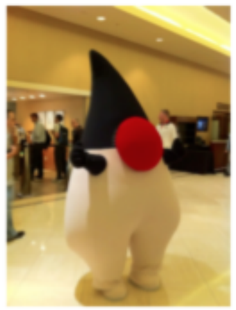
BDX I/O
Bordeaux Developer eXperience

Boite à outils - Living Documentation

Me

@binout https://github.com/binout



Il y un an …

Living Documentation : vous allez aimer la documentation !

Cyril Martraire, BDX/IO 2015

Et aussi le livre

970 READERS | 362 PAGES | 98,489 WORDS

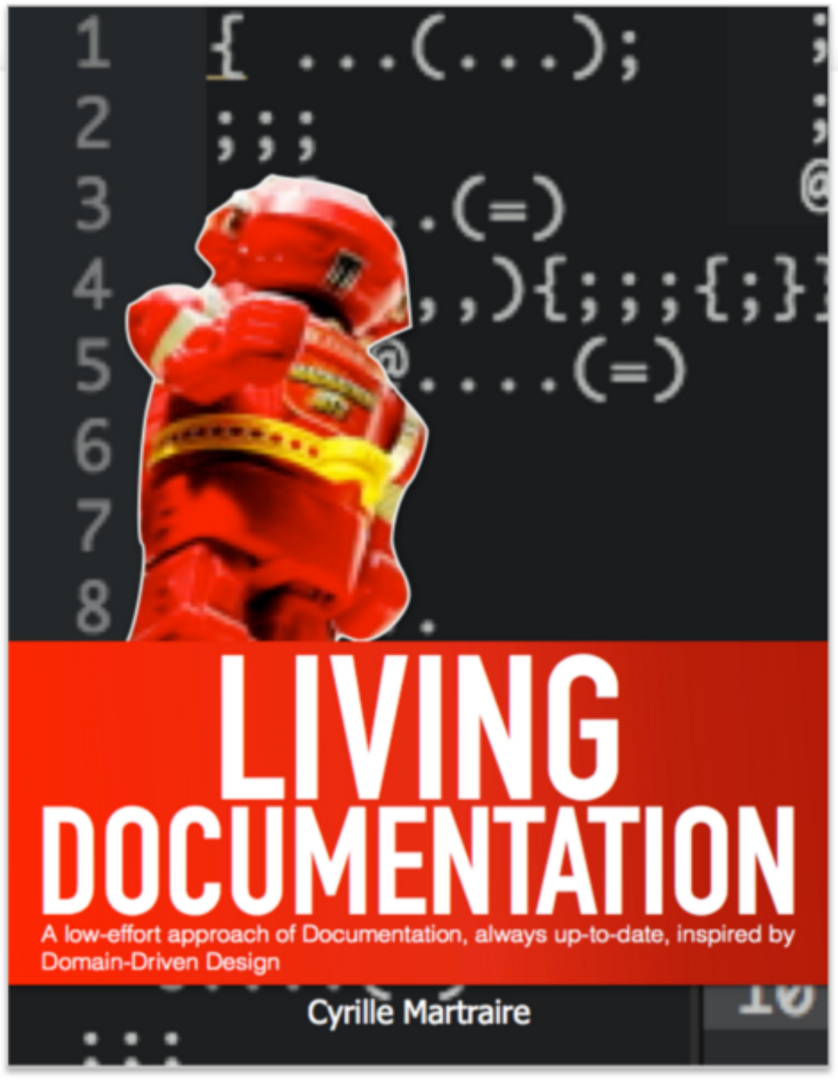
Living Documentation by design, with Domain-Driven Design

