What Embedded can learn from IT Testing Techniques

Parasoft

Rix Groenboom (rixg@parasoft.com)

Mirosław Zielinski (mirek@parasoft.com)

Agenda

- Introduction
- Industry trends
- Inspiration from IT
- Suggestions and (research) questions
- Examples

Introduction

Publisher of software test-solution (test-automation)

Private company, founded 30 years ago

- Started in domain of Parallel Software
- Mission: Perfecting Software

Company Offices:

- Los Angeles (US) / Krakow (PL)
- EMEA HQ in The Hague (NL)
- DACH office in Berlin (DE)

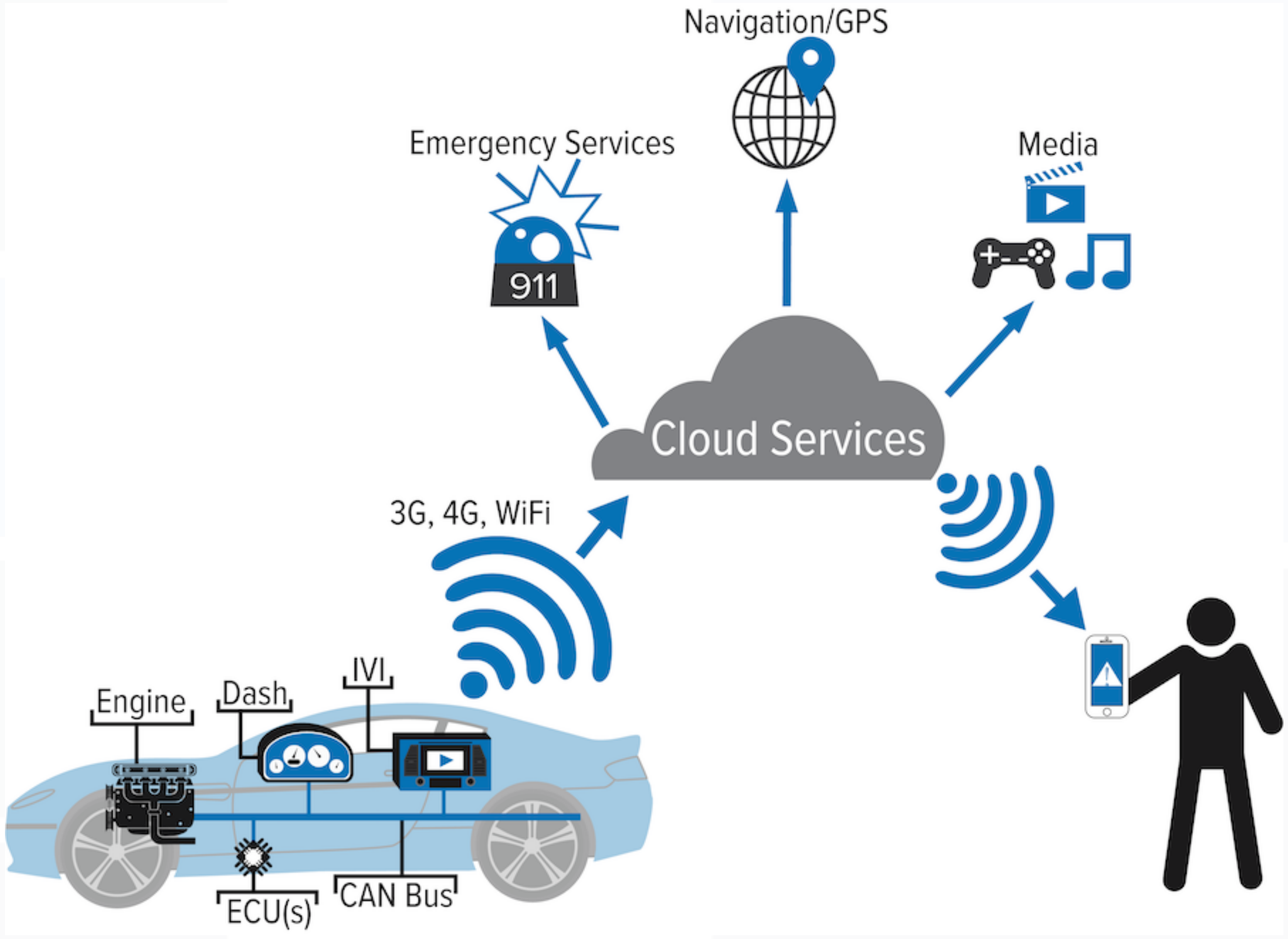
Introduction

- Tools: Static analysis, Unit-testing, Coverage
  - Team level
- Solutions: API testing, service virtualization
  - Architecture level
- Platforms: Development & Continuous testing
  - Process level
- Analytics: Compliance & Metrics
  - Research level

Industry Trends

- Open architecture (standards & protocols)
- Connectivity (24x7) is a game changer:
  - System of Systems
  - Shifts attention of testing
- New trends emerge in automotive, medical, …
  - Software = 21% of vehicle value
  - Software configurable & remote updates
- Safety, security, quality
  - “Computer got out of its cage”

Reference picture



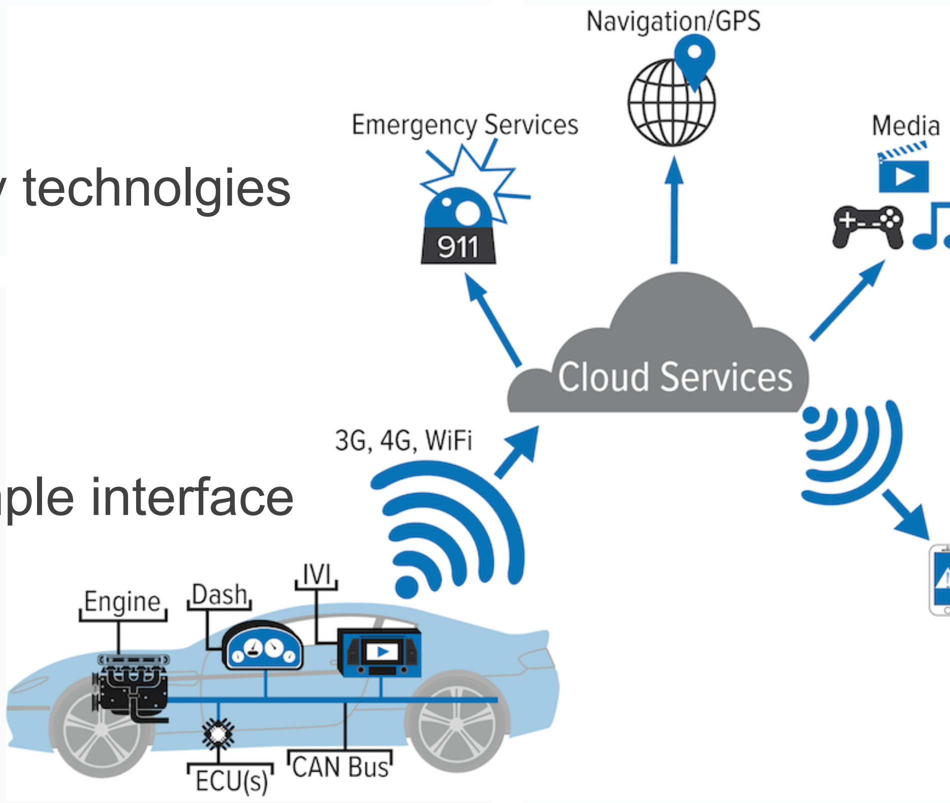
Reference picture: but a car is not static

