PERFORMANCE TESTING IN JAVA

Andres Almiray @aalmiray
Ixchel Ruiz @ixchelruiz

PERFORMANCE

speed
resiliency
quality
capacity
scalability
sustainability
load
robustness
elasticity
stability
consistency
QoS
repeatability
resiliency
recoverability
resource utilisation
responsiveness
reliability

A defined measure of performance in a system.

"Endurance of systems and processes"

An assessment of how well a delivered service conforms to the client's expectations.

"Long-lived and healthy systems"

PERFORMANCE TESTING IS IN GENERAL, A **TESTING PRACTICE** PERFORMED TO DETERMINE HOW A **SYSTEM PERFORMS** IN TERMS OF **RESPONSIVENESS** AND **STABILITY** UNDER A PARTICULAR **WORKLOAD**. IT CAN ALSO SERVE TO **INVESTIGATE**, **MEASURE**, **VALIDATE** OR VERIFY OTHER QUALITY ATTRIBUTES OF THE SYSTEM, SUCH AS **SCALABILITY**, **RELIABILITY** AND **RESOURCE USAGE**.

https://en.wikipedia.org/wiki/Software_performance_testing

PERFORMANCE TESTING IS THE PROCESS OF DETERMINING THE **SPEED** OR **EFFECTIVENESS** OF A COMPUTER, NETWORK, SOFTWARE PROGRAM OR DEVICE. THIS PROCESS CAN INVOLVE **QUANTITATIVE** TESTS DONE IN A LAB, SUCH AS MEASURING THE **RESPONSE TIME** OR THE NUMBER OF **MIPS** (MILLIONS OF INSTRUCTIONS PER SECOND) AT WHICH A SYSTEM FUNCTIONS. **QUALITATIVE ATTRIBUTES** SUCH AS **RELIABILITY**, **SCALABILITY** AND **INTEROPERABILITY** MAY ALSO BE EVALUATED. PERFORMANCE TESTING IS OFTEN DONE IN CONJUNCTION WITH **STRESS** TESTING.

