
- Maximal metric measurement with afterward analysis
- Maximal automatize of routine activities (data preparation and first analysis)
- Tight communication with onshore development team and customer teams

Scope:

Project Timeline

Initiation

- Define critical business values and goals
- Project schedule

Information gathering

- Infrastructure analysis
- Test Strategy
- Performance requirements analysis
- Design check-lists

Manual testing

- Script generating
- Manual preparation
- Manual run and bottleneck searching

Result Analysis & Reporting

- Recommendations how to fix performance issues
- Summary report

Automated run

- CI (TFS + teamcity)
- Distributed run
- Configuring test plan from CI tool
- CI pipeline

Connect monitoring

- Jmeter monitoring tool
- Dynatrace api
- Ckeck_mk monitoring

Automatization routine operation

- Data preparation
- Data cleaning
- Data saving
- Data wrangling

Report generation

- Static report
- Web-report

Automatized comparation with requirements

- Define metrics for comparations
- Defined requirements

Automatized requirement generation

- Arima forecasting expected results

Semiautomatic bottleneck detection

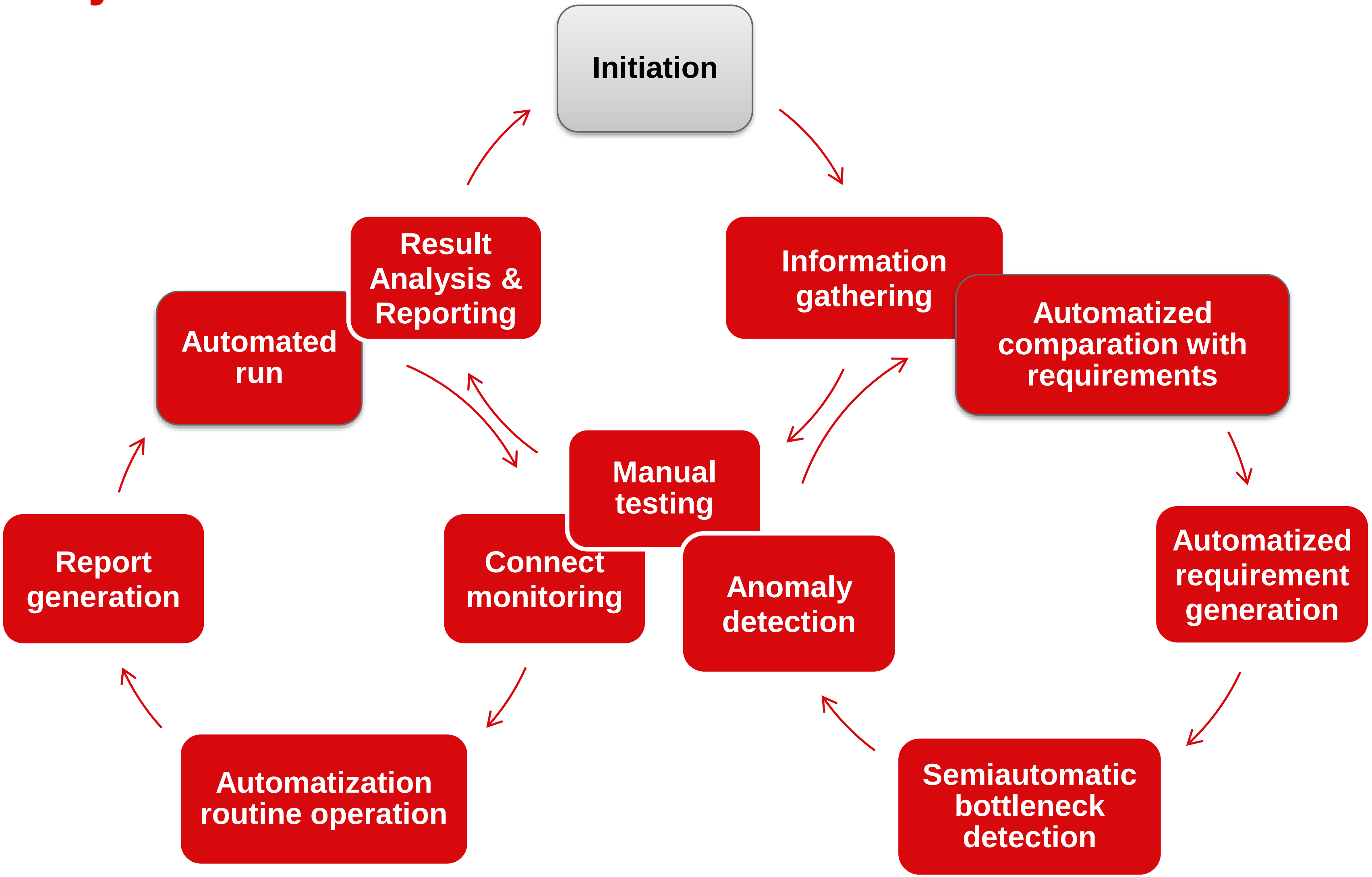
- Cluster analysis
- Correlation heatmaps