RDW

PARASOFT

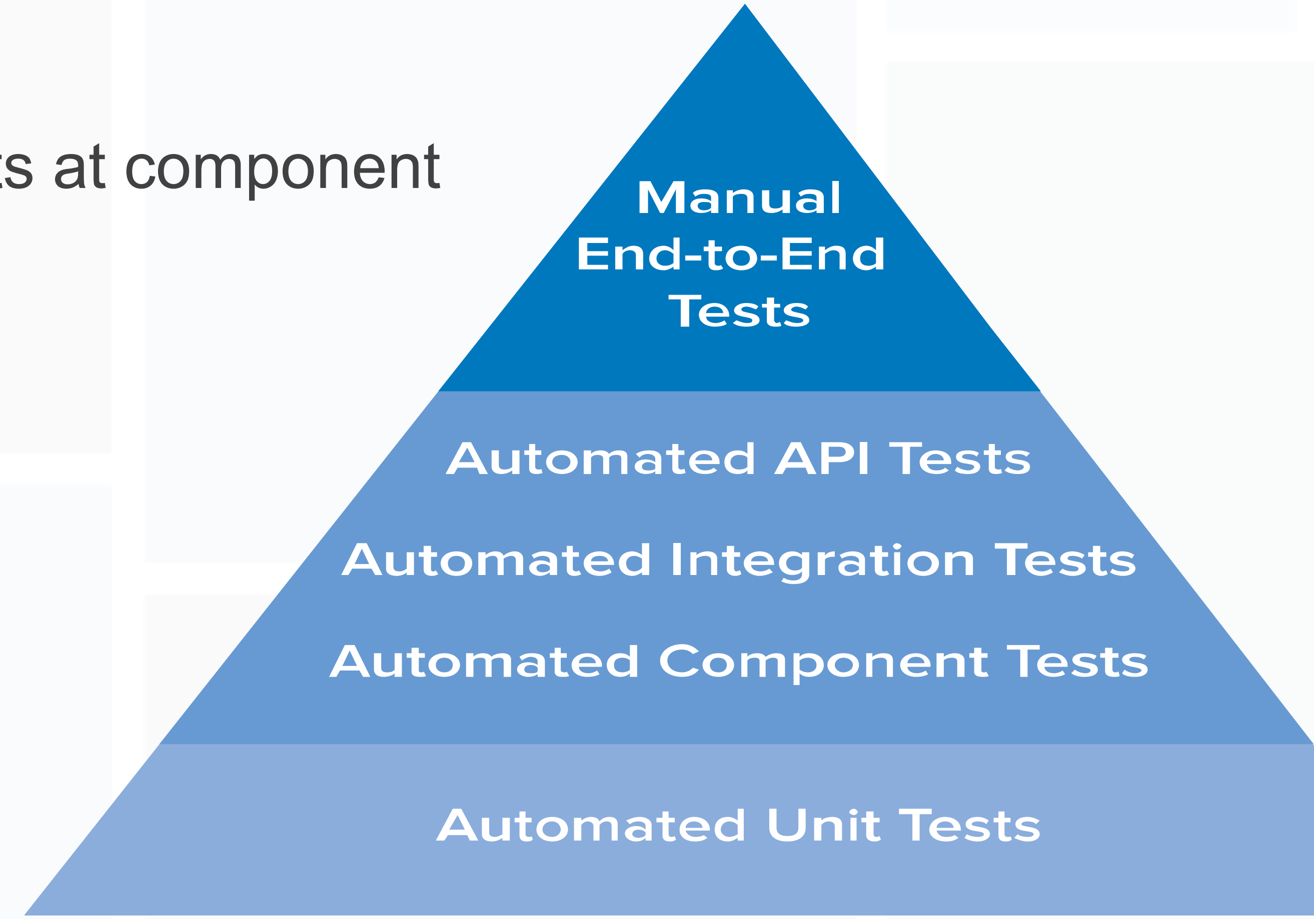
Technology specifics

- Multi-layered systems
  - Even simple functionality may require different teams to develop
- Disparate technologies
  - Debugging requires solid understanding of many technolgies
- Functionality spanning across many layers
  - Complex functionality may be hidden behind simple interface



How to do it better ?

