

Quality assessment in DevOps:

Automated Analysis of a Tax Fraud Detection System

Diego Perez-Palacin, Youssef Ridene, José Merseguer

University of Zaragoza, Nefective Technology



DICE Developing Data-Intensive Cloud Applications
With Iterative Quality Enhancements

Big Blu

**eGov Tax Fraud Detection System
Under Development by Netfective Technology**



Big Blu

eGov Tax Fraud Detection System Under Development by Netfective Technology



- ◆ Tax fraud represent a huge problem for governments.



https://ec.europa.eu/taxation_customs/fight-against-tax-fraud-tax-evasion/missing-part_en

- ◆ EU has estimated tax evasion to be of the order of 1 trillion euros

Big Blu

Big Blu is a Data Intensive Application in the field of *Big Data*

- **Volume:** millions of daily tax operations – combined with history
- **Variety:** data coming from different sources:
 - VAT
 - Banks
 - Incomes
 - ...
- **Velocity:** data arrives too fast in some periods of time (tax declaration periods)
- **Variability and Volatility:** data changes fast:
 - Relocations of people
 - Updated banking details
- **Veracity, Validity and Value:** data contains noise such as legitimate exemptions

Big Blu

Big Blu is developed following Agile and DevOps principles

- Iterative process with incremental iterations pursuing
 - Quick design
 - Quick delivery of enhancements
 - Quick feedback

- Bring closer Development and Operations activities to improve the effectiveness of each iteration
 - Faster iterations since both activities share the same UML models
 - Higher proportion of iterations achieve satisfactory results

Big Blu

- ◆ Software Architecture composed of 3 main layers:
 - GUI: web based application. Unique interface
 - Web Services: implement RESTful interoperability and deployed on Tomcat
 - Back-end: Data processing elements

Big Blu

◆ Software Architecture composed of 3 main layers:

- GUI: web based application. Unique interface

The image displays two screenshots of the Big Blu web application. The top screenshot shows the home page with a welcome message and a list of bullet points describing the system's capabilities. The bottom screenshot shows the 'Frauds Detection' page, which includes a table of detection records and a 'Kill Job' button for each record.

Home

Welcome to Big Blu : an eGov Tax Fraud Detector

Big Blu is designed and developed in the context of the DICE project (<http://www.dice-h2020.eu>). Big Blu is an MVP to prove the capabilities of Big Data frameworks in the context of governmental applications.

Tax evasion and frauds represents a huge problem for governments, causing them a big loss of money each year. Governments are increasingly using Big Data in multiple sectors to help their agencies manage their operations, and to improve the services they provide to citizens and businesses. In this case, Big Data has the potential to make tax agencies faster and more efficient. However, detecting fraud is actually a difficult task because of the high number of tax operations performed each year and the differences inherent in the way the taxes are calculated. NETF plans to build a software prototype to demonstrate the capabilities of Big Data in e-government. Our demonstrator aims to build a model of "fraudulent conduct" from the automatic operations on existing tax data files such as business creation.

- Identifying taxpayers who are registered in different regions in order to collect fraudulently social money.
- The fictitious relocation of the taxpayer who improperly claim domiciled abroad in order to not pay taxes.
- Companies normally collect VAT from their customers but "forget" to pay back to the Treasury a companies do not hesitate to include the VAT debt liabilities in their balance sheet, which proves they are not paying.
- We may even consider detecting ID tax fraud since it has been the most attractive type of identity fraud.

The demonstrator will facilitate the task of filtering and gathering data for fiscal agents in order to increase the efficiency of the tax collection process. Exploring and analyzing high volumes of data from various heterogeneous sources should be scalable and efficient. This demonstrator will avoid any privacy/confidentiality issues since processed random data which are generated specifically for being realistic (features could apply to a real system), generic (features and data model are not specific to a particular country) and the data generator produces real data.

This demonstrator will avoid any privacy/confidentiality issues since processed random data which are generated specifically for being realistic (features could apply to a real system), generic (features and data model are not specific to a particular country) and the data generator produces real data.

© 2016 Big Blu

Frauds Detection

Successfully submitted a new detection with the id **detection20170124171050420**

Detection ID	Launch Date	Database	Indicators	Status	Actions
detection20170124171007574	24/01/2017 17:10:07	database201701117164508855	F11[Income decrease = 23]	RUNNING	Kill Job
detection20170124171050420	24/01/2017 17:10:50	database201701117154626276	F11[Income decrease = 23]	SUBMITTED	Kill Job

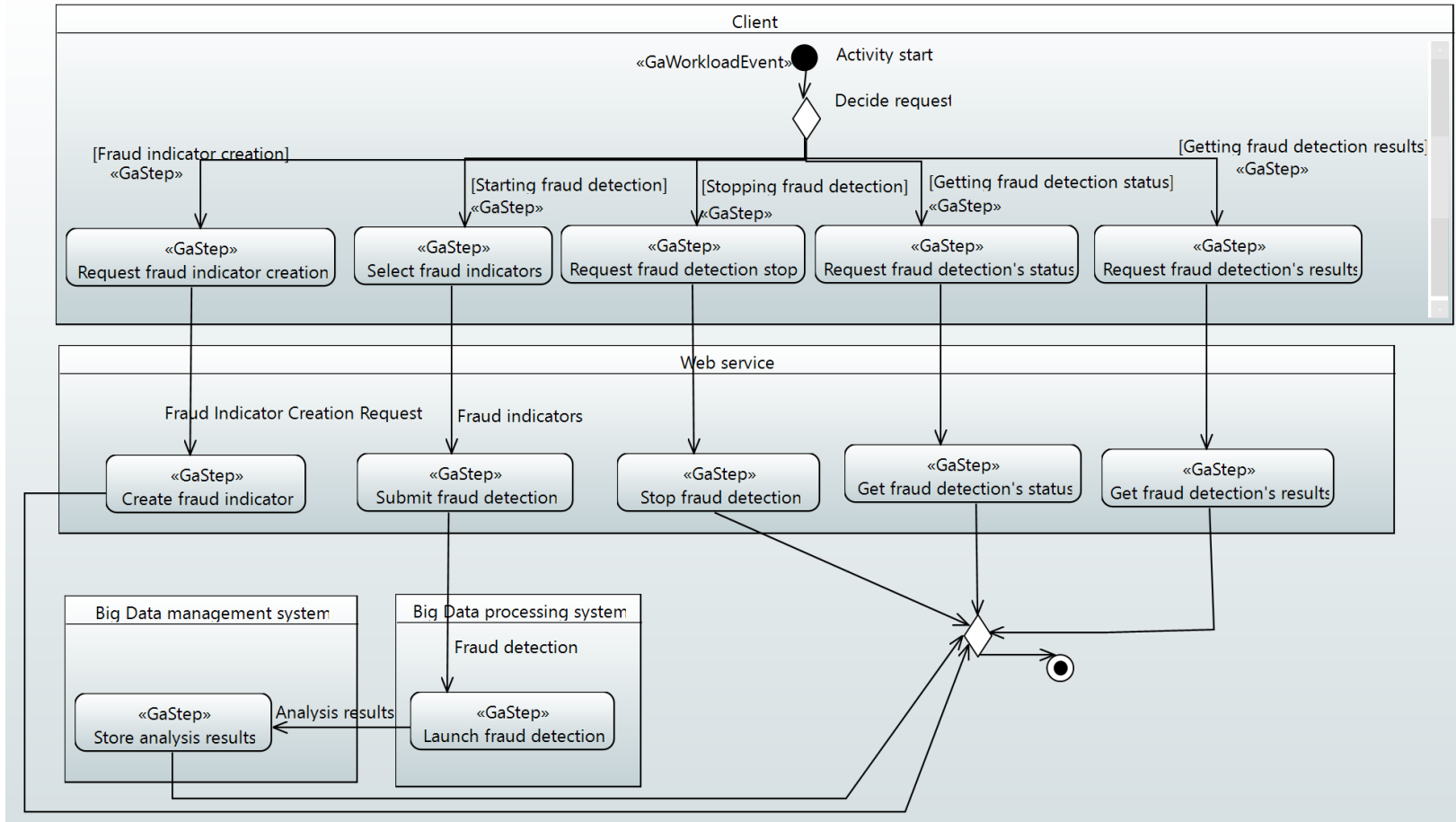
operability and deployed

Big Blu

- ◆ Software Architecture composed of 3 main layers:
 - GUI: web based application. Unique interface
 - Web Services: implement RESTful interoperability and deployed on Tomcat
 - Back-end: Data processing elements

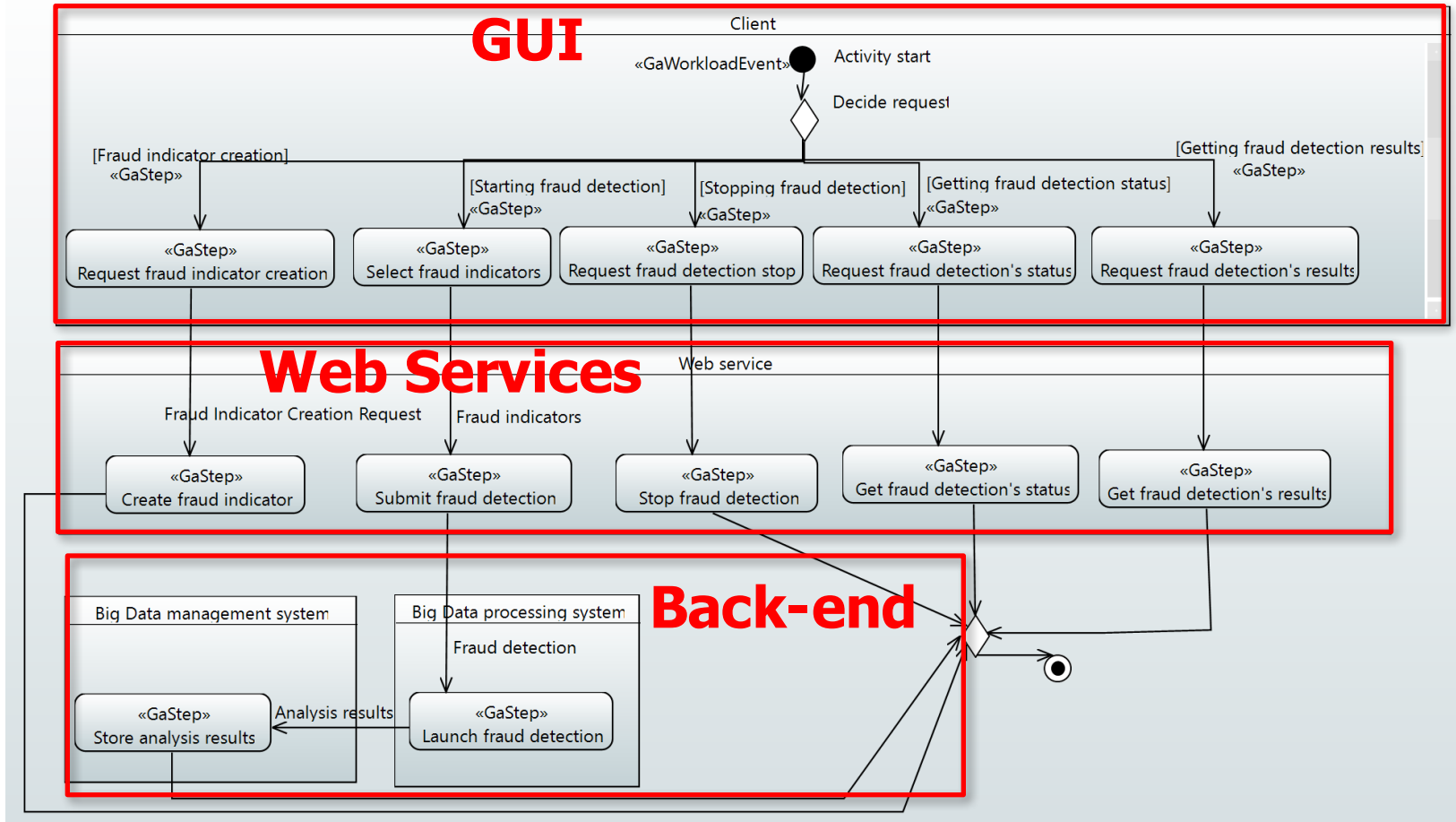
Big Blu

- ◆ Software Architecture composed of 3 main layers:



Big Blu

- ◆ Software Architecture composed of 3 main layers:



DICE approach



Researches towards building an quality-driven framework for development, deployment, monitoring and continuous improvement of Data-Intensive Cloud Applications.