Anomaly detection

- Context depending
- Moving median
- Robust statistic

itera
MAKE A DIFFERENCE

Project Timeline in real life



Initiation

Information gathering

Automatized comparation with requirements

Automatized requirement generation

Semiautomatic bottleneck detection

Anomaly detection

Manual testing

Connect monitoring

Automatization routine operation

Report generation

Automated run

Result Analysis & Reporting

TaaS: Performance testing

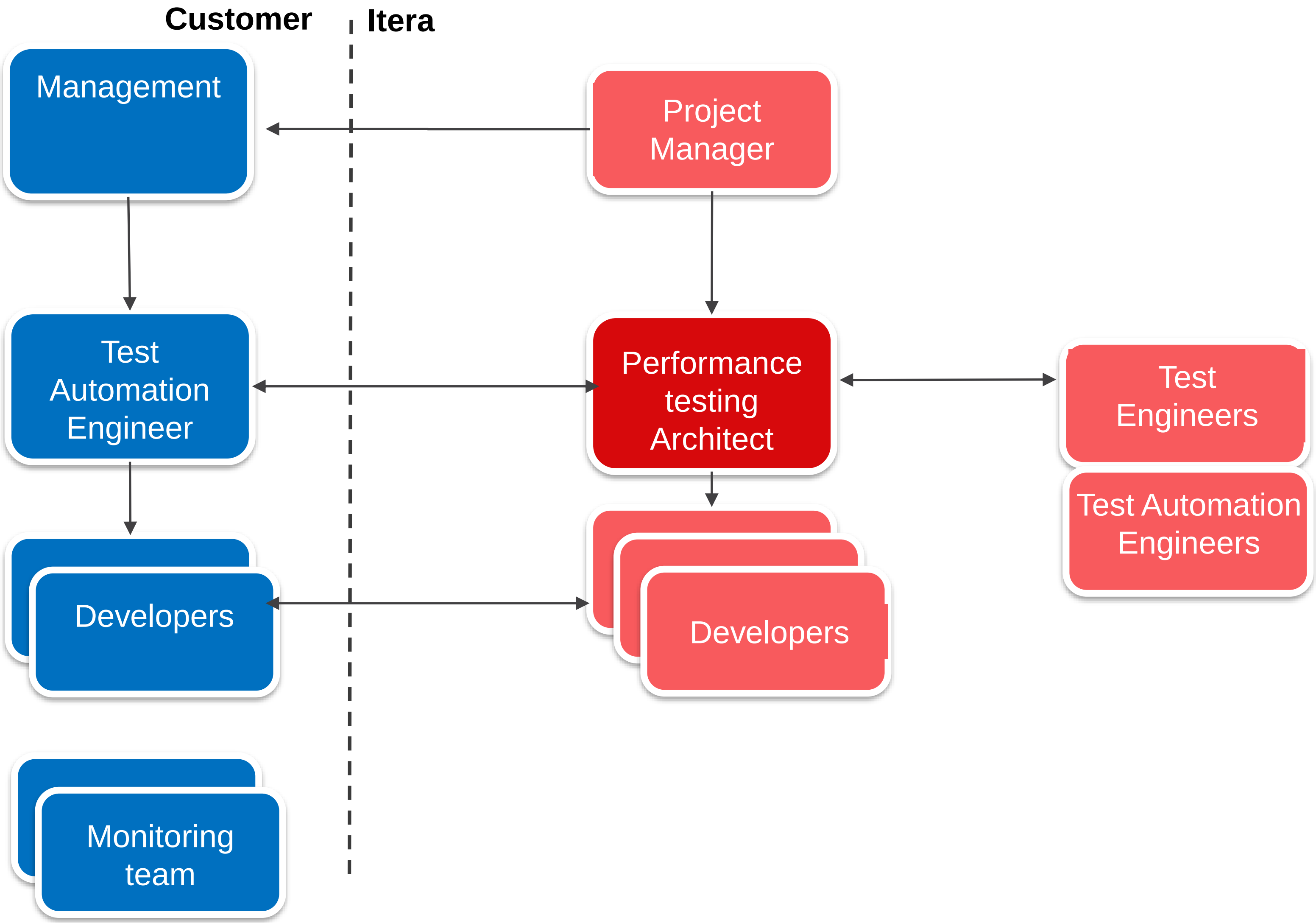
Results:

- Performance testing framework
- Performance testing CI
- Automated analysis
- Automated report generation
- Semiautomatic bottleneck detection (using ML)
- Integration with monitoring system

Deliverables:

- Test Strategy is introduced and followed
- Overall Test Scope is identified
- Performance testing framework (description and documentation)
- Scenarios and scripts automated (Smoke, load, scalability, stability)
- Customer issues analyzed (application, DB, infrastructure)
- Recommendation provided
- CI configured
- Daily ongoing automated reports and recommendation

