LTB 2022
Load Testing & Benchmarking

A Review of Modern Challenges in Performance Testing

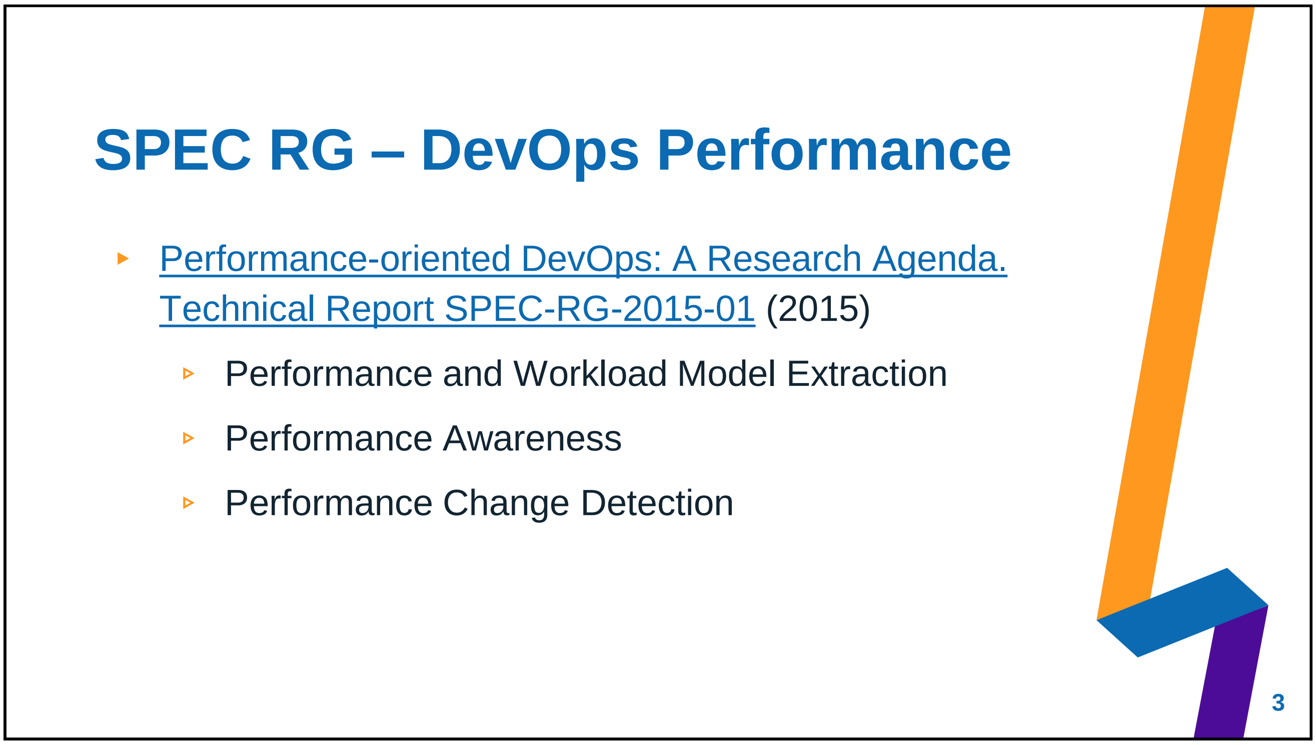
Alexander Podelko
Staff Performance Engineer
MongoDB

1

The Current Trends in Industry and Academia

2

2

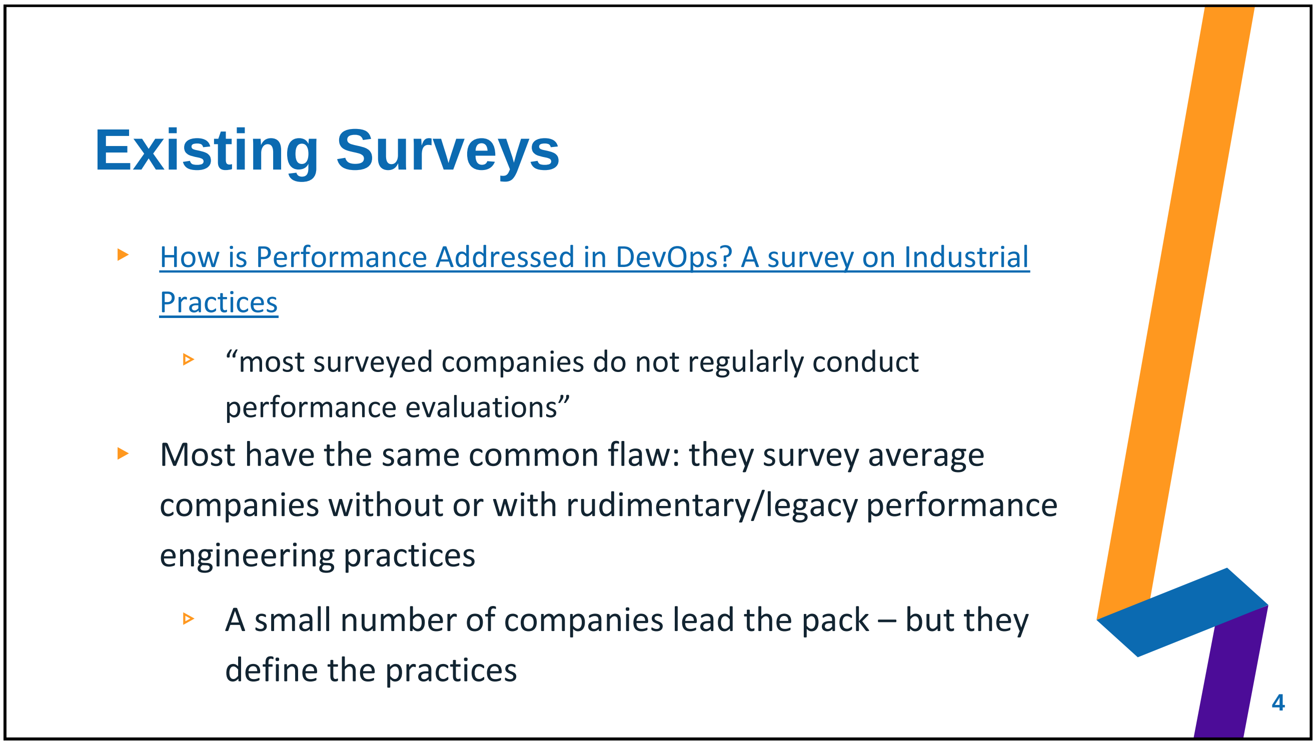


SPEC RG – DevOps Performance

- Performance-oriented DevOps: A Research Agenda. Technical Report SPEC-RG-2015-01 (2015)
  - Performance and Workload Model Extraction
  - Performance Awareness
  - Performance Change Detection

3

3

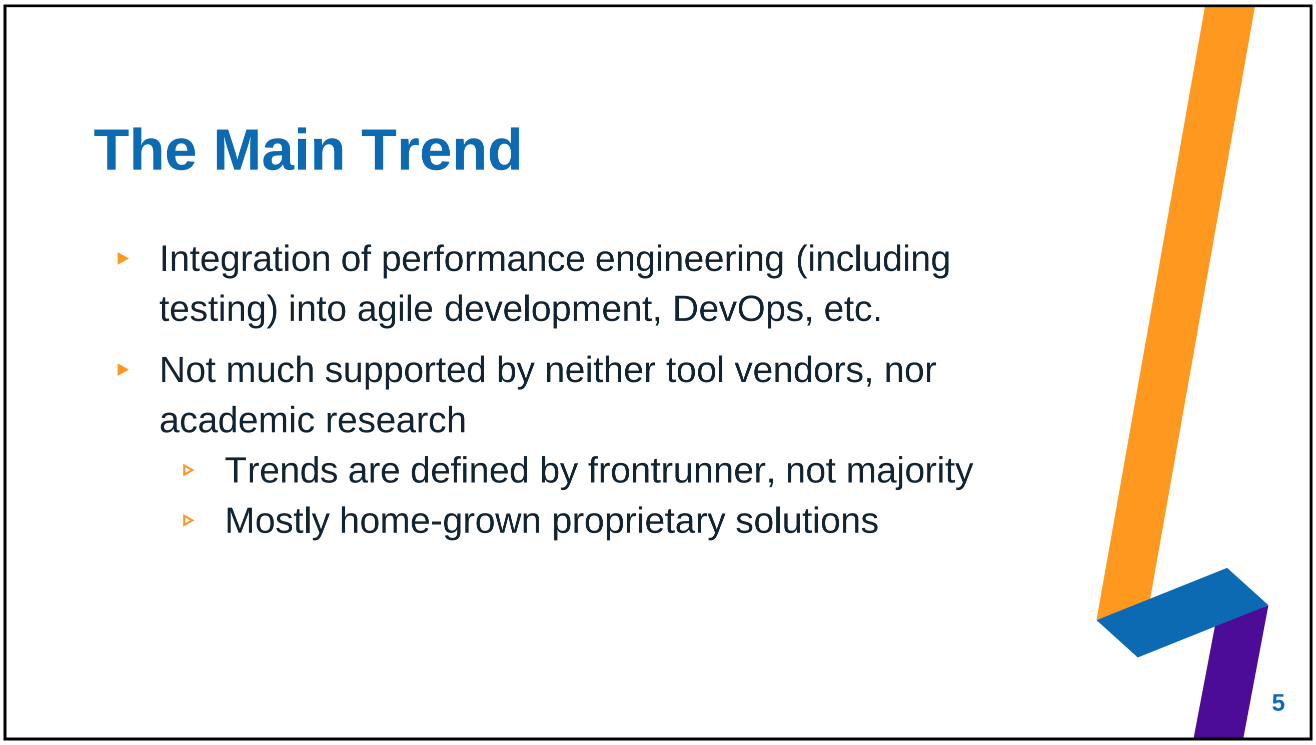


Existing Surveys

- How is Performance Addressed in DevOps? A survey on Industrial Practices
  - “most surveyed companies do not regularly conduct performance evaluations”
- Most have the same common flaw: they survey average companies without or with rudimentary/legacy performance engineering practices
  - A small number of companies lead the pack – but they define the practices

4

4

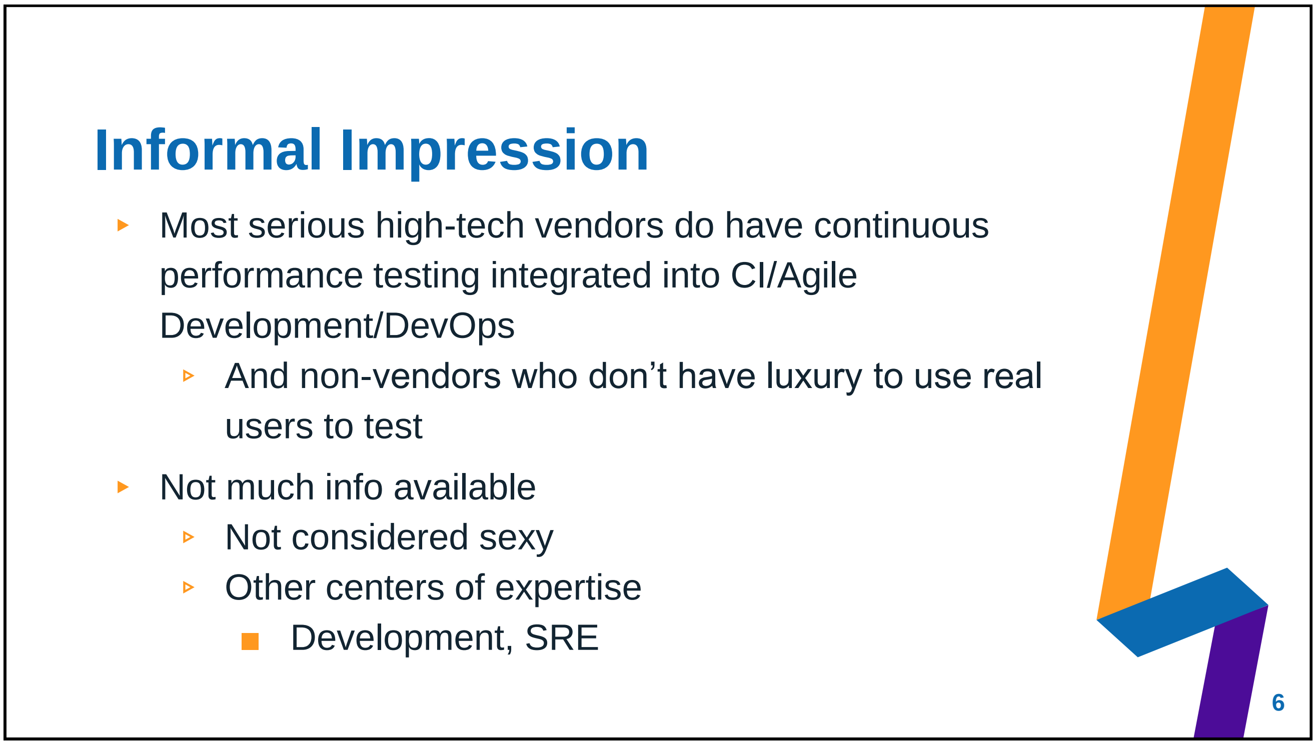


The Main Trend

- Integration of performance engineering (including testing) into agile development, DevOps, etc.
- Not much supported by neither tool vendors, nor academic research
  - Trends are defined by frontrunner, not majority
  - Mostly home-grown proprietary solutions