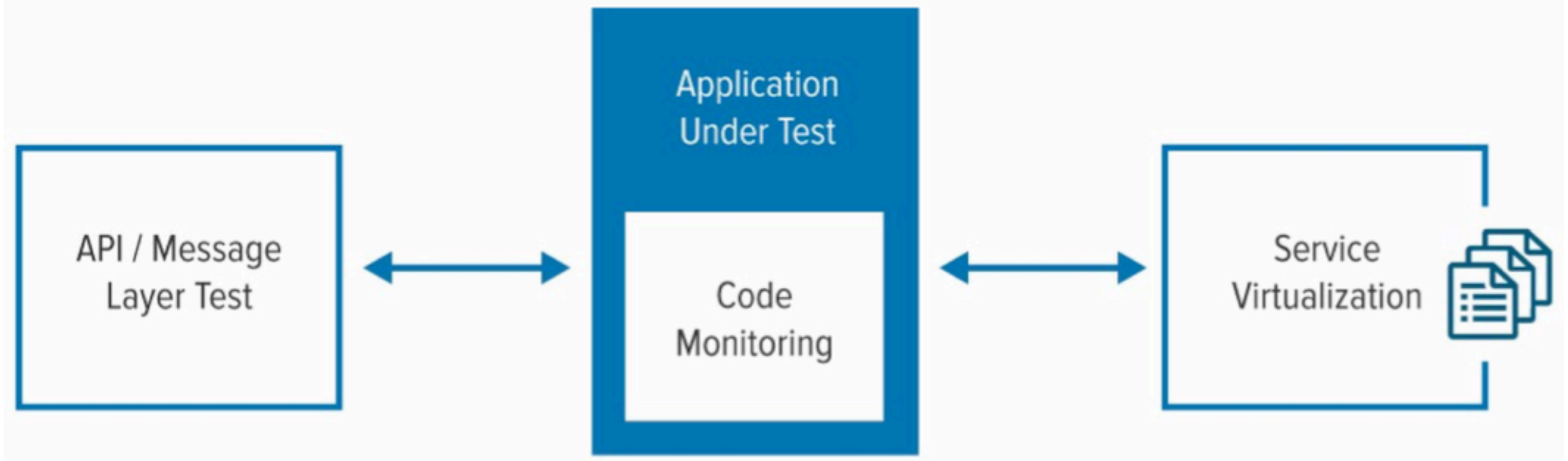
- Limit manual end-to-end testing
- Invest more into automated tests at component
- Follow testing pyramid rules
- Measure:
  - test results
  - test effectiveness



PARASOFT

How to do it better ?

- Give absolute priority to automated tests
- Invest time into designing interfaces (API)
- Use API (service) testing tools to cover interfaces
- Measure the quality of the test



PARASOFT

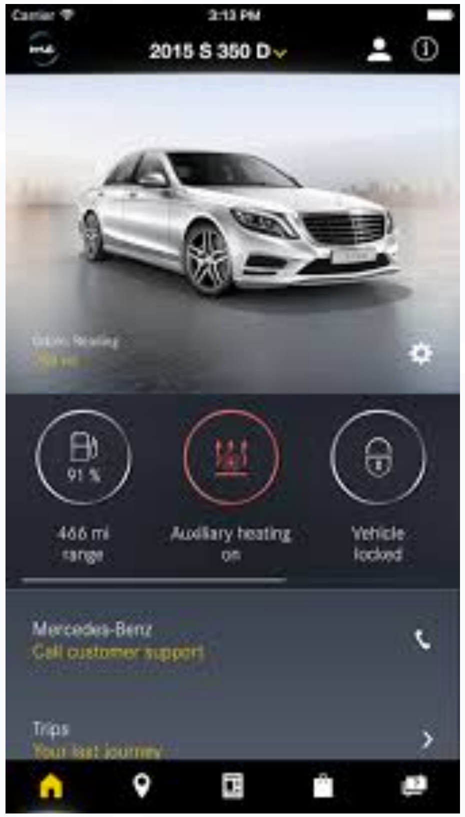
How to do it better ?

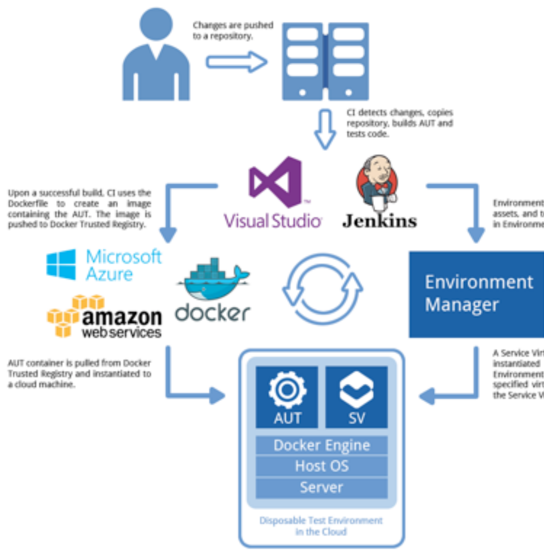
- **Emulate dependent systems with service virtualization, which removes testing constraints and increases flexibility across complex test environments.**

- **Research question:** *What is the right level of abstraction to intro- duce simulation and how "realistic" do these virtual services have to be? What is the impact of the factor "real-time" when using Service Virtualization for cyber-physical systems?*

Example: Service Virtualization

- Car and App
- Connect via gateway
  - Backend connection
- Agile development
  - Simulated test environment
- Load and performance testing




Changes are pushed to a repository.
CI detects changes, copies repository, builds AUT and tests code.
Upon a successful build, CI uses the Dockerfile to create an image containing the AUT. The image is pushed to Docker Trusted Registry.
Visual Studio
Jenkins
Microsoft Azure
amazon web services
docker
Environment Manager
AUT container is pulled from Docker Trusted Registry and instantiated to a cloud machine.
AUT
SV
Docker Engine
Host OS
Server
Disposable Test Environment in the Cloud

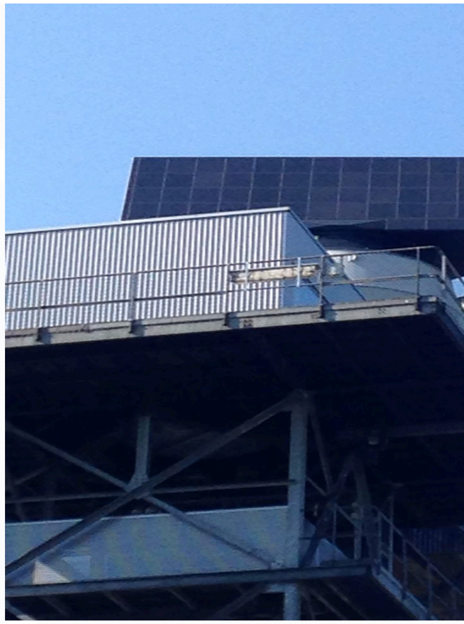
How to do it better ?

