University of Oviedo

School of Computer Science

EN
English

Universidad de Oviedo

School of
Computer
Science

Quality attributes

SOFTWARE
ARCHITECTURE

Course 2022/23

Jose E. Labra Gayo

Quality attributes

Also known as:

Architecture characteristics

Non-functional requirements

Types of requirements

Requirements can be categorized as:

Functional requirements: state what the system must do, how it must behave or react to run-time stimuli.

Quality attribute requirements. Annotate (qualify) functional requirements

Examples: Availability, modifiability, usability, security,...

Also called: non-functional requirements

Constraints. A design decision that has already been made
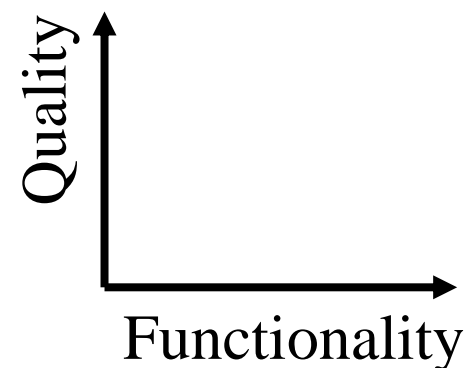
Functional requirements

Functionality = ability of the system to do the work for which it was intended

Functionality has a strange relationship to architecture:

It does not determine architecture;

If functionality were the only requirement, the system could exist as a single monolithic module with no internal structure at all

Functionality and quality attributes are orthogonal

Quality

Functionality

Quality attributes (QA)

Quality attributes annotate (qualify) the functionality

If a functional requirement is "when the user presses the green buttom an options dialog appears"

- A performance QA could describe how quickly

- An availability QA could describe this option could fail, or how often it will be repaired

- A usability QA could describe how easy is to learn this function

Quality attributes influence the architecture

What is Quality?

Degree to which a system satisfies the stated and implied needs of its stakeholders, providing value

- Degree (not Boolean)
- Quality = Fitness for purpose (*stakeholders* needs)
- Conform to requirements (stated and implied)
- Providing value

Several definitions of quality

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

Quality attributes and trade-offs

QAs are all good

...but value depends on the project & stakeholder

"Best quality"...for what?, for whom?

QAs are not independent

Some qualities can conflict

What matters most?

Example: A very secure system can be less usable

There is always a price

What is your budget?

☛ There is no single *perfect* system or architecture!

Specifying quality attributes

2 considerations:

QAs by themselves are not enough

They are non-operational

Example: It is meaningless to say that a system shall be modifiable or maintainable

The vocabulary describing QAs varies widely

It is necessary to describe each attribute separately

QA scenarios: A common form to specify QA requirements
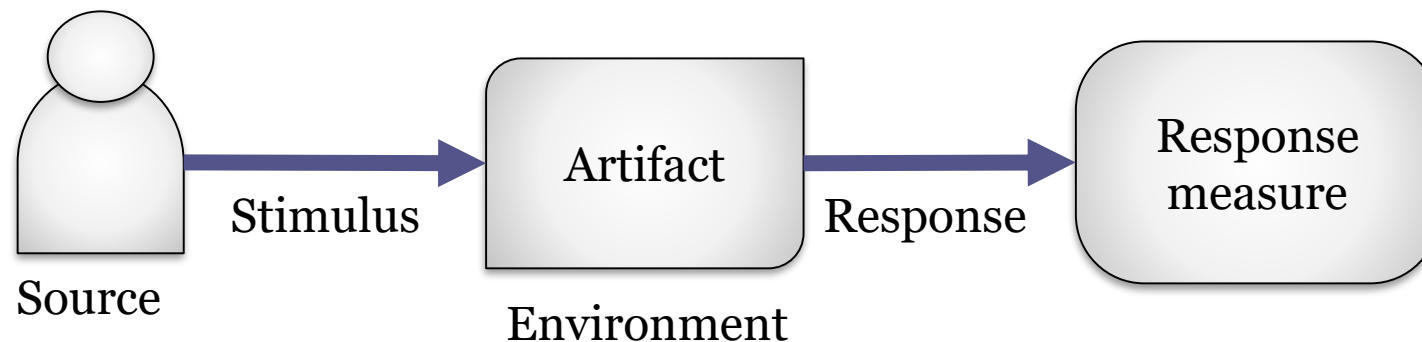
Quality attribute scenario