5

5

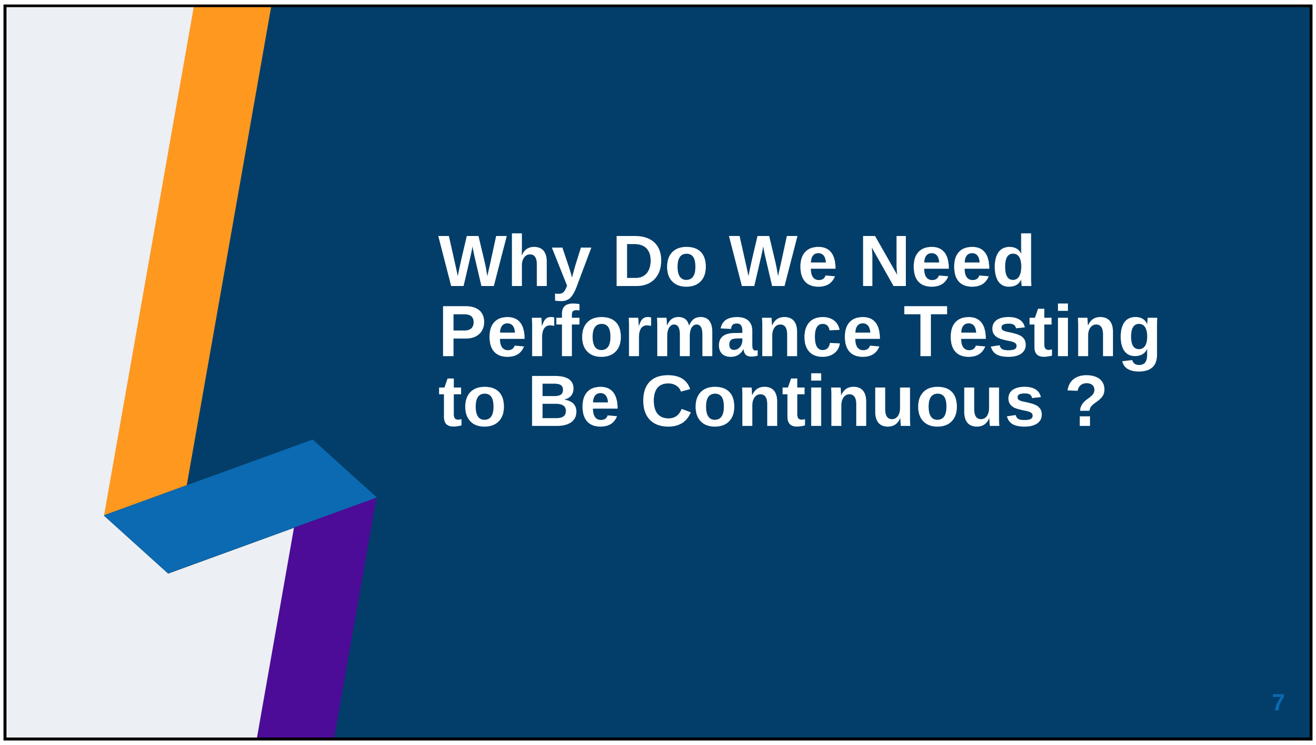


Informal Impression

- Most serious high-tech vendors do have continuous performance testing integrated into CI/Agile Development/DevOps
  - And non-vendors who don't have luxury to use real users to test
- Not much info available
  - Not considered sexy
  - Other centers of expertise
    - Development, SRE

6

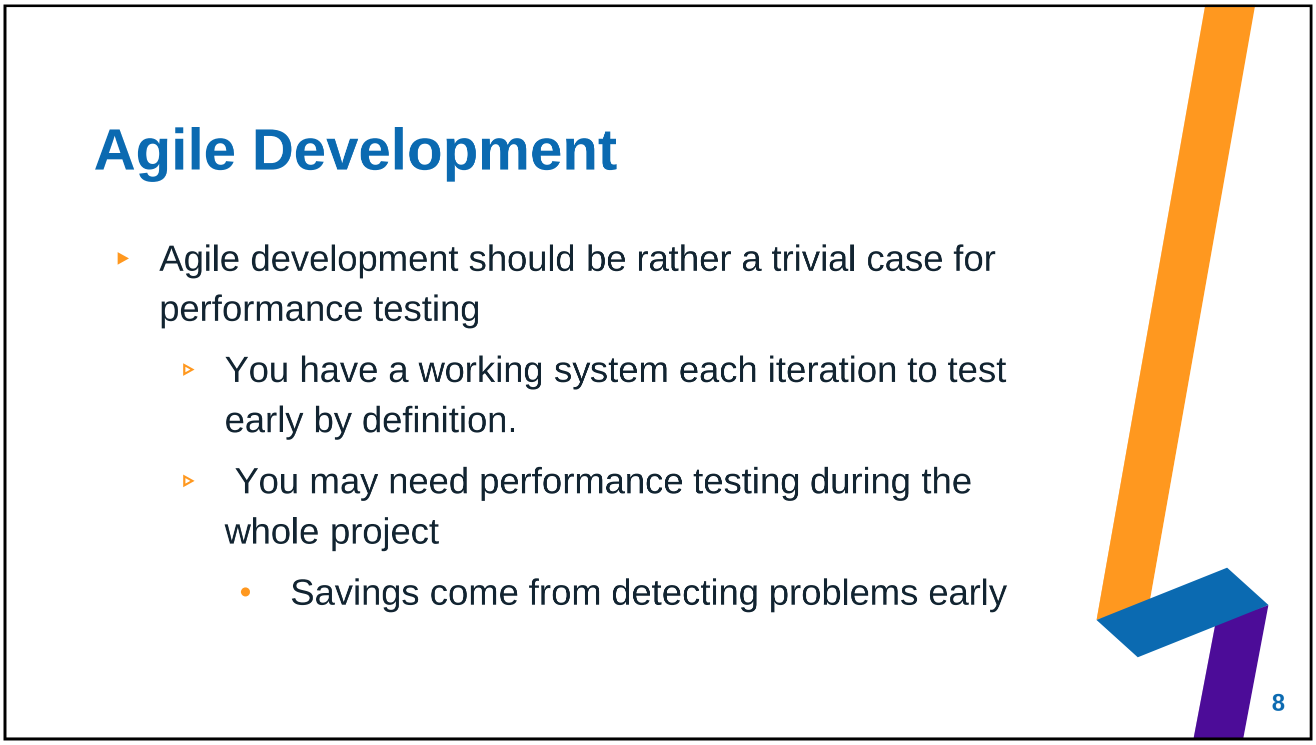
6



Why Do We Need Performance Testing to Be Continuous ?

7

7



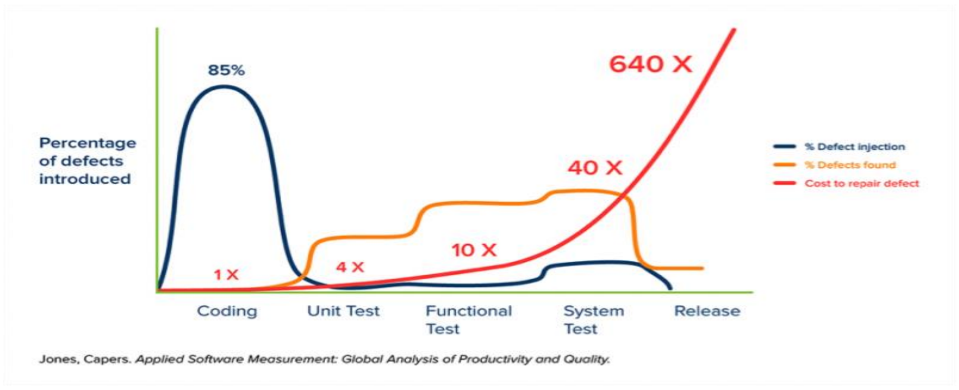
Agile Development

- Agile development should be rather a trivial case for performance testing
  - You have a working system each iteration to test early by definition.
  - You may need performance testing during the whole project
    - Savings come from detecting problems early

8

8

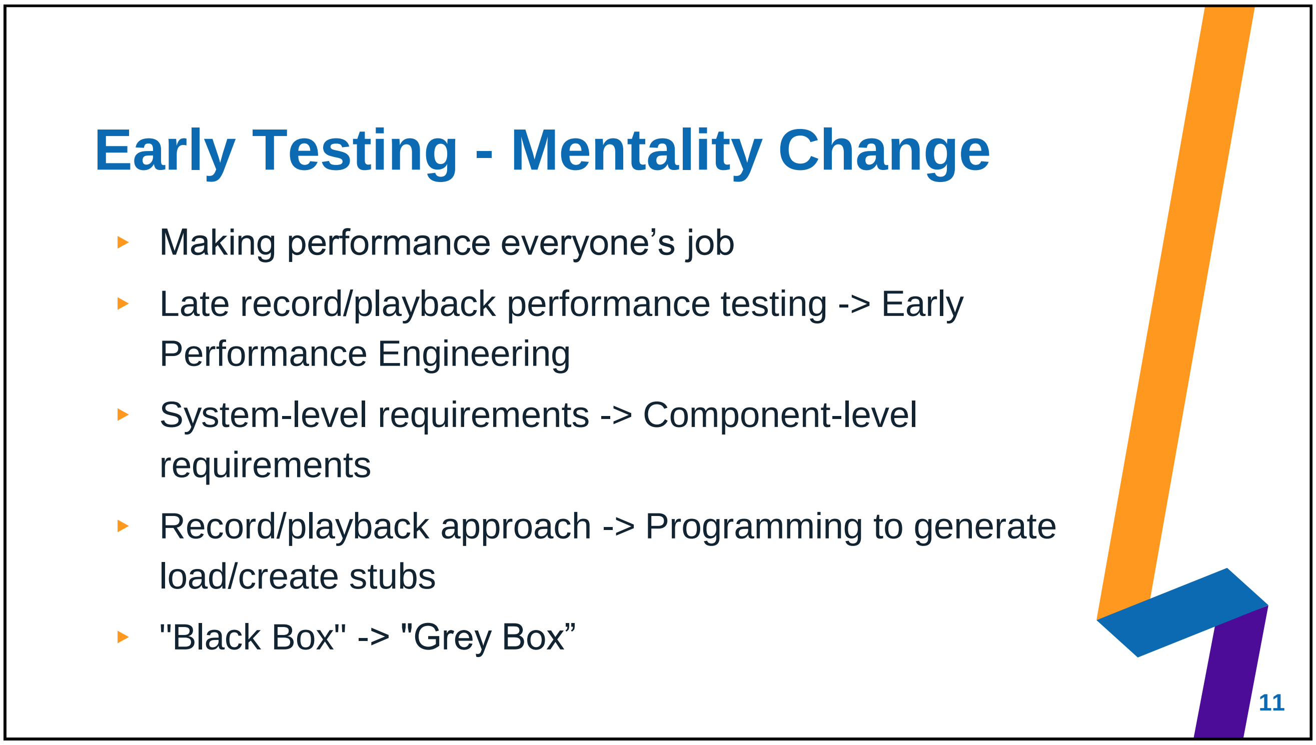
Cost of fixing defects during earlier phases of application life cycle is significantly lower



Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality.*

Paradigm Change

- Traditional performance approach don’t scale well
  - Even having competent and effective engineers
- Increased volume exposes the problem
  - Early testing
  - Each iteration
- Remedies: **automation, making performance everyone’s job**

