

# SOFTWARE ENGINEERING FOUNDATIONS – SYNTHESIS / Q&A

Matthieu Tixier – #EOF

# Outline



- A possible synthesis
- How to prepare for the GL02 exam?
- The software project
- Specification
- Implementation
- Maintenance and evolution
- Perspectives

# A synthesis?

- GL02 pedagogical aims
  - ▣ Experience the links between specification, implementation and evolution at the heart of the software life-cycle
    - ➔ A practice based perspective
  - ▣ Knowledge (connaissances)
    - Defining a software project specification (data, process)
    - Developing programs from a specification
  - ▣ Skills (compétences)
    - Organize the cooperative work of developing a software solution
    - Check the conformity between a software and its SRS (Software Requirements Specification – Cahier des charges)
    - Organize the maintenance and evolution of a software project

# GL02 exam preparation?

- Software engineering as a practice
- Overall understanding
  - Application of knowledge through TP/TD and the project
- ➔ (Re)call several core subjects together
- What will not be on the exam...
  - ▣ Historical references and figures → culture générale
  - ▣ Craft exact copies of schema or diagrams
  - ▣ Giving the complete man page for a SVN command
    - → your own understanding of the command will be sufficient
  - ▣ English and French terms are welcome as well

# What is software?

5

## □ Software (Logiciel) :

*A set of computer programs that aims to support general or specific functions (related to an organizational context) as well as the associated documentation that allows its use, maintenance and evolution.*

## □ A set of resources

- Source code
- Software Requirements Specification (SRS – Cahier des charges)
- Licence
- Test
- Documentation
  - Product
  - Process

# Software life-cycle

6

- The software project
    - ▣ **Analysis** : understanding the problem or needs, state of the art of available and missing technology and solutions
    - ▣ **Design** (Conception) : define the features of the solution, the data process and formats, the performance and security criterion.
    - ▣ **Implementation** : programming and realization of the solution
    - ▣ **Tests** : check the conformity of the solution against the features specifications (alpha)
    - ▣ **Déploiement/Evaluation** : Assess the problem and needs are responded in context (bêta)
    - ▣ **Improvement** (Amélioration) : take into account the evaluation results and users feedback, plan future version
- ➔ At the scale of a project/contract or through several iterations (@see the different software project process #1)

# Specification

- Software Requirements Specification - Cahier des charges
  - ▣ Functional/Non functional requirements
  - ▣ User/System requirements
- Different methods for requirements specification
  - ▣ Natural language
  - ▣ Structured natural language
  - ▣ Scenario / Users stories
  - ▣ Mockups
  - ▣ Models
  - ▣ **Formal specification**
- Data format specification
- Abstract Data Type (ADT - Spécification algébriques)

# Types of formal grammar

8

## □ Chomsky hierarchy (1959)

▣ Gathers 4 classes of decreasing expressivity (from 0 to 3)

■ For  $\gamma \in (N \cup \Sigma)^*$  and  $\varepsilon$  the empty symbol

Type <i>Type</i>	Nom <i>Name</i>	Forme des règles <i>Constraints on grammar rules</i>
0	Récurivement énumérable <i>Recursively enumerable</i>	$\alpha \rightarrow \gamma$ tel que $\gamma \neq \varepsilon$
1	Sensible au contexte <i>Context sensitive grammar</i>	$\alpha A \beta \rightarrow \alpha \gamma \beta$ tel que $\gamma \neq \varepsilon$
2	CFG (« Context Free Grammar ») <i>Grammaire algébrique</i>	$A \rightarrow \gamma$
3	<b>Grammaire régulière</b> <i>Regular grammar</i>	$A \rightarrow \gamma B$ ou $A \rightarrow \gamma$ avec $\gamma \in \Sigma^*$



# Application for software engineering - BNF

9

## □ BNF (Backus–Naur Form)

### ▣ Notation and formalism equivalent to CFG

`<symbol> ::= __expression__`

- The non terminal symbols are used to be noted with `<>`
    - The non terminal symbol are virtually all left hand part of rules symbols
  - `__expression__`
    - One or several symbol sequences (that can be separated by `|`)
  - `::=`
    - The left hand part of the rule can be substituted by the right hand (cf. )
- ▣ Used to specify network protocols and file formats (cf. IETF) as well as the syntax of many programming languages (cf. ALGOL, Javascript, Python ...)