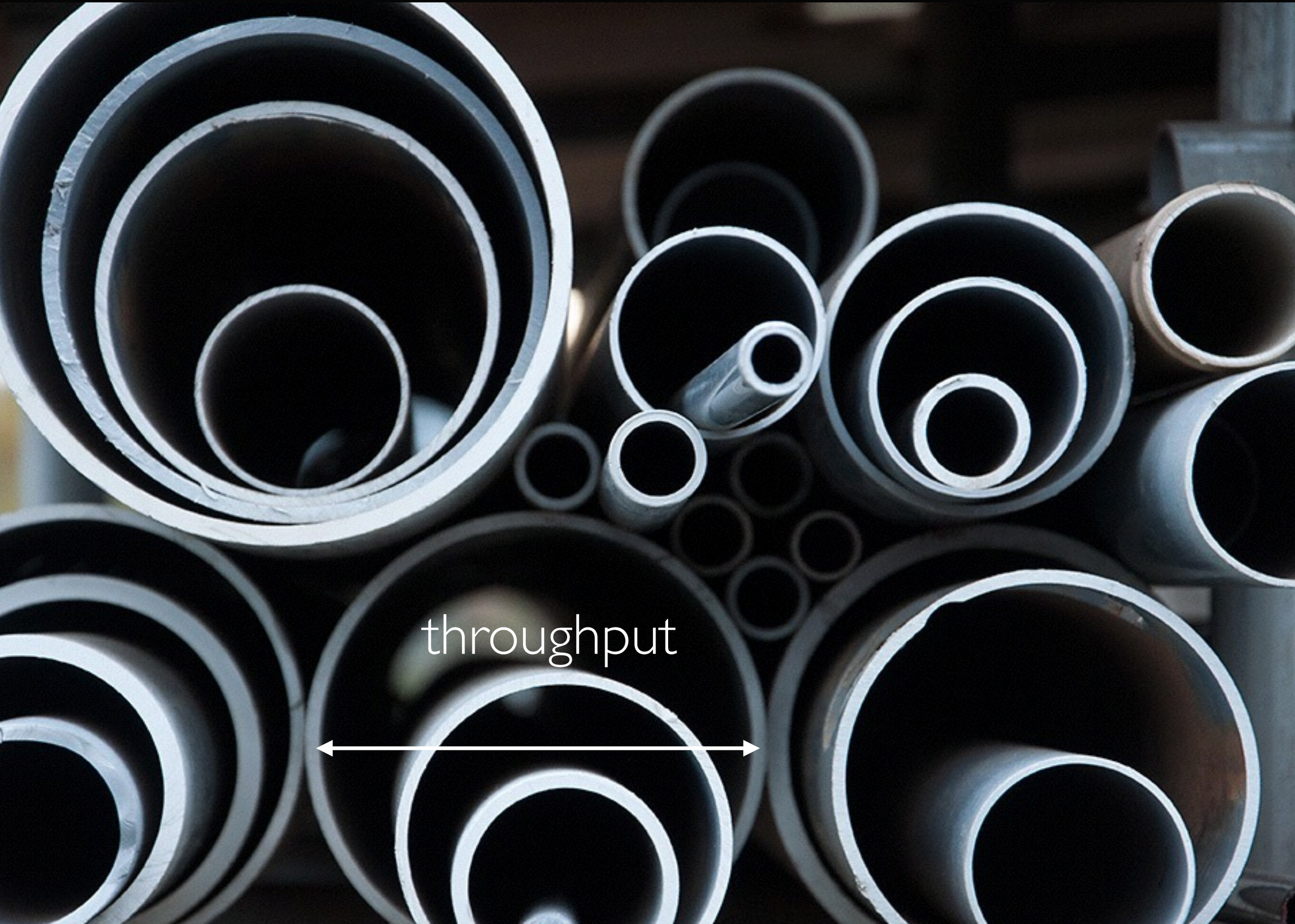
http://searchsoftwarequality.techtarget.com/definition/performance-testing

LOAD TESTING IS THE PROCESS OF PUTTING **DEMAND** ON A SOFTWARE SYSTEM OR COMPUTING DEVICE AND **MEASURING ITS RESPONSE**. LOAD TESTING IS PERFORMED TO DETERMINE A SYSTEM'S BEHAVIOR UNDER BOTH **NORMAL** AND **ANTICIPATED PEAK LOAD** CONDITIONS. IT HELPS TO IDENTIFY THE **MAXIMUM OPERATING CAPACITY** OF AN APPLICATION AS WELL AS ANY **BOTTLENECKS** AND DETERMINE WHICH ELEMENT IS CAUSING DEGRADATION.

https://en.wikipedia.org/wiki/Load_testing

THE EFFICIENCY WHICH SOMETHING REACTS OR FULFILLS ITS INTENDED PURPOSE.

CHARACTERIZED BY THE TIME AND RESOURCES NEEDED TO COMPLETE A UNIT OF WORK



https://www.flickr.com/photos/chimble/11771338823/in/photolist-iWcbVD-athKPm-GJSkr-9JF3PH-5f4sbF-7UpH9X-8JKoMf-mFYzke-RVyug1-a9Qfe9-QPSTtQ-e6bRp-59WzKR-9su9go-dhKdAr-7jyDw4-4KtZwe-4AcveW-gZ3akJ-dhKcN3-aW8vgn-8VXqj-kV27U3-8hTtG2-898mcx-hd2Cvi-ceY5Ko-jzDEQH-ifK92F-aqkH7-4BS4eZ-2Ca3qN-dQ95W4-f7Jiy7-pfAtR2-w9ZHBR-AogT-4GGNw-9i713i-7U4kBN-mBB5Vq-jLuzrQ-3W7fD-dvmNF-no8AEv-vV2L2-4mHG8A-bydWoY-7kewfY-7xek4J