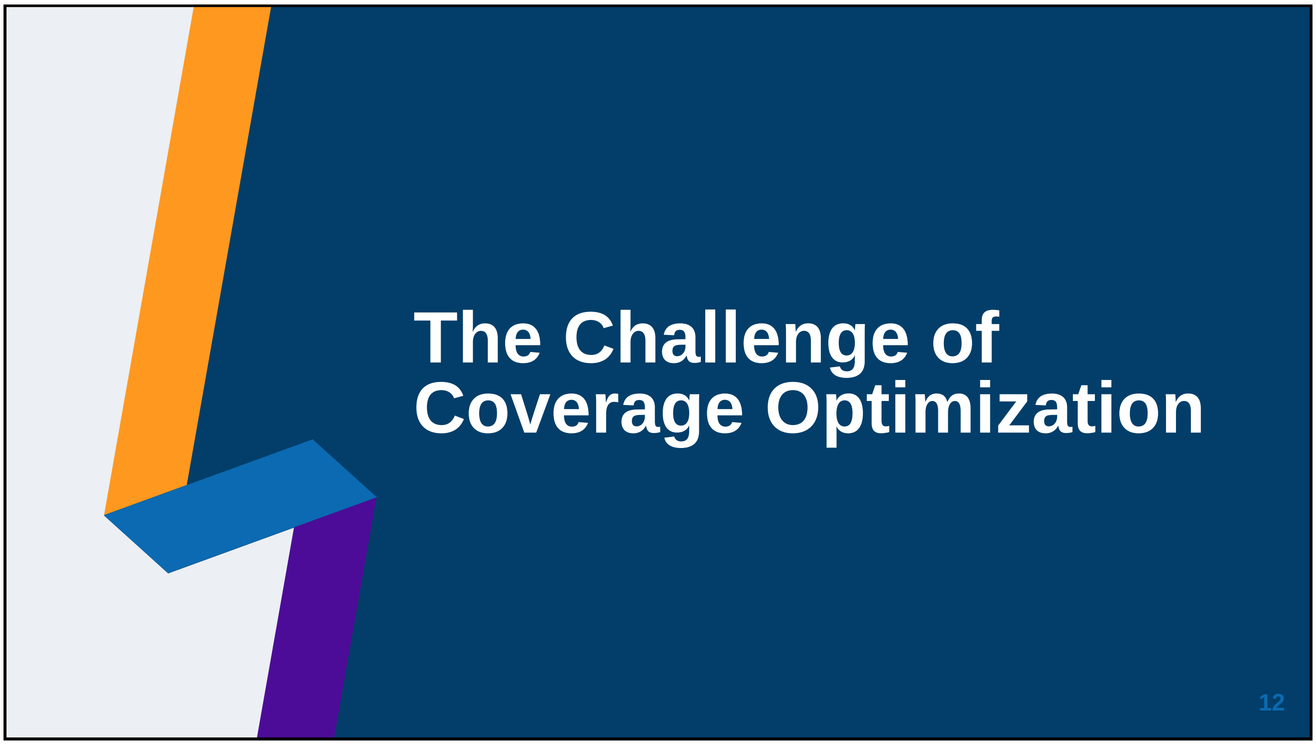


Early Testing - Mentality Change

- Making performance everyone’s job
- Late record/playback performance testing -> Early Performance Engineering
- System-level requirements -> Component-level requirements
- Record/playback approach -> Programming to generate load/create stubs
- "Black Box" -> "Grey Box”

11

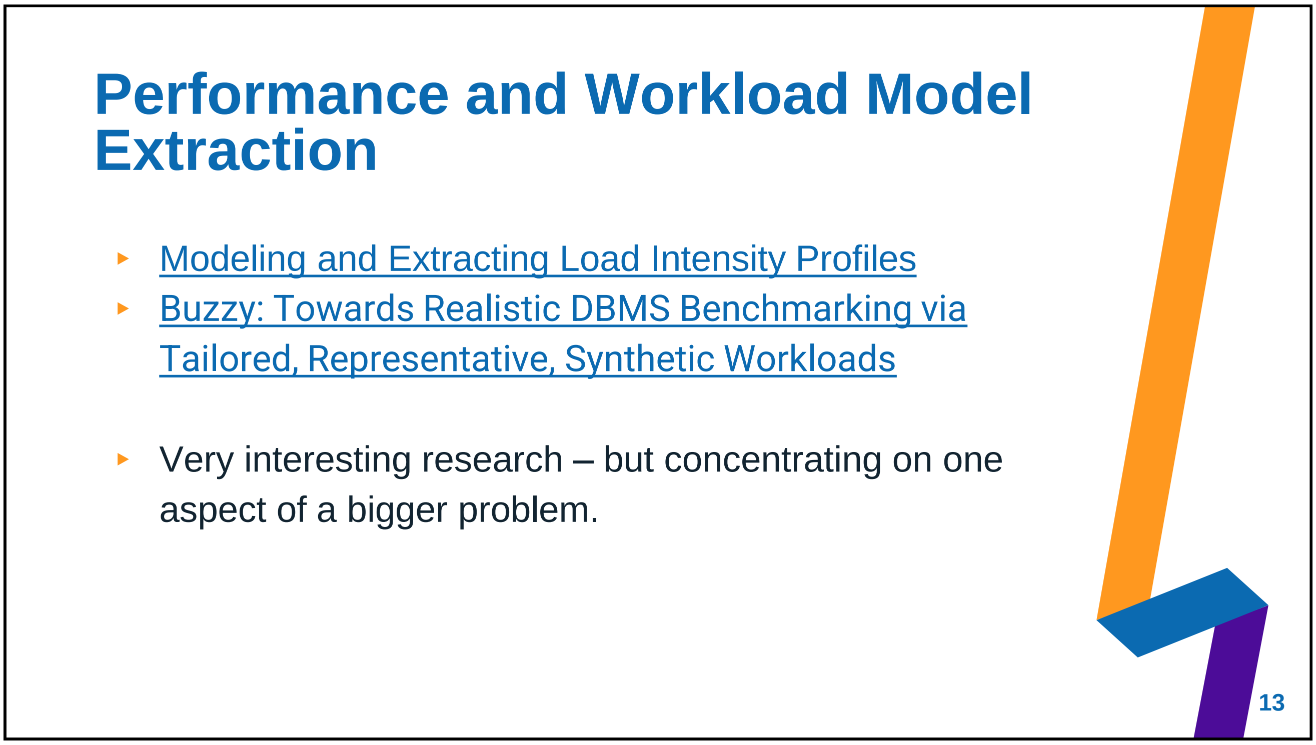
11



The Challenge of Coverage Optimization

12

12



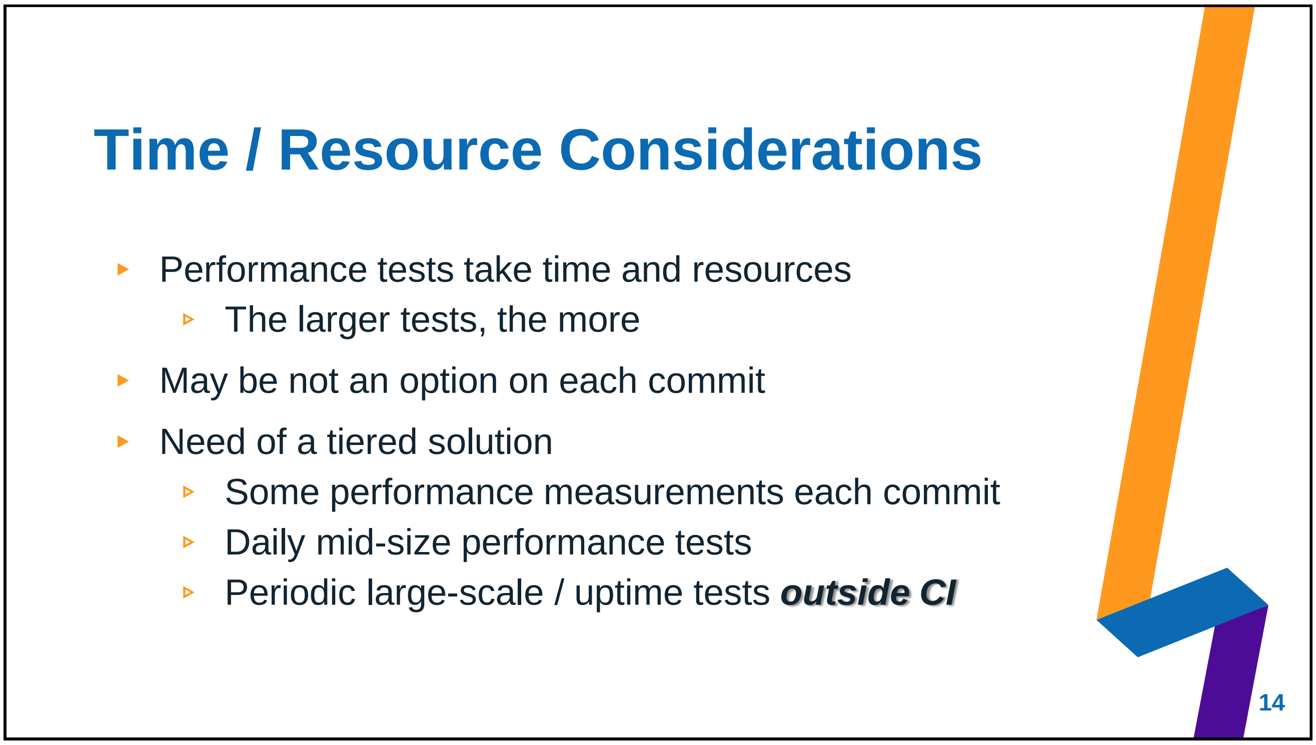
Performance and Workload Model Extraction

- Modeling and Extracting Load Intensity Profiles
- Buzzy: Towards Realistic DBMS Benchmarking via Tailored, Representative, Synthetic Workloads

- Very interesting research – but concentrating on one aspect of a bigger problem.

13

13

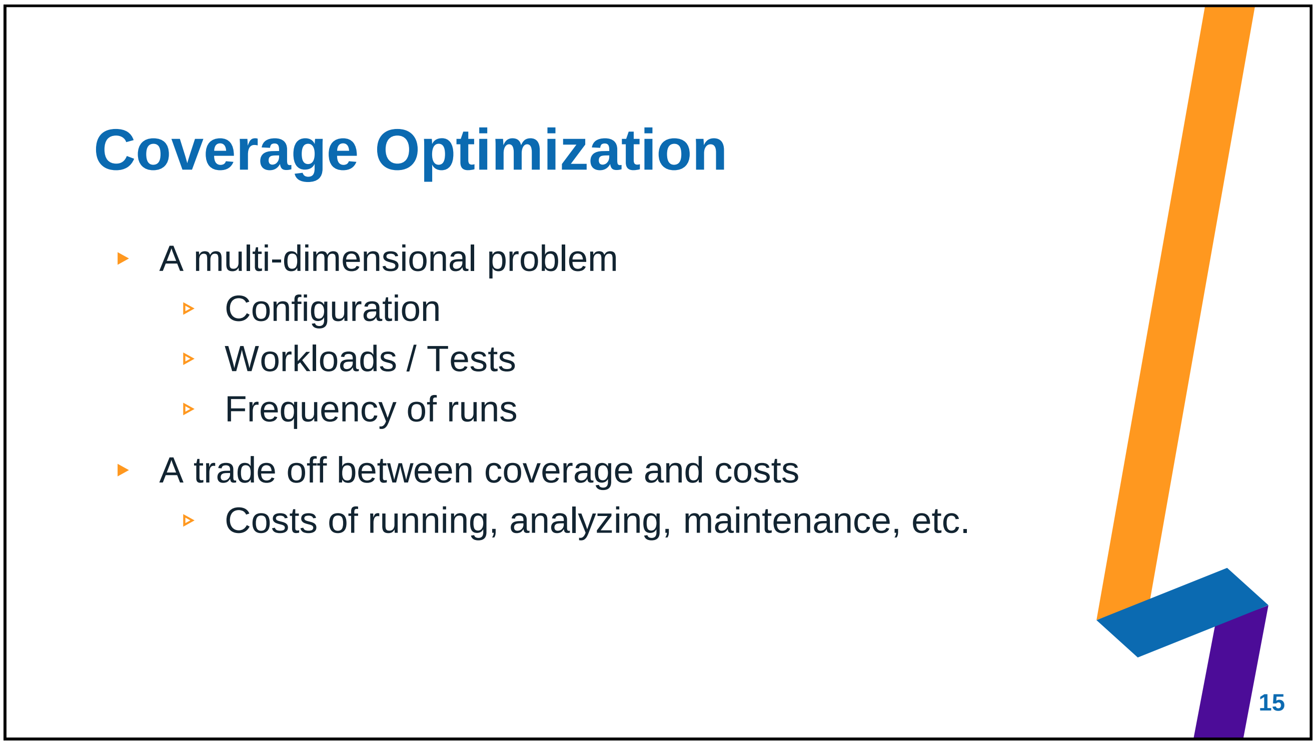


Time / Resource Considerations

- Performance tests take time and resources
  - The larger tests, the more
- May be not an option on each commit
- Need of a tiered solution
  - Some performance measurements each commit
  - Daily mid-size performance tests
  - Periodic large-scale / uptime tests ***outside CI***