Accelerate Delivery Through Continuous Investment on Knowledge

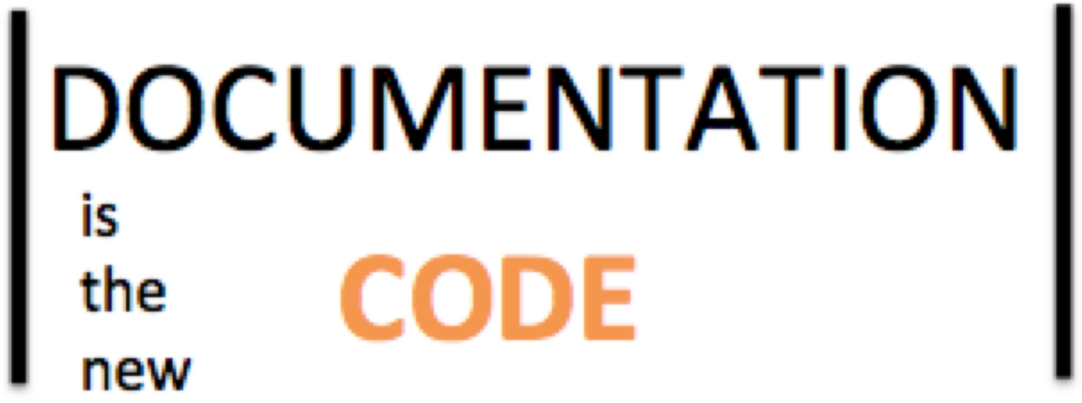
Cyrille Martraire

You don't necessarily have to chose between Working Software and Extensive Documentation! Discover how a Living Documentation can help you in all aspects of your projects, from the business goals to the business domain knowledge, architecture and design, processes and deployment, even if you hate writing documentation.

FREE SAMPLE



Documentation 2.0



- La ***documentation*** est avec le ***code***
- La ***documentation*** se build avec le ***code***
- La ***documentation*** se teste comme le ***code***

Mais…

- La documentation doit rester **vivante** !
- La **maintenance** reste l'enjeu majeur.

Mais…

- La documentation doit rester **vivante** !
- La **maintenance** reste l'enjeu majeur.

> *Le code doit être la source de la documentation !*

Ma boîte à outils

Documentation Technique

BDX I/O
Bordeaux Developer eXperience

README

- description d'un projet, du contexte
- outils de build

TUTORIAL

- getting started
- examples de codes

API REST

- documentation des ressources

Asciidoc et Asciidoctor !

Asciidoc
langage de ***balisage*** (aka Markdown mais en mieux)

Asciidoctor
toolchain pour convertir l'asciidoc en html, pdf, epub, …

Pourquoi ?

1. on se concentre plus sur le fond que sur la forme
2. c'est du texte, donc un éditeur classique suffit
3. on peut gérer l'historique avec un SCM
4. on a un peu l'impression de coder 👍

Macro `include` pour du code

```
= Tutorial

== Code examples

[source,java]
---
include:{basedir}/src/main/java/io/github/binout/Geek.java[]
---
```

> *La documentation présente le code à jour et sans erreur !*

Macro `include` pour un morceau code

```
= Tutorial

== Code examples

[source,java]
---
include:{basedir}/src/main/java/io/github/binout/Geek.java[tags=hype]
---
```

```java
public class Geek  {

    String getName();

    //tag::hype[]
    Hype computeHype(int age, String language);
    //end::hype[]
}
```

Macro `include` pour un morceau code

```
= Tutorial

== Code examples

[source,java]
---
include:{basedir}/src/main/java/io/github/binout/Geek.java[tags=hype]
---
```

Tutorial

Code examples

```
Hype computeHype(int age, String language);
```

Macro `include` pour un fichier de test

```
== Body

[source,asciidoc]
---
include:{basedir}/src/test/resources/io/github/binout/geek.json[]
---
```

geek.json

```
{
  "name" : "binout",
  "language" : "java",
  "hype" : 100
}
```