https://www.flickr.com/photos/pigpogm/9169357228/in/photolist-eYgms1-J6pc4-7jYXdk-bAqYbT-6V2FNB-a8EGB2-47WPEX-4nHmpX-3nW8yu-bAr8VR-5SamAk-doondU-dAk1n8-NzqwG-4dx6An-RRRnge-nc4XL-54i56V-6g488W-hy9wsT-6K6rNp-5S6Niu-5diEuj-wS2CS-5bwaNq-9dc4PE-aLpot-cxexe-7skkJV-6Xx8K5-S2ZdCR-4LXETs-36WEBT-qWVKM-4jDwZm-7DzPNG-69Nkdy-2nN3z-qduAS-CzUn-5tG28U-7ZKRAd-bArof6-7XhcJ6-92HMwD-n7dc-4fgWJN-9u4uvu-bnwBNE-dFKrp

https://www.flickr.com/photos/waterarchives/5811736921/

Emergent behaviour

https://www.flickr.com/photos/98609459@N08/27830655732/in/photolist-JpiokW-mbisQn-cUFA4Y-beanUn-9LGjPf-8KQ2EF-e1EqCC-oyV1xg-oUxYBf-h72jZG-bEfZVW-aaez5V-psr4Mv-nrTvVc-HDw8AW-dpdB9r-9AX7GM-dCfzV6-6GhhNj-j3jioj-dMvYpA-iwaPsY-8rM9GN-pT3Qq6-ozhiGd-8sJcTY-oGzF3f-fmJyKu-dkv6F-bavkA2-bgtJe8-bcTZfD-RtX4hY-pJuw3T-dZ7MU4-gKAEag-dBfcfo-9jdBQ-6LMTBg-e6ThuD-bBD5k4-bDMUJ1-znkb1-9n6a7L-98RB1R-nNcReV-8YYpNC-ndjBN4-aatUAA-6nRDVD

"A collection of **strategies** applied to a given system to **verify** its **performance requirements**"