14

14

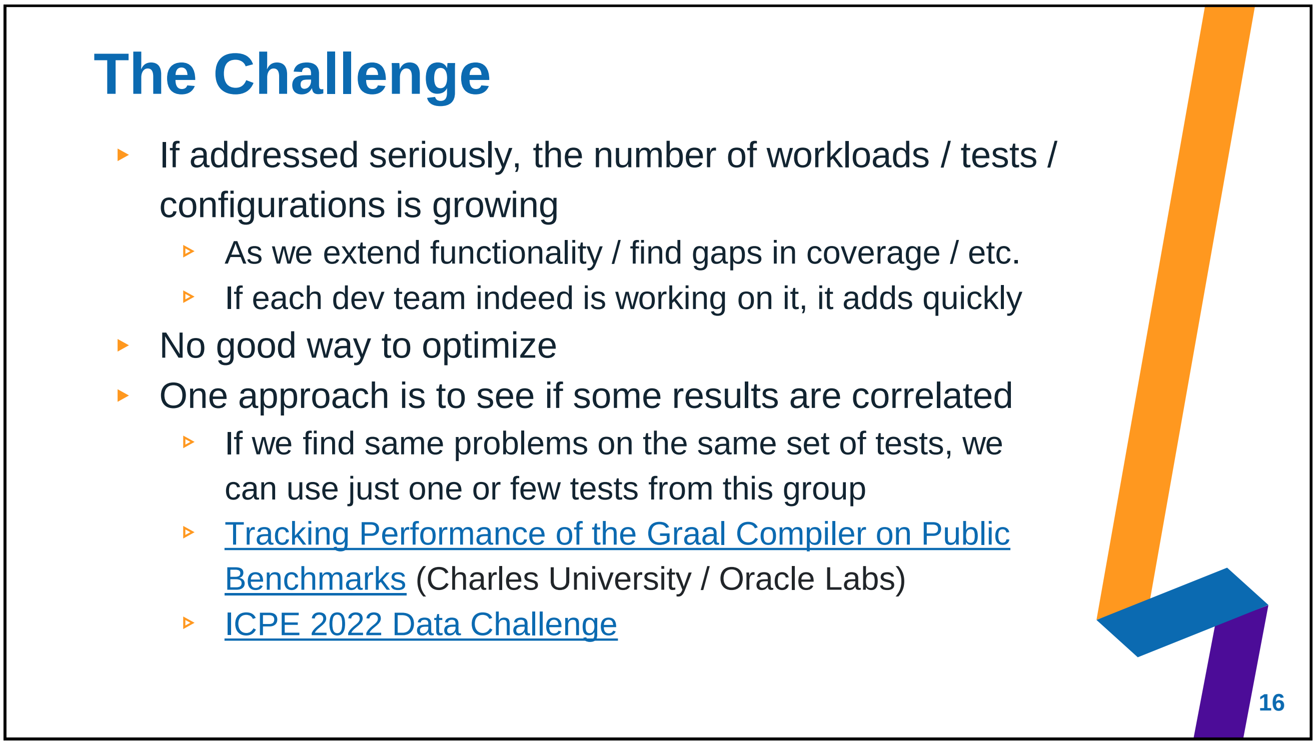


Coverage Optimization

- A multi-dimensional problem
  - Configuration
  - Workloads / Tests
  - Frequency of runs
- A trade off between coverage and costs
  - Costs of running, analyzing, maintenance, etc.

15

15



The Challenge

- If addressed seriously, the number of workloads / tests / configurations is growing
  - As we extend functionality / find gaps in coverage / etc.
  - If each dev team indeed is working on it, it adds quickly
- No good way to optimize
- One approach is to see if some results are correlated
  - If we find same problems on the same set of tests, we can use just one or few tests from this group
  - Tracking Performance of the Graal Compiler on Public Benchmarks (Charles University / Oracle Labs)
  - ICPE 2022 Data Challenge

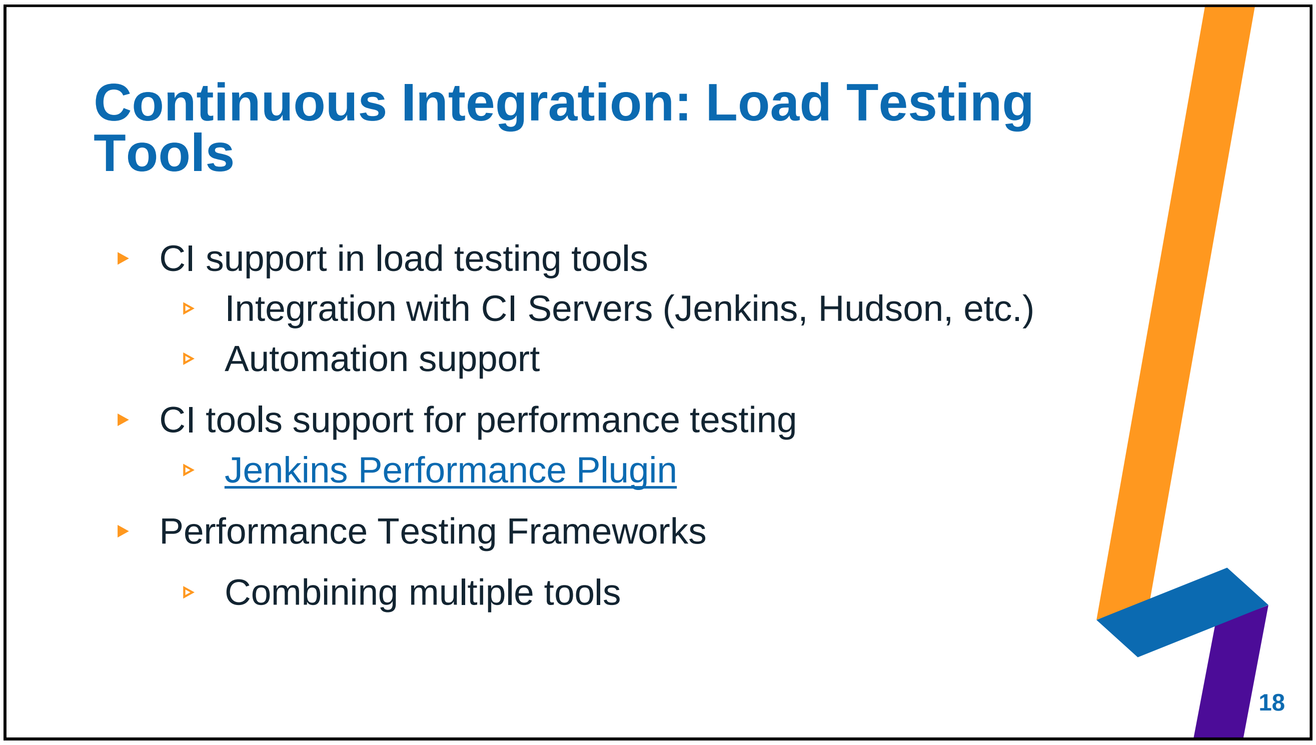
16

16

The Challenge of Integration

17

17



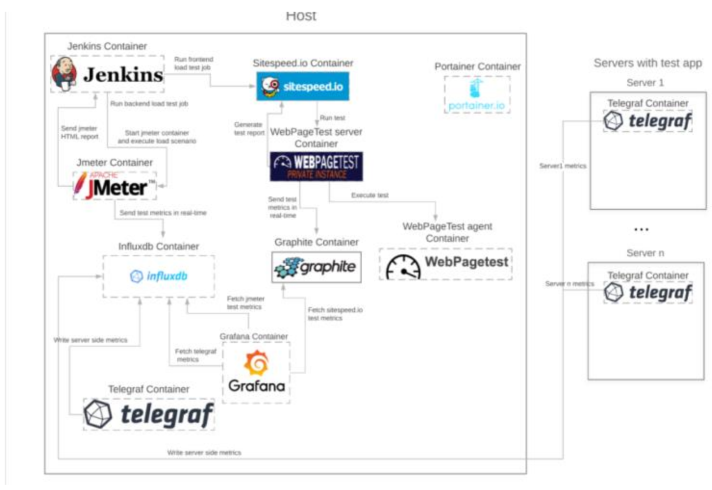
Continuous Integration: Load Testing Tools

- CI support in load testing tools
  - Integration with CI Servers (Jenkins, Hudson, etc.)
  - Automation support
- CI tools support for performance testing
  - Jenkins Performance Plugin
- Performance Testing Frameworks
  - Combining multiple tools

18

18

A Performance Testing Framework



https://github.com/serputko/performance-testing-framework

19

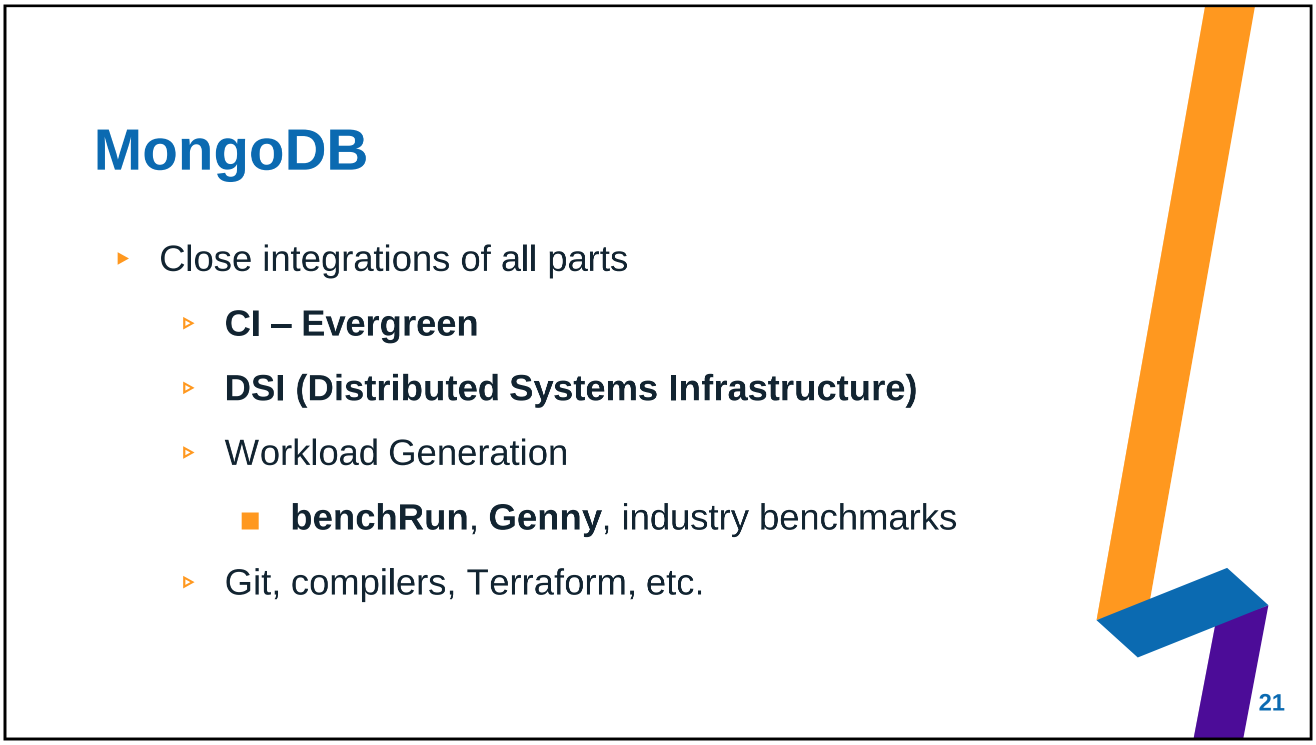
19

Closely Integrated Systems

- Creating a Virtuous Cycle in Performance Testing at MongoDB
- Fallout: Distributed Systems Testing as a Service (DataStax)
- Tracking Performance of the Graal Compiler on Public Benchmarks (Charles University / Oracle Labs)
- Introducing Ballast: An Adaptive Load Test Framework (Uber)

20

20



MongoDB

- Close integrations of all parts
  - **CI – Evergreen**
  - **DSI (Distributed Systems Infrastructure)**
  - Workload Generation
    - **benchRun**, **Genny**, industry benchmarks
  - Git, compilers, Terraform, etc.