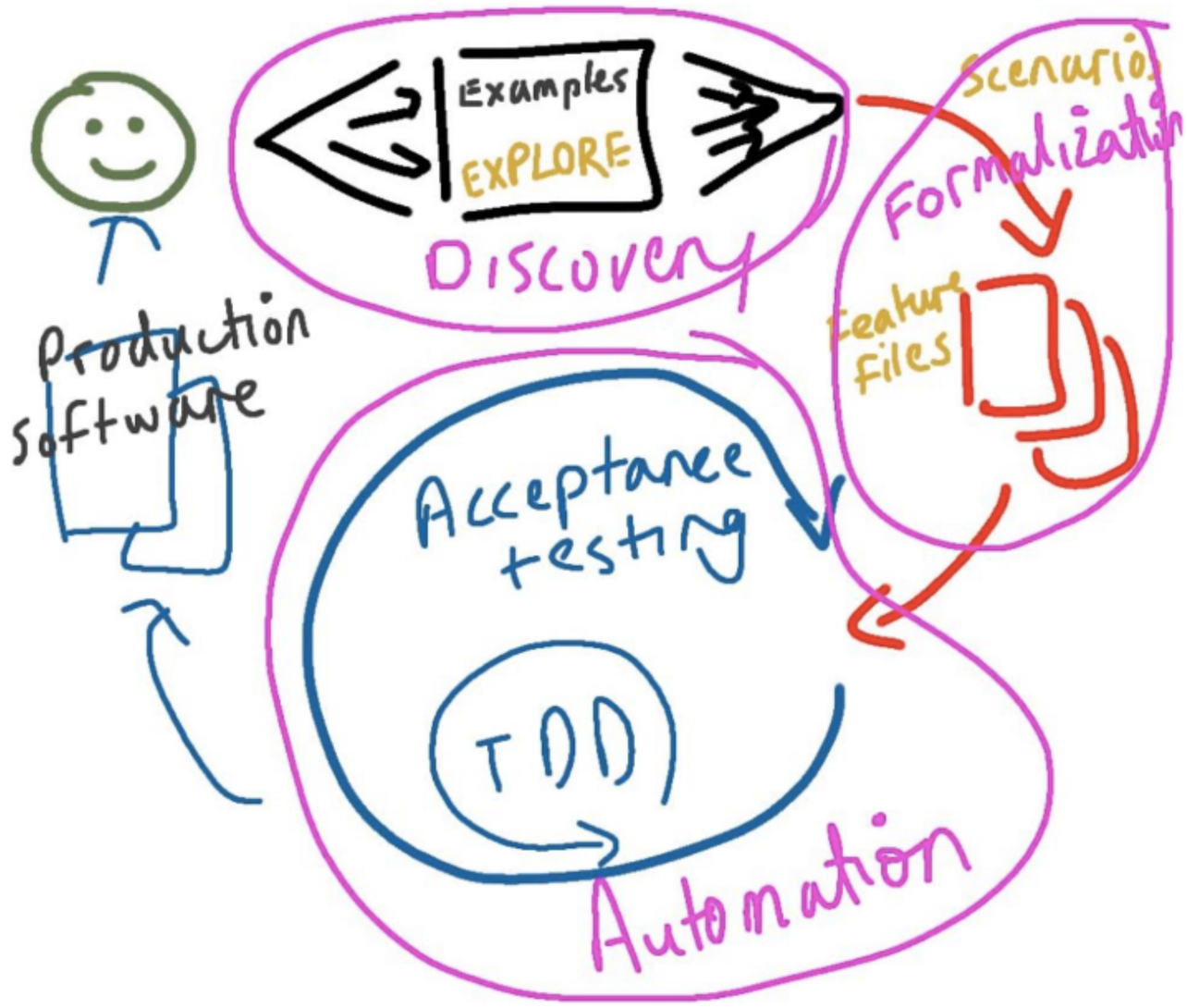
Documentation Métier

> *BDD (Behavior Driven Development)*

- Utilisation du langage naturel et du domaine métier pour décrire les spécifications
- Les spécifications permettent de créer des tests automatisés

Cucumber

https://cucumber.io/



cucumber-java en 4 temps

1. Ajouter les dépendances cucumber :
    - `info.cukes:cucumber-java`
    - `info.cukes:cucumber-junit`
2. Ecrire un fichier `.feature` en `Gherkin`
3. Coder une classe Java faisant la ***glue*** entre les ***features*** et les assertions de test
4. Coder une classe Java pour ajouter le runner Cucumber Junit

Fichier feature

Syntaxe gherkin

```
Feature: Features of dropbox command line

  Scenario: Command whoami
    Given a dropbox api key
    When i type "whoami"
    Then it should return my name

  Scenario: Command ls
    Given a dropbox api key
    When i type "ls"
    Then it should return a list of path
```