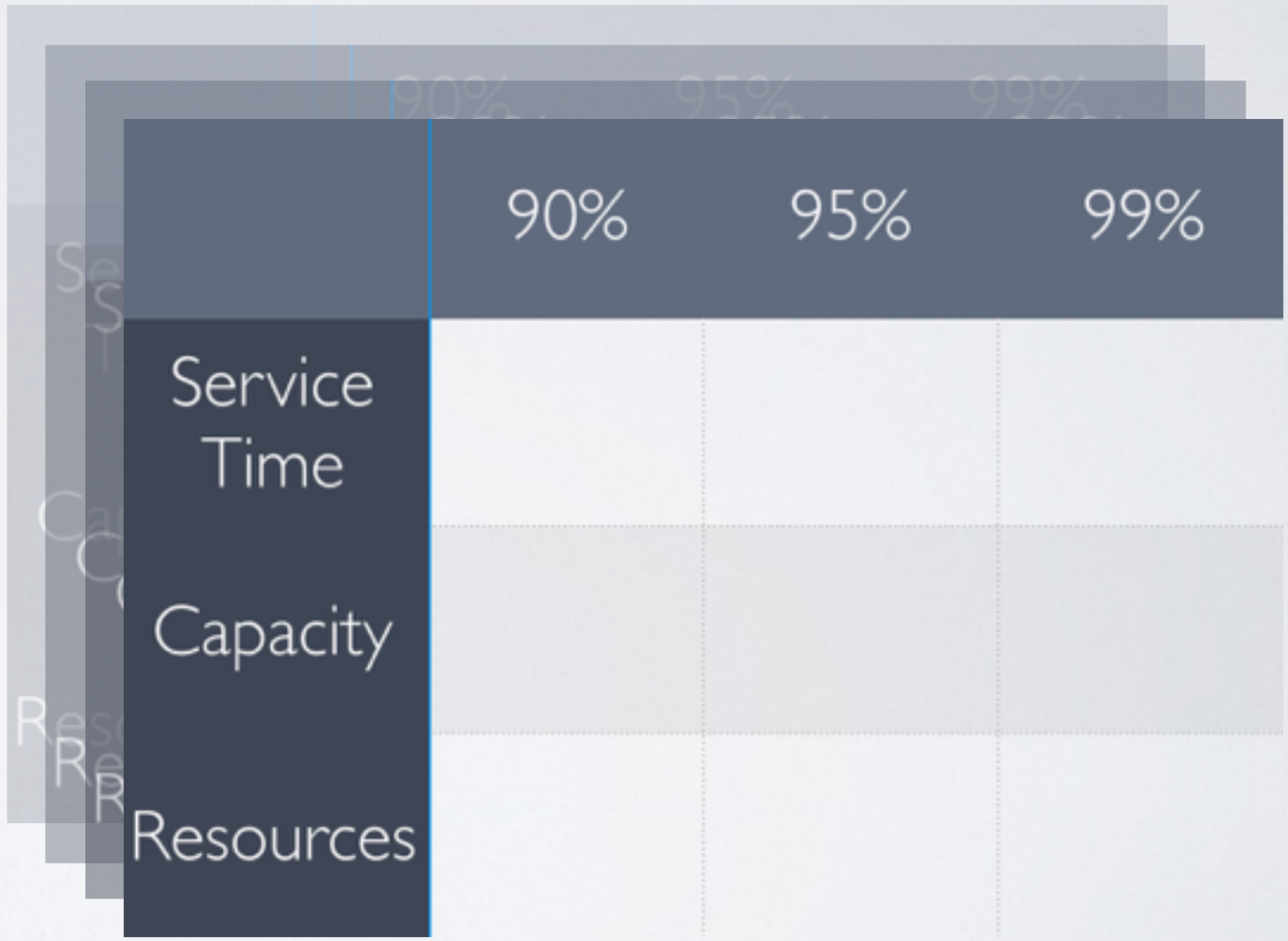
PERFORMANCE CONCERNS

- Quantitative

  Requirements

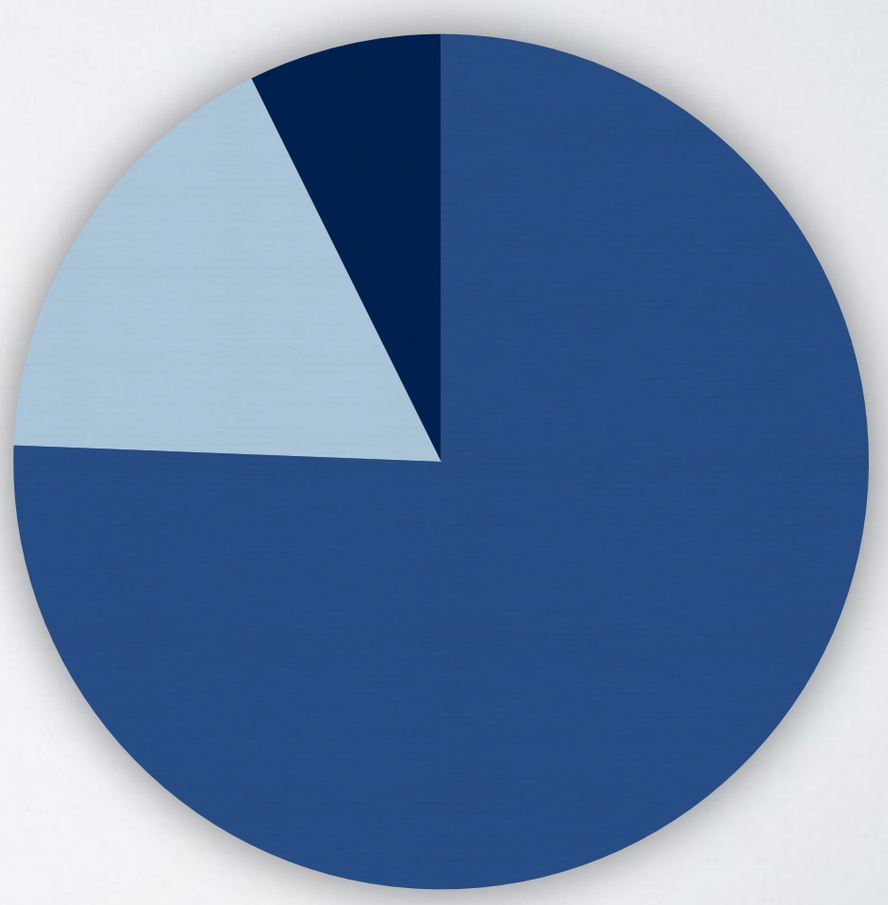
  - SLAs

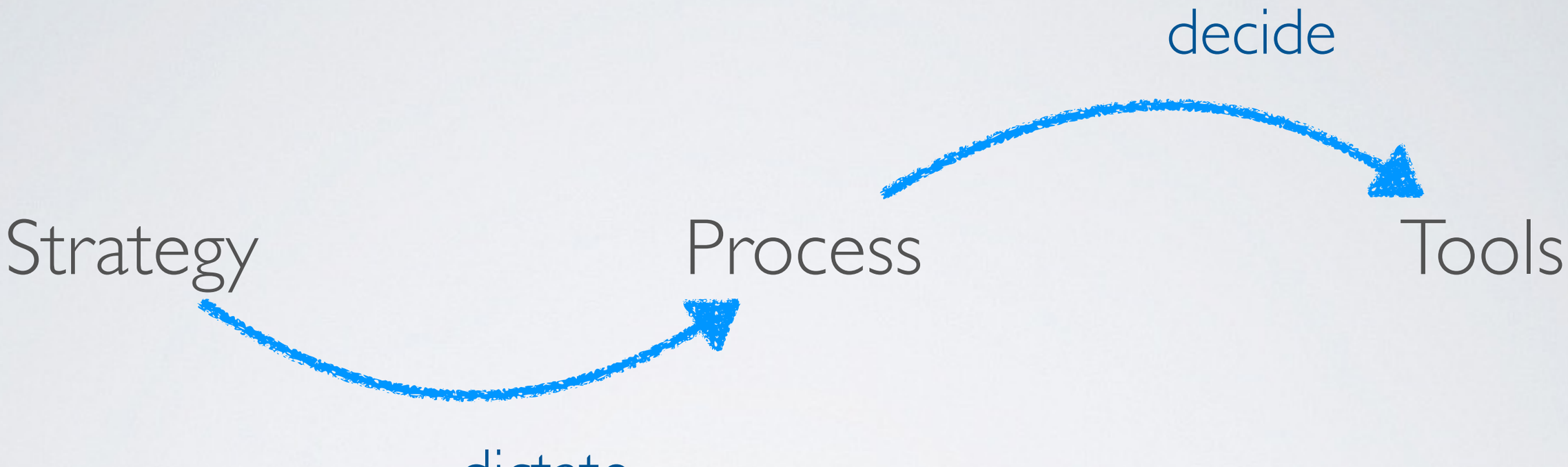
- Execution environment



BUDGET

- Computation / Execution
- Environmental Constraints
  - I/O Latency
  - CPU




decide
Strategy
Process
Tools
dictate

STRATEGY

Preemptive

- The right information in the right time
- Fail early and fail fast
- Reduce overtime
- Evaluate tradeoffs earlier based on measurements.

Corrective

- Performance as an afterthought
- Disruptive
- Usually more expensive
- Tuning

TESTING PROCESS

- Performance Objectives : Response time, throughput, resource utilization.
- Workload Goals
- Budgets : Constraints (maximum execution time and resource utilization levels)
- Execution and measurement tools.
- Generate different sets of inputs.
- Execute Tests & Store measurement data.
- Review and compare results.