21

21

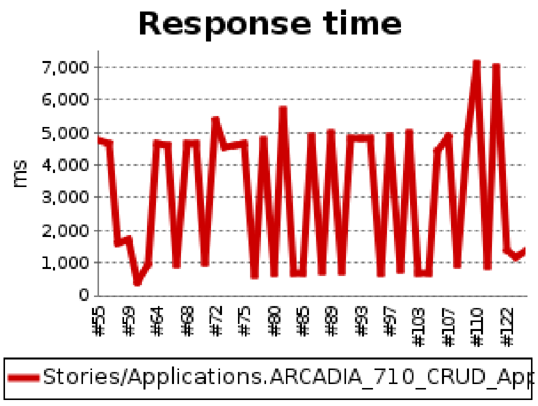
The Challenge of Variability

22

22

Variability - Environment

- Due to difference in environments

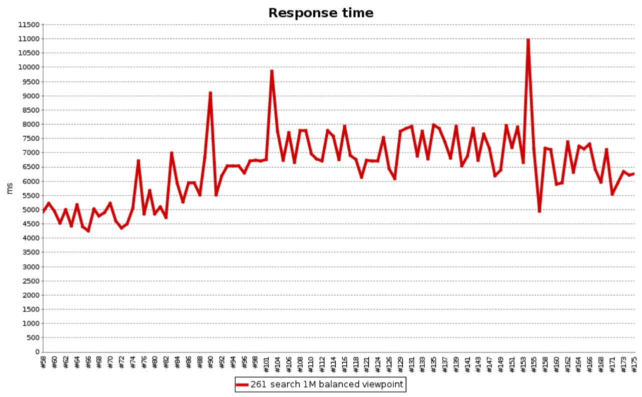


23

23

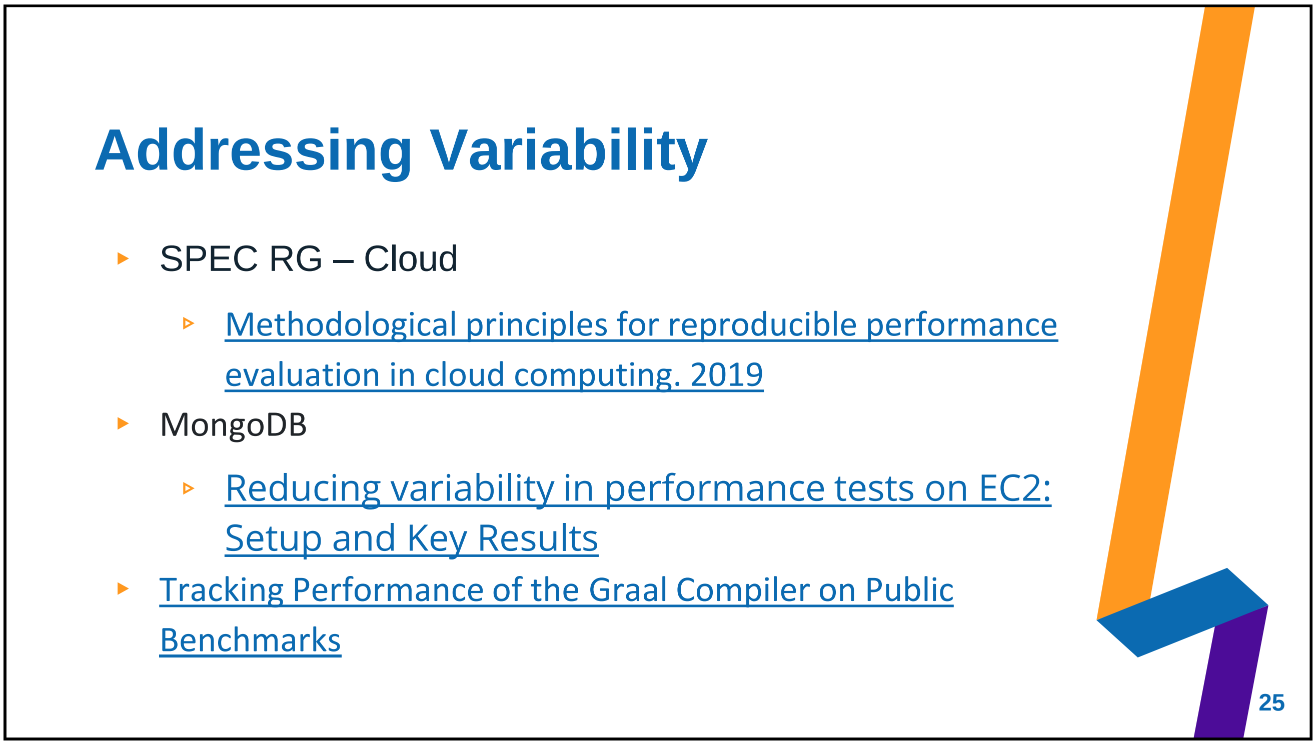
Variability - System

- Inherent to the test setup



24

24

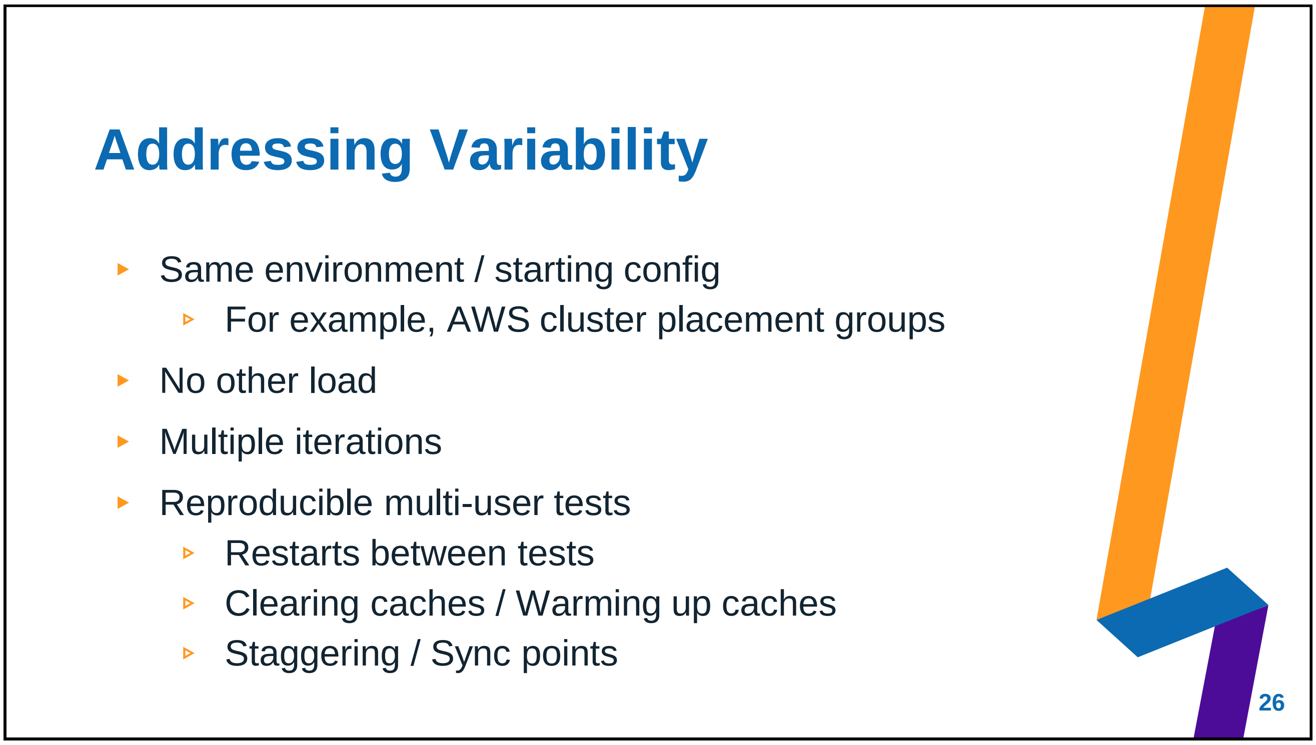


Addressing Variability

- SPEC RG – Cloud
  - Methodological principles for reproducible performance evaluation in cloud computing. 2019
- MongoDB
  - Reducing variability in performance tests on EC2: Setup and Key Results
- Tracking Performance of the Graal Compiler on Public Benchmarks

25

25



Addressing Variability

- Same environment / starting config
  - For example, AWS cluster placement groups
- No other load
- Multiple iterations
- Reproducible multi-user tests
  - Restarts between tests
  - Clearing caches / Warming up caches
  - Staggering / Sync points

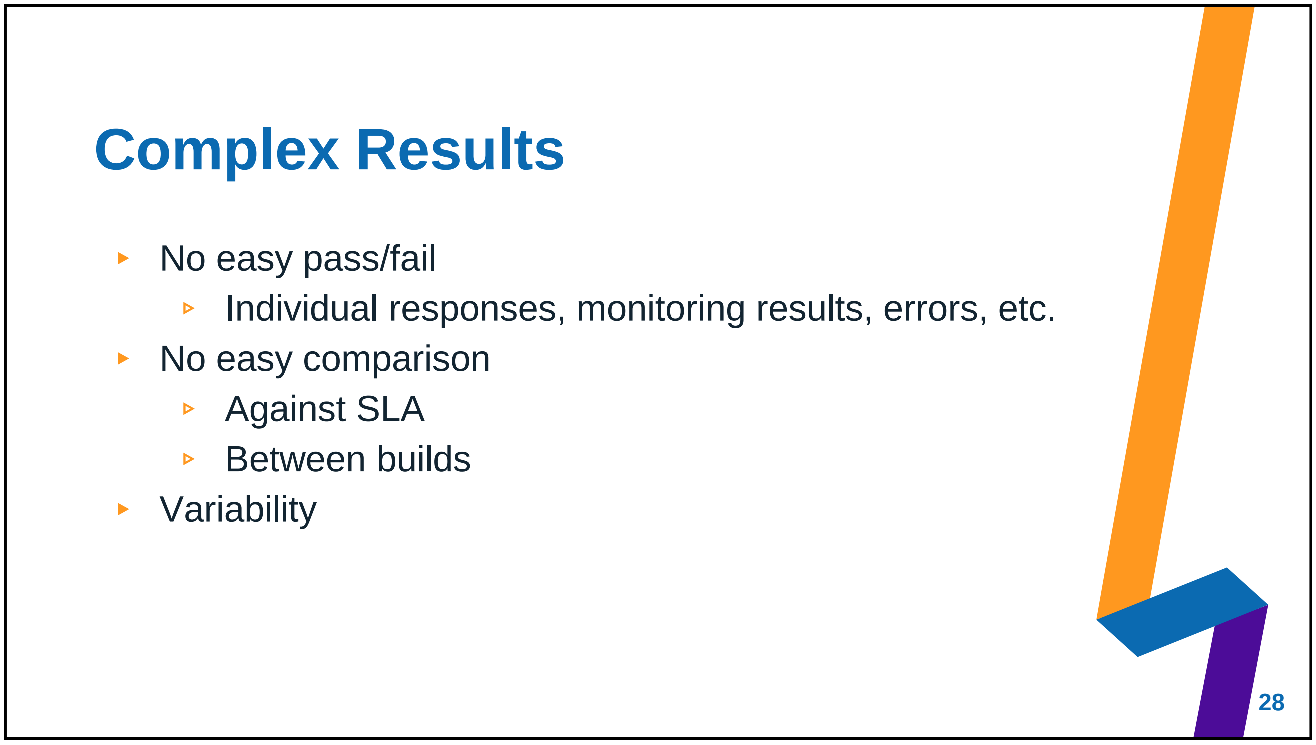
26

26

The Challenge of Change Detection

27

27



Complex Results

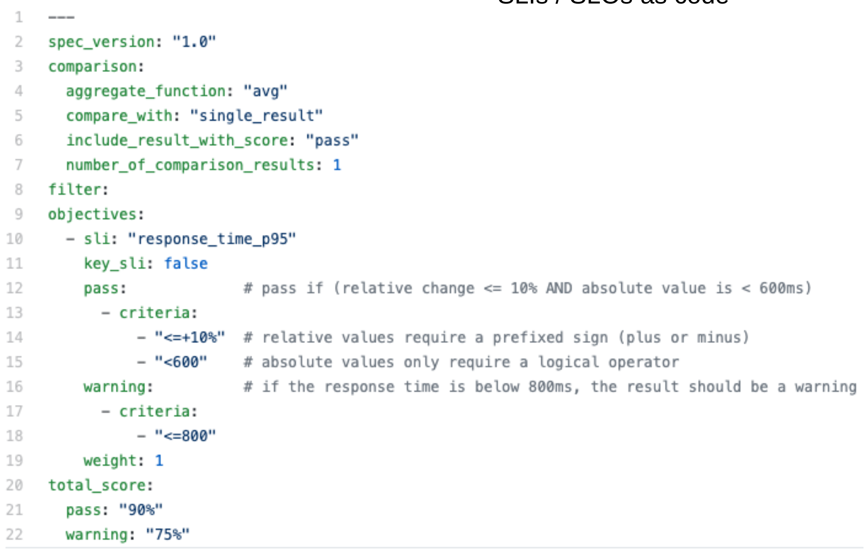
- No easy pass/fail
  - Individual responses, monitoring results, errors, etc.
- No easy comparison
  - Against SLA
  - Between builds
- Variability