Describe a stimulus of the system and a measurable response to the stimulus

- Stimulus = event triggered by a person or system
- The stimulus generates a response

Must be testable

- Response must be externally visible and measurable

Components of QA scenario

Source: Person or system that initiates the stimulus

Stimulus: Events that requires the system to respond

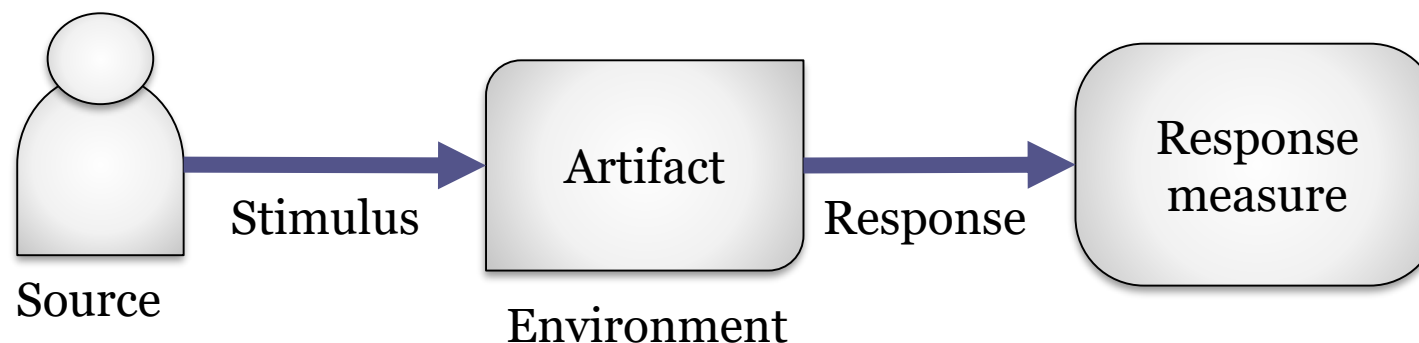**Artifact**: Part of the system or the whole system

Response: Externally visible action

Response measure: Success criteria for the scenario

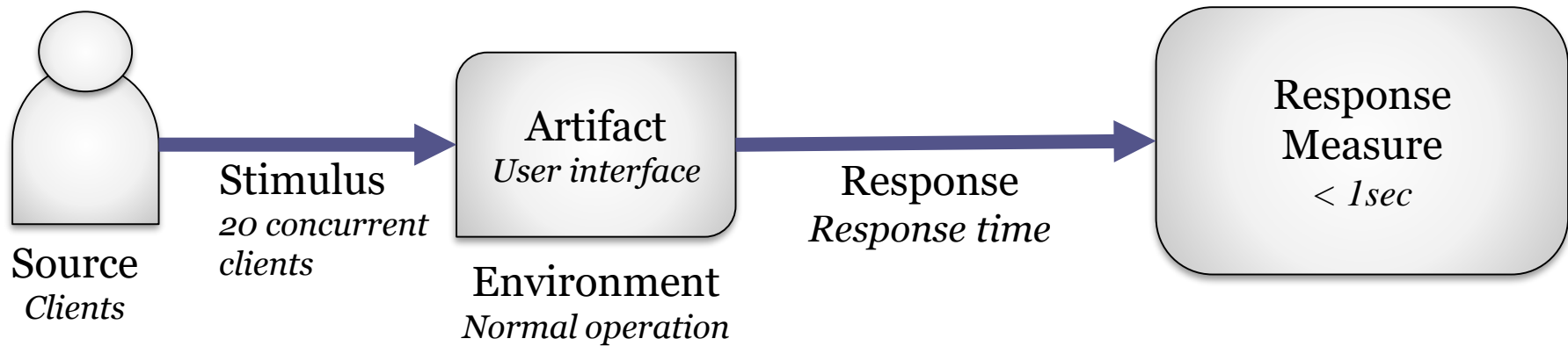   Should be specific and measurable

Environment: Operational circumstances

   Should be always defined (even if it is "normal")