- ◆ Pursues developments through Iterative Quality enhancements

- ◆ Delivers a toolchain for DevOps:
 - Design
 - Quality analysis
 - Deployment
 - Testing
 - Monitoring (collect data, visualization, anomaly detection, trace checking)
 - Enhancement

DICE approach



Researches towards building an quality-driven framework for development, deployment, monitoring and continuous improvement of Data-Intensive Cloud Applications.

◆ Pursues developments through Iterative Quality enhancements

◆ Delivers a toolchain for DevOps:

- Design
- Quality analysis
- Deployment
- Testing
- Monitoring (collect data, visualization, anomaly detection, trace checking)
- Enhancement



DICE approach



Researches towards building an quality-driven framework for development, deployment, monitoring and continuous improvement of Data-Intensive Cloud Applications.

◆ Pursues developments through Iterative Quality enhancements

◆ Delivers a toolchain for DevOps:

- Design
- **Quality analysis**
- Deployment
- Testing
- Monitoring (collect data, visualization, anomaly detection, trace checking)
- Enhancement

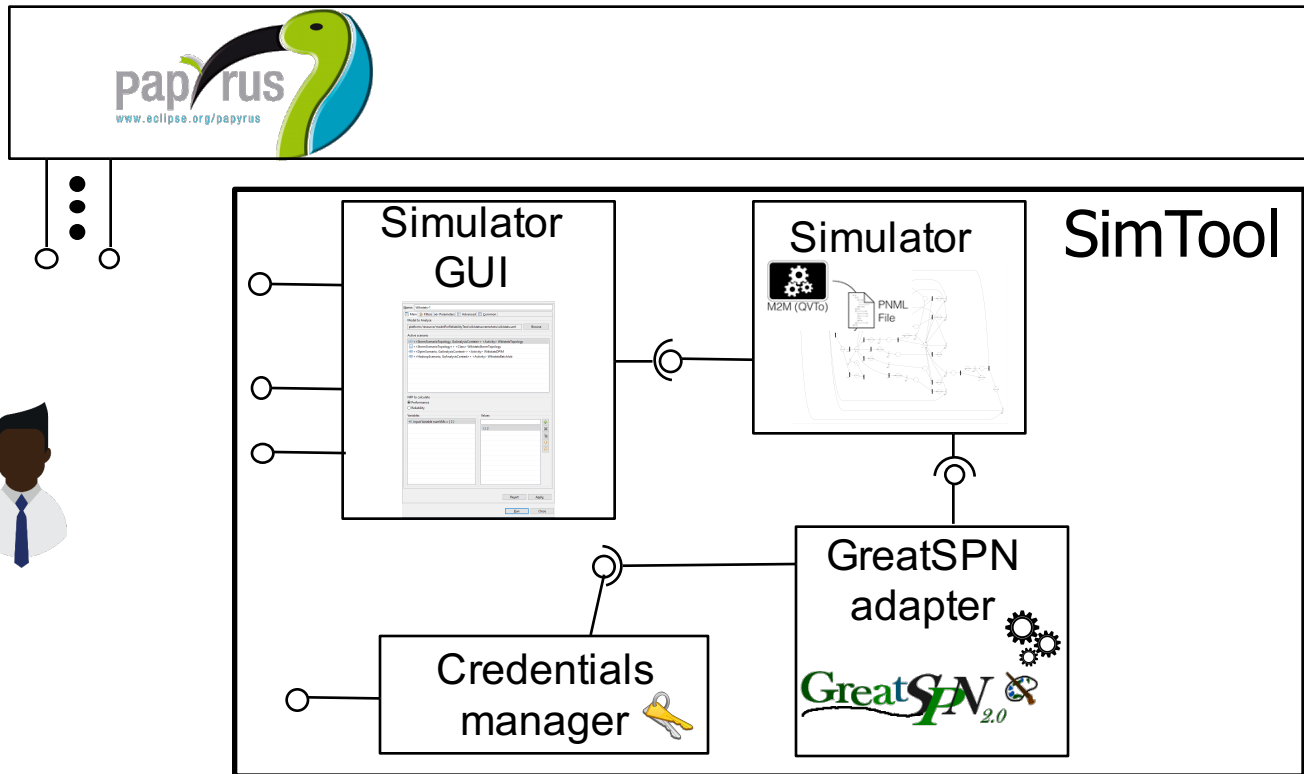


DICE Simulation tool

+ PROFILING

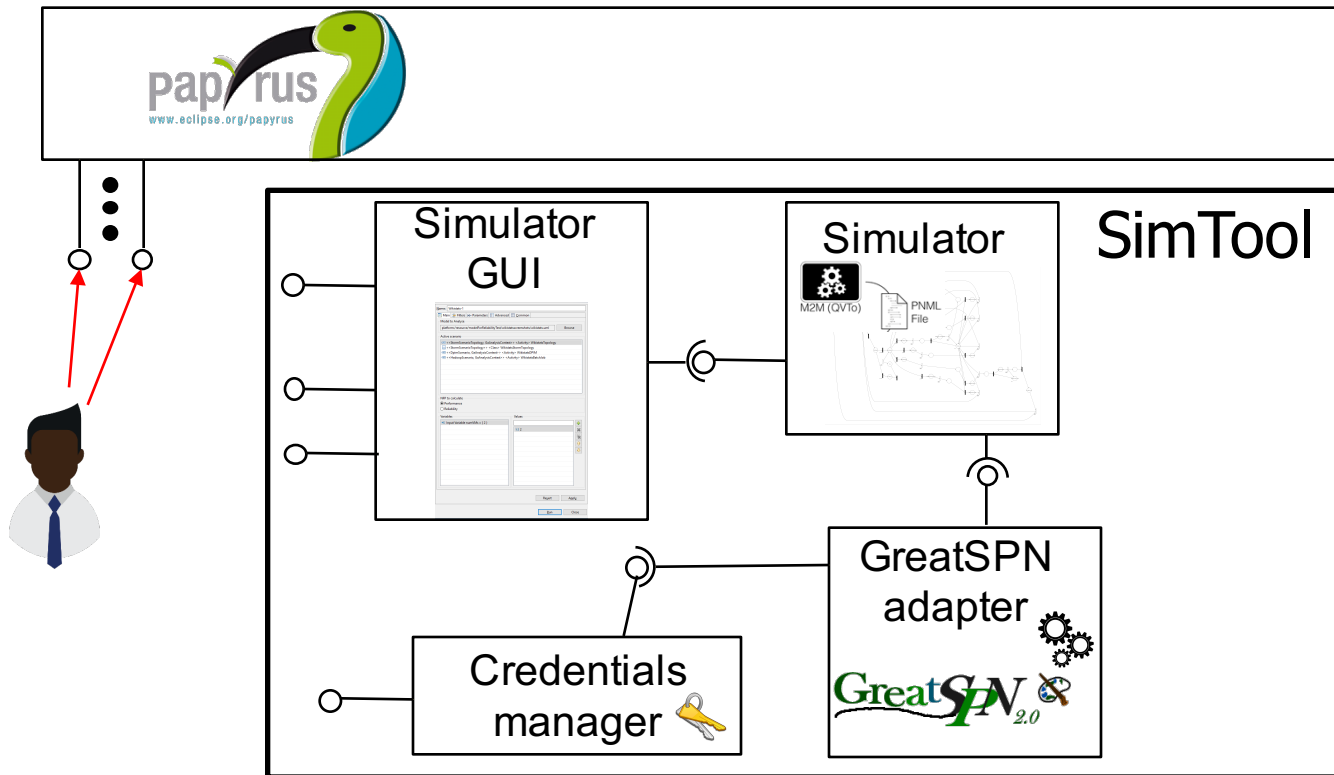
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



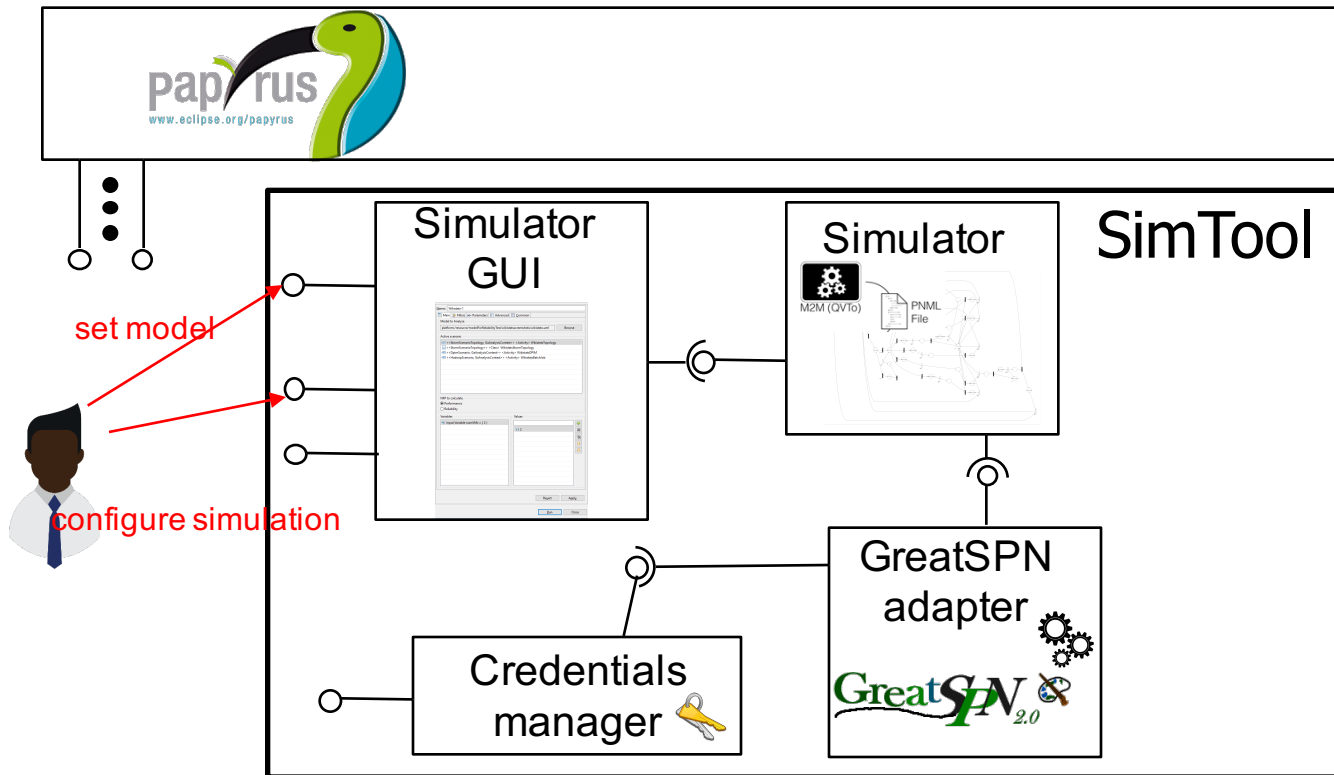
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