Fichier Steps


```
@Given("^a dropbox api key$")
    public void a_dropbox_api_key() throws Throwable {
        assertThat(Dropbox.apiKey()).isNotNull();
    }

    @When("i type \"([^\"]*)\"")
    public void i_type_a_command(String command) throws Throwable {
        this.result = executeCommand(command);
    }

    @Then("^it should return my name$")
    public void it_should_return_my_name() throws Throwable {
        assertThat(result).contains("Benoît Prioux");
    }

    @Then("^it should return a list of path$")
    public void it_should_return_a_list_of_path() throws Throwable {
        Arrays.stream(result.split(System.lineSeparator())).forEach(p ->
assertThat(p).startsWith("/"));
    }
}
```

Fichier Runner

```
package io.github.binout.dropbox.bdd;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(strict=true, plugin = {"json:target/cucumber.json"} )
public class DropboxCliTest {
}
```

Cukedoctor - Living Documentation

http://rmpestano.github.io/cukedoctor/

Features of dropbox command line

> *In order to manage my Dropbox account,*
> *As a Dropbox user*
> *I want to use a CLI.*

You need to retrieve a API key for Dropbox API v2

Scenario: Command whoami

Given

a dropbox api key (099ms)

When

i type "whoami" (02s 837ms)

Then

it should return my name (000ms)

Autre exemple de doc métier

Glossaire

- vocabulaire métier de l'application
- est utilisé par les experts métier

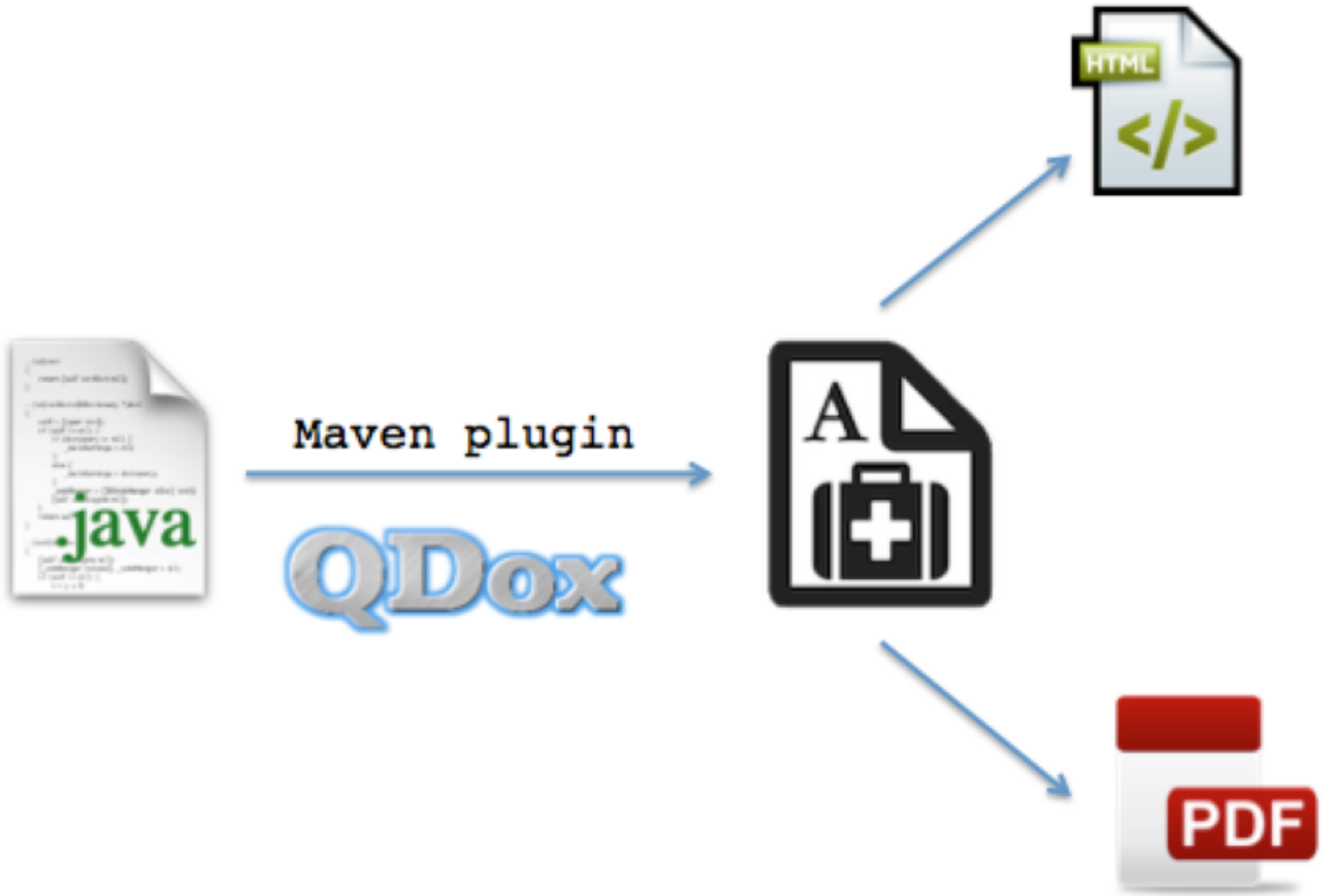
> *On doit retrouver le même vocabulaire dans le code !*