QA scenario, example 1
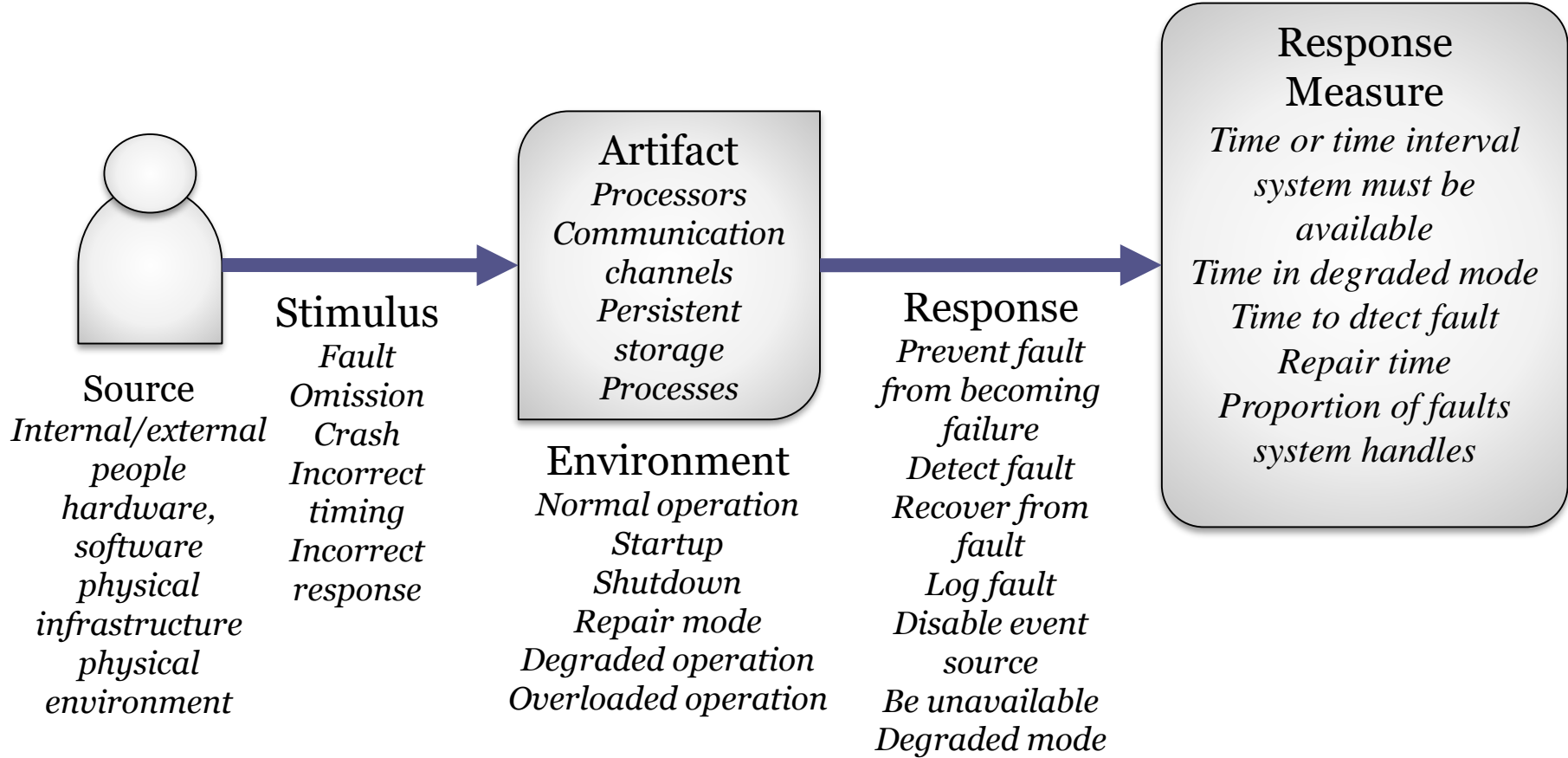
Performance: *If there are 20 concurrent clients, the response time should be less than 1 sec. under normal operation*

Source of stimulus	Stimulus	Artifact	Environment	Response	Response measure
Clients	20 concurrent clients	User interface	Normal operation	Response time	<1sec
. . .	. . .	. . .	. . .	. . .	. . .