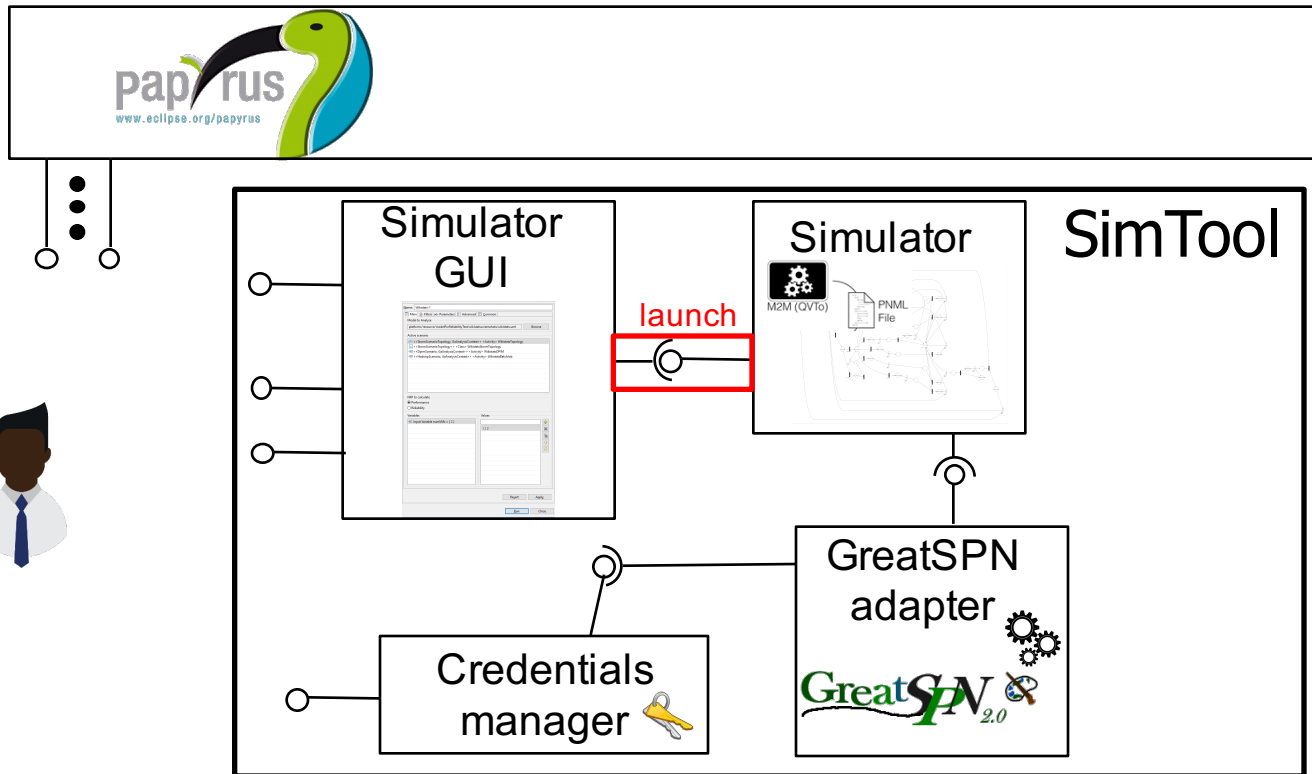
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



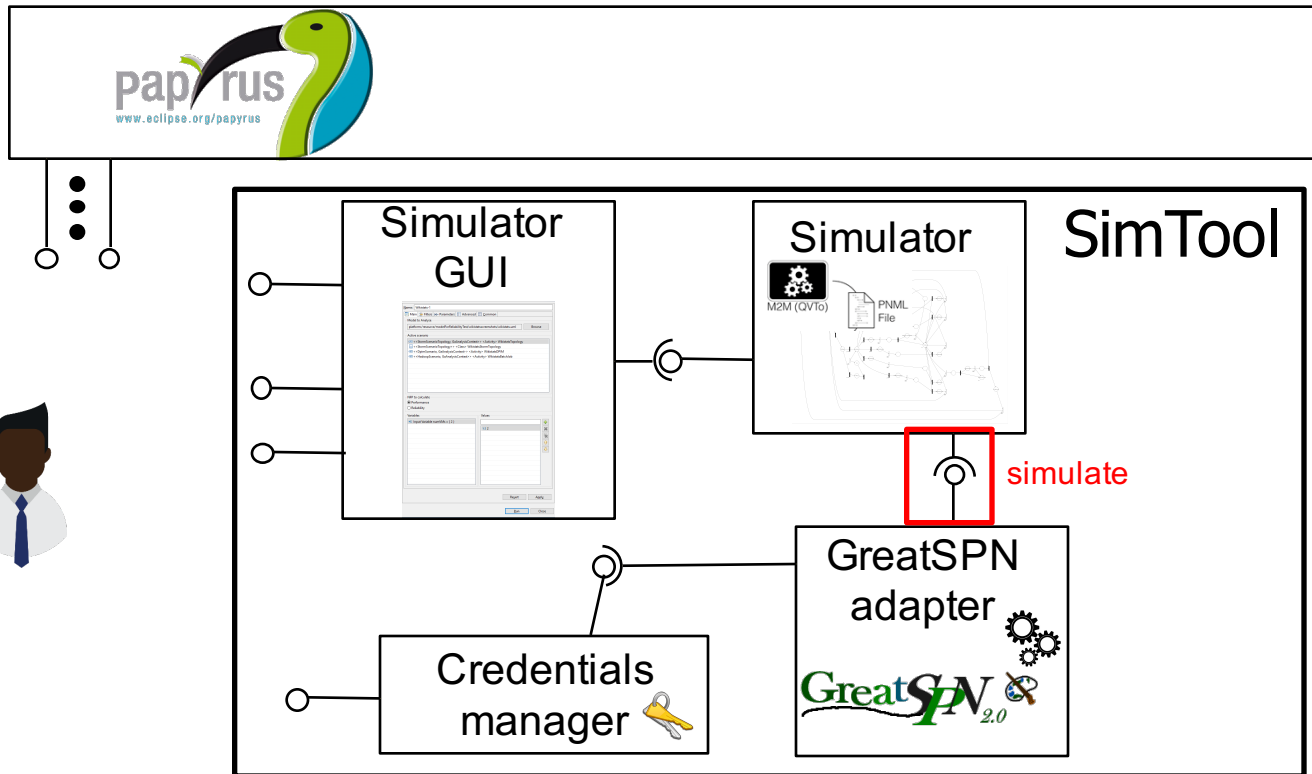
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



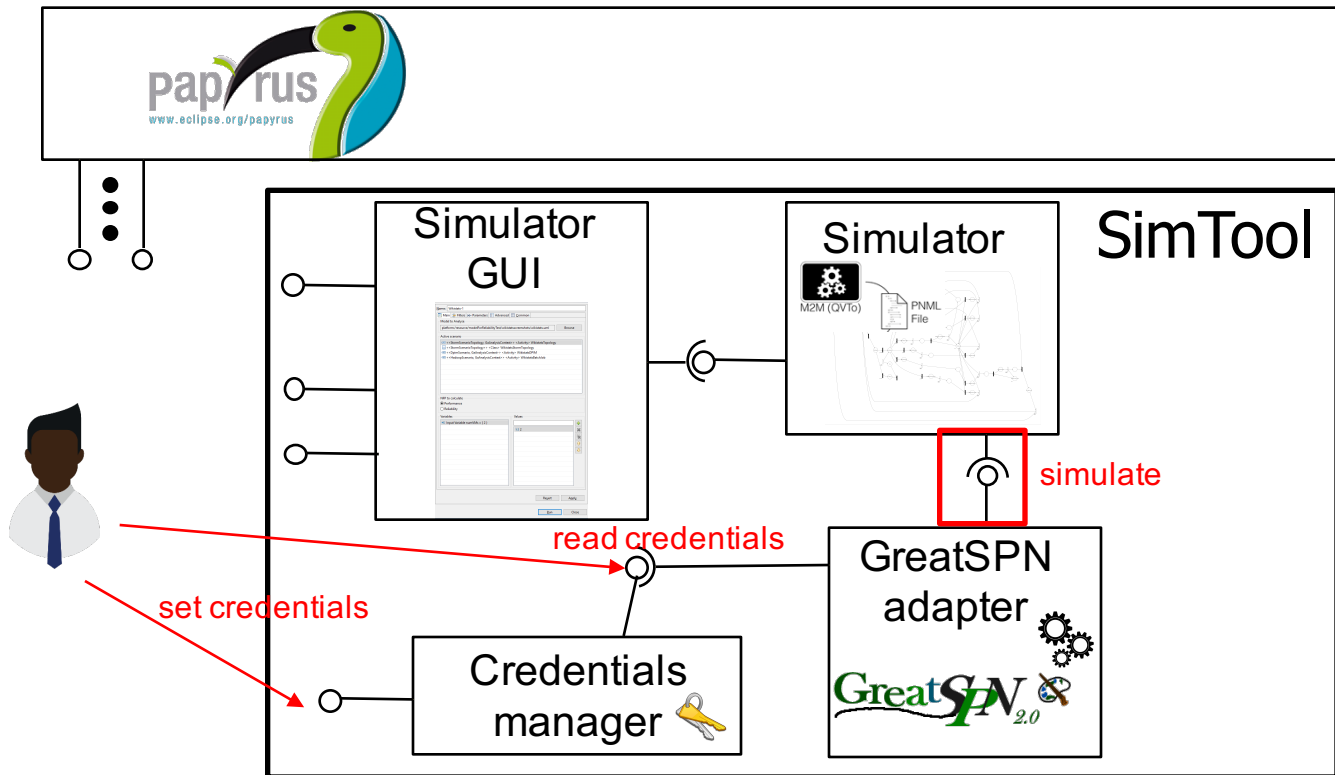
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



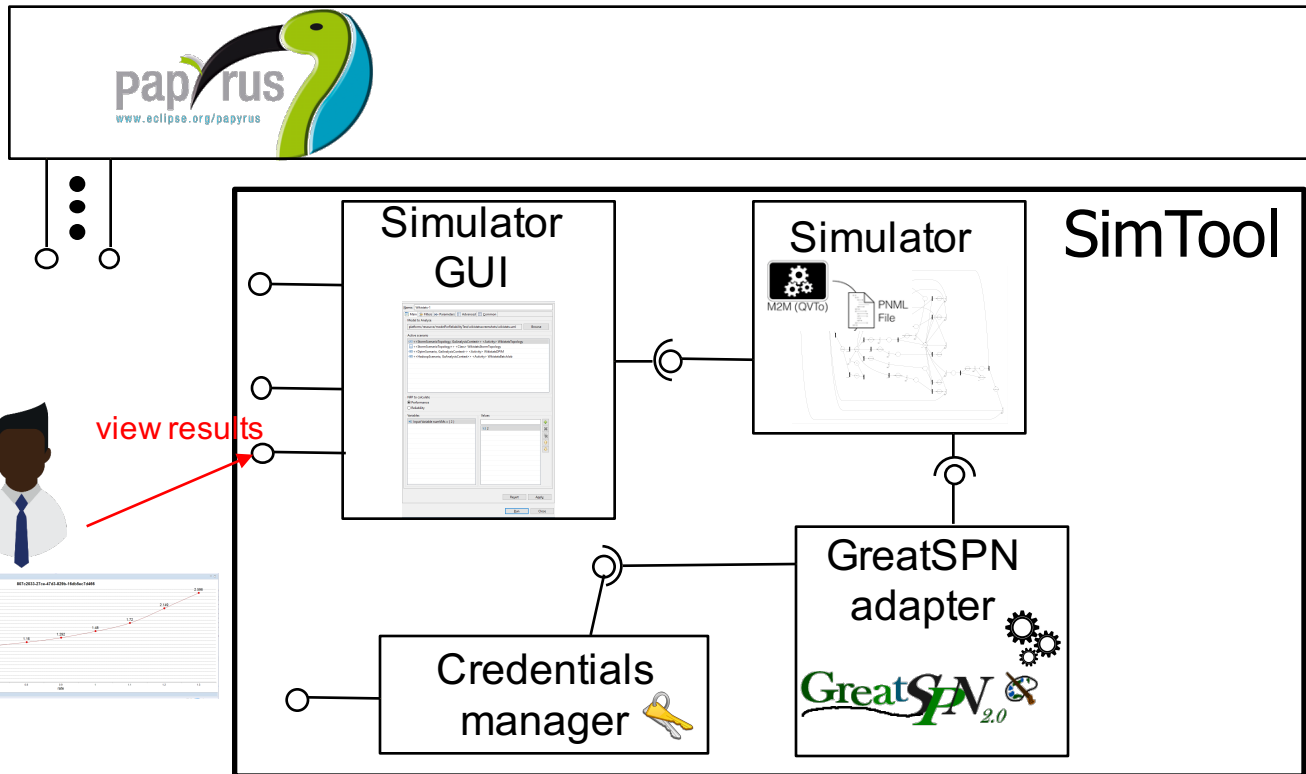
DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