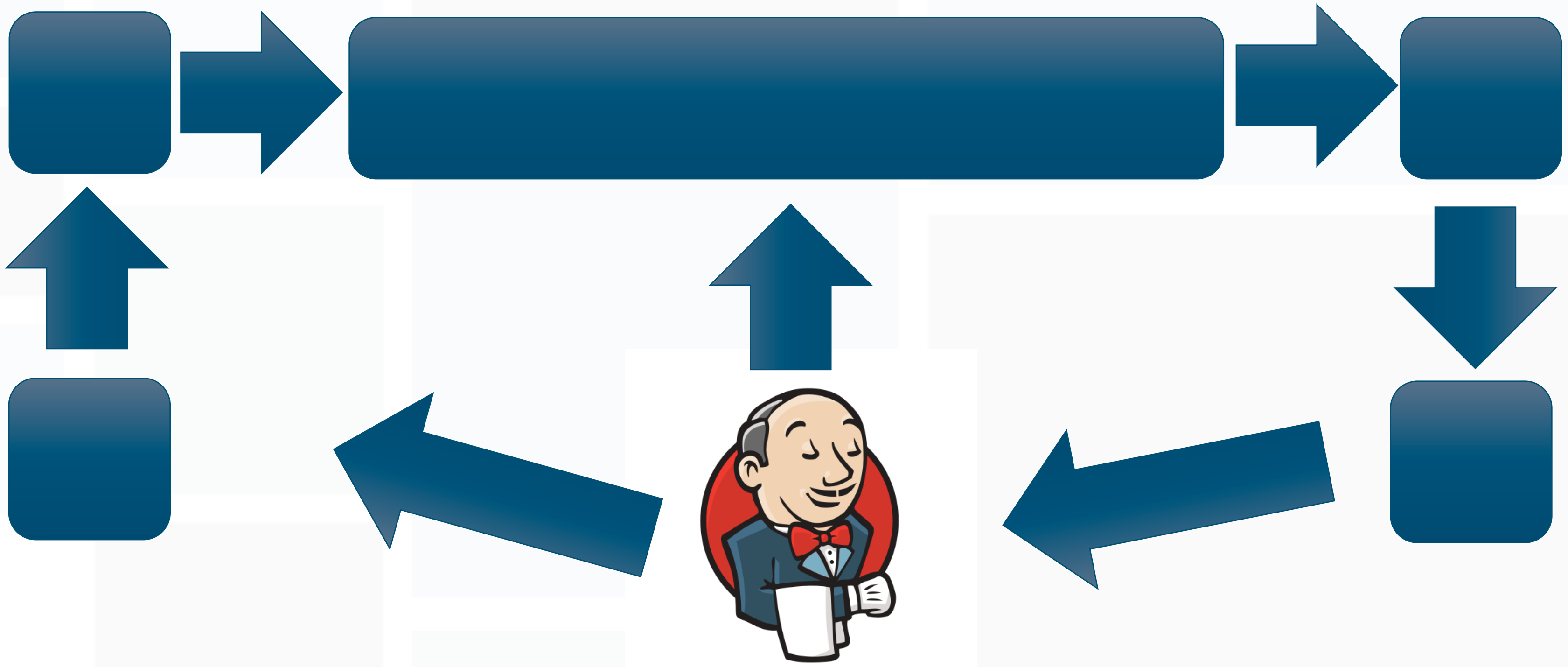
- **Test at scale when scalable hardware is unavailable or impractical.**

- ***Research questions***: *How can we efficiently create these virtual environments and to include "fault models" to simulate the required test conditions?*

Example: API testing

- Radar tracking software (SMART-L)
- Data feed & Matlab verification
- API-driven regression testing





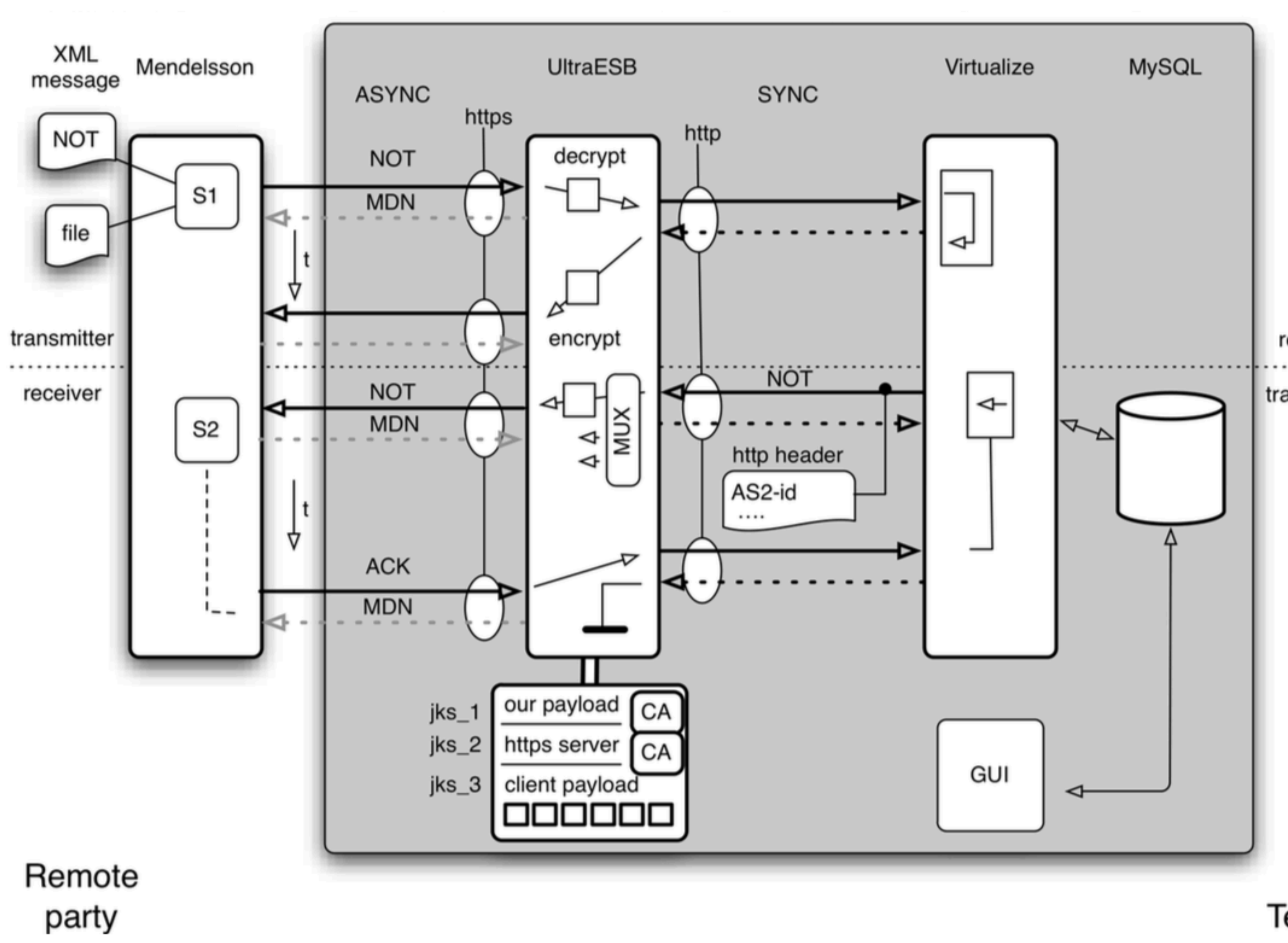
How to do it better ?

