# Continuous Performance

## Load testing for developers with Gatling

Tim van Eijndhoven

@TimvEijndhoven

# Let's meet

Tim
van Eijndhoven

Bert Jan
Schrijver

jpoint

MALMBERG
a Sanoma company

.nl.
jug

# Outline

- Performance testing process
- Introduction to Gatling
- Demo
- Results
- Looking forward
- Summary
- Q&A

# How it all started…

Performance testing should be part of the process

# Performance testing traditionally…

…happens several times per year

…and/or at major releases

…is performed by specialists

**.. which is far from ideal:**

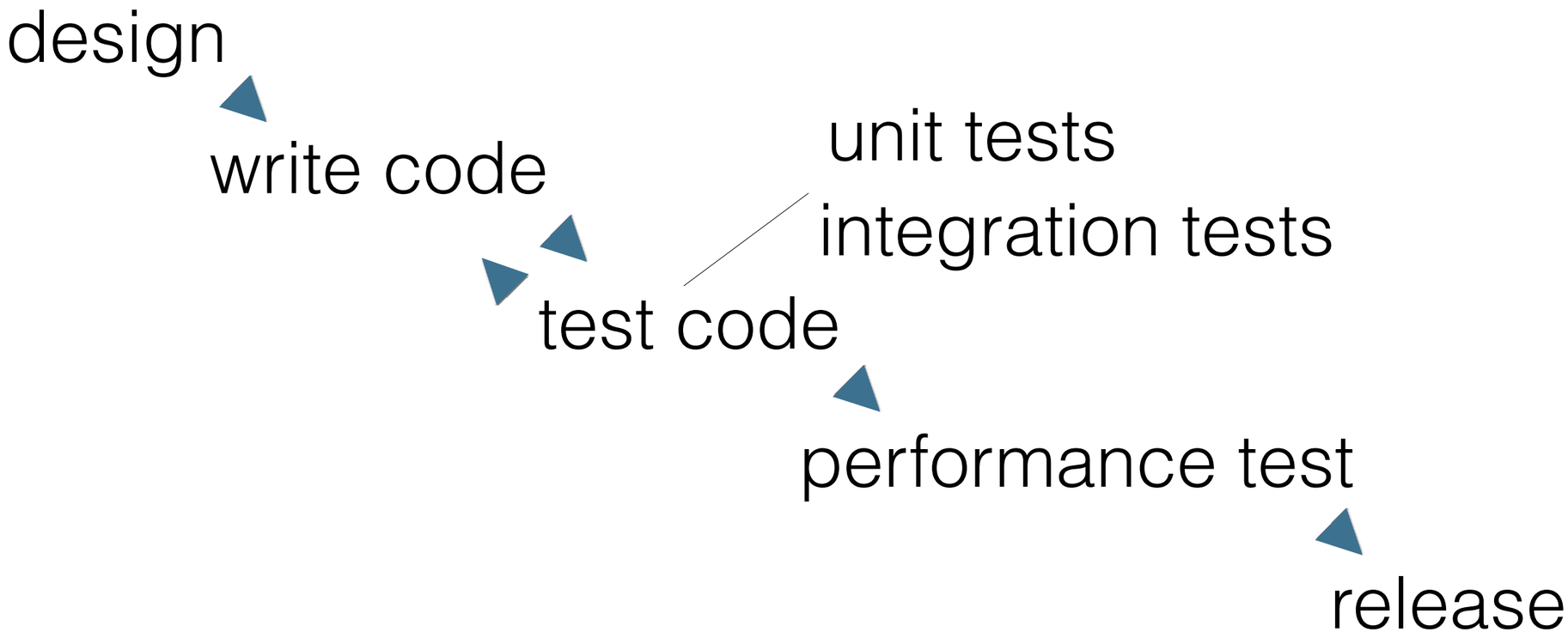
- changes were made long ago

- many different code changes

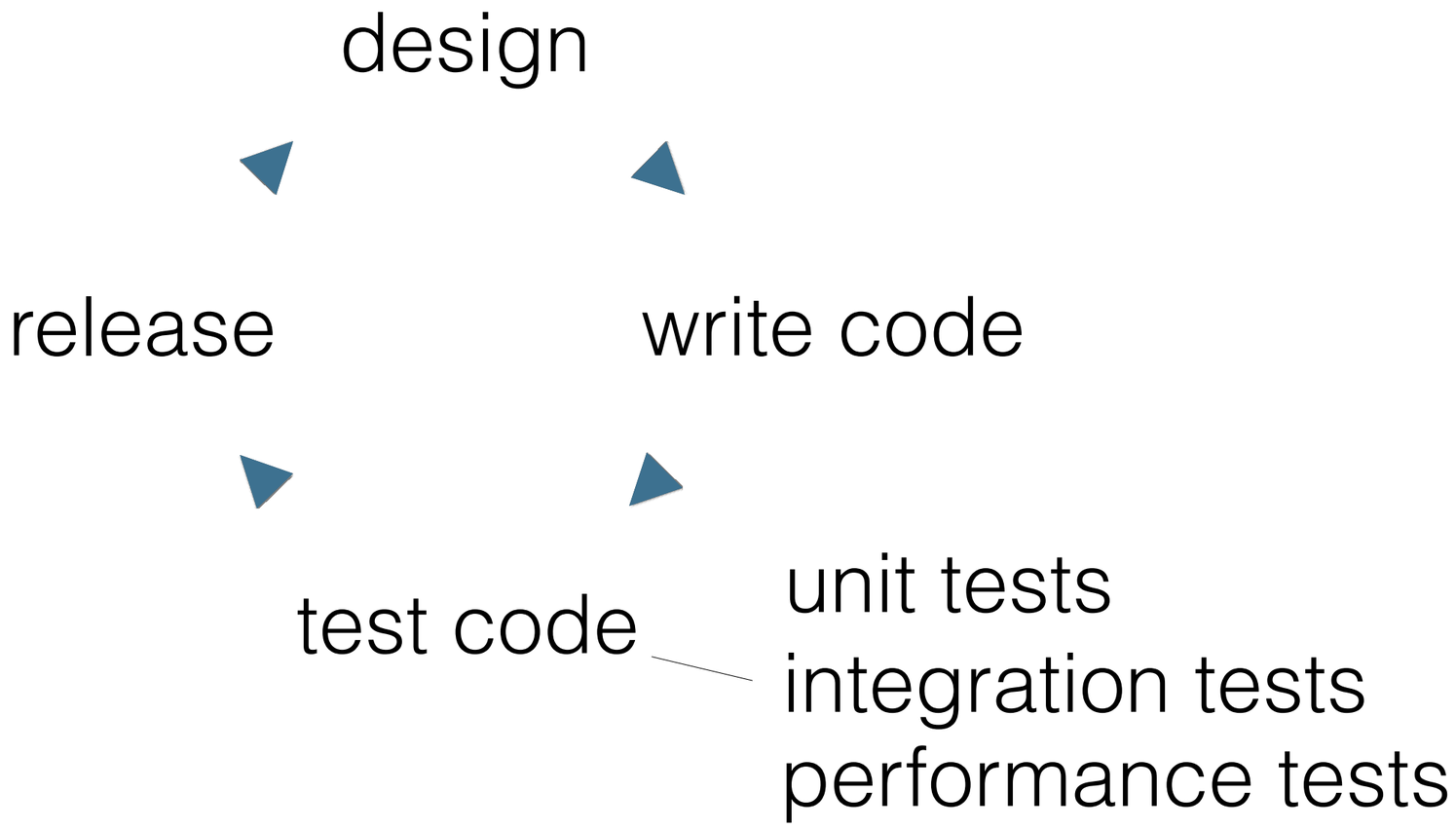
- at a certain moment in time

- when is a test required?