DICE Simulation Tool

- ◆ Based on eclipse plugins  eclipse Delivered with



DICE Simulation Tool

Usefulness in Agile cycles following DevOps

◆ Scenario 1: Development of new functionalities

PROBLEM

- In agile cycles, the required quality of the new functionalities may not be clear for developers
 - The quality requirements refer to the overall system qualityMonitored information is available

CONSEQUENCES

- Obtained quality is not good enough and development cycle has to be repeated

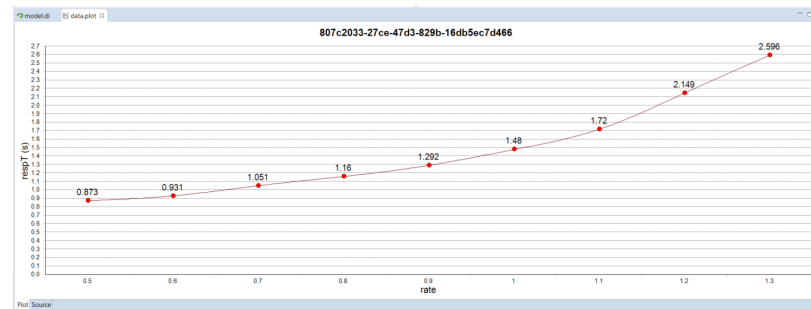
DICE Simulation Tool

Usefulness in Agile cycles following DevOps

◆ Scenario 1: Development of new functionalities

APPROACH TO SOLUTION

- Obtain values for “appropriate quality” of the new functionality that can be already asserted during the unit tests
 - Analyze the expected system quality based on *what-if* values of the quality offered by the new functionality.
E.g., predict system response time considering different resource demands of the new functionality

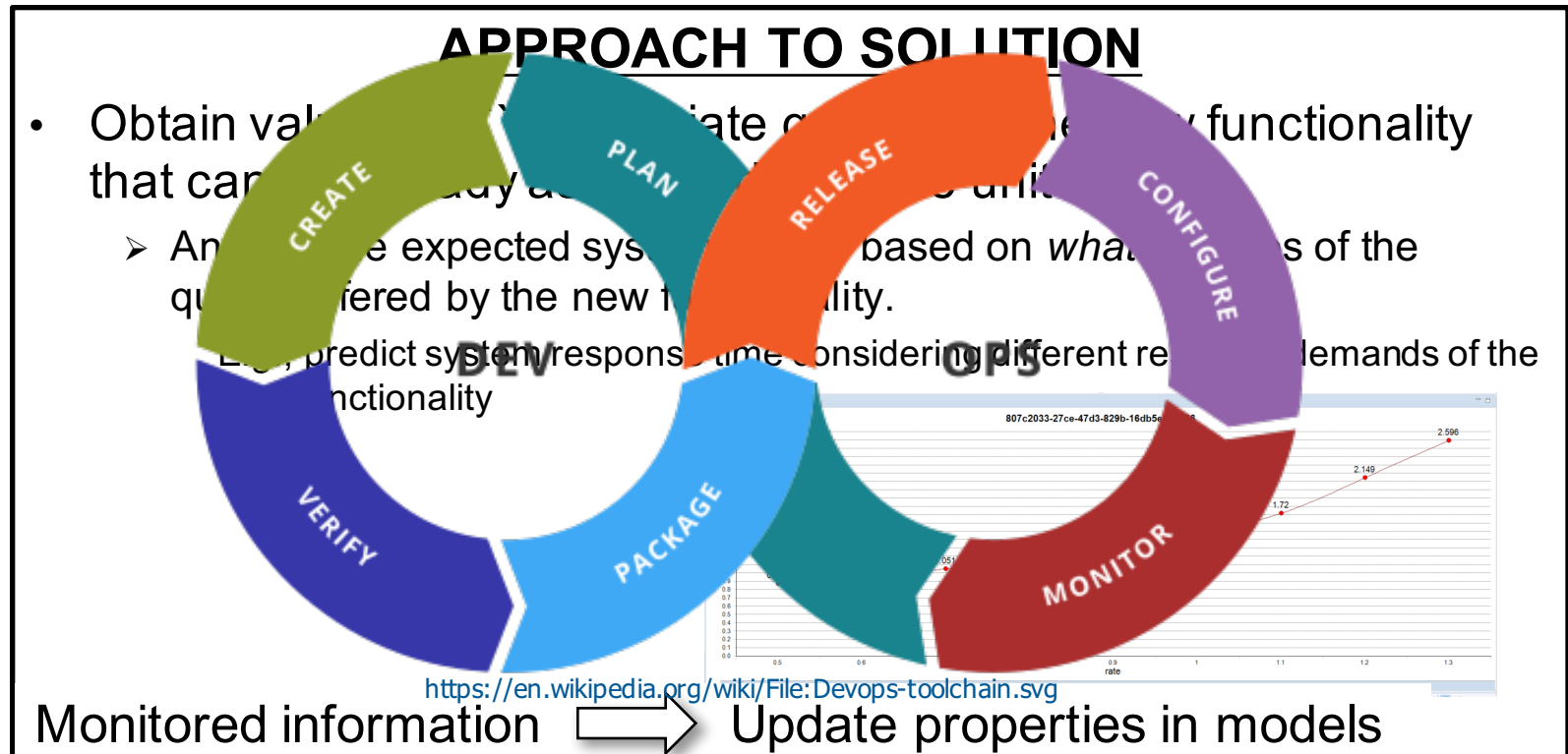


- Deliver a functionality that passes these unit tests

DICE Simulation Tool

Usefulness in Agile cycles following DevOps

◆ Scenario 1: Development of new functionalities



DICE Simulation Tool

Usefulness in Agile cycles following DevOps

◆ Scenario 2: Maintenance of functionalities

PROBLEM

- Quality of a functionality has to be improved...
 - Due to changes in the characteristics or utilization profile
 - Due to improvable designs and new time restrictions
- ...and can be improved in different phases of DevOps toolchain

CONSEQUENCES

- Modifications do not achieve the expected quality
- Modifications result more expensive than necessary

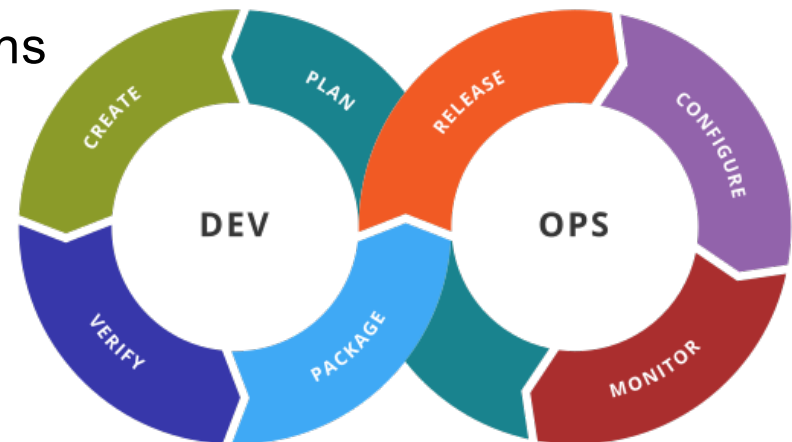
DICE Simulation Tool

Usefulness in Agile cycles following DevOps

◆ Scenario 2: Maintenance of functionalities

APPROACH TO SOLUTION

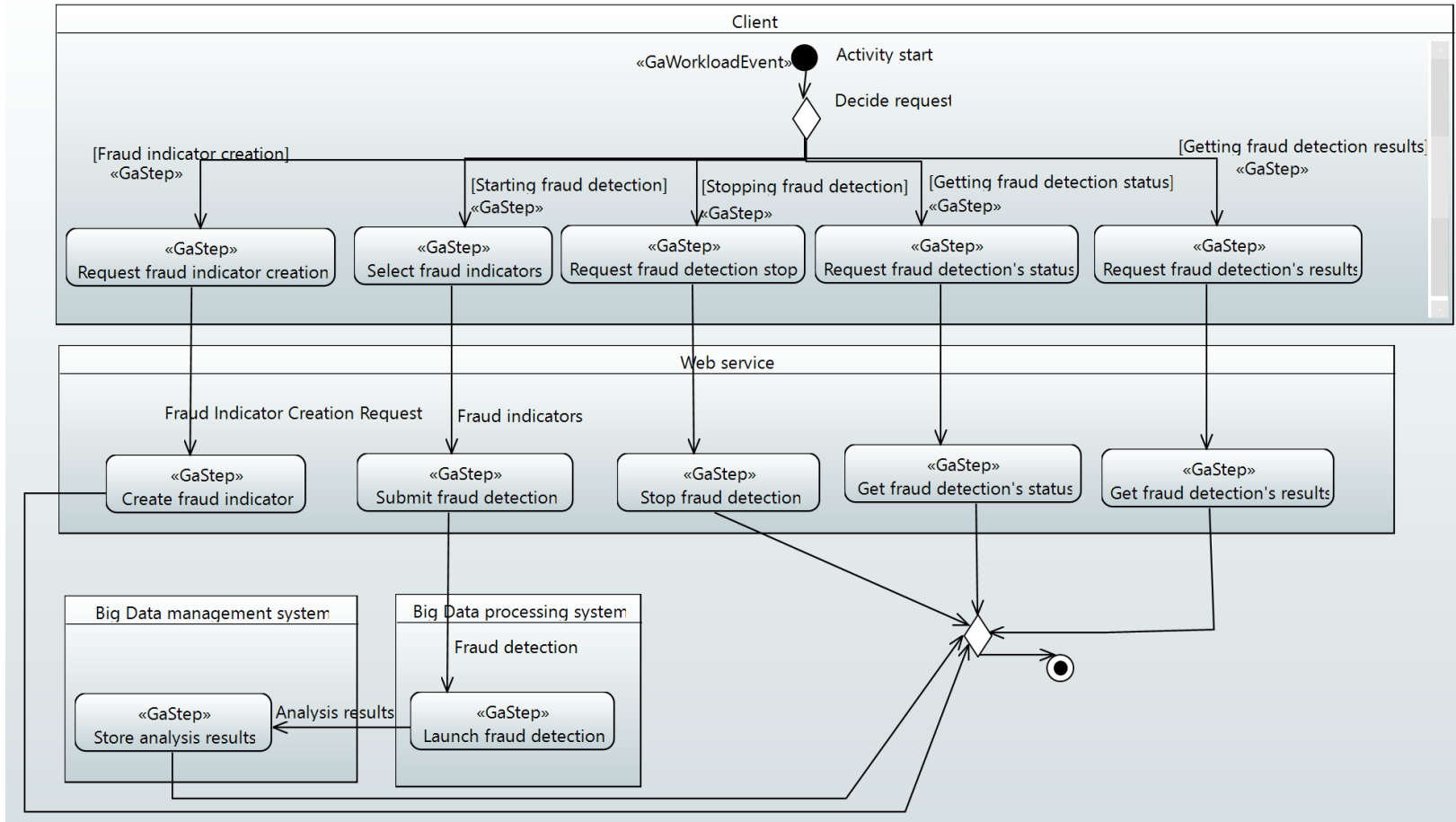
- Update the models with recent monitored data
- Identify quality issues
- Evaluate different possibilities to solve the issues
- Decide best maintenance actions