28

28

keptn.sh

Quality Gates
SLIs / SLOs as code



```
---
spec_version: "1.0"
comparison:
  aggregate_function: "avg"
  compare_with: "single_result"
  include_result_with_score: "pass"
  number_of_comparison_results: 1
filter:
objectives:
  - sli: "response_time_p95"
    key_sli: false
    pass:             # pass if (relative change <= 10% AND absolute value is < 600ms)
      - criteria:
          - "<=+10%"  # relative values require a prefixed sign (plus or minus)
          - "<600"    # absolute values only require a logical operator
    warning:          # if the response time is below 800ms, the result should be a warning
      - criteria:
          - "<=800"
    weight: 1
total_score:
  pass: "90%"
  warning: "75%"
```

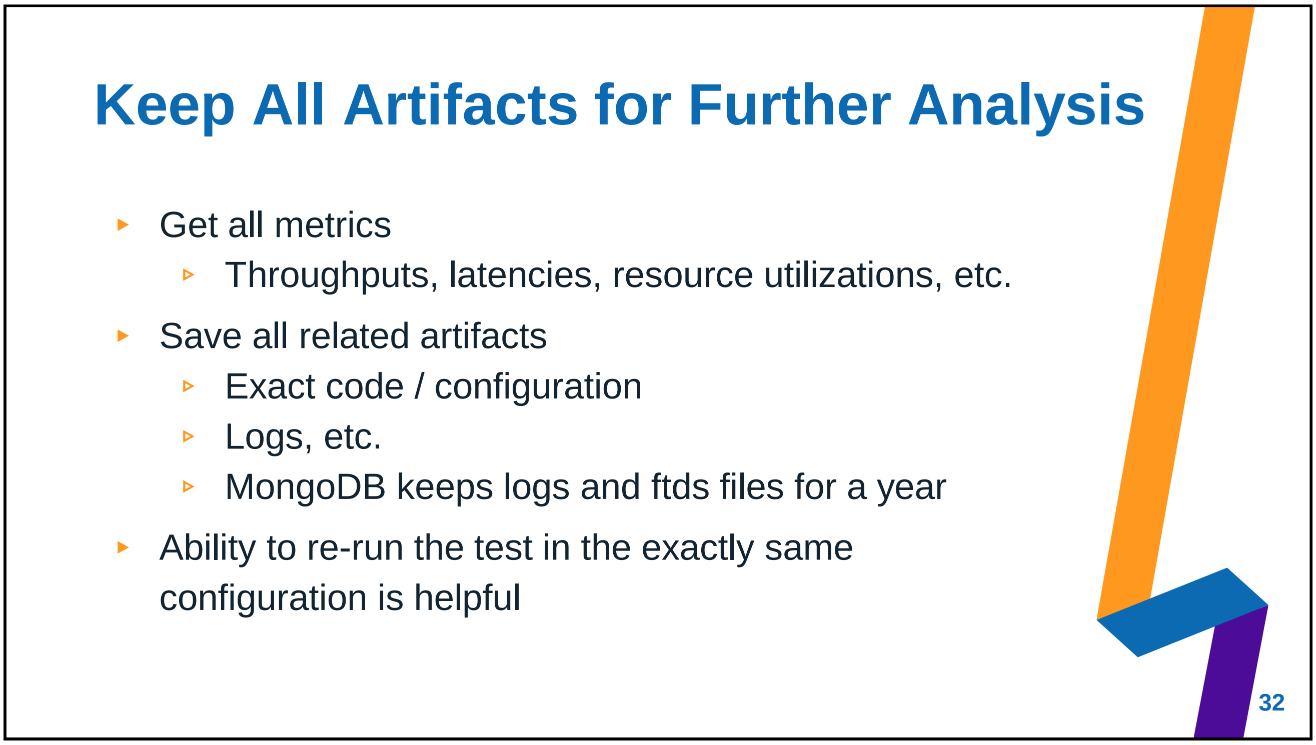
Change Point Detection

- Statistical methods taking noise in consideration
  - E-Divisive means algorithm
    - See ICPE Paper: Change Point Detection in Software Performance Testing
      - Fixing Performance Regressions Before they Happen, Netflix Technology Blog
    - https://github.com/mongodb/signal-processing-algorithms
      - Open sourced, generic
    - Need several data points. May miss a gradual degradation.
    - ICPE 2022 Data Challenge

The Challenge of Advanced Analysis

31

31



Keep All Artifacts for Further Analysis

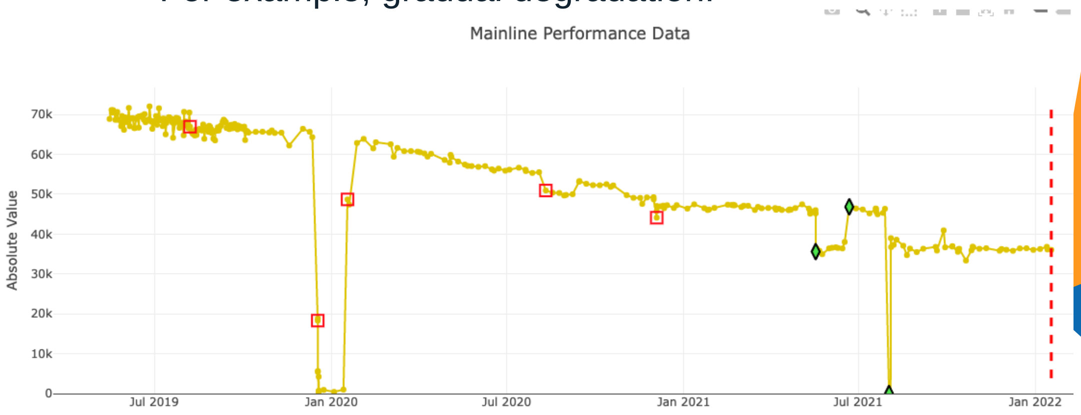
- Get all metrics
  - Throughputs, latencies, resource utilizations, etc.
- Save all related artifacts
  - Exact code / configuration
  - Logs, etc.
  - MongoDB keeps logs and ftds files for a year
- Ability to re-run the test in the exactly same configuration is helpful

32

32

Visualization

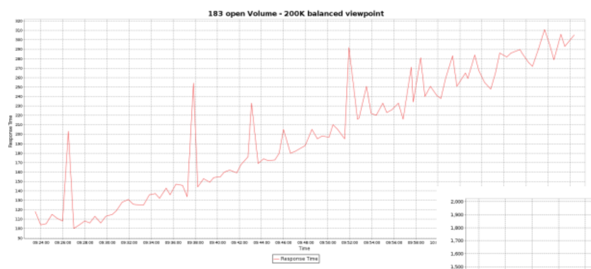
- [Visualizing systems and software performance - Report on the GI-Dagstuhl]()
- Sometimes helps to catch an issue
  - For example, gradual degradation:

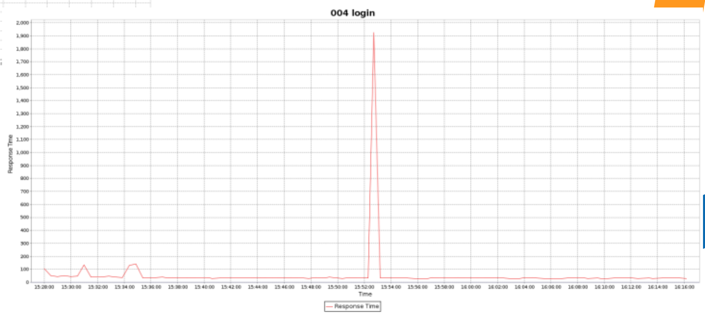


33

33

Looking Beyond Aggregate Info



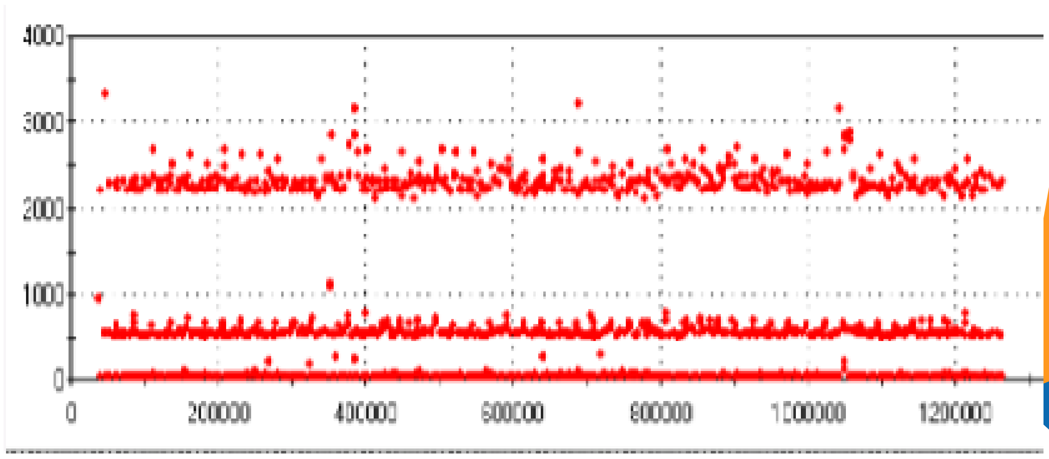


34

34

Looking at Individual Results Patterns

Scatter charts – a "banding" pattern from http://www.perftestplus.com/resources/BPT6.pdf