Performance team integration



- Joint team
- Tight collaboration

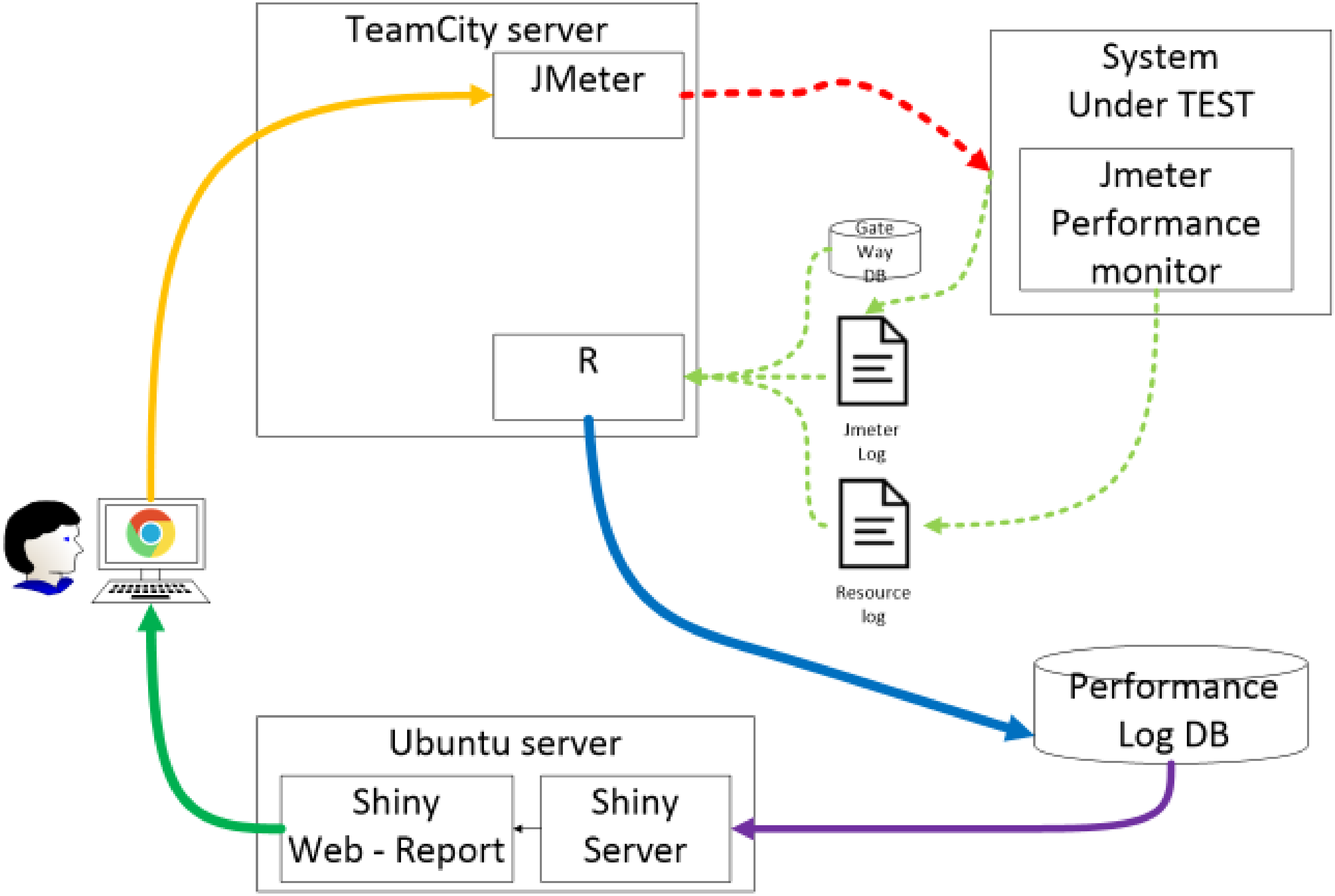
Continuous PT

Report Generation

Machine learning in PT

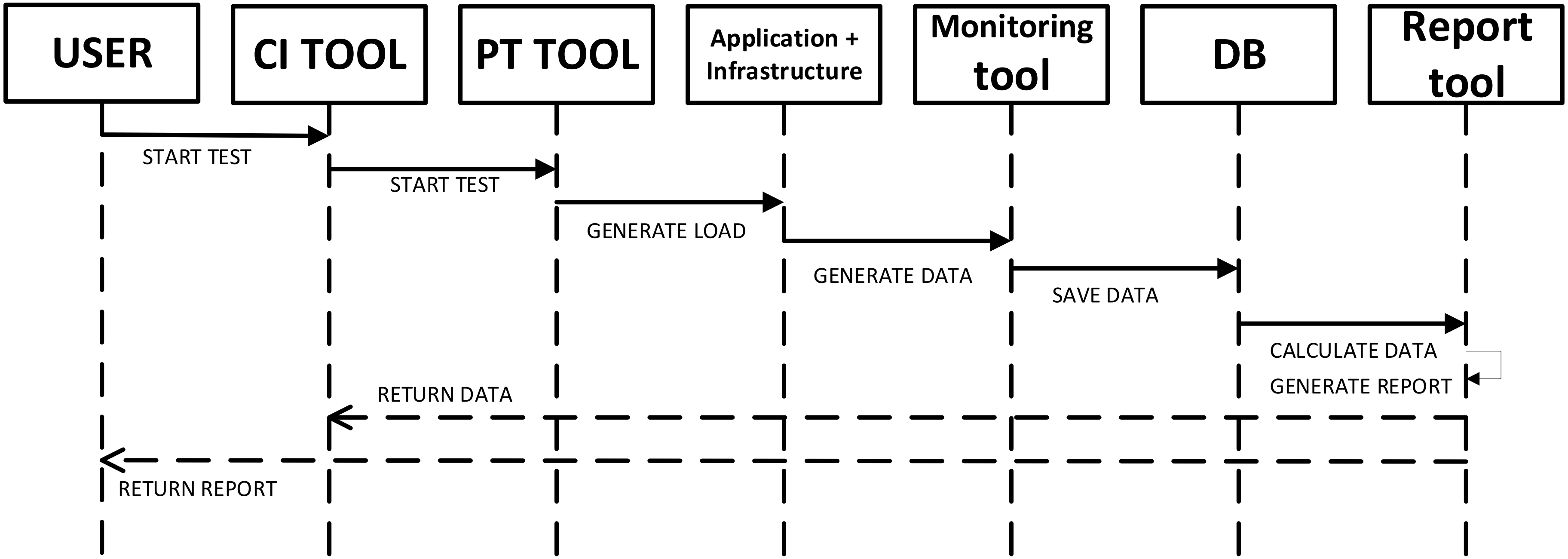
Recommendation for bottleneck detection

PT fairy ~~fails~~ tales