<https://en.wikipedia.org/wiki/File:Devops-toolchain.svg>

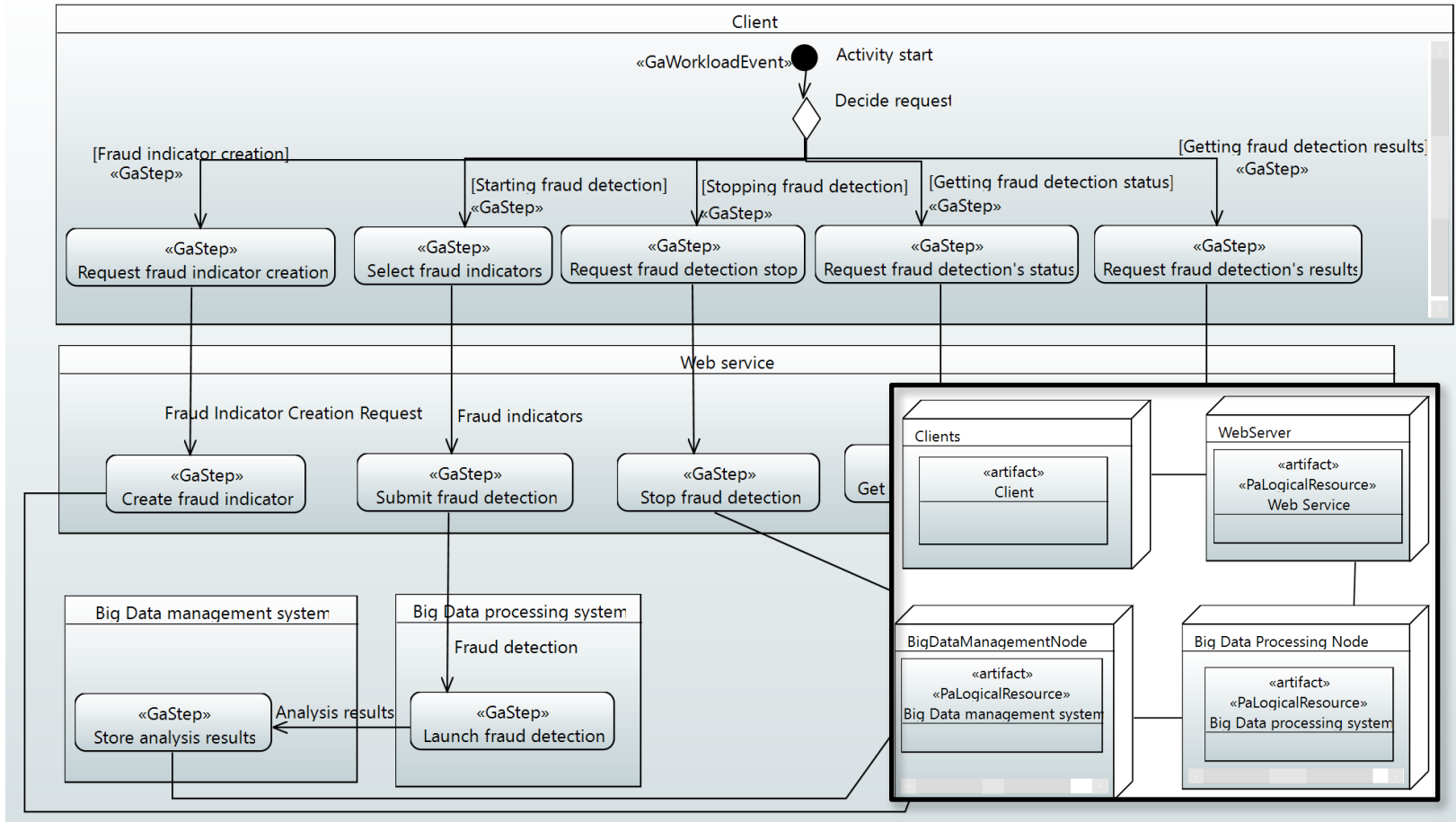
Big Blu Quality assessment

◆ Quality malfunction reported



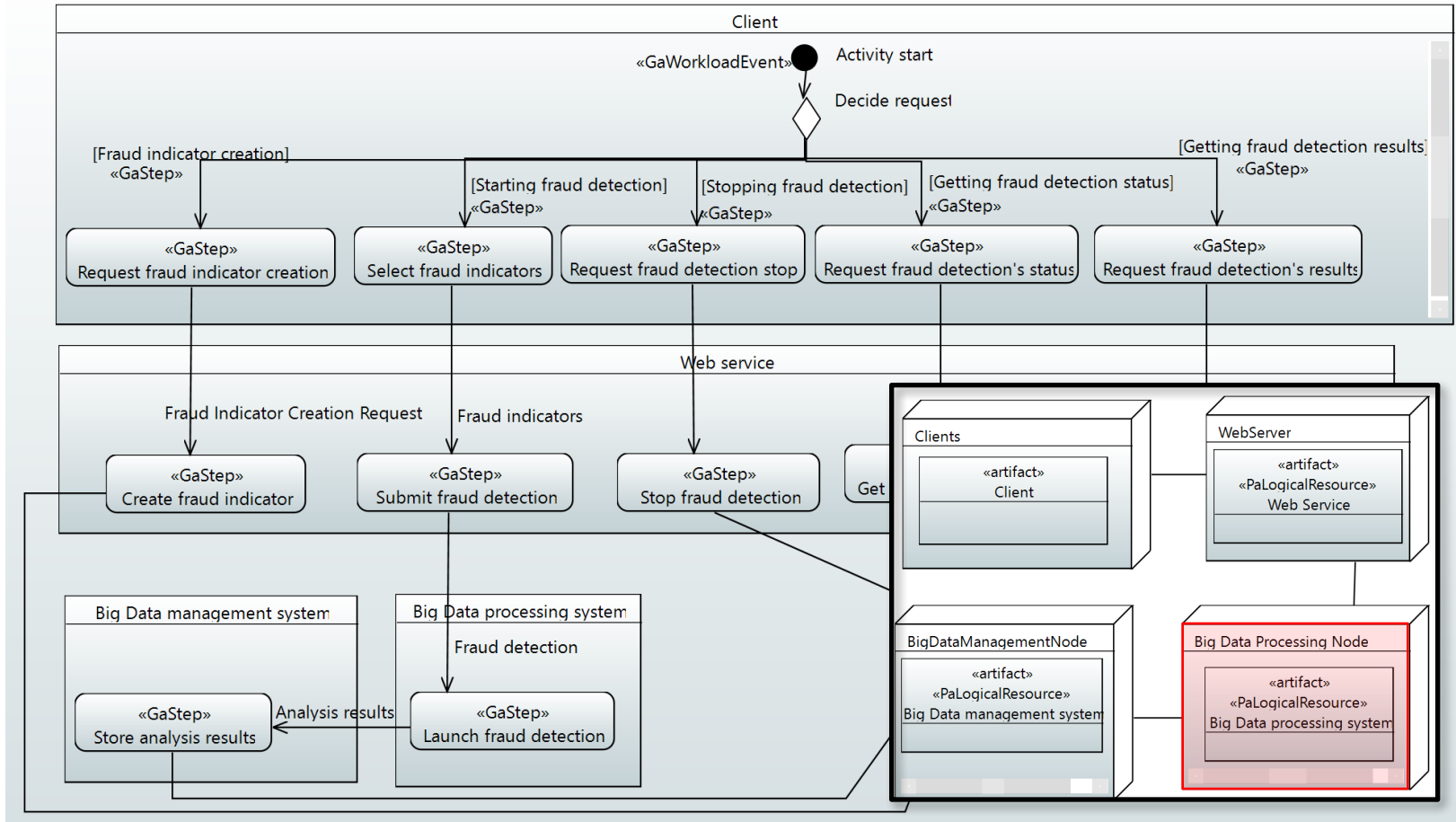
Big Blu Quality assessment

◆ Quality malfunction reported



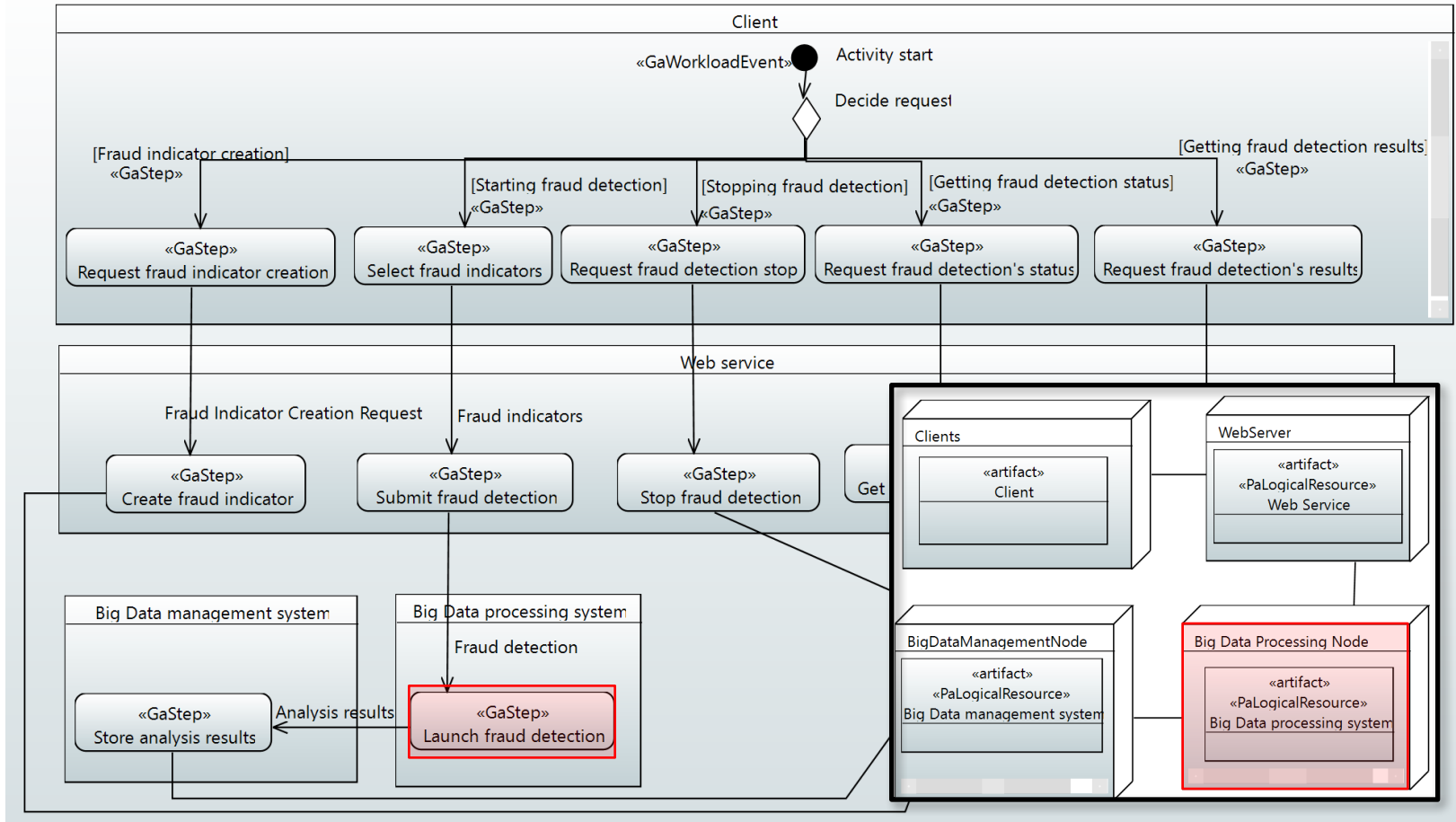
Big Blu Quality assessment

◆ Quality malfunction reported



Big Blu Quality assessment