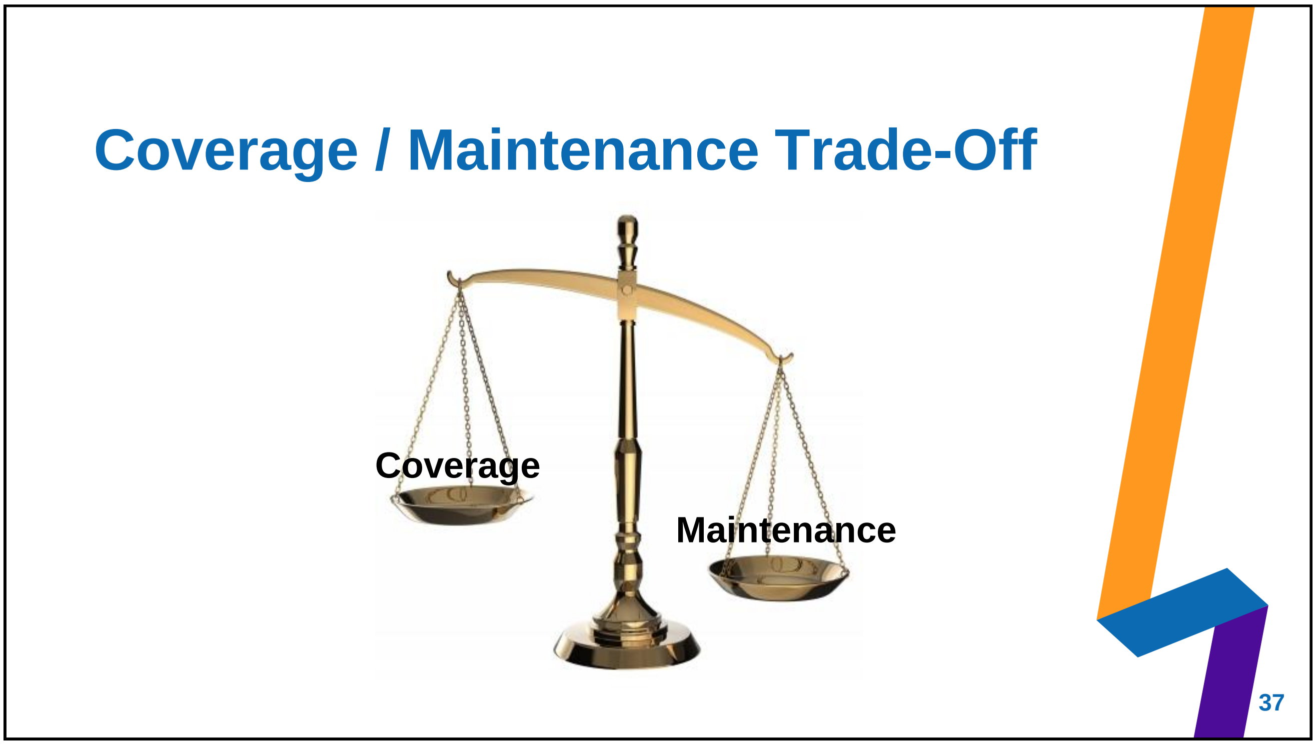
35

35

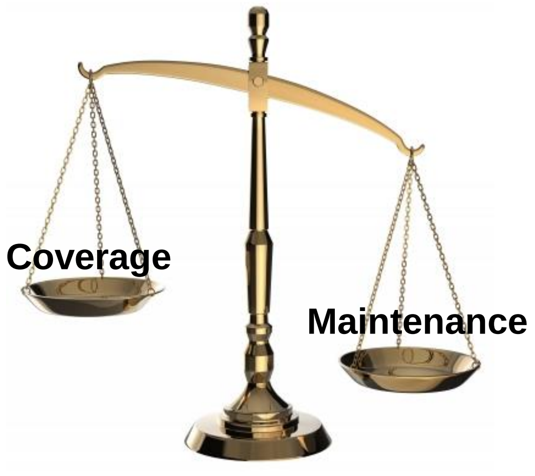
The Challenge of Maintenance

36

36

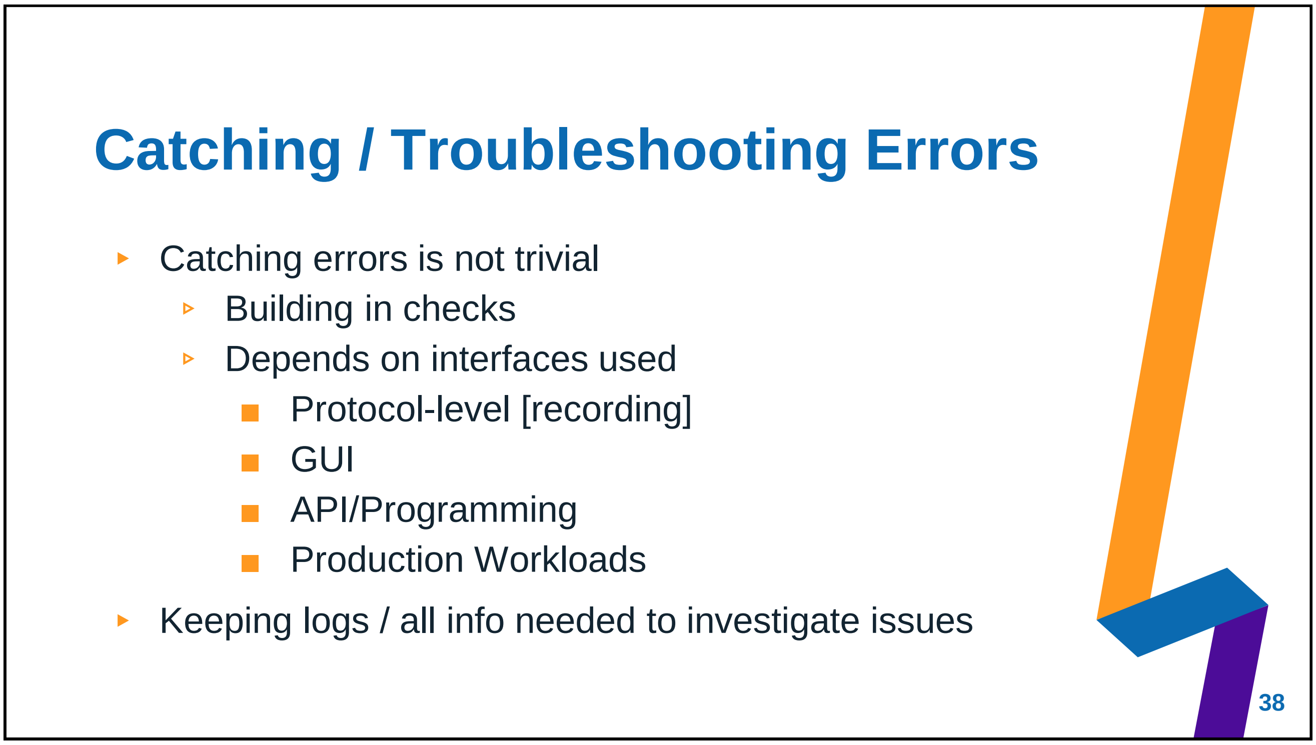


Coverage / Maintenance Trade-Off



37

37

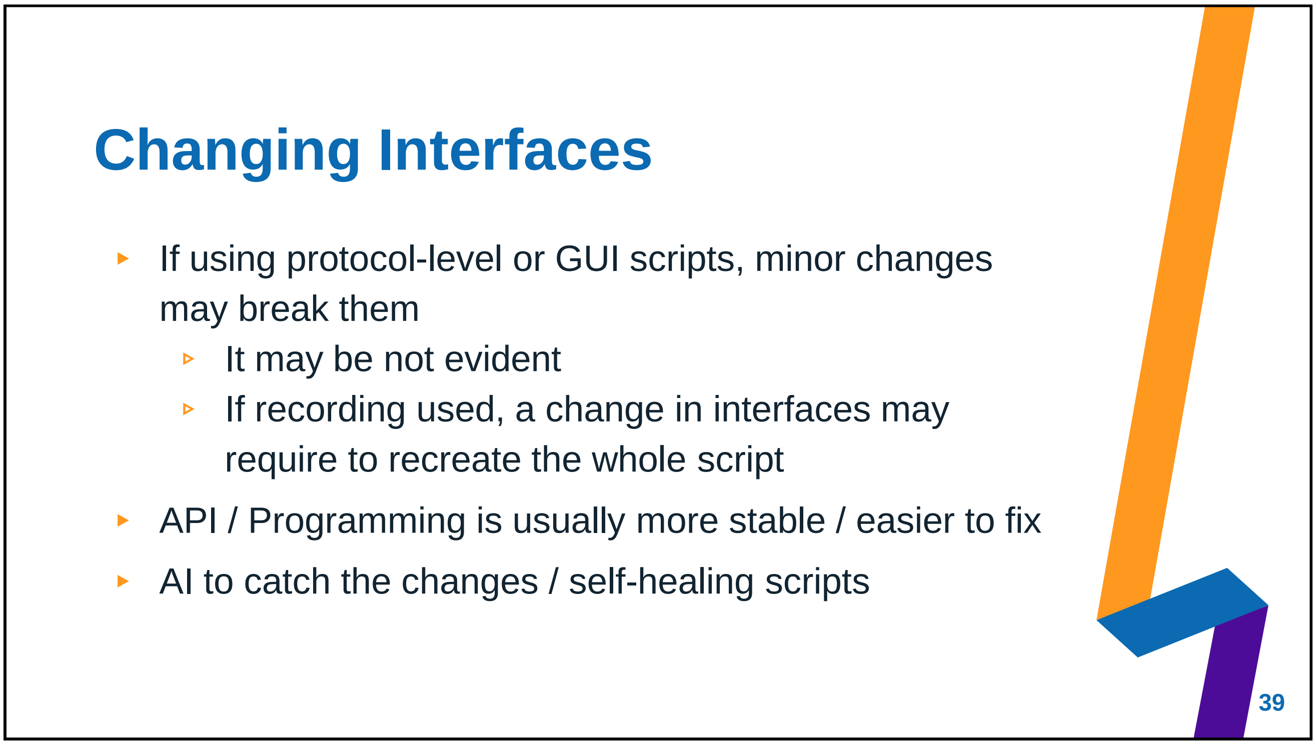


Catching / Troubleshooting Errors

- Catching errors is not trivial
  - Building in checks
  - Depends on interfaces used
    - Protocol-level [recording]
    - GUI
    - API/Programming
    - Production Workloads
- Keeping logs / all info needed to investigate issues

38

38

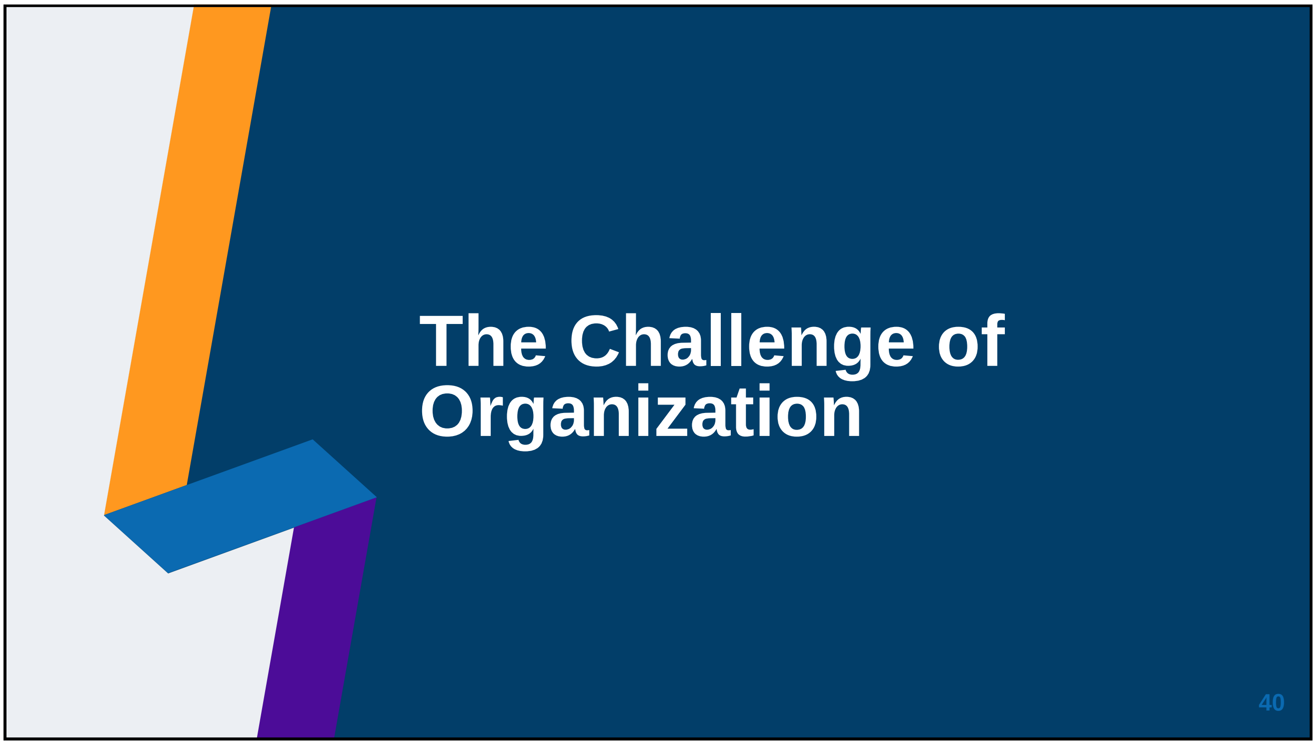


Changing Interfaces

- If using protocol-level or GUI scripts, minor changes may break them
  - It may be not evident
  - If recording used, a change in interfaces may require to recreate the whole script
- API / Programming is usually more stable / easier to fix
- AI to catch the changes / self-healing scripts