QA Scenario structure for availability

Source
Internal/external people hardware, software physical infrastructure physical environment

Stimulus
Fault Omission Crash Incorrect timing Incorrect response

Artifact
Processors Communication channels Persistent storage Processes

Environment
Normal operation Startup Shutdown Repair mode Degraded operation Overloaded operation

Response
Prevent fault from becoming failure Detect fault Recover from fault Log fault Disable event source Be unavailable Degraded mode

Response Measure
Time or time interval system must be available Time in degraded mode Time to dtect fault Repair time Proportion of faults system handles

Types of QA scenarios

Usage scenarios

The system is used (any use case or system function is executed)

Describe how the system behaves in these cases

Runtime, performance, memory consumption, throughput,...

Change (or modification) scenarios

Any component within the system, its environment or its operational infrastructure changes

Failure scenarios:

Some part of the system, infrastructure or neighbours fail

Prioritizing QA scenarios

Scenarios should be prioritized

2 values (Low/Medium/High)

How important it is for success (ranked by customer)

How difficult it is to achieve (ranked by architect)

Ref	Quality Attribute	Scenario	Priority
1	Availability	When the database does not respond, the system should log the fault and respond with stale data during 3 seconds	High, High
2	Availability	A user searches for elements of type X and receives a list of Xs 99% of the time on average over the course of the year	High, Medium
3	Scalability	New servers can be added during the planned maintainance window (less than 7 hours)	Low, Low
4	Performance	A user sees search results within 5 seconds when the system is at an average load of 2 searches per second	High, High
5	Reliability	Updates to external elements of type X should be reflected on the application within 24 hours of the change	Low, Medium
	. . .	. . .	. . .

Identifying quality attributes

Finding QAs

Most of the time, QAs are not explicit

They're only verbally said alongside func. requirements

Usually implicit or said without much thought

Software architect must do educated guesses

Quality Attribute Workshops

Meetings where stakeholders specify QAs

Formal checklists

ISO25010

Wikipedia: https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Typical quality attributes

Availability

Modifiability

Performance

Security

Testability

Maintainability

Usability

Scalability

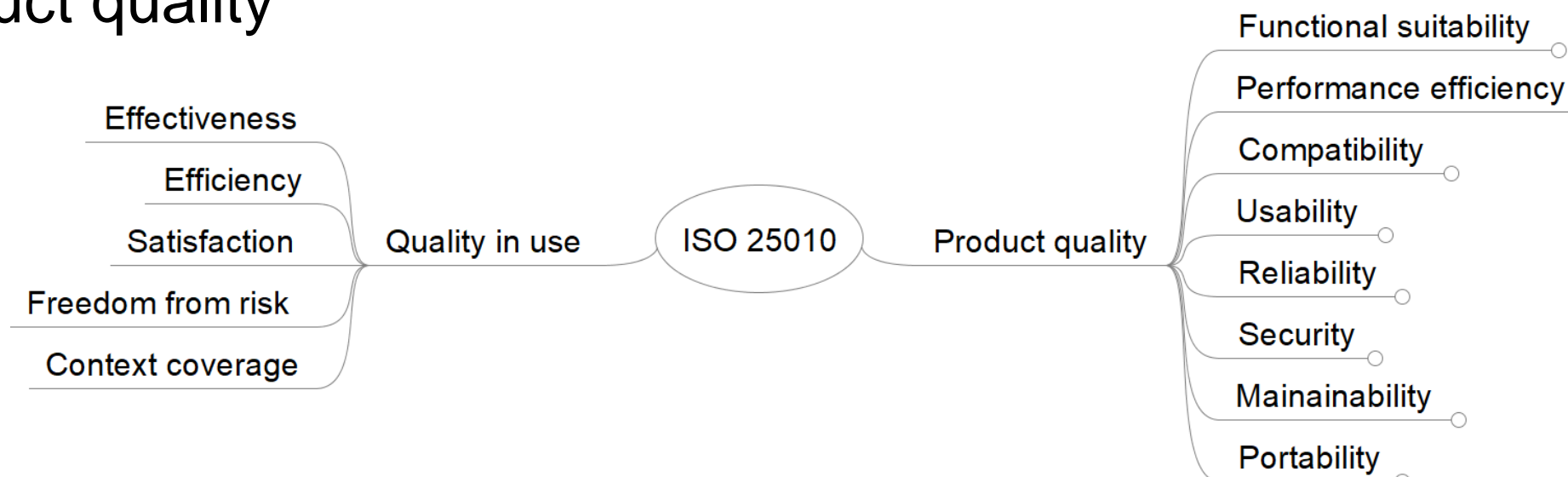Interoperability

Portability

Changeability

. . .