Types

PERFORMANCE TESTING

- Bandwidth (Throughput)
- Response (Latency)
- Concurrency (Contention)

Types

PERFORMANCE TESTING

- Stress
- Endurance
- Load

WHEN?

ALWAYS (FREQUENCY)

PERFORMANCE CONCERNS

- What are the **relevant code paths** and how do they affect performance?
    - Where does the resource utilisation or **computation** affect performance?

JAVA8'S OPTIONAL VS IF(NULL)

https://struberg.wordpress.com/2017/01/28/optional-vs-if-null/

```
String var = val;
if (val == null) { var = compute(); }

String var = Optional.ofNullable(val)
                     .orElse(this::compute)
```

JMH

JMH is a Java harness for building, running, and analysing **nano/micro/milli/macro** benchmarks written in Java and other languages targeting the JVM.

http://openjdk.java.net/projects/code-tools/jmh/

https://github.com/melix/jmh-gradle-plugin

http://www.rationaljava.com/2015/02/jmh-how-to-setup-and-run-jmh-benchmark.html

```
plugins {
    id 'me.champeau.gradle.jmh' version '0.3.1'
    id 'io.morethan.jmhreport' version '0.1.0'
}

repositories {
    jcenter()
}

jmhReport {
    jmhResultPath = project.file('build/reports/jmh/results.json')
    jmhReportOutput = project.file('build/reports/jmh')
}
tasks.jmh.finalizedBy tasks.jmhReport

jmh {
    jmhVersion = '1.17.5'
    iterations = 10
    fork = 2
    warmupIterations = 5
    resultFormat = 'json'
}
```

JAVA8'S OPTIONAL VS IF(NULL)

Several iterations with JMH, reducing the test to its very core

- Blog suggested integers. Must consider autoboxing cost.
- Switched to Strings. Consider null & non-null values.
- 3 ways to query for an Optional's value.

Tweak JMH settings, such as fork, warmup iterations, etc.