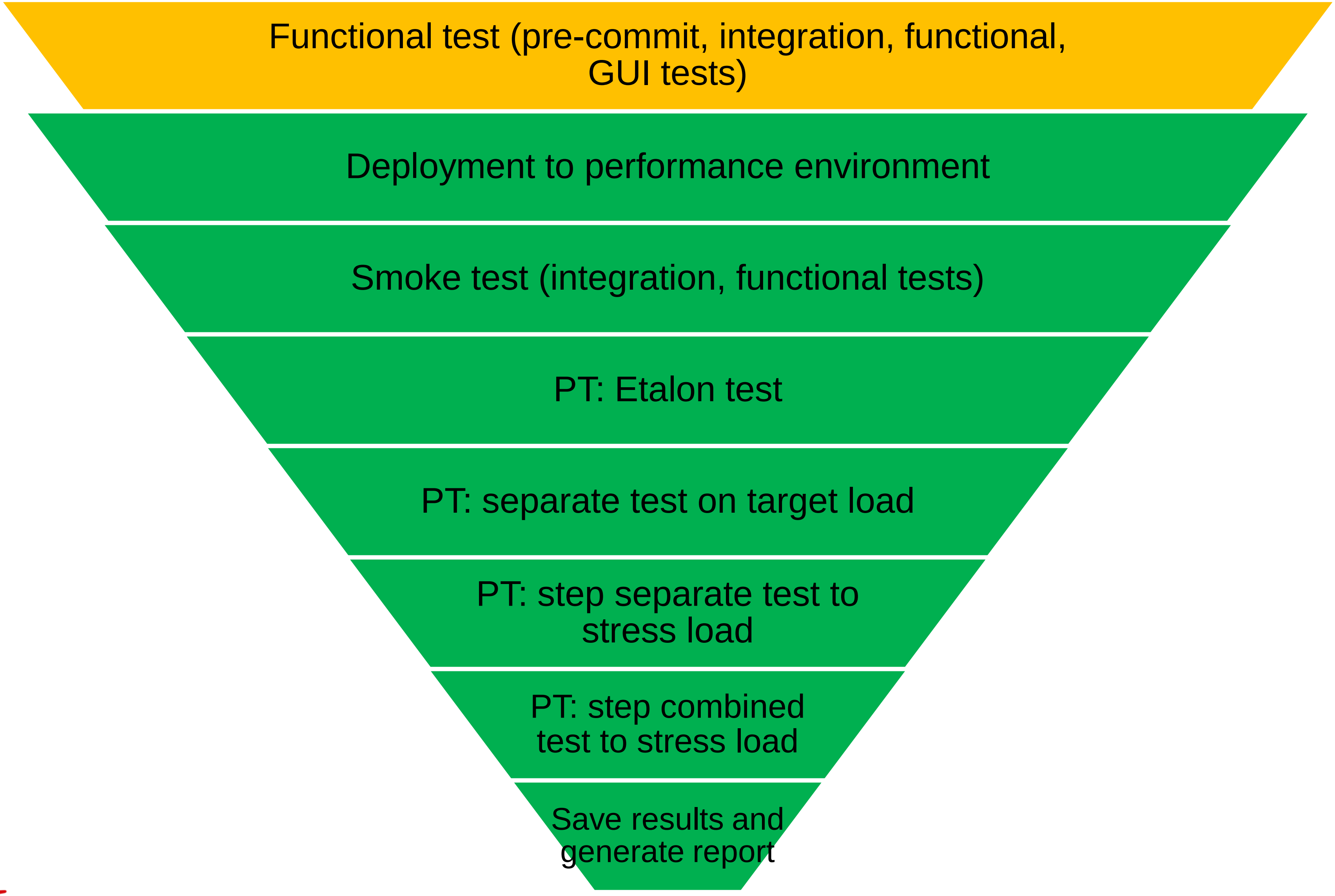


itera
MAKE A DIFFERENCE

Continuous performance testing



Continuous performance testing



Functional test (pre-commit, integration, functional, GUI tests)

Deployment to performance environment

Smoke test (integration, functional tests)