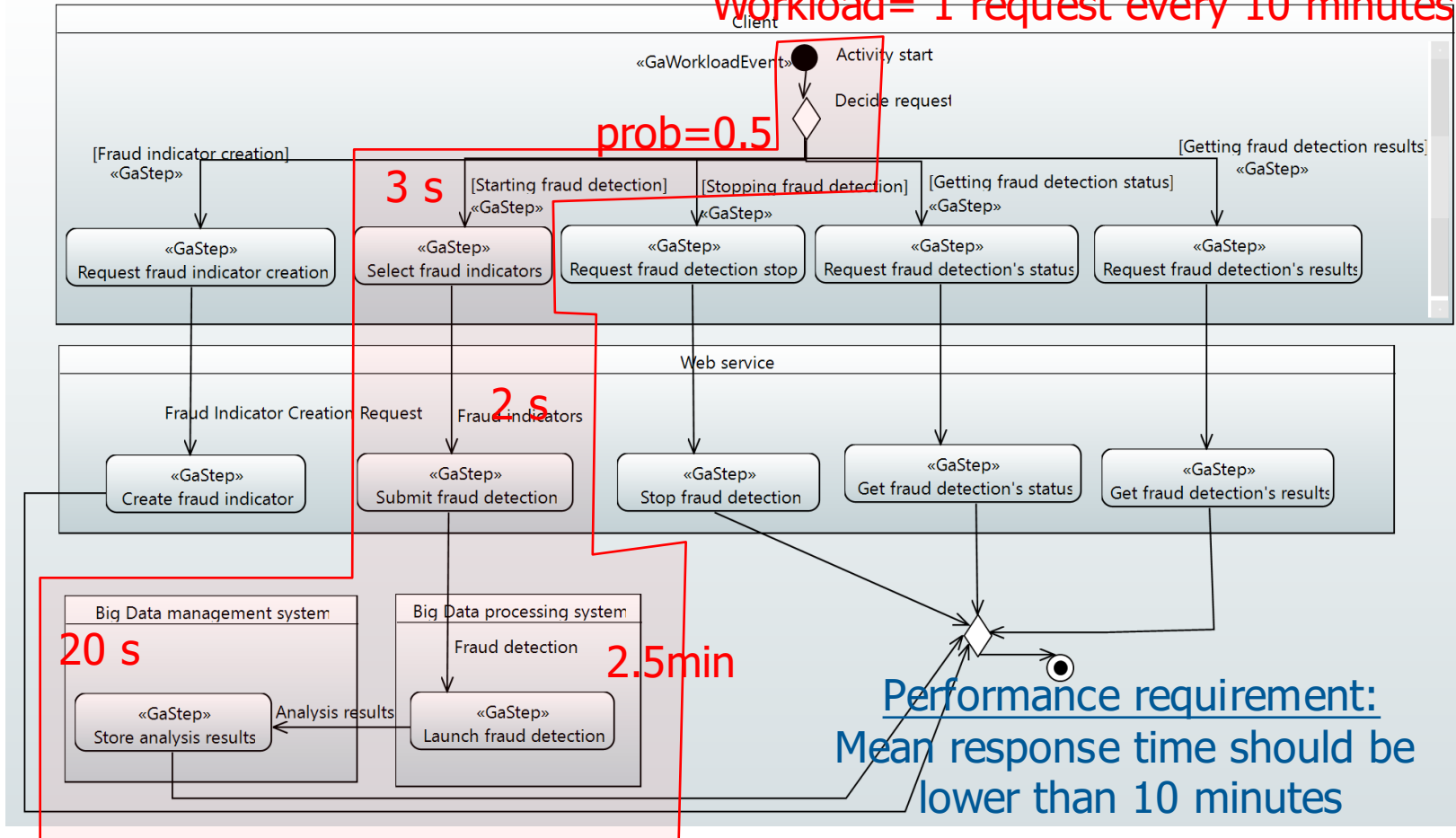
◆ Quality malfunction reported



Big Blu Quality assessment

◆ Quality malfunction reported

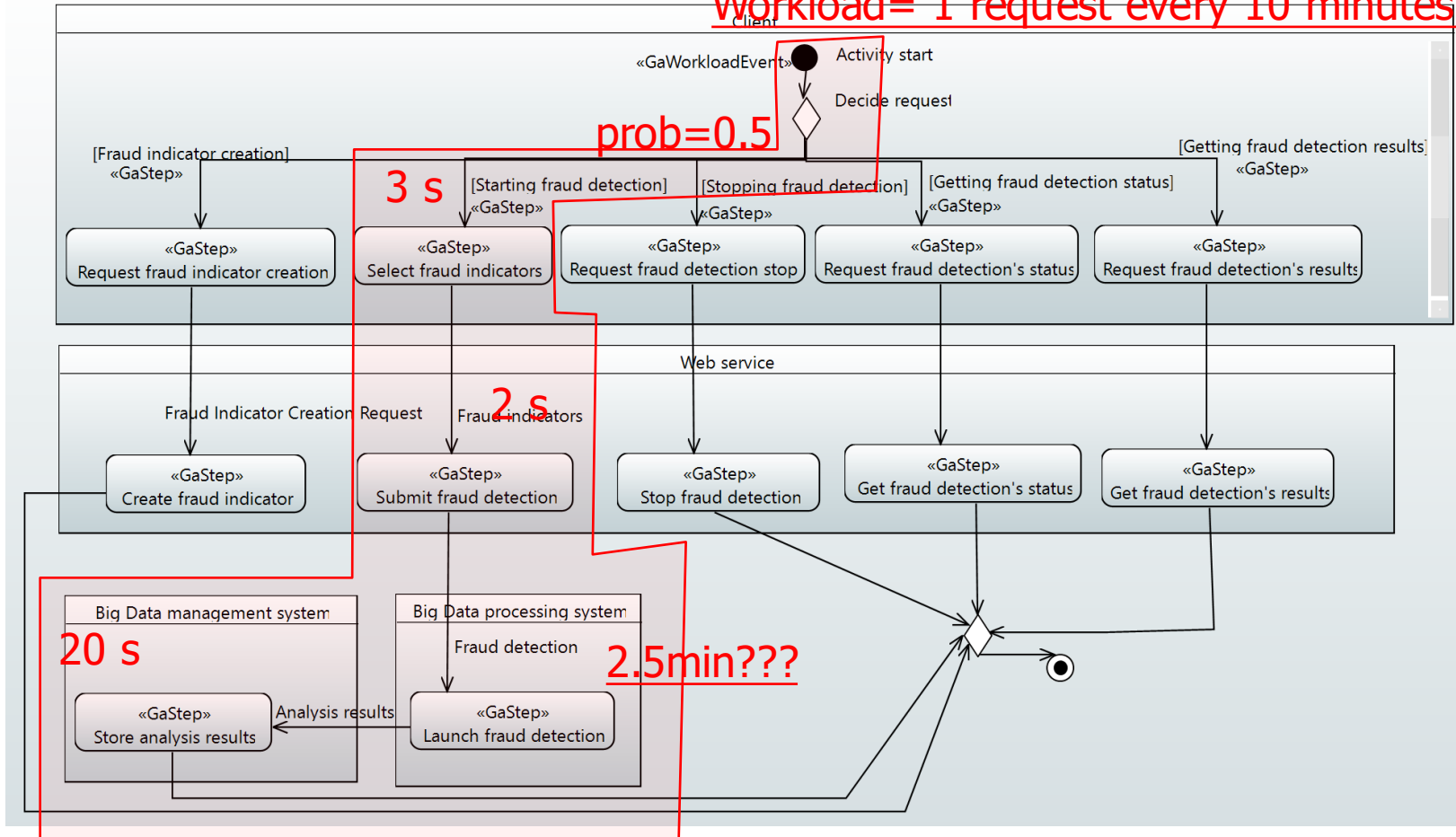
Workload = 1 request every 10 minutes???



Big Blu Quality assessment

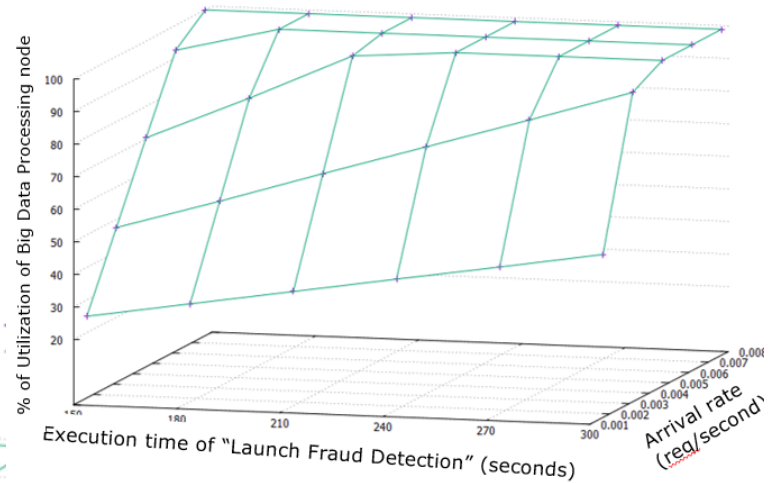
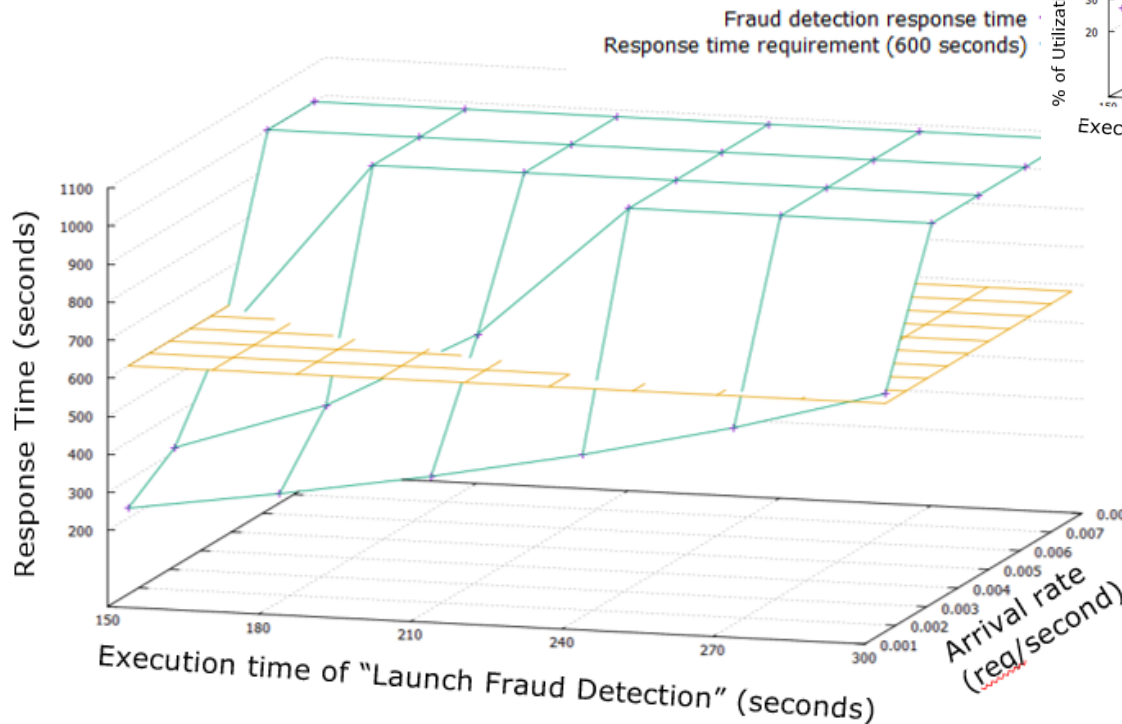
◆ Quality malfunction reported

Workload = 1 request every 10 minutes???