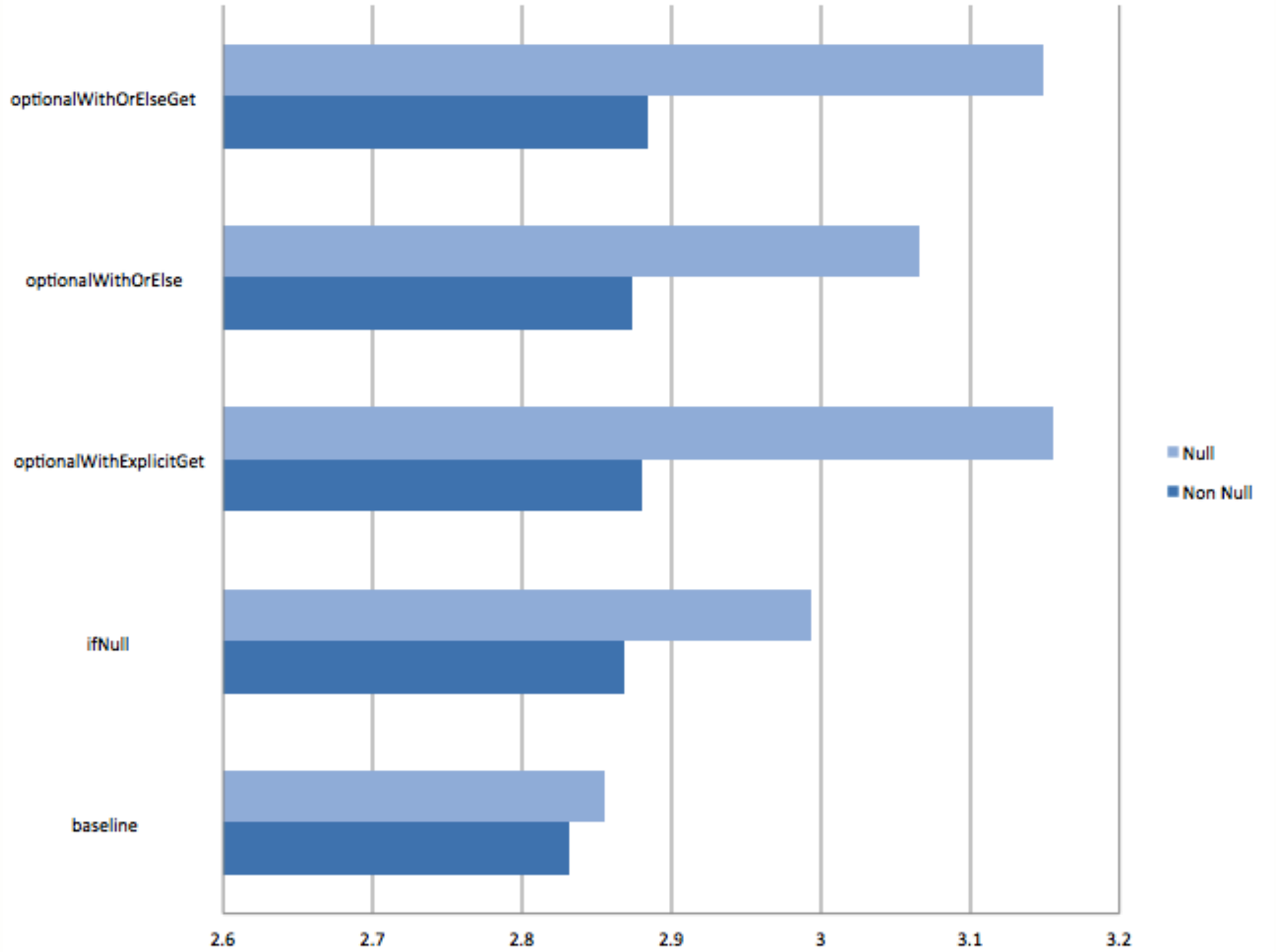
```java
@State(Scope.Benchmark)
@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.NANOSECONDS)
public class OptionalVsIfNull_Parameterized {
    @Param({"false", "true"})
    private boolean isNull;

    private String value;
    private String answerWhenNull = "NULL";

    @Setup(Level.Trial)
    public void setUp() { value = isNull ? null : "something"; }

    @Benchmark
    public String baseline() { return value; }

    @Benchmark
    public String ifNull() {
        String val = value;
        if (val != null) {
            return val;
        }
        return answerWhenNull;
    }

    @Benchmark
    public String optionalWithExplicitGet() {
        Optional<String> optional = Optional.ofNullable(value);
        if (optional.isPresent()) {
            return optional.get();
        }
        return answerWhenNull;
    }

    @Benchmark
    public String optionalWithOrElse() { return Optional.ofNullable(value).orElse(answerWhenNull); }

    @Benchmark
    public String optionalWithOrElseGet() { return Optional.ofNullable(value).orElseGet(() -> answerWhenNull); }
}
```

JAVA8'S OPTIONAL VS IF(NULL)

Benchmark	(isNull)	Mode	Cnt	Score	Error	Units
OptionalVsIfNull_AlwaysNonNull.baseline	N/A	avgt	20	**2.929**	± 0.054	ns/op
OptionalVsIfNull_AlwaysNonNull.ifNull	N/A	avgt	20	**2.838**	± 0.028	ns/op
OptionalVsIfNull_AlwaysNonNull.optionalWithExplicitGet	N/A	avgt	20	**2.831**	± 0.026	ns/op
OptionalVsIfNull_AlwaysNonNull.optionalWithOrElse	N/A	avgt	20	**2.863**	± 0.068	ns/op
OptionalVsIfNull_AlwaysNonNull.optionalWithOrElseGet	N/A	avgt	20	**2.850**	± 0.042	ns/op
OptionalVsIfNull_AlwaysNull.baseline	N/A	avgt	20	**2.785**	± 0.049	ns/op
OptionalVsIfNull_AlwaysNull.ifNull	N/A	avgt	20	**2.768**	± 0.058	ns/op
OptionalVsIfNull_AlwaysNull.optionalWithExplicitGet	N/A	avgt	20	**2.911**	± 0.035	ns/op
OptionalVsIfNull_AlwaysNull.optionalWithOrElse	N/A	avgt	20	**2.832**	± 0.027	ns/op
OptionalVsIfNull_AlwaysNull.optionalWithOrElseGet	N/A	avgt	20	**2.900**	± 0.066	ns/op