PT: Etalon test

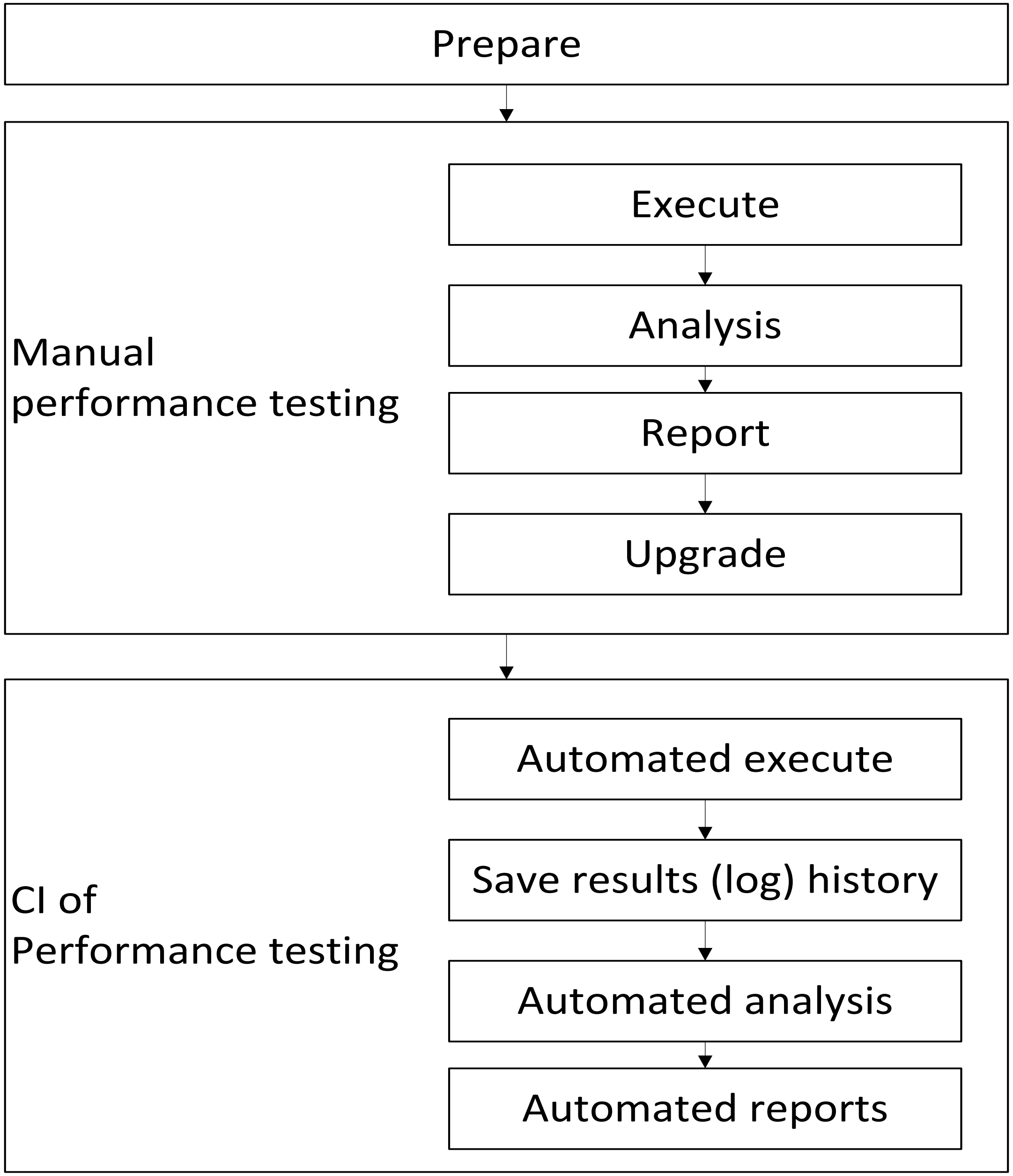
PT: separate test on target load

PT: step separate test to stress load

PT: step combined test to stress load

Save results and generate report

Evolution Continuous performance testing



Prepare

Manual
performance testing

Execute

Analysis

Report

Upgrade

CI of
Performance testing

Automated execute

Save results (log) history

Automated analysis

Automated reports

Evolution Continuous performance testing

- Hardcoded scripts
- Manual executions
- Manual (eye analysis)
- Manual checks
- Manual report generators
- Configured scripts
- Configured runs
- Automatic executions
- Configured tests
- Results saving
- Automatic results analysis
- Automatic report generation
- Automatic anomaly detection
- Automatic recommendation generation

Continuous PT

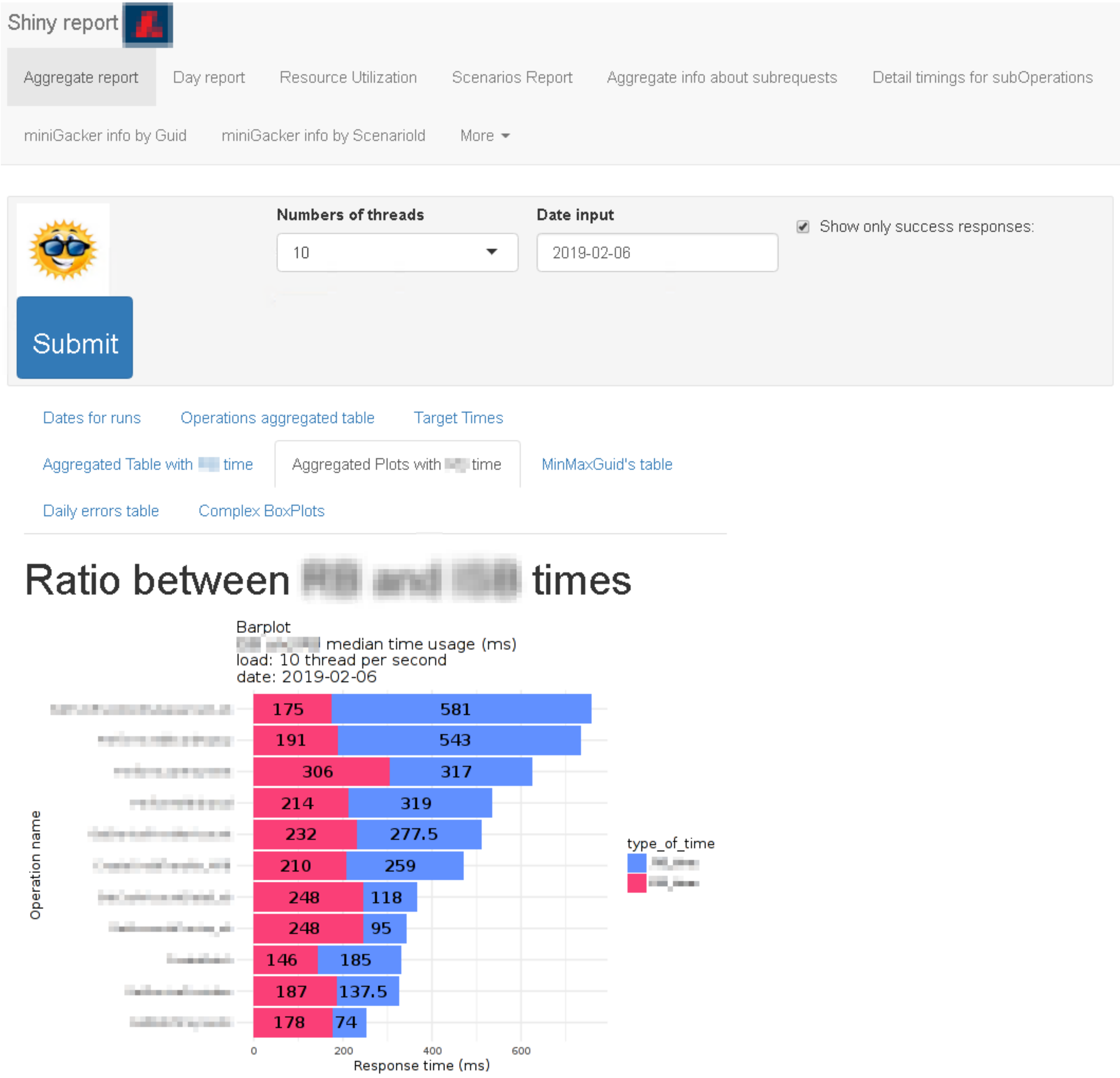
Report Generation

Machine learning in PT

Recommendation for bottleneck detection

PT fairy ~~fails~~ tales

Report generation



Shiny report
Aggregate report
Day report
Resource Utilization
Scenarios Report
Aggregate info about subrequests
Detail timings for subOperations
miniGacker info by Guid
miniGacker info by ScenarioId
More
Submit
Dates for runs
Operations aggregated table
Target Times
Aggregated Table with [redacted] time
Aggregated Plots with [redacted] time
MinMaxGuid's table
Daily errors table
Complex BoxPlots