- **Leverage penetration testing techniques to expose security vulnerabilities and trace their impact directly to the code base.**
  - **“Unintended Easter-eggs”, or for some a “Christmas present”**

- ***Research questions****: How can we support the discovery of the (code level) rules that prevent possible vulnerabilities, in particular in case of highly connected systems with diverse technology stack?*

Example: Compliance testing

- Energy market
- B2B communication
- Protocol verification



How to do it better ?

- **Combine data from coverage analysis and message layer testing technologies**

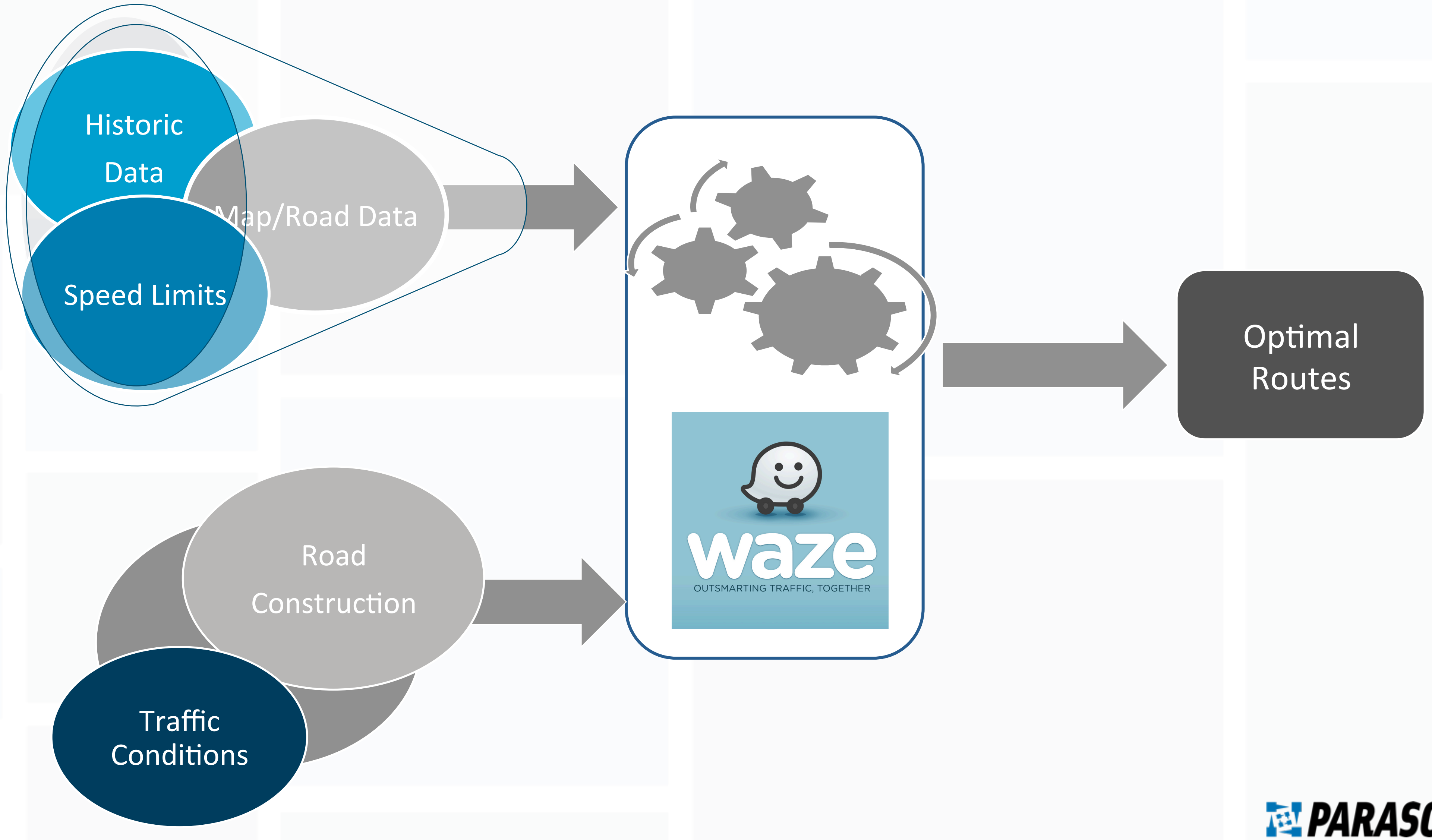
- ***Research questions****: What are the right metrics to collect and how can the used to act as quality gates to answer strategic questions on “what is the quality of the current build of my system”?*

Harnessing "Big" Data from SW development

- Aggregate data
- Correlate data
- Mine data
- Create
  - Reports
  - Dashboards
  - Tasks
  - Alerts
- Continuous testing/delivery/release