JAVA8'S OPTIONAL VS IF(NULL)

Benchmark	(isNull)	Mode	Cnt	Score	Error	Units
OptionalVsIfNull_Parameterized.baseline	false	avgt	20	**2.832**	± 0.024	ns/op
OptionalVsIfNull_Parameterized.baseline	true	avgt	20	**2.855**	± 0.032	ns/op
OptionalVsIfNull_Parameterized.ifNull	false	avgt	20	**2.868**	± 0.035	ns/op
OptionalVsIfNull_Parameterized.ifNull	true	avgt	20	**2.994**	± 0.036	ns/op
OptionalVsIfNull_Parameterized.optionalWithExplicitGet	false	avgt	20	**2.881**	± 0.057	ns/op
OptionalVsIfNull_Parameterized.optionalWithExplicitGet	true	avgt	20	**3.156**	± 0.036	ns/op
OptionalVsIfNull_Parameterized.optionalWithOrElse	false	avgt	20	**2.874**	± 0.031	ns/op
OptionalVsIfNull_Parameterized.optionalWithOrElse	true	avgt	20	**3.066**	± 0.052	ns/op
OptionalVsIfNull_Parameterized.optionalWithOrElseGet	false	avgt	20	**2.884**	± 0.027	ns/op
OptionalVsIfNull_Parameterized.optionalWithOrElseGet	true	avgt	20	**3.149**	± 0.051	ns/op


optionalWithOrElseGet
optionalWithOrElse
optionalWithExplicitGet
ifNull
baseline
Null
Non Null
2.6
2.7
2.8
2.9
3
3.1
3.2

MORE INFO ON JMH

Using Java Microbenchmark Harness (JMH) in a real world project

http://2015.jokerconf.com/talks/vyazelenko/

@DVyazelenko

PRAGMATIC PERFORMANCE

Pragmatic: dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations.

QCon San Paulo 2015 Keynote by Gil Tene

http://qconsp.com/sp2015/system/files/keynotes-slides/Qcon_Sao_Paulo_Keynote_March2015.pdf

PARTING THOUGHTS

- Beware of preconceptions and confirmation bias.
- Be mindful of the shape of the input data.
- Always mind the system's environment.
- Approach results skeptically.
- Iterate, iterate, iterate.

INPUT DATA

For any given scenario, is the system able to handle data

- sorted in natural order
- sorted in reverse order
- random order

"If we have data, let's look at data. If all we have are opinions, let's go with mine."

— Jim Barksdale

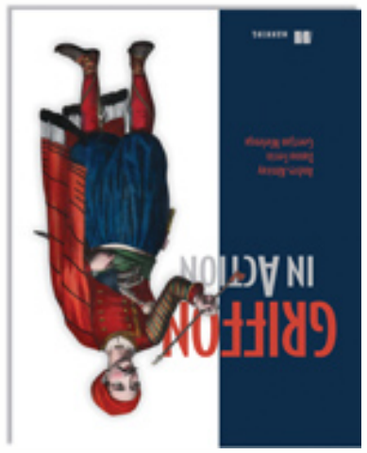
“Measure, don’t guess!”®

–Kirk Pepperdine

AXIOMS

- It all depends
- Failure it's guaranteed, we're only humans!

Fail **gracefully** & plan for how to **recover**.



canoo

THANK YOU!

Andres Almiray
Ixchel Ruiz

@aalmiray
@ixchelruiz