Numbers of threads: 10

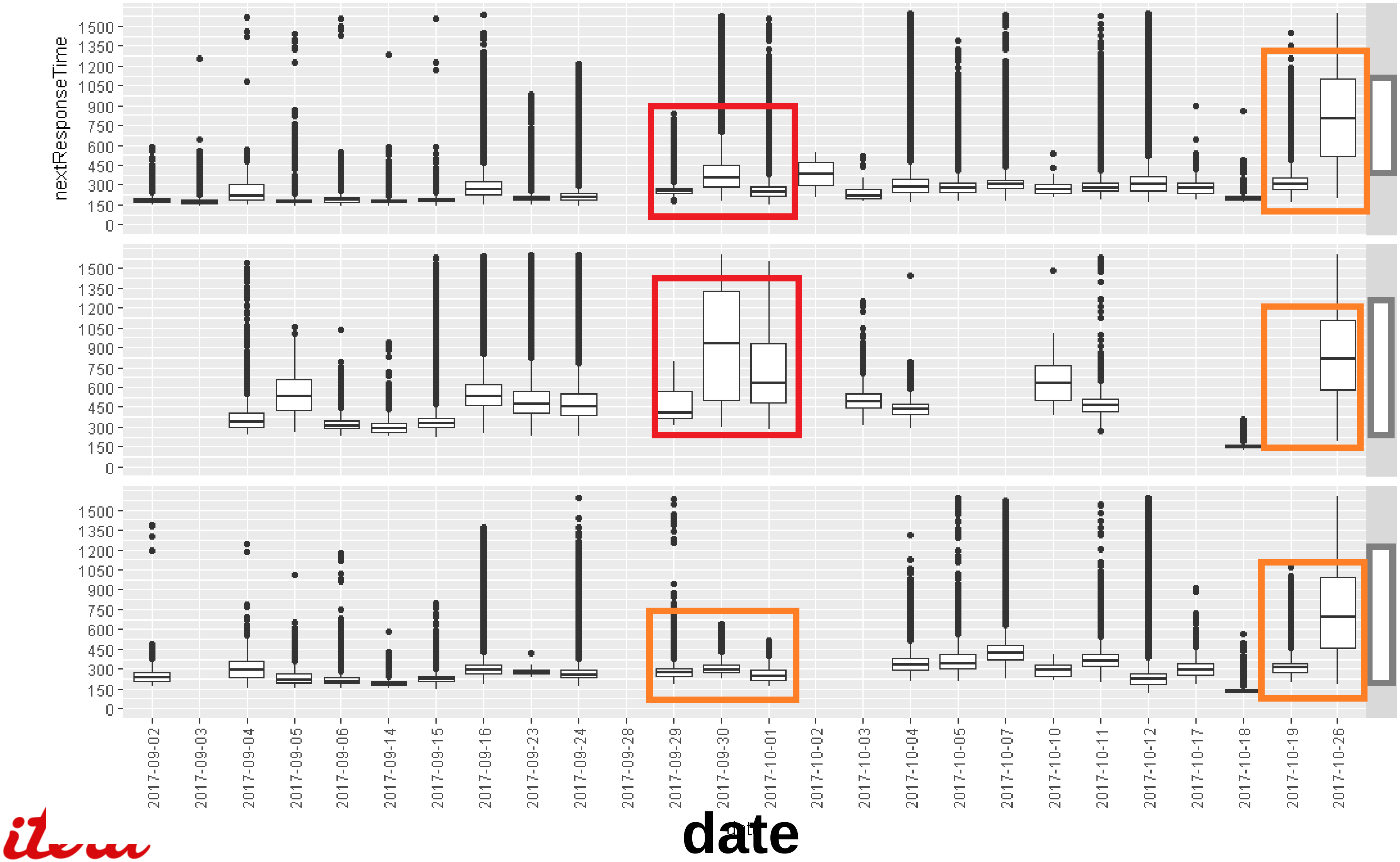
Date input: 2019-02-06

☑ Show only success responses:

Ratio between [redacted] times

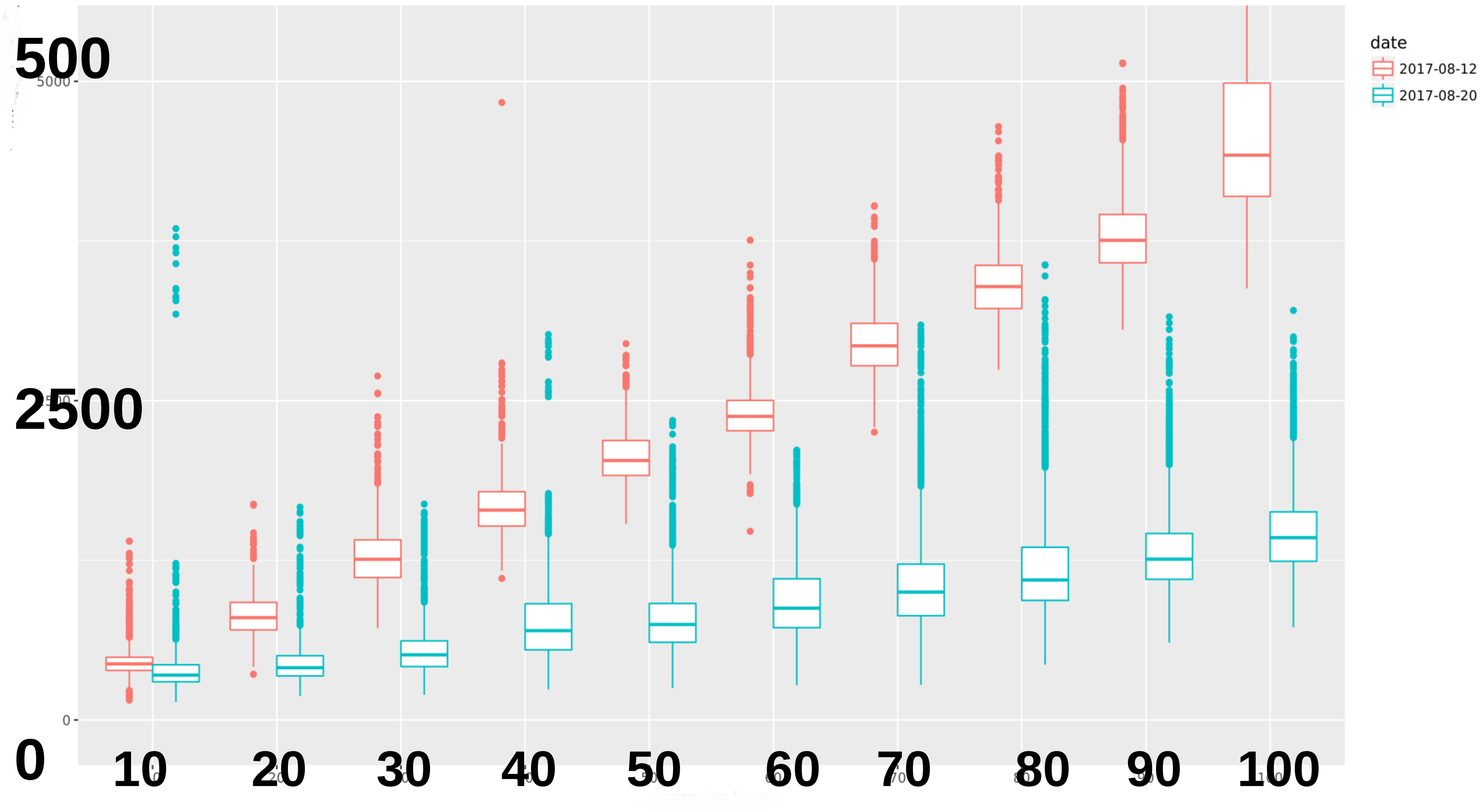
Barplot
[redacted] median time usage (ms)
load: 10 thread per second
date: 2019-02-06