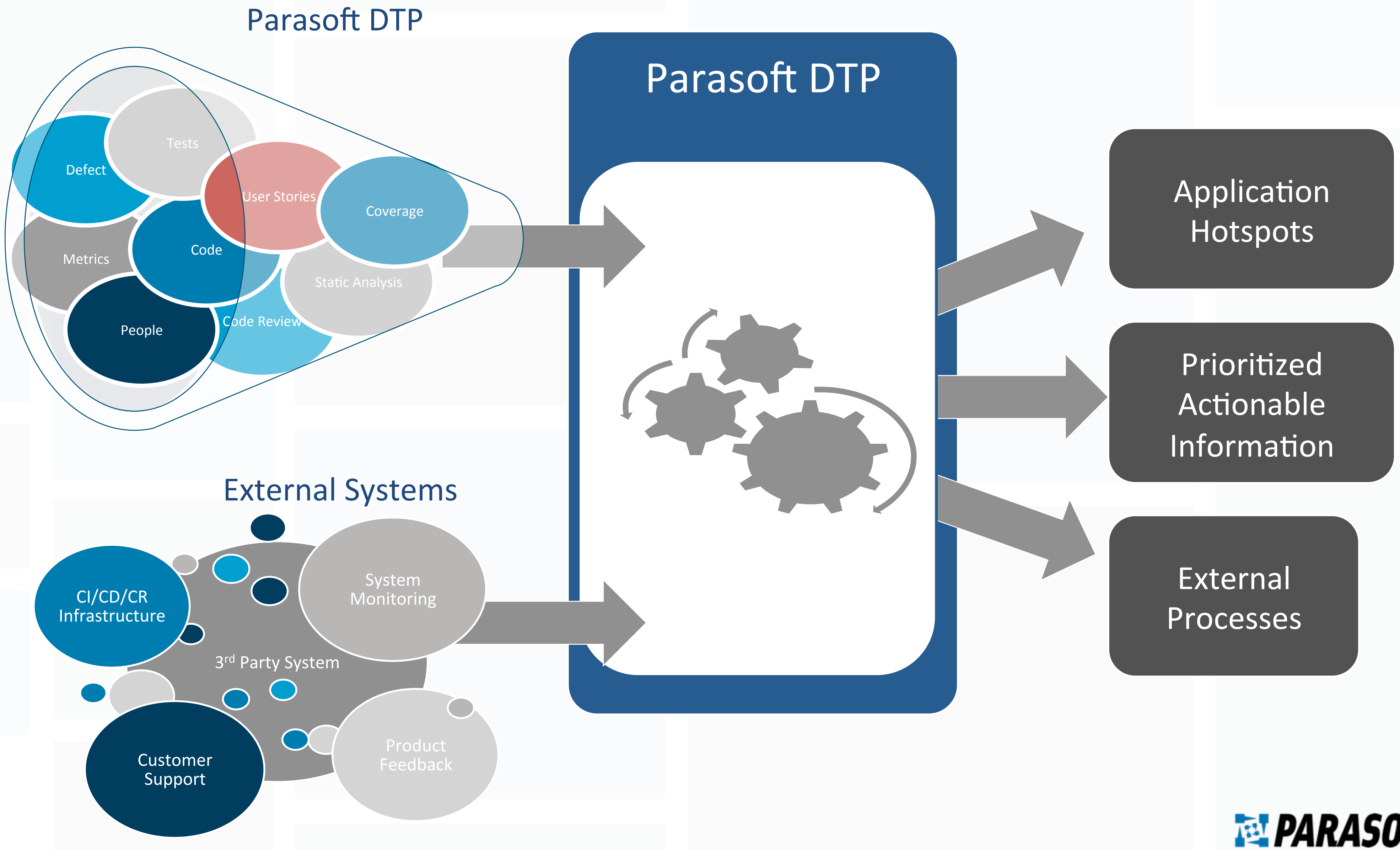
PARASOFT

Software Assisted Decisions: Navigation



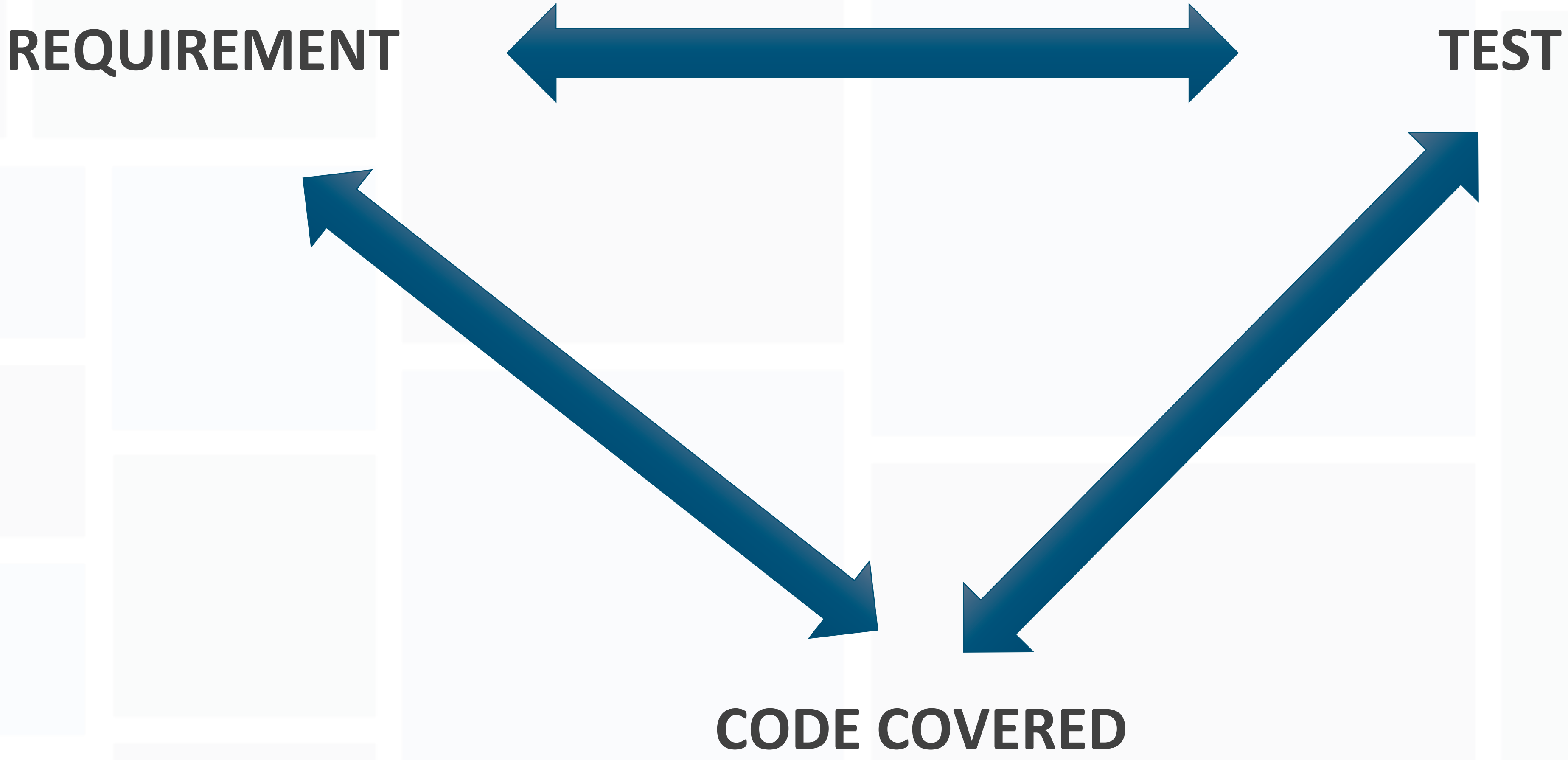
PARASOFT

Software Assisted Decisions: SW release

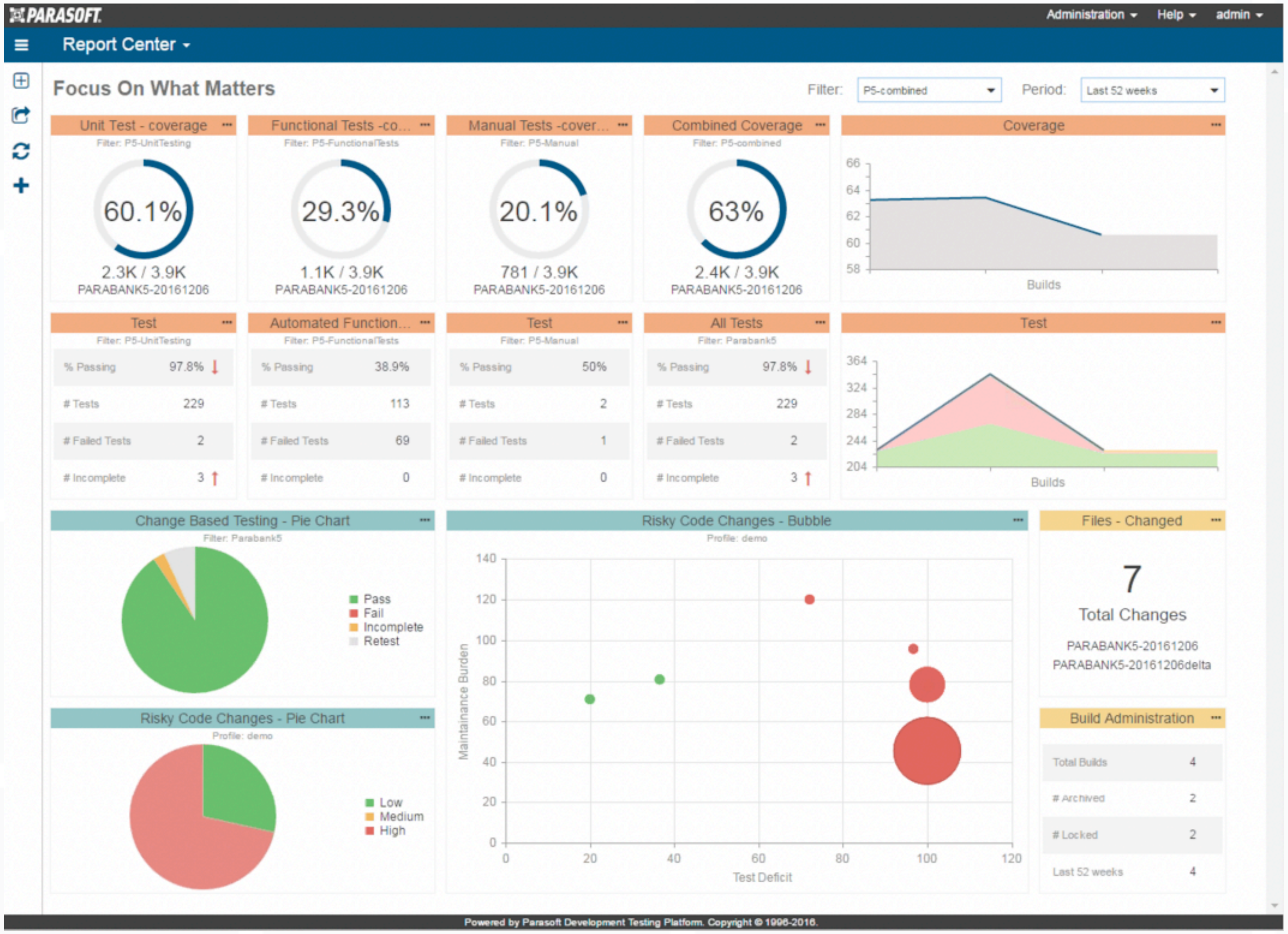


PARASOFT

Example: Change Based Testing



Example: Change Based Testing



PARASOFT | Administration | Help | admin

Report Center

Focus On What Matters

Filter: P5-combined | Period: Last 52 weeks

Unit Test - coverage	Functional Tests -co...	Manual Tests -cover...	Combined Coverage
Filter: P5-UnitTesting	Filter: P5-FunctionalTests	Filter: P5-Manual	Filter: P5-combined
60.1%	29.3%	20.1%	63%
2.3K / 3.9K	1.1K / 3.9K	781 / 3.9K	2.4K / 3.9K
PARABANK5-20161206	PARABANK5-20161206	PARABANK5-20161206	PARABANK5-20161206

Coverage

Test		Automated Function...		Test		All Tests	
Filter: P5-UnitTesting		Filter: P5-FunctionalTests		Filter: P5-Manual		Filter: Parabank5	
% Passing	97.8% ↓	% Passing	38.9%	% Passing	50%	% Passing	97.8% ↓
# Tests	229	# Tests	113	# Tests	2	# Tests	229
# Failed Tests	2	# Failed Tests	69	# Failed Tests	1	# Failed Tests	2
# Incomplete	3 ↑	# Incomplete	0	# Incomplete	0	# Incomplete	3 ↑