Ajouter la documentation dans le code

Annotation *marqueur*

```
@Glossary
/*
A peculiar person, especially one who is perceived to be overly
intellectual, unfashionable, or socially awkward
*/
public class Geek {
   ....
}
```

Génération à partir du code



- https://github.com/paul-hammant/qdox

Un joli glossaire à chaque build

Glossary

Geek

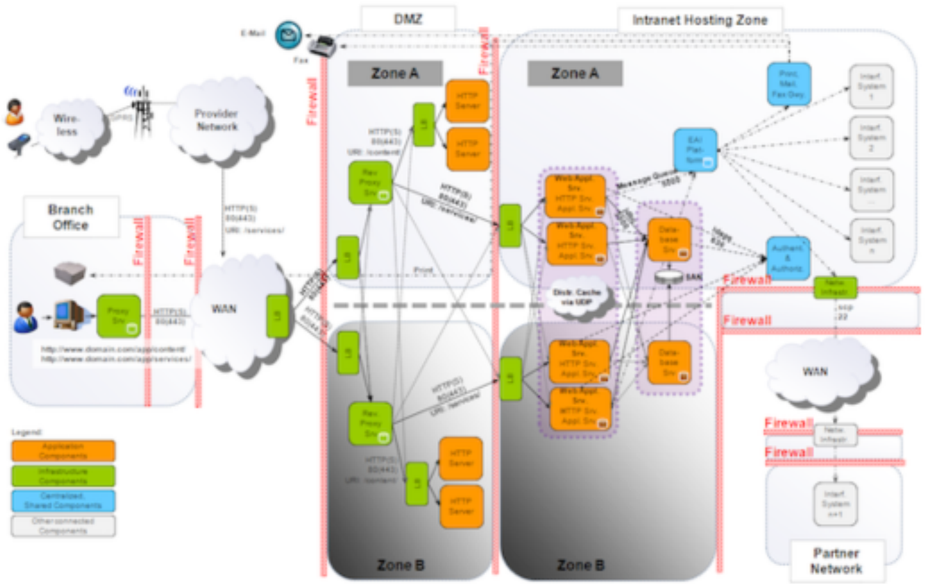
A peculiar person, especially one who is perceived to be overly intellectual, unfashionable, or socially awkward