Report generation



MAKE A DIFFERENCE

Report generation



Report generation

- Based on typical log files reports should be divided on next parts:
- Summary report – show only things which go bad
- Aggregated report -- describe all measurements with all statistics and graphs
- Interactive report – should give full access to all historical data with possibility to compare, filter, split and unite different data types and measurements.
  - It should be some application with visualization of data
  - Should provide access to all history of data
  - Should provide access for all involved person
  - Should provide export raw data for analysis in external tools
  - Should be possibility to compare results and measurement between dates, loads, layers and others….

Continuous PT

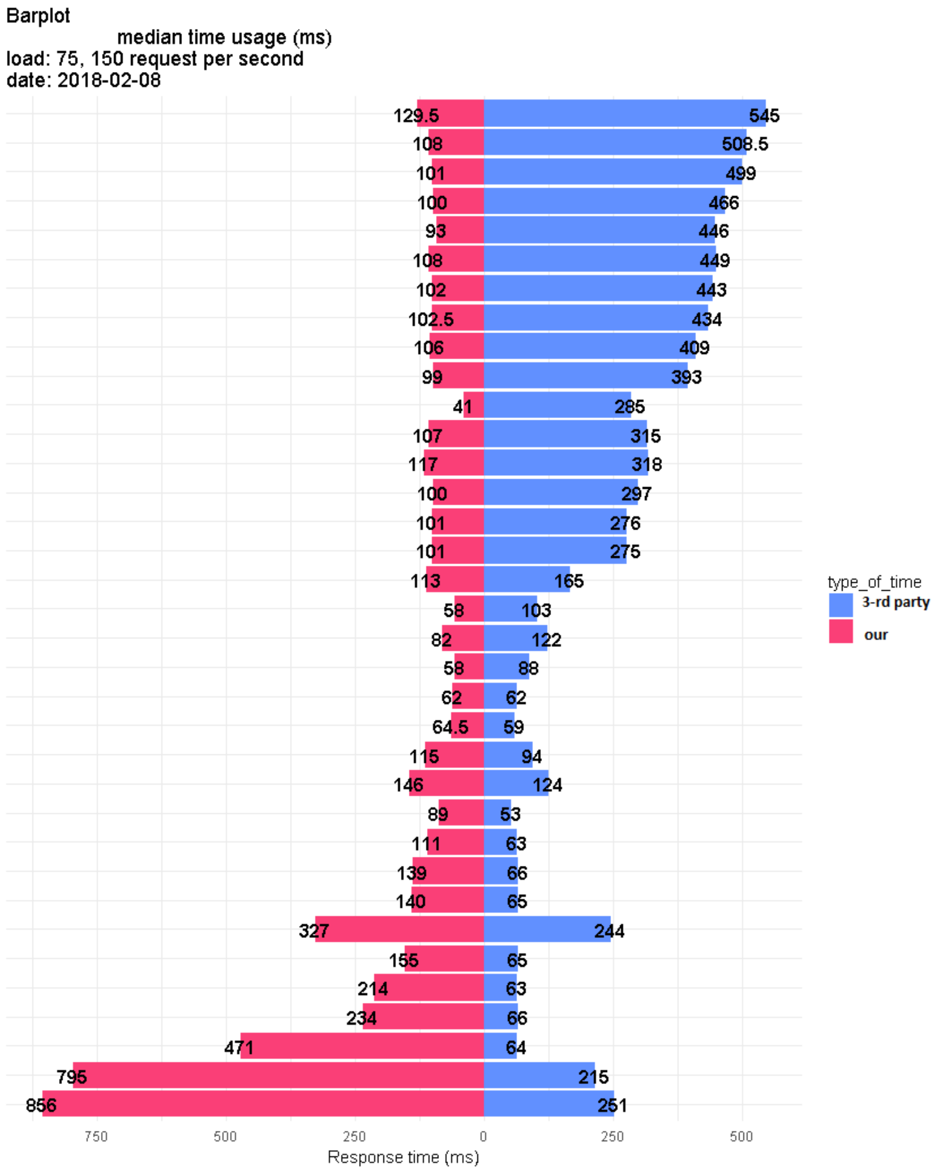
Report Generation

Machine learning in PT

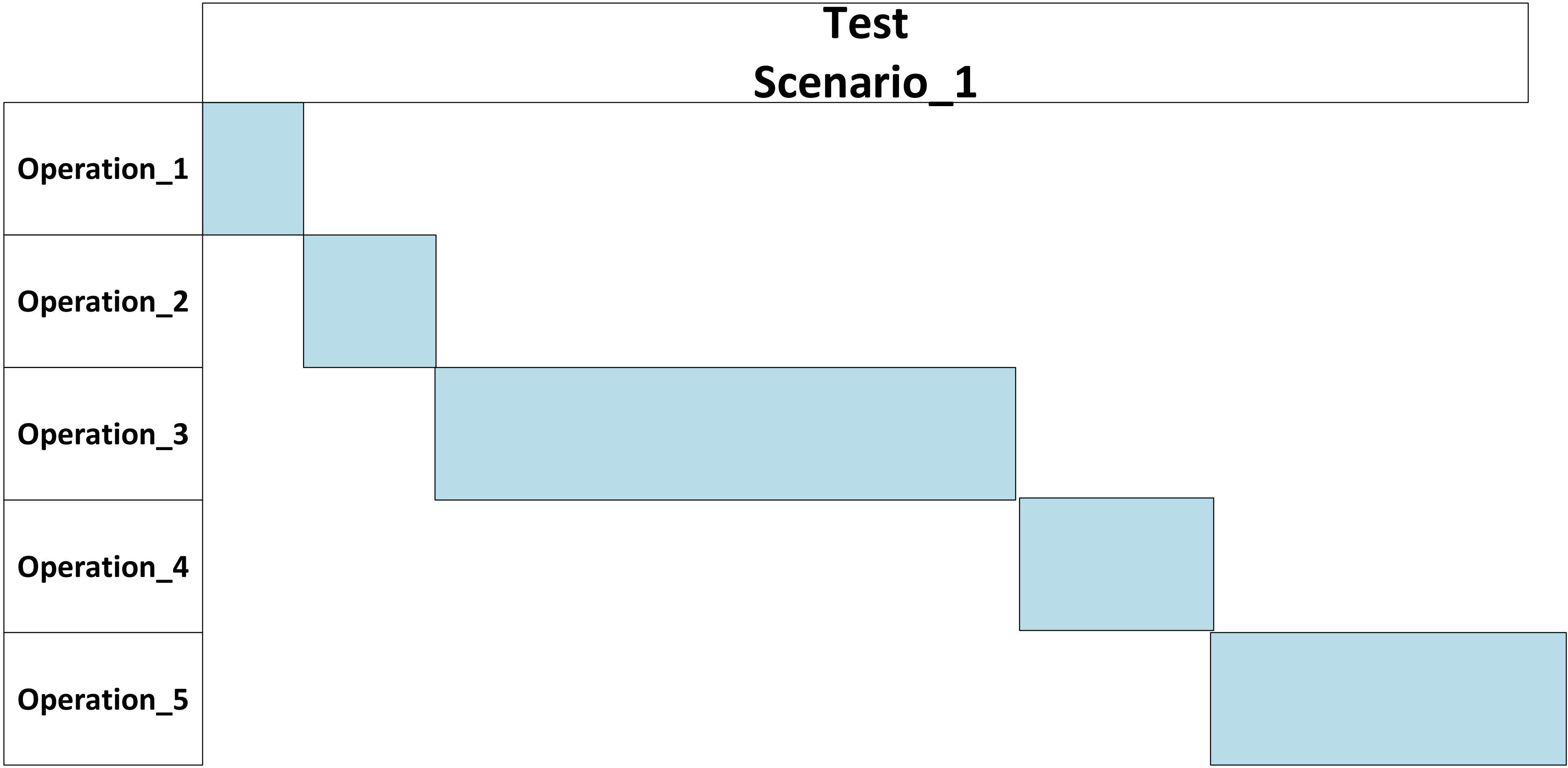
Recommendation for bottleneck detection

PT fairy ~~fails~~ tales

Recommendation for bottleneck detections



Recommendation for bottleneck detections



Test Scenario_1

Operation_1

Operation_2

Operation_3

Operation_4

Operation_5

Recommendation for bottleneck detections

	Test Scenario_1
Operation_1	0.1
Operation_2	0.15
Operation_3	0.95
Operation_4	0.2
Operation_5	0.7

Recommendation for bottleneck detections

	Test Scenario_1
Operation_1	0.1
Operation_2	0.15
Operation_3	0.95
Operation_4	0.2
Operation_5	0.7

Recommendation for bottleneck detections

	Test Scenario_1	Test Scenario_2	Test Scenario_3	Test Scenario_4	Test Scenario_5