# Traditional performance testing



# Performance testing in continuous delivery



# Continuous Delivery

**Demands code to be**

Always production ready

Short feedback cycles

Maintained by self-supporting teams

**In regard to performance**

Has to be under control

Effects should be clear ASAP

No external specialists

# Part of the process

**With the same level of support as**

- Unit tests and integration tests

- Continuous Integration

- Zero-downtime deployments

Performed by the development team

# Performance testing process

Design ▶ Record ▶ Operationalise ▶ Execute ▶ Report

# Designing scenarios

Generic tests used to test core functionality

Specialised tests used to test specific features

# Tool support is key for performance test adoption

# Gatling

```scala
package computerdatabase // 1

import io.gatling.core.Predef._ // 2
import io.gatling.http.Predef._
import scala.concurrent.duration._

class BasicSimulation extends Simulation { // 3

  val httpConf = http // 4
    .baseURL("http://computer-database.gatling.io") // 5
    .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8") // 6
    .doNotTrackHeader("1")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .acceptEncodingHeader("gzip, deflate")
    .userAgentHeader("Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.0")

  val scn = scenario("BasicSimulation") // 7
    .exec(http("request_1")  // 8
    .get("/")) // 9
    .pause(5) // 10

  setUp( // 11
    scn.inject(atOnceUsers(1)) // 12
  ).protocols(httpConf) // 13
}
```