Big Blu Quality assessment

- ◆ Quality malfunction reported
- ◆ Using the SimTool we obtain

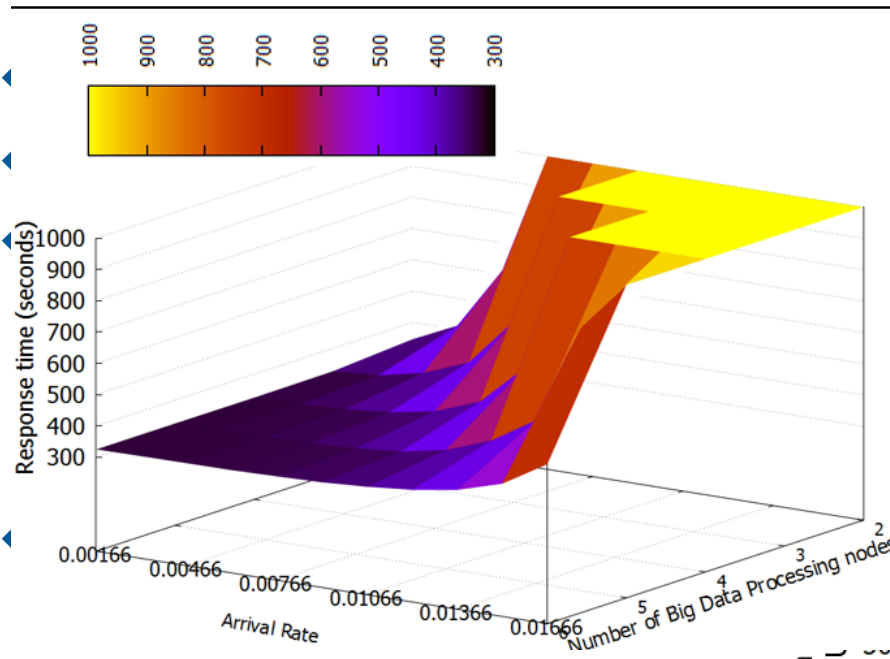


Big Blu Quality assessment

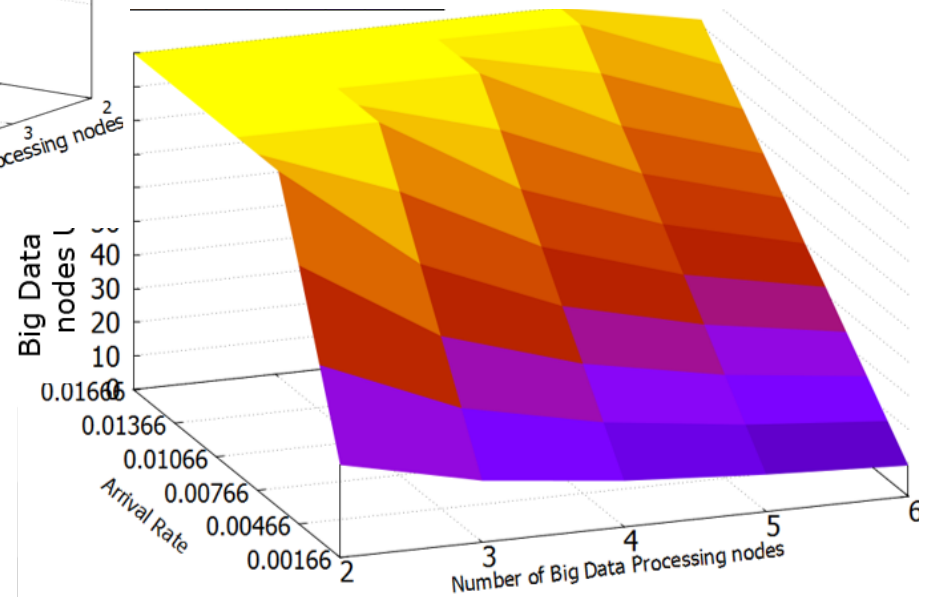
- ◆ Quality malfunction reported
- ◆ Using the SimTool we obtain
- ◆ Developers see two possible solutions
 - Acquire more computing nodes for to parallelise requests
 - Reengineer *Launch Fraud Detection* activity to make it faster

- ◆ Using the SimTool we obtain

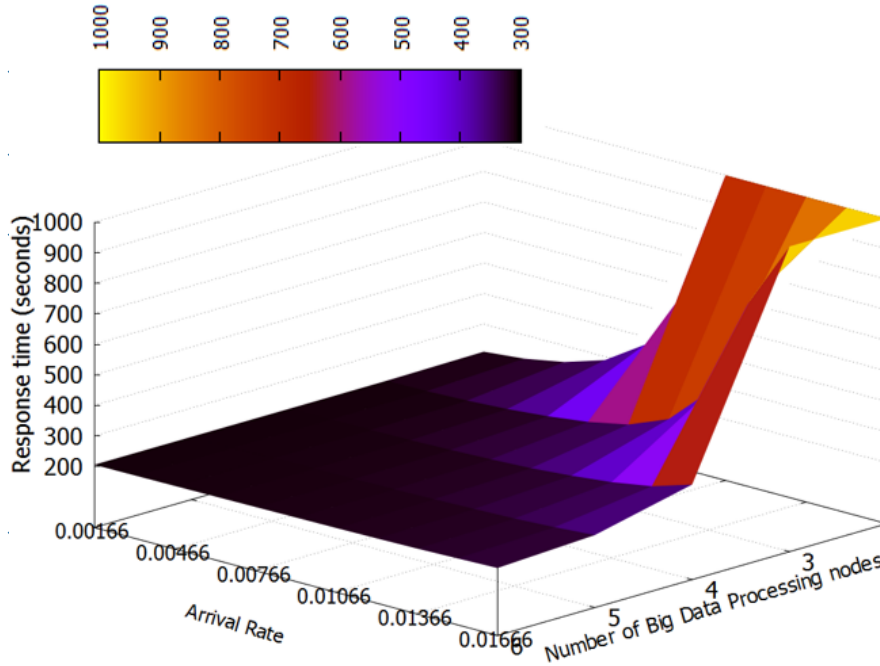
Big Blu Quality assessment



ations
or to parallelise requests
tion activity to make it faster



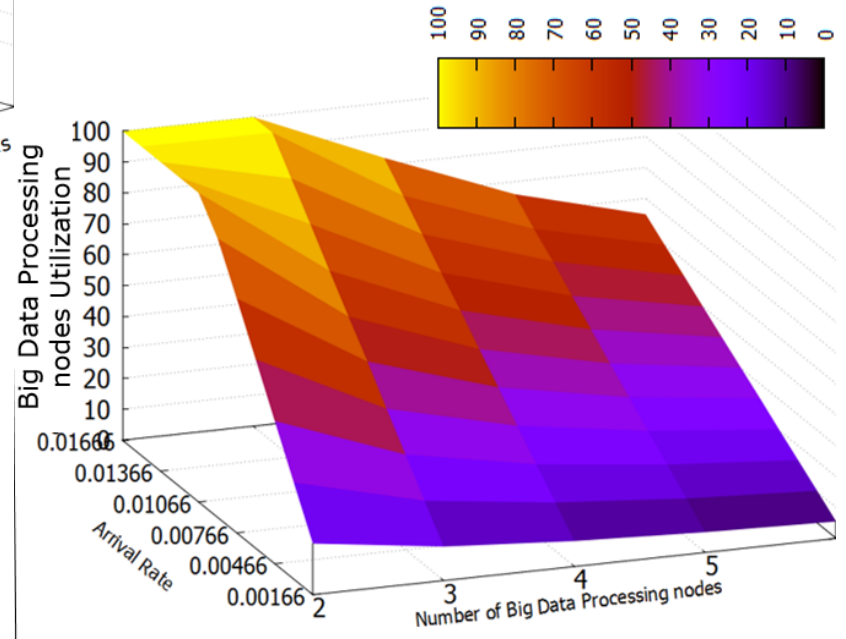
Big Blu Quality assessment



ions

to parallelise requests

on activity to make it faster



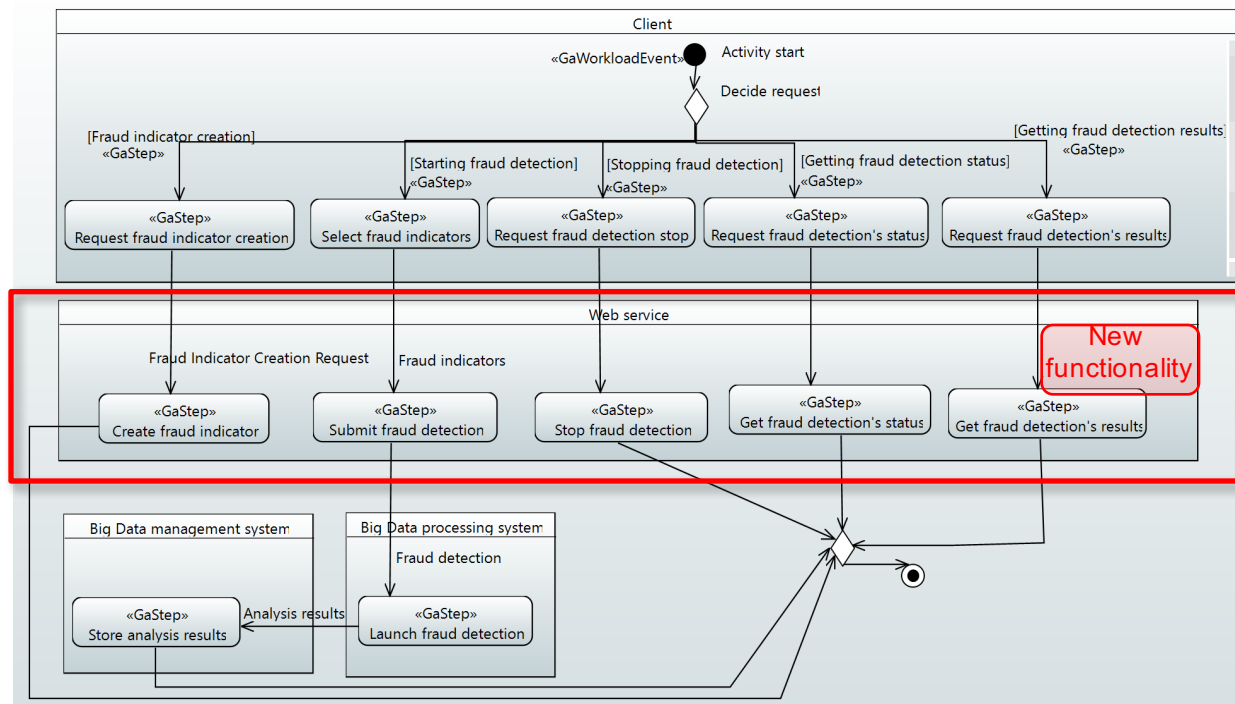
Big Blu Quality assessment

- ◆ Adding a new functionality
 - API that is invoked frequently
 - Provides volatile information to all clients

Big Blu Quality assessment

◆ Adding a new functionality

- API that is invoked frequently
- Provides volatile information to all clients
- It executes in the Web Services layer



Big Blu Quality assessment

◆ Adding a new functionality

- API that is invoked frequently
- Provides volatile information to all clients
- It executes in the Web Services layer
- Each of the current 200 clients will make 1 request per minute
- Developers believe that they can make it to demand between 10 and 20 milliseconds for execution, but there are no new performance requirements

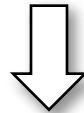
“The mean mean response time of all paths should be less than 10 seconds, except for the 10 minutes allowed for Launch Fraud Detection path”