39

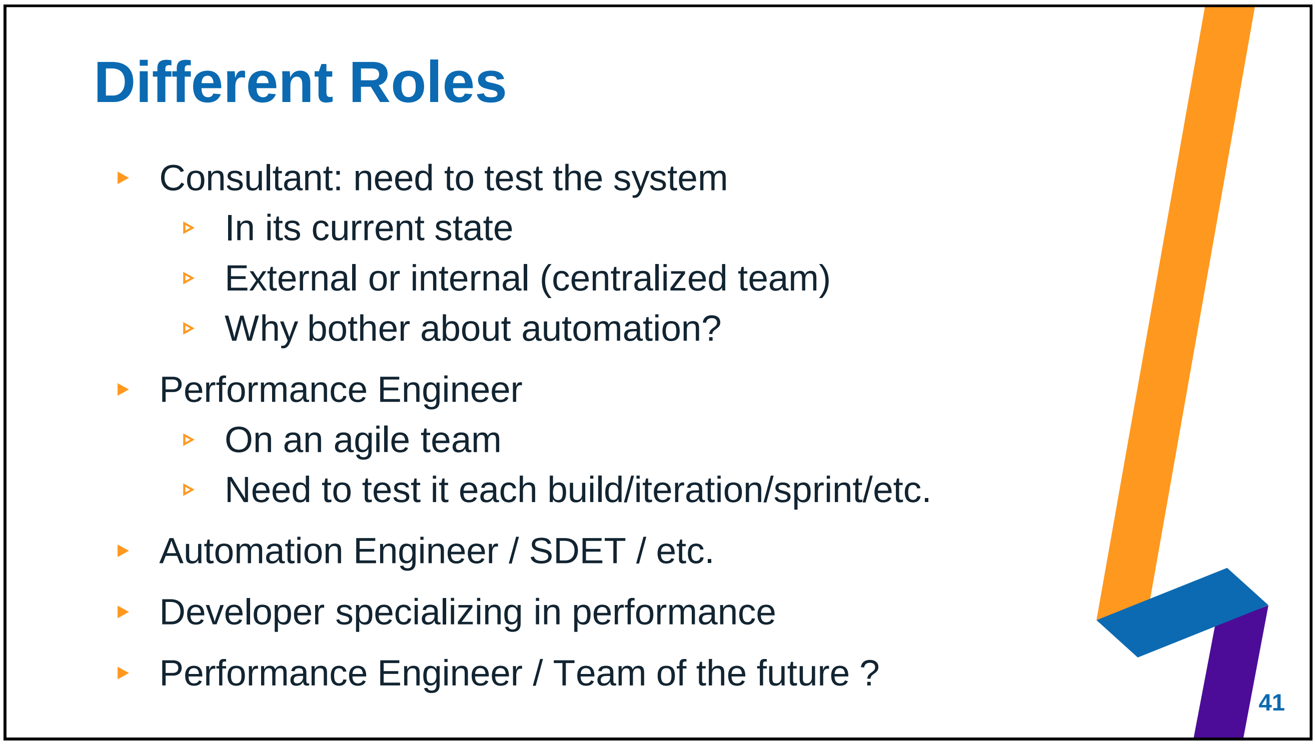
39



The Challenge of Organization

40

40

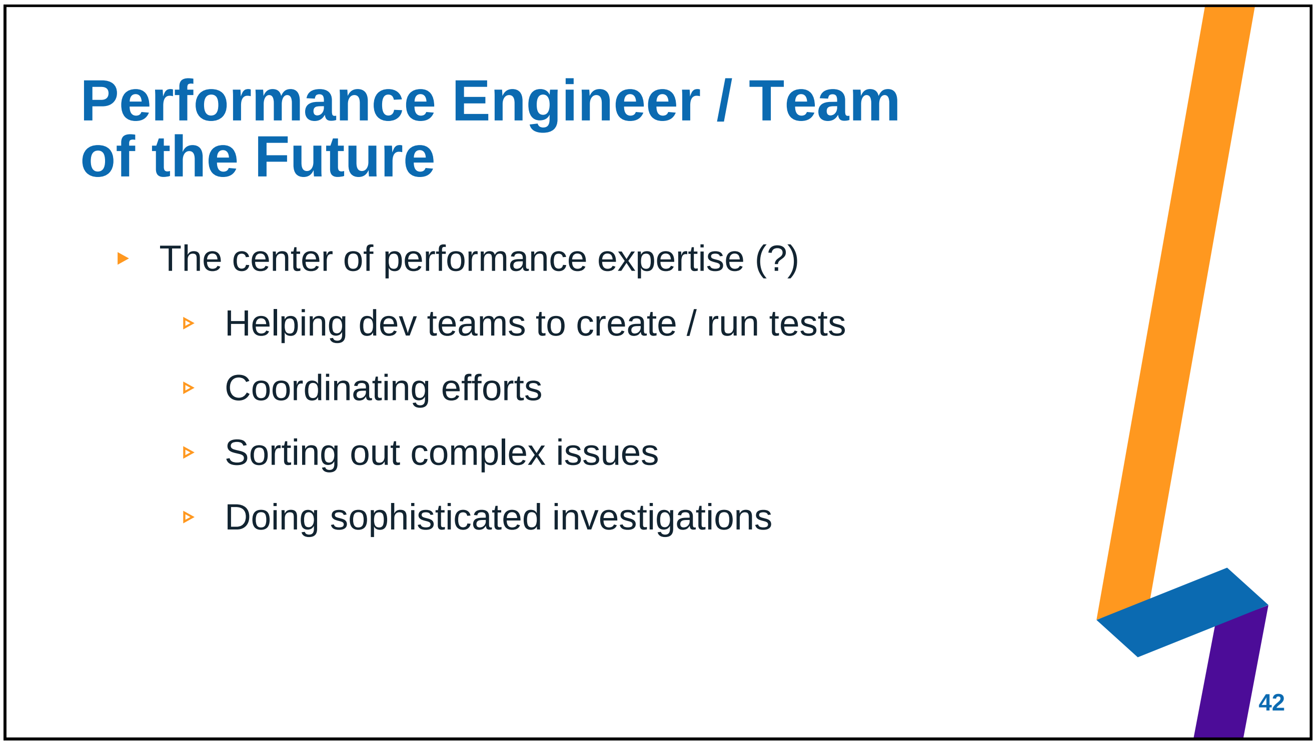


Different Roles

- Consultant: need to test the system
  - In its current state
  - External or internal (centralized team)
  - Why bother about automation?
- Performance Engineer
  - On an agile team
  - Need to test it each build/iteration/sprint/etc.
- Automation Engineer / SDET / etc.
- Developer specializing in performance
- Performance Engineer / Team of the future ?

41

41

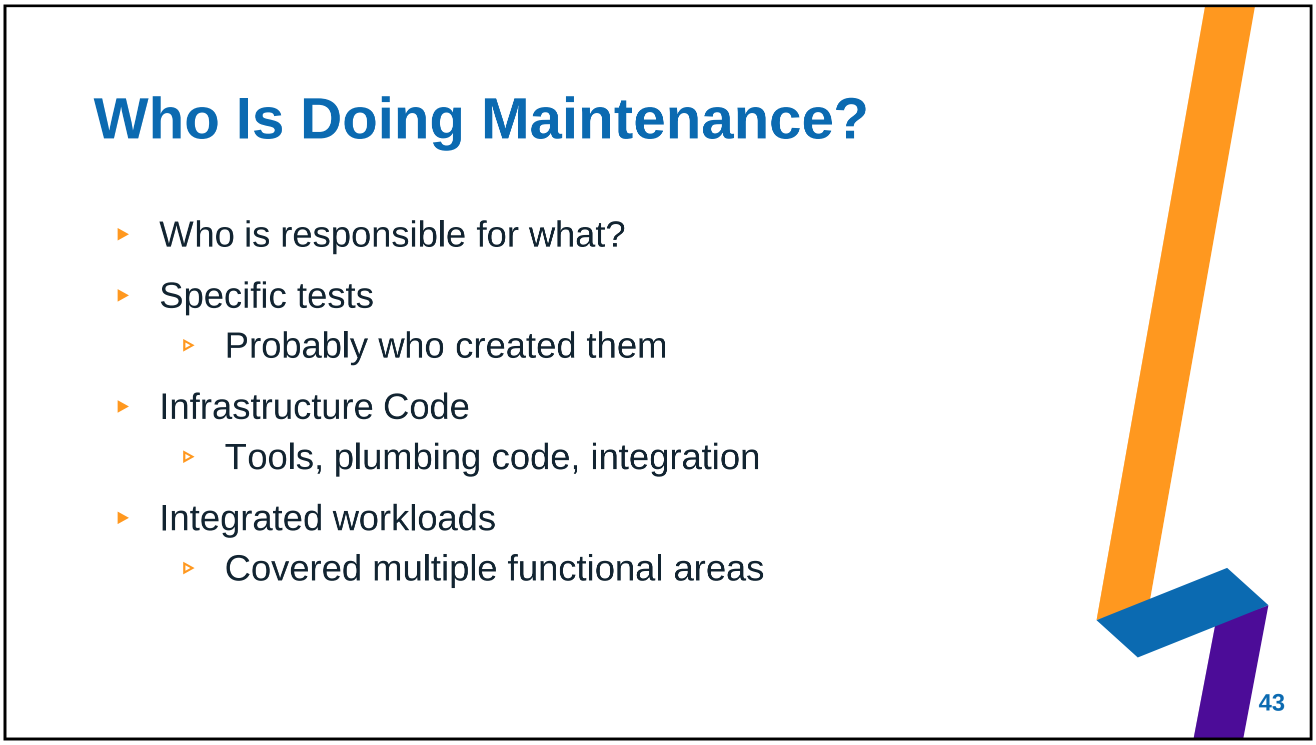


Performance Engineer / Team of the Future

- The center of performance expertise (?)
  - Helping dev teams to create / run tests
  - Coordinating efforts
  - Sorting out complex issues
  - Doing sophisticated investigations

42

42

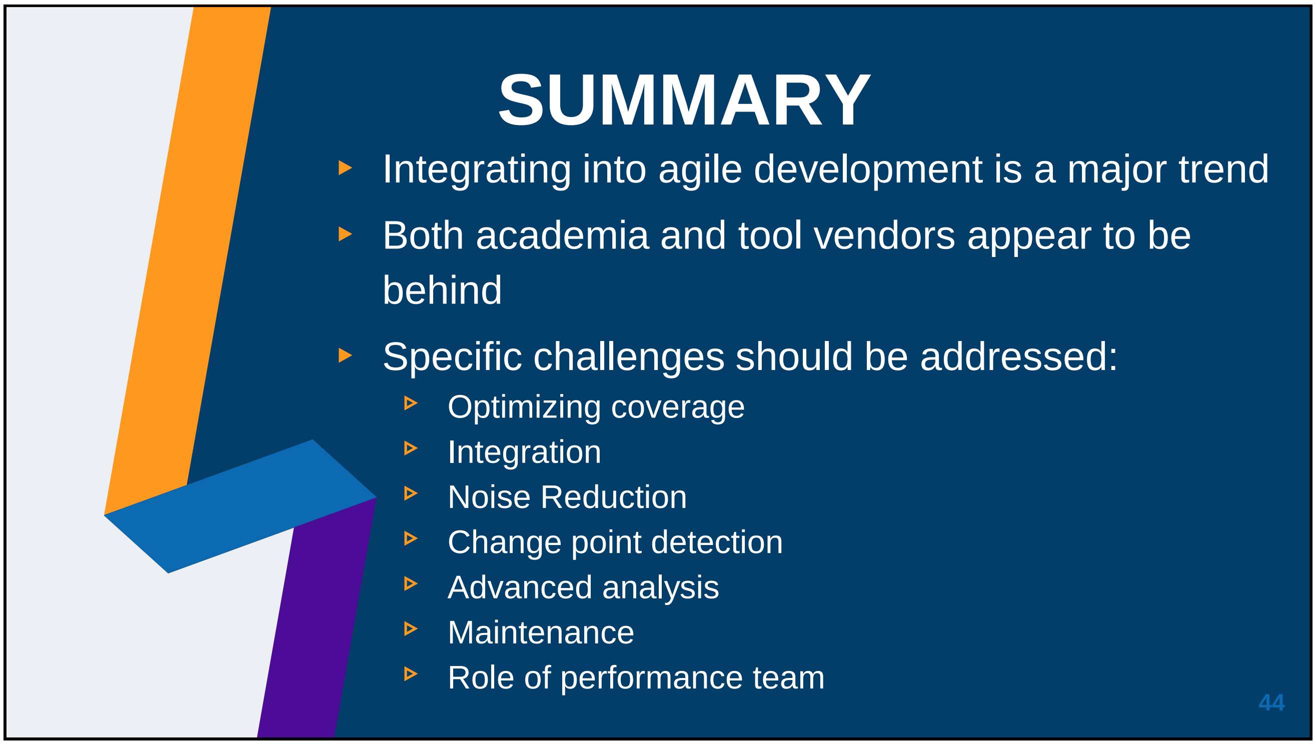


Who Is Doing Maintenance?

- Who is responsible for what?
- Specific tests
  - Probably who created them
- Infrastructure Code
  - Tools, plumbing code, integration
- Integrated workloads
  - Covered multiple functional areas

43

43



SUMMARY

- Integrating into agile development is a major trend
- Both academia and tool vendors appear to be behind
- Specific challenges should be addressed:
  - Optimizing coverage
  - Integration
  - Noise Reduction
  - Change point detection
  - Advanced analysis
  - Maintenance
  - Role of performance team

44

44

Questions ?

apodelko@yahoo.com
@apodelko
https://alexanderpodelko.com/blog/
https://www.linkedin.com/in/alexanderpodelko/

45

45