Legacy

Surviving computer systems, hardware, or software

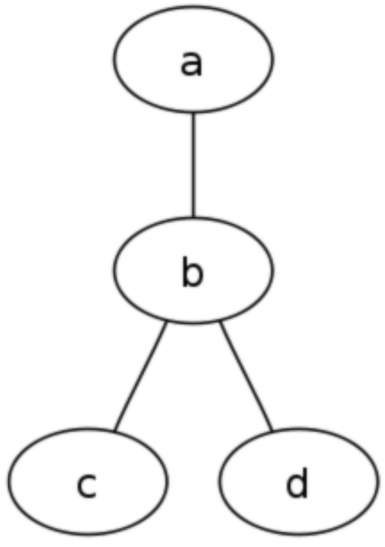
Documentation Architecture

> *L'architecte, c'est celui qui fait les diagrammes !*

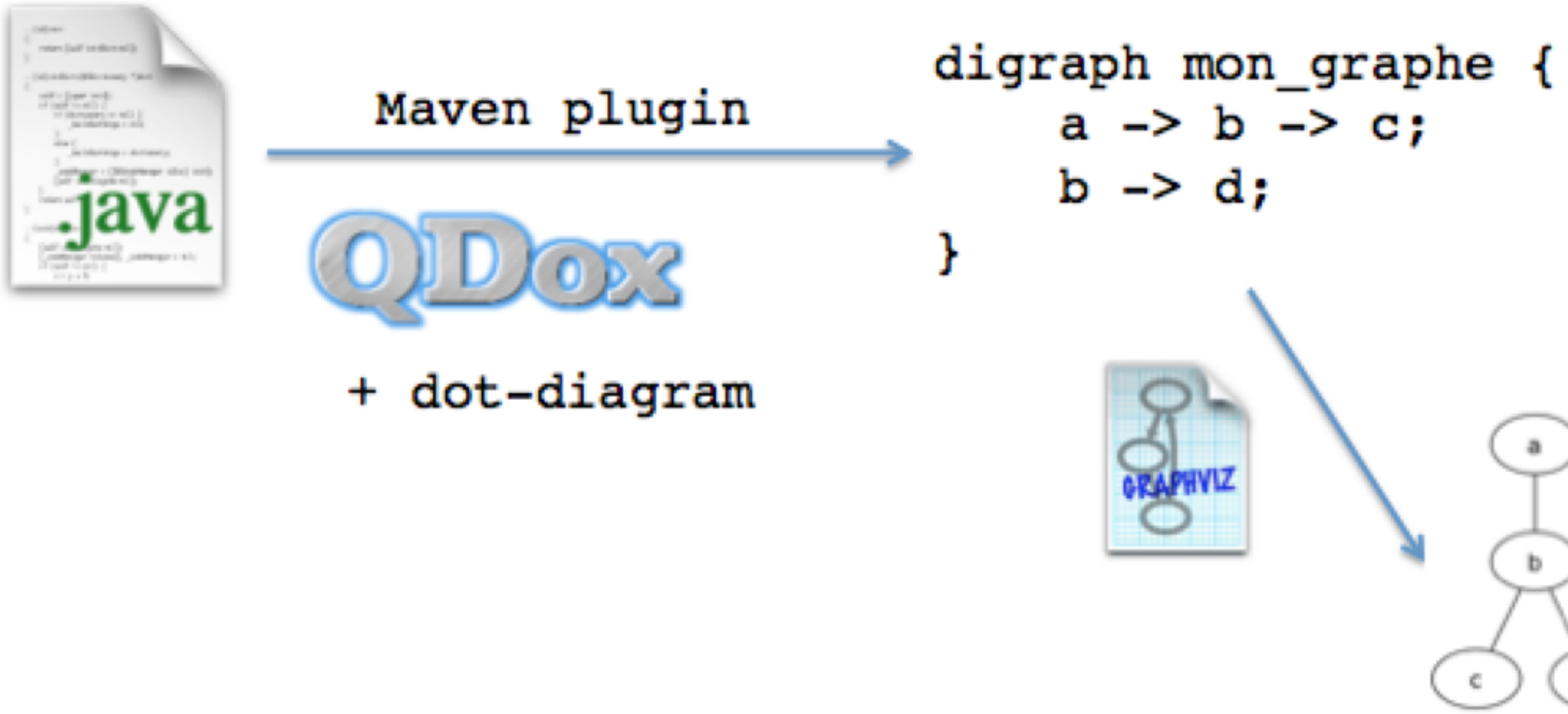


Syntaxe dot

```
digraph mon_graphe {
    a -> b -> c;
    b -> d;
}
```



Génération depuis le code



- https://github.com/cyriux/dot-diagram

Visualisation du graphe

- http://www.graphviz.org/
- https://vincenthee.github.io/DotEditor/
- Plantuml supporte `dot`

```
@startdot
...
@enddot
```

Plugins Google Chrome et atom

Diagramme du domaine

- marquer avec des annotations vos entités du domaine
- représenter avec un graphe les interactions entre entités