University of Oviedo

School of Computer Science

ISO-25010 Software Quality Model

Systems and software Quality Requirements and
  Evaluation (SQuaRE)

2 parts:
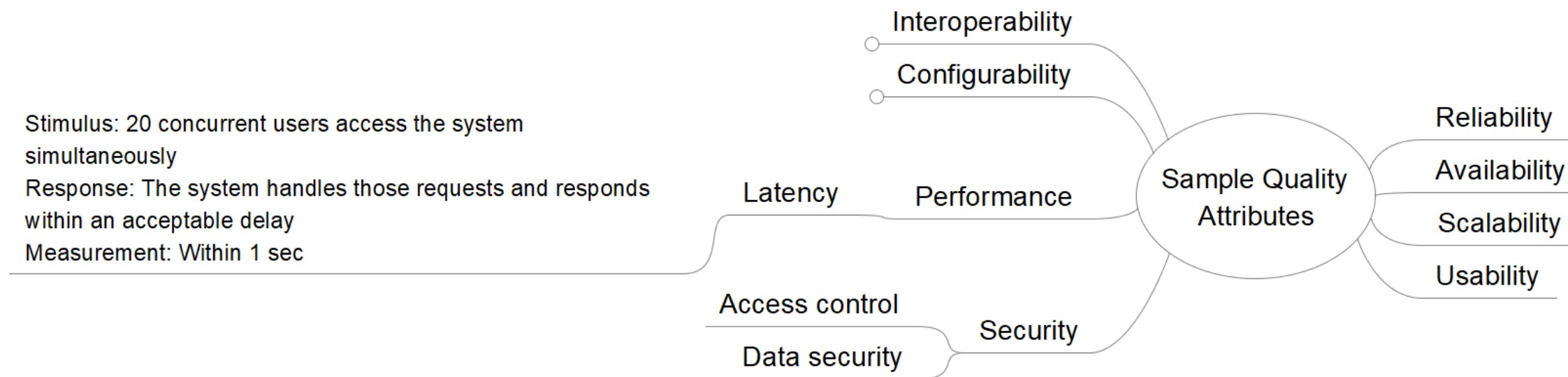
- Quality in-use

- Product quality

University of Oviedo

School of Computer Science

Quality Attribute tree

Mindmap representations can be useful to visualize QA scenarios

Measuring quality attributes

QA scenarios require response measures

Common problems:

Many QA have vague meanings

Example: deployability, scalability

Wildly varying definitions

No universal definitions

Too composite

Usually, a QA is composed of several features

Software metrics

Goal: Objective measures which can be automatically computed

Numerous metrics for different aspects

More information:

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

Operational measures

Lots of metrics

Absolute values

Examples: Number of users, number of errors

Statistical values and models

Example

$$availability = \frac{MTBF}{MTBF + MTTR}$$ where

$MTBF = mean\ time\ between\ failures\ (uptime)$

$MTTR = mean\ time\ to\ recover\ (downtime)$

Availability	Downtime/90days	Downtime/year
99,0%	21hours, 36 minutes	3 days, 15.6 hours
99,99%	12 minutes, 58secs	52min, 34 secs
99,9999%	8 secs	32 secs

Structural measures

Measure the structure of the modules

Examples:

Cyclomatic complexity (CC) from McCabe, 1976

Goal: Objective measure of complexity of code

It is obtained from the program control flow graph

$CC = E - N + 2$ where $\begin{matrix} E = number\ of\ edges \\ N = number\ of\ nodes \end{matrix}$