Test

Change Based Testing - Pie Chart
Filter: Parabank5

Risky Code Changes - Bubble
Profile: demo

Files - Changed

7
Total Changes
PARABANK5-20161206
PARABANK5-20161206delta

Risky Code Changes - Pie Chart
Profile: demo

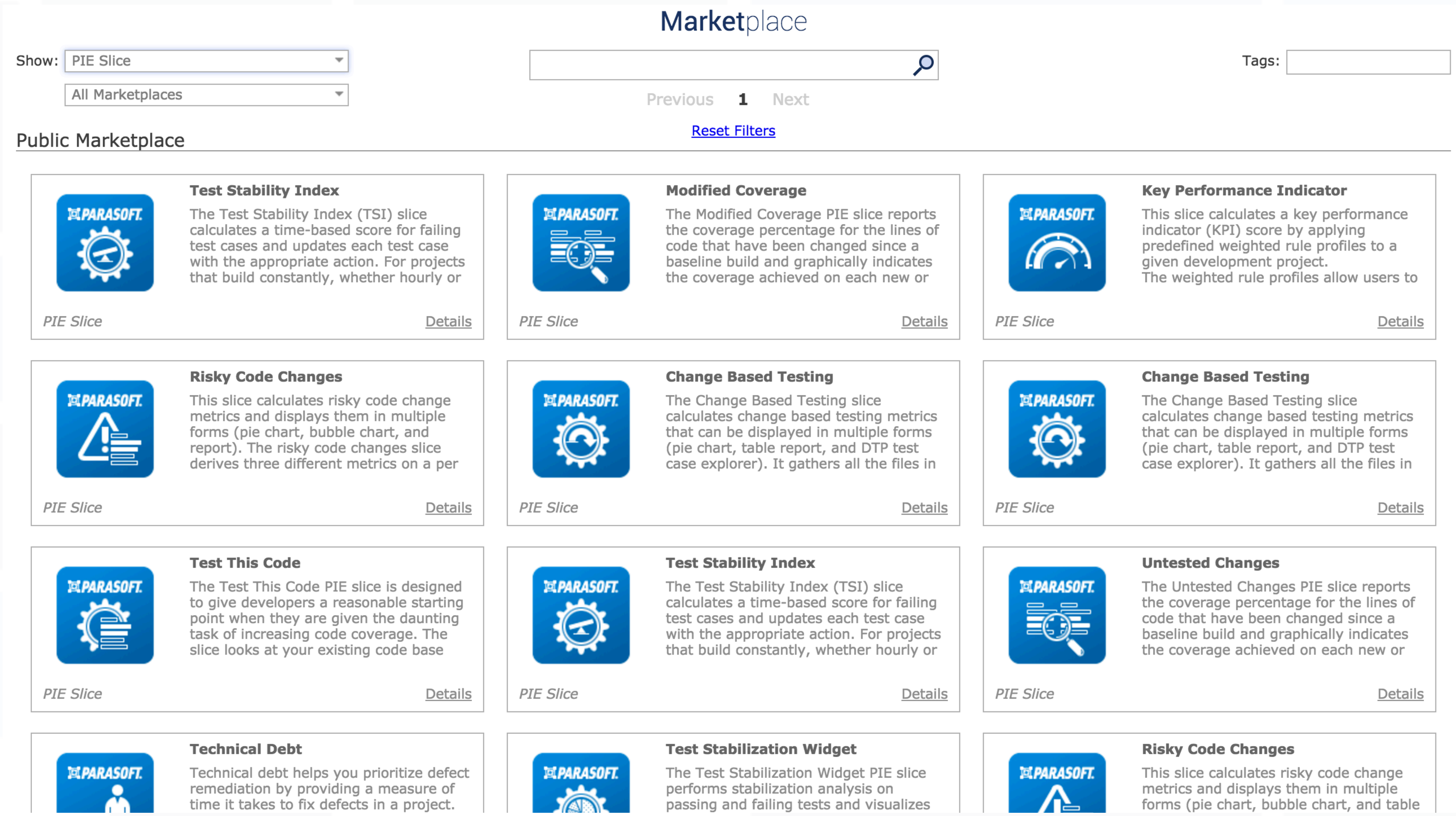
Build Administration

Total Builds	4
# Archived	2
# Locked	2
Last 52 weeks	4

Powered by Parasoft Development Testing Platform. Copyright © 1996-2016.

PARASOFT

Example: More derived metrics



Marketplace

Show: PIE Slice

All Marketplaces

Previous 1 Next

Reset Filters

Tags:

Public Marketplace

Test Stability Index
The Test Stability Index (TSI) slice calculates a time-based score for failing test cases and updates each test case with the appropriate action. For projects that build constantly, whether hourly or
PIE Slice Details

Modified Coverage
The Modified Coverage PIE slice reports the coverage percentage for the lines of code that have been changed since a baseline build and graphically indicates the coverage achieved on each new or
PIE Slice Details

Key Performance Indicator
This slice calculates a key performance indicator (KPI) score by applying predefined weighted rule profiles to a given development project.
The weighted rule profiles allow users to
PIE Slice Details

Risky Code Changes
This slice calculates risky code change metrics and displays them in multiple forms (pie chart, bubble chart, and report). The risky code changes slice derives three different metrics on a per
PIE Slice Details

Change Based Testing
The Change Based Testing slice calculates change based testing metrics that can be displayed in multiple forms (pie chart, table report, and DTP test case explorer). It gathers all the files in
PIE Slice Details

Change Based Testing
The Change Based Testing slice calculates change based testing metrics that can be displayed in multiple forms (pie chart, table report, and DTP test case explorer). It gathers all the files in
PIE Slice Details

Test This Code
The Test This Code PIE slice is designed to give developers a reasonable starting point when they are given the daunting task of increasing code coverage. The slice looks at your existing code base
PIE Slice Details

Test Stability Index
The Test Stability Index (TSI) slice calculates a time-based score for failing test cases and updates each test case with the appropriate action. For projects that build constantly, whether hourly or
PIE Slice Details

Untested Changes
The Untested Changes PIE slice reports the coverage percentage for the lines of code that have been changed since a baseline build and graphically indicates the coverage achieved on each new or
PIE Slice Details

Technical Debt
Technical debt helps you prioritize defect remediation by providing a measure of time it takes to fix defects in a project.

Test Stabilization Widget
The Test Stabilization Widget PIE slice performs stabilization analysis on passing and failing tests and visualizes

Risky Code Changes
This slice calculates risky code change metrics and displays them in multiple forms (pie chart, bubble chart, and table

Summary

- We recognize parallel trends from “IT” to “IoT” and “IIoT”
- Automated API Testing:
  - Moving up the testing pyramid enables a cost-effective way to validate components individually and as part of the system as a whole.
- Service Virtualization:
  - Using service virtualization eliminates system constraints and enables automated tests to run continuously as part of CI workflows.
- Automated Penetration Testing:
  - Expose vulnerabilities that can be traced back to the underlying code
- Test, Requirement and Code Traceability:
  - Capturing coverage for individual tests and correlating back to original requirements enables organizations to choose the most cost-effective testing techniques.

PARASOFT.

Q & A