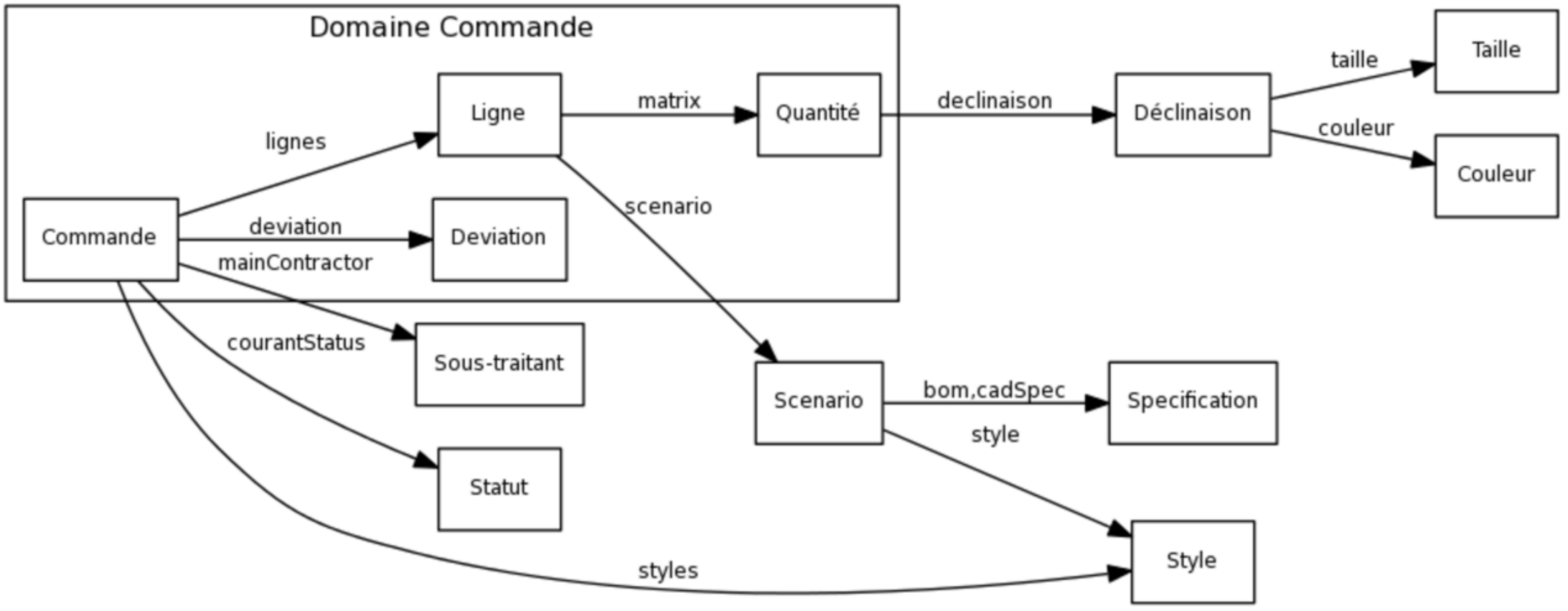
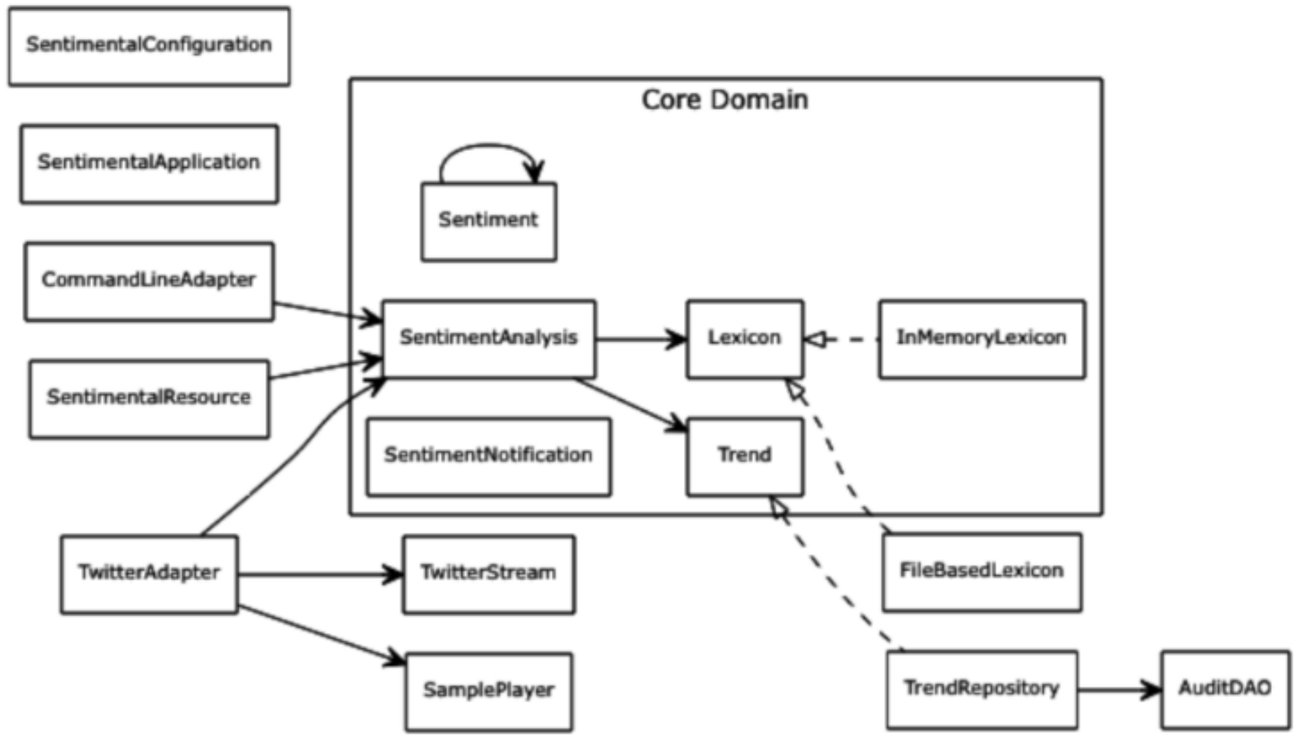