Example:

```java
public static void main(String[] args) {
    int max = 10 ;
    int next = 0 ;
    while (next < max) {
      next++;
      if (next % 2 == 0)
        System.out.println("Even " + next);
      else
        System.out.println("Odd " + next);
    }
  }
```

$$CC = 10 - 9 + 2 = 3$$

```
int max = 10 ;
int next = 0 ;
```

```
while (next < max) {
```

```
next++;
```

```
if (next % 2 == 0)
```

```
println("Odd "+next);        println("Even "+next);
```
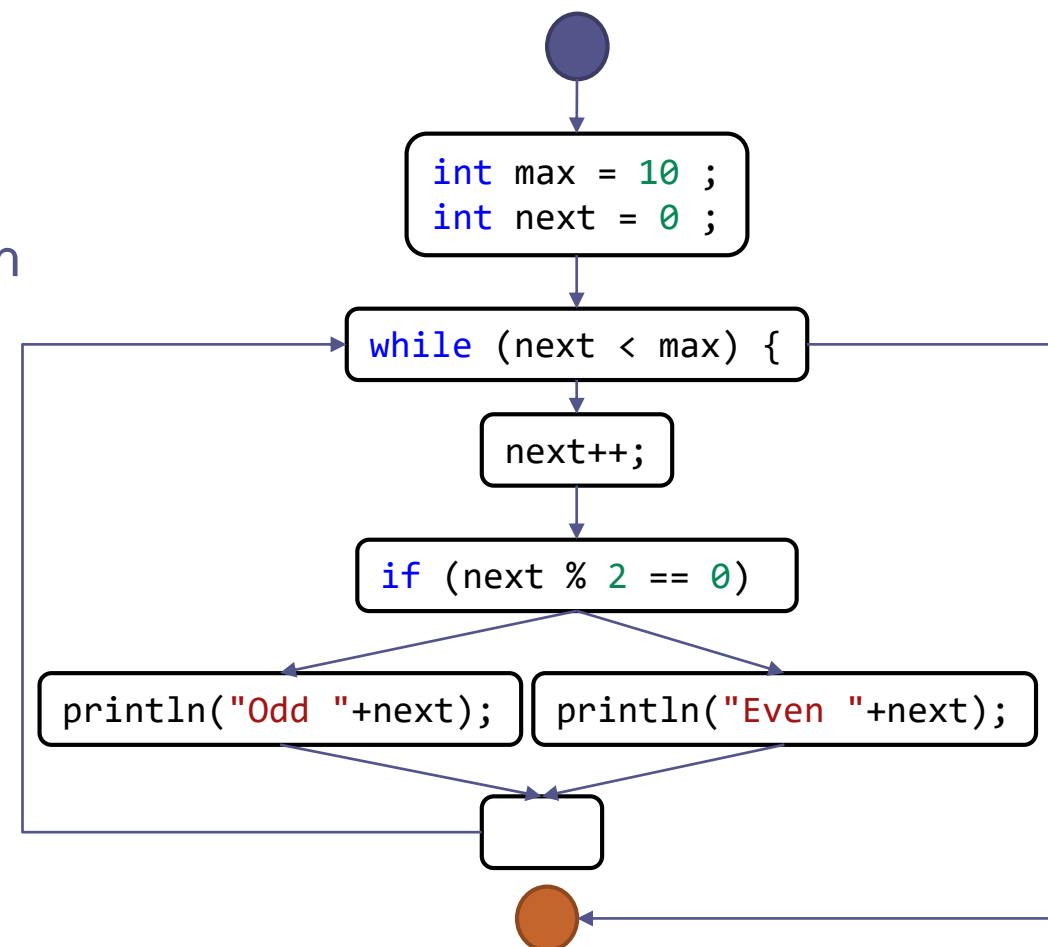
Process measures

Measure process aspects like agility, testability, maintainability, etc.

Examples:

Testability: Code-coverage during testing

Percentage of lines of code that have been run during the test phase

Deployability:

% of successful to failed deploys

How long deployment takes

Issues/bugs in deployment

...

4 key metrics or DORA metrics

Origin: State of DevOps report by DORA (DevOps Research and Assessment team)
Measure High delivery performance

1. Lead Time to change

   Time from code committed to code successfully running in production

2. Deployment frequency

   How often does an organization deploy code to production

3. Mean time to restore (MTTR)

   how long does it take to restore service after an incident

4. Change failure rate

   Percentage of changes to production that result in degraded service

https://www.devops-research.com/quickcheck.html

Governing quality attributes

Evolutionary architectures

Fitness functions: mechanism that provides objective integrity assessment of some combination of quality attributes

Make QA measures and evolve the architecture

1. Geospiza magnirostris.
3. Geospiza parvula.

2. Geospiza fortis.
4. Certhidea olivasca.