# Alternative tools



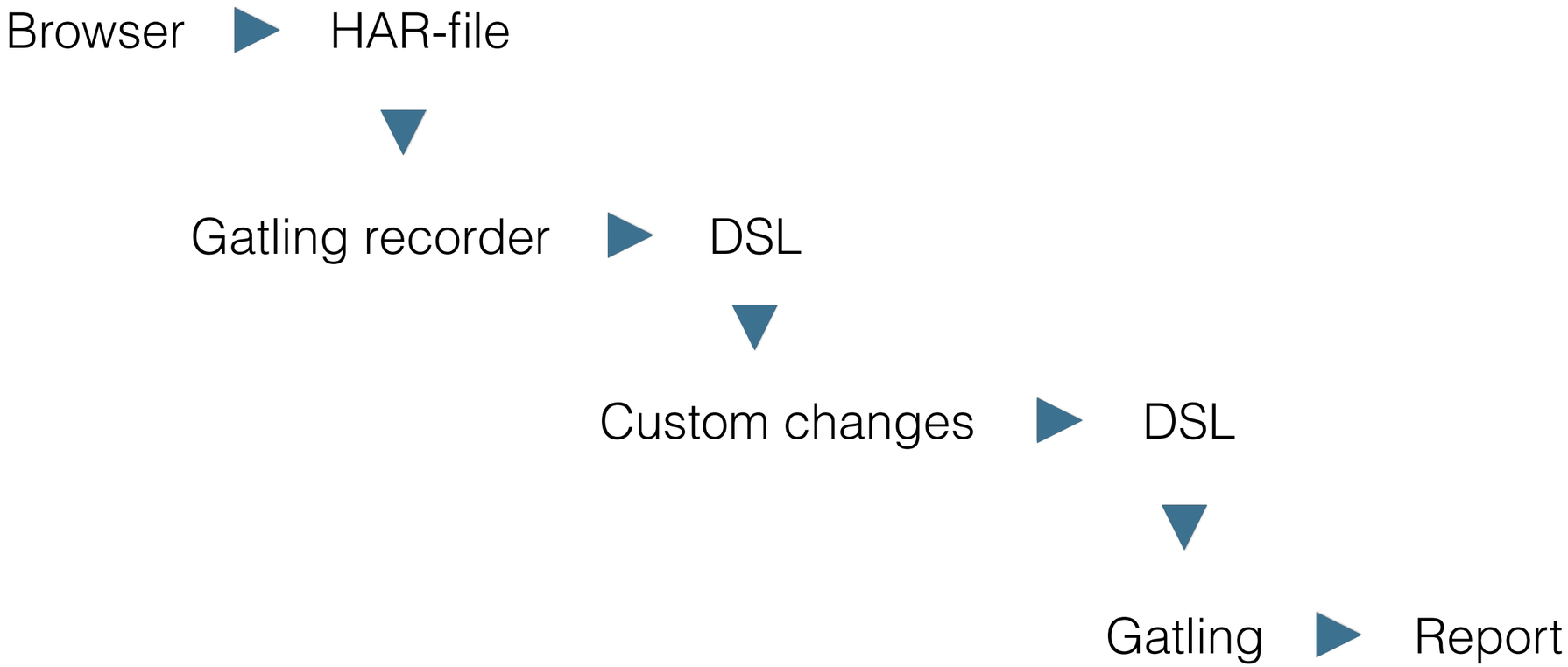
And many more…

# Gatling core concepts

| | |
|---|---|
| **Gatling DSL** | Easy-to-read, developer-friendly way of defining tests |
| **Scenario** | A sequence of http requests used to simulate application usage |
| **Feeder** | A tool used to fill request parameters |
| **Recorder** | The tool used to record http requests or take a HAR-file and convert it to Gatling DSL |