Big Blu Quality assessment

◆ Adding a new functionality

- API that is invoked frequently
- Provides volatile information to all clients
- It executes in the Web Services layer
- Each of the current 200 clients will make 1 request per minute
- Developers believe that they can make it to demand between 10 and 20 milliseconds for execution, but there are no new performance requirements

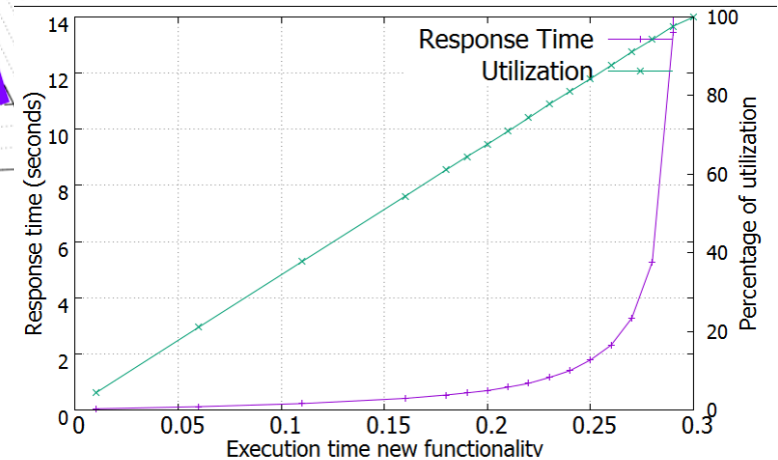
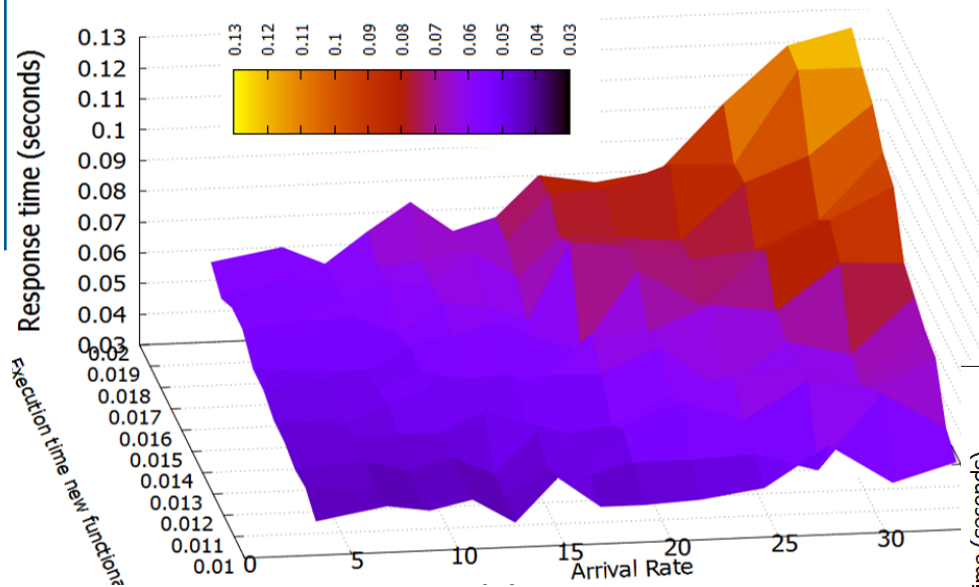
“The mean mean response time of all paths should be less than 10 seconds, except for the 10 minutes allowed for Launch Fraud Detection path”



- Developers do not know if their development will be good enough until integration or operation

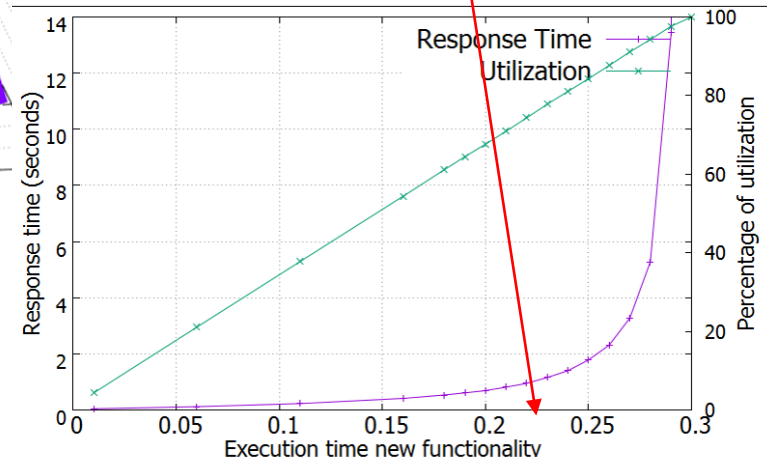
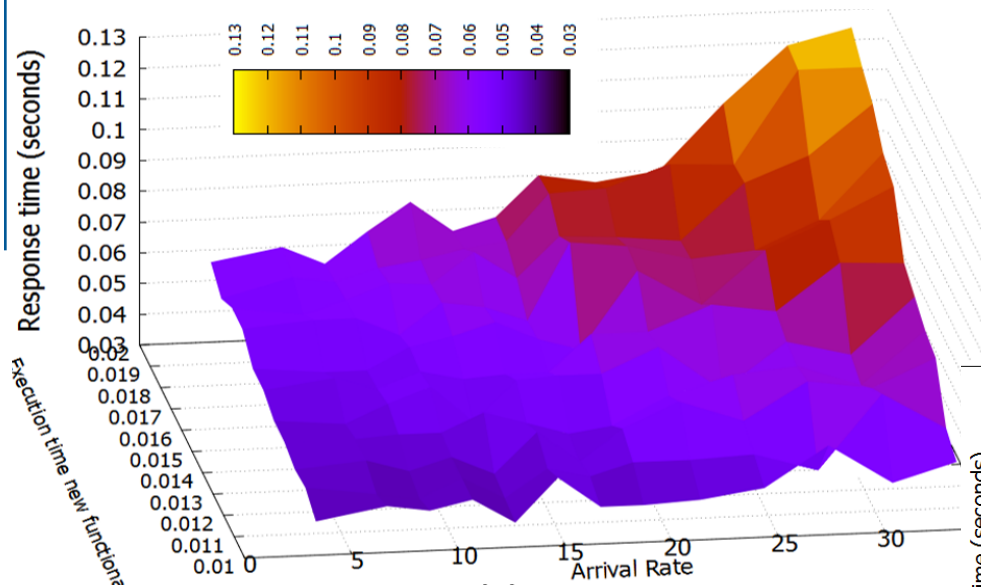
Big Blu Quality assessment

- ◆ Adding a new functionality
 - Using the SimTool we obtain



Big Blu Quality assessment

- ◆ Adding a new functionality
 - Using the SimTool we obtain



Conclusions

- ◆ Report of an experience on usage of software quality evaluation during DevOps-oriented software development
- ◆ Reduce the number of development cycles until satisfactory modification of the system
- ◆ Reported two common scenarios in development cycles
 - Maintenance activity
 - Development of new functionality

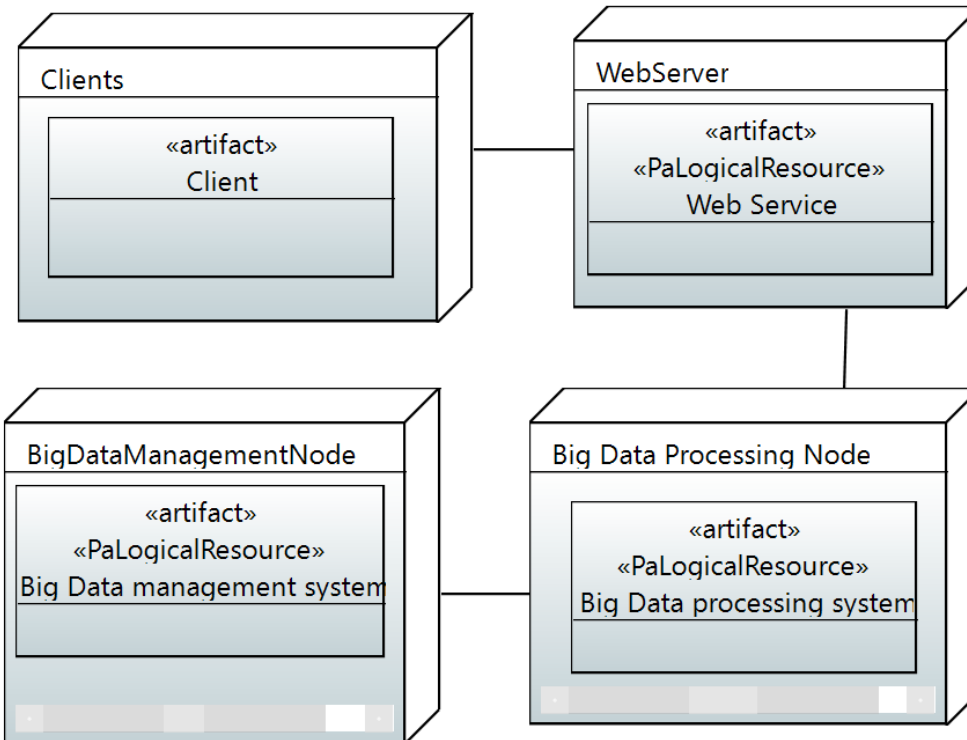
FUTURE:

- Big Blu uses Big Data technologies such as Apache Spark
- SimTool will incorporate characteristics of Big Data technologies

Quality assessment in DevOps:

Automated Analysis of a Tax Fraud Detection System

THANK YOU FOR YOUR ATTENTION!



Probabilities. <i>Prob. of...</i>		Workload	
Fraud indicator creation	0.11	1 request every 600s	
Starting fraud detection	0.5	Resources. <i>Num. of...</i>	
Stopping fraud detection	0.02	Web Server	1
Getting fraud detection status	0.12	Big Data Management	1
Getting fraud detection results	0.25	Big Data Processing	1
Activities Service Times			
Request fraud indicator creation			2s
Select fraud indicators			3s
Request fraud detection stop			3s
Request fraud detection status			1s
Request fraud detection results			2s
Create fraud indicator			3s
Submit fraud detection			2s
Stop fraud detection			1s
Get fraud detection's status			0.1s
Get fraud detection's results			3s
Store analysis results			20s
Launch fraud detection			150s



Tax Compliance

Tax Non-compliance

Normal

Optimization (Avoidance)

Difference in interpretation of the law

Unintended mistakes

Intended mistakes

Evasion

Fraud



Big Blu

Motivation

- ◆ DICE initiative. SimTool is part of it
- ◆ Netfective Technology is developing BIG BLU
- ◆ Big Blu - Performance effective
- ◆ Iterations of new and modified characteristics: Agile
- ◆ New developments may not be appropriate for Operations: DevOps
- ◆ Reduce the number of incorrect cycle iterations
- ◆ DevOps and Agile images here....

DICE Simulation Tool

- **A DB has...** %A second typical usage scenario happens when an new Dev cycle starts and the **quality improvement** of a functionality is planned, assigned to a developer, and monitored information from the Ops is received remarking the poor quality of the current system. This necessity of a new development may come from an inappropriate previous development of the functionality, but not necessarily. The necessity may also come because it happens that the usage of the system has exceeded initial expectations and has increased considerably since the initial development of the functionality. For instance, accesses to a database that has grown in content and concurrent utilization have created a performance bottleneck in the system. The developer can update the quality values in the design model with those ones coming from the monitoring of the application and execute the simulation tool to identify bottlenecks. By the **what-if** analysis, the developer can hypothesize about different quality values several parts of the design in order to identify how much the quality of concrete parts of the system should improve in order to make the overall system satisfy again its requirements. The developer can also play the **what-if** analysis with different designs. Following the previous example, the developer can evaluate what would happen if a cache is created that can resolve 60% of accesses to the database in very little time.