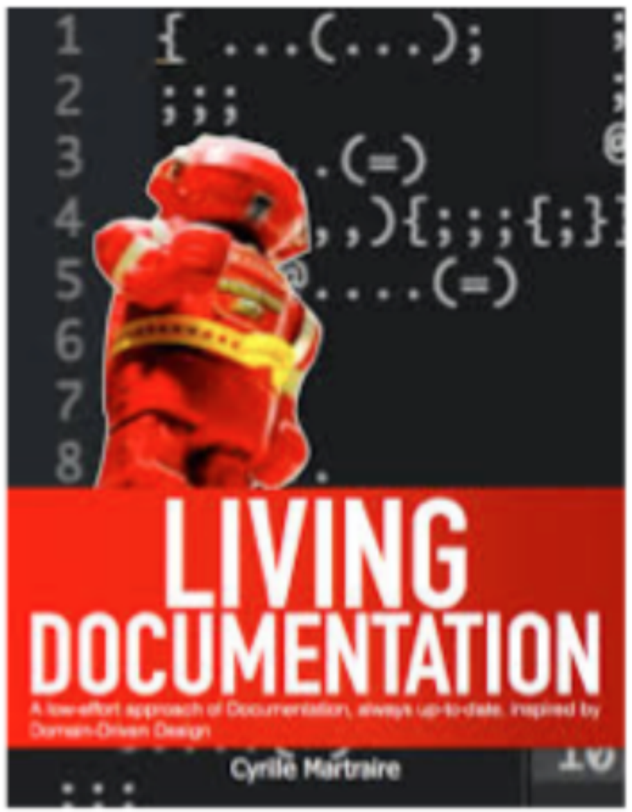
Diagramme d'architecture

- Peut permettre de ***visualiser*** les choix de design
    - "*mon domaine ne doit pas dépendre de mon infrastructure*"



Hexagonal Architecture Living Diagram generated from the source code

Cukedoctor



cucumber

Java

@nnotations

QDox

Overview

Maven™

Organisation Github

Living Documentation

An organization to gather all contributions about Living Documentation

Repositories People 1 Teams 0 Settings

Filters ▾ Find a repository... New repository

awesome-living-documentation

★ 0 0

A curated list of stuff about Awesome Living Documentation

Updated 38 minutes ago