Operation_1	0.1	0.99	0.2	0.99	0.3
Operation_2	0.15	0.9	0.25	0.1	0.2
Operation_3	0.95	0.1	0.95	0.3	0.86
Operation_4	0.2	0.5	0.75	0.4	0.7
Operation_5	0.7	0.1	0.99	0.77	0.25

Old build

	REST Response time	SOAP Response time	SQL Response time	CPU usage	RAM usage
REST Response time	▨	**0.75**	0.6	**0.7**	0.3
SOAP Response time		▨	**0.8**	**0.78**	0.2
SQL Response time			▨	0.4	0.2
CPU usage				▨	0.32
RAM usage					▨

−

New build

	REST Response time	SOAP Response time	SQL Response time	CPU usage	RAM usage
REST Response time	▨	**0.73**	0.6	**0.7**	0.3
SOAP Response time		▨	**0.9**	**0.58**	0.2
SQL Response time			▨	0.4	0.5
CPU usage				▨	0.32
RAM usage					▨

=

Difference

	REST Response time	SOAP Response time	SQL Response time	CPU usage	RAM usage
REST Response time	▨	0.02	0	**0**	0
SOAP Response time		▨	**-0.1**	**0.2**	0
SQL Response time			▨	0	**-0.3**
CPU usage				▨	0
RAM usage					▨

itera
MAKE A DIFFERENCE

Recommendation for bottleneck detections

- To decrease time for bottleneck detection it is necessary gather information about all layers and all resources.
- Next it is necessary build correlation matrix predictor and predicted variables.
- Next step it is visualize it in heat map – and it shows what impact more on dependent variable.
- It is easiest way to drop all unimportant variables in variety of variables.
- For detecting what was changes with previous releases it just needed count differences between previous matrix and newest

Questions

Presentation Here