# Demo

# Scenario to outcome

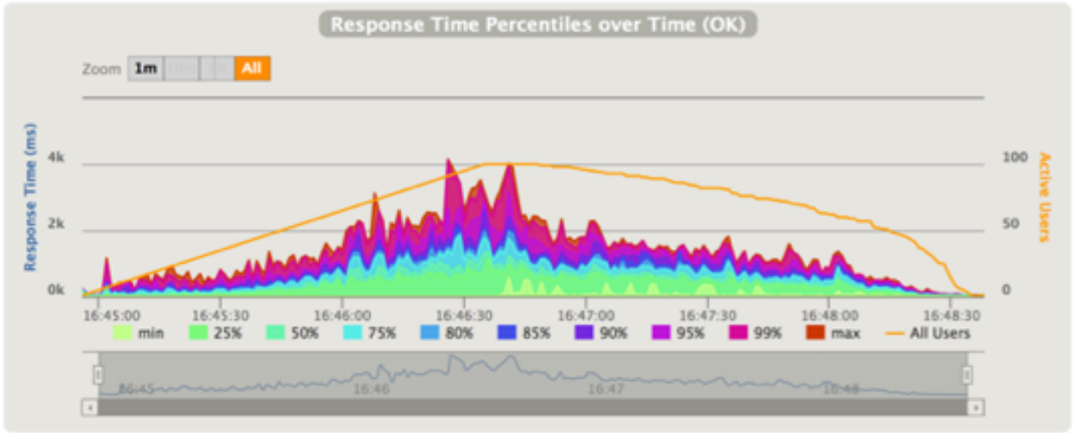


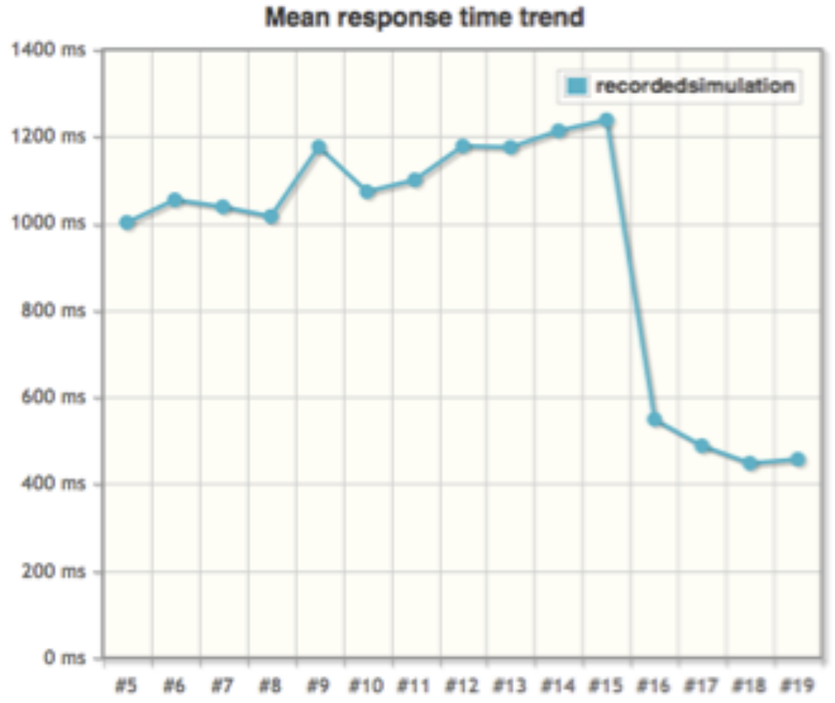
# Demo

# Reports

STATISTICS (Click here to show more) — Expand all groups | Collapse all groups

| Requests | Executions | | | | | Response Time (ms) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total | OK | KO | % KO | Req/s | Min | 50th pct | 75th pct | 95th pct | 99th pct | Max | Mean | Std Dev |
| Global Information | 25800 | 25787 | 13 | 0% | 115.878 | 1 | 455 | 738 | 1370 | 2081 | 30436 | 545 | 804 |
| GET / Redirect 2 | 100 | 100 | 0 | 0% | 0.449 | 47 | 56 | 68 | 95 | 177 | 240 | 64 | 25 |
| GET / Redirect 1 | 100 | 100 | 0 | 0% | 0.449 | 1 | 3 | 3 | 6 | 10 | 14 | 3 | 1 |
| GET / | 200 | 200 | 0 | 0% | 0.898 | 40 | 44 | 46 | 51 | 121 | 163 | 46 | 11 |
| GET /dtdi/select | 100 | 100 | 0 | 0% | 0.449 | 2 | 5 | 7 | 16 | 23 | 25 | 6 | 4 |
| GET /dtd...direct 1 | 100 | 100 | 0 | 0% | 0.449 | 10 | 29 | 40 | 54 | 66 | 82 | 31 | 13 |
| GET /dtd...direct 2 | 100 | 100 | 0 | 0% | 0.449 | 29 | 280 | 489 | 766 | 1017 | 1319 | 328 | 248 |
| GET /dtd....student | 100 | 100 | 0 | 0% | 0.449 | 2 | 4 | 6 | 10 | 21 | 26 | 5 | 3 |
| GET /dtd...direct 3 | 100 | 100 | 0 | 0% | 0.449 | 2 | 3 | 4 | 7 | 23 | 36 | 3 | 3 |
| GET /api/course | 200 | 200 | 0 | 0% | 0.898 | 11 | 268 | 600 | 1021 | 1321 | 1729 | 372 | 341 |
| POST /api/state | 3800 | 3800 | 0 | 0% | 17.067 | 10 | 392 | 609 | 1038 | 1515 | 2417 | 433 | 330 |



# Interpretations

**We can**

See how changes affect performance

Have feedback on performance in short amount of time

**We cannot**

See which load the application can endure in production

# Results so far

- Prevented multiple issues in production
- Helped testing performance fixes and database tuning
- Discovered configuration error in test infrastructure
- Helped track down and validate a fix for a memory leak
- Helped tweak database indexes

# Further development

- Automate (re-)recording process
- Re-use functional test scenarios as performance tests
- Eliminate custom code changes

# Summary

# Some take-aways

- Performance testing should be a first class citizen in your development cycle
- Frontend changes can impact backend performance
- Gatling is an awesome programmer friendly tool for load testing
- The approach we shared monitors performance trends - it does NOT determine the maximum load a production environment can endure

https://github.com/timve/continuous-performance-demo
https://github.com/bertjan/gatling-seed


Questions?

# Thanks for your time!

*Liked it? Tweet it!*