➔ ABNF standard follows the same principles (RFC5324) and is more precise and better

# ABNF – Remarks

10

## □ [B.1] Core Rules (all in uppercase) – very useful

ALPHA = %x41-5A / %x61-7A  
; A-Z / a-z  
CHAR = %x01-7F  
; any 7-bit US-ASCII character,  
; excluding NUL  
CR = %x0D  
; carriage return  
CRLF = CR LF  
; Internet standard newline  
WSP = SP / HTAB

DIGIT = %x30-39  
DQUOTE = %x22  
HEXDIG = DIGIT / "A" / "B" / "C" /  
"D" / "E" / "F"  
HTAB = %x09  
LF = %x0A  
; linefeed  
SP = %x20  
VCHAR = %x21-7E  
; caractères visibles

## □ [2.4] Character encoding

- ▣ ABNF is defined for the ASCII characters (ie, vs UTF-8)
- ▣ Flexibility for the GL02 project (accents included in VCHAR etc)
  - No need to re-define the « core rules » for UTF-8 or else...

# Examples

11

## □ The JSON format – Javascript Object Notation

```
{
  "Image":
  {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": "http://www.example.com/image/481989943",
      "Height": 125,
      "Width": 100
    },
    "Animated" : false,
    "IDs": [116, 943, 234, 38793]
  }
}
```

# Examples

12

## □ ABNF of the JSON format (RFC7159)

JSON-text = ws value ws

begin-array = ws %x5B ws

; [ left square bracket

begin-object = ws %x7B ws

; { left curly bracket

end-array = ws %x5D ws

; ] right square bracket

end-object = ws %x7D ws

; } right curly bracket

name-separator = ws %x3A ws

; : colon

ws = \*(

%x20 / ; Space

%x09 / ; Horizontal tab

%x0A / ; Line feed

%x0D ) ; Carriage return

value = false / null / true / object /  
array / number / string

false = %x66.61.6c.73.65

; false

null = %x6e.75.6c.6c

; null

true = %x74.72.75.65

; true

# ADT Format

13

- Specification title
- The « sort » name (abstract type)
- References to other sort
- A full text description of the operations
- Preconditions (optional)
- Operations signatures
- Axioms
  - Construction vs inspection operation

< SPECIFICATION NAME >

sort < name >

imports < LIST OF SPECIFICATION NAMES >

Informal description of the sort and its operations

Operation signatures setting out the names and the types of the parameters to the operations defined over the sort

Axioms defining the operations over the sort

# Example - List

14

Title : LIST(E)

Sort : List

References : INTEGER

Description :

Defines a list where elements are added at the end and removed from the front. The operations are Create, which brings an empty list into existence, Cons, which creates a new list with an added member, Length, which evaluates the list size, Head, which evaluates the front element of the list, and Tail, which creates a list by removing the head from its input list. Undefined represents an undefined value of type E.

-----

Create :  $\rightarrow$  List

Cons : List  $\times$  E  $\rightarrow$  List

Head : List  $\rightarrow$  E

Length : List  $\rightarrow$  Integer

Tail : List  $\rightarrow$  List

-----

Head(Create) = Undefined (error empty list)

Head(Cons(L, v)) = **if** L = Create **then** v **else** Head(L)

Length(Create) = 0

Length(Cons(L, v)) = Length(L) + 1

Tail(Create) = Create

Tail(Cons(L, v)) = **if** L = Create **then** Create **else** Cons(Tail(L), v)

# Implementation

- Configuration Management System
  - ▣ Sourceforge
  - ▣ Version management: SVN
- Software Configuration Item (SCI)
  - ▣ Any project resource can be placed under configuration control
    - Source code
    - Design document
    - Dataset
    - User manual
    - Licence
    - [...]
  - ▣ Uniquely identified resources
  - ▣ That can exist in different versions
- Important distinction between distant « repository » and local working copy


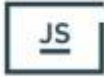




# Implementation

## □ Javascript



### Compétences de langage

Apprenez les concepts et syntaxes de base pour les langages de programmation les plus populaires.

 <h4>HTML &amp; CSS</h4> <p>Apprenez les bases du développement web avec l'HTML et le CSS, et créez votre propre site internet dès la fin du cours !</p>	 <h4>JavaScript</h4> <p>Dernière activité il y a 5 days</p> <div><div></div>9%</div>	 <h4>jQuery</h4> <p>jQuery utilise JavaScript pour faire facilement des sites interactifs. Apprenez à faire des animations et des petits événements sur une page !</p>
 <h4>PHP</h4> <p>PHP est le langage "côté serveur" le plus populaire, et est relativement facile à apprendre.</p>	 <h4>Python</h4> <p>Apprenez les fondamentaux de la programmation pour développer des applications Web et gérer des données.</p>	 <h4>Ruby</h4> <p>Ruby est un outil puissant, et pourtant il est facile de commencer à l'apprendre ; il est utilisé pour le développement des sites, partout dans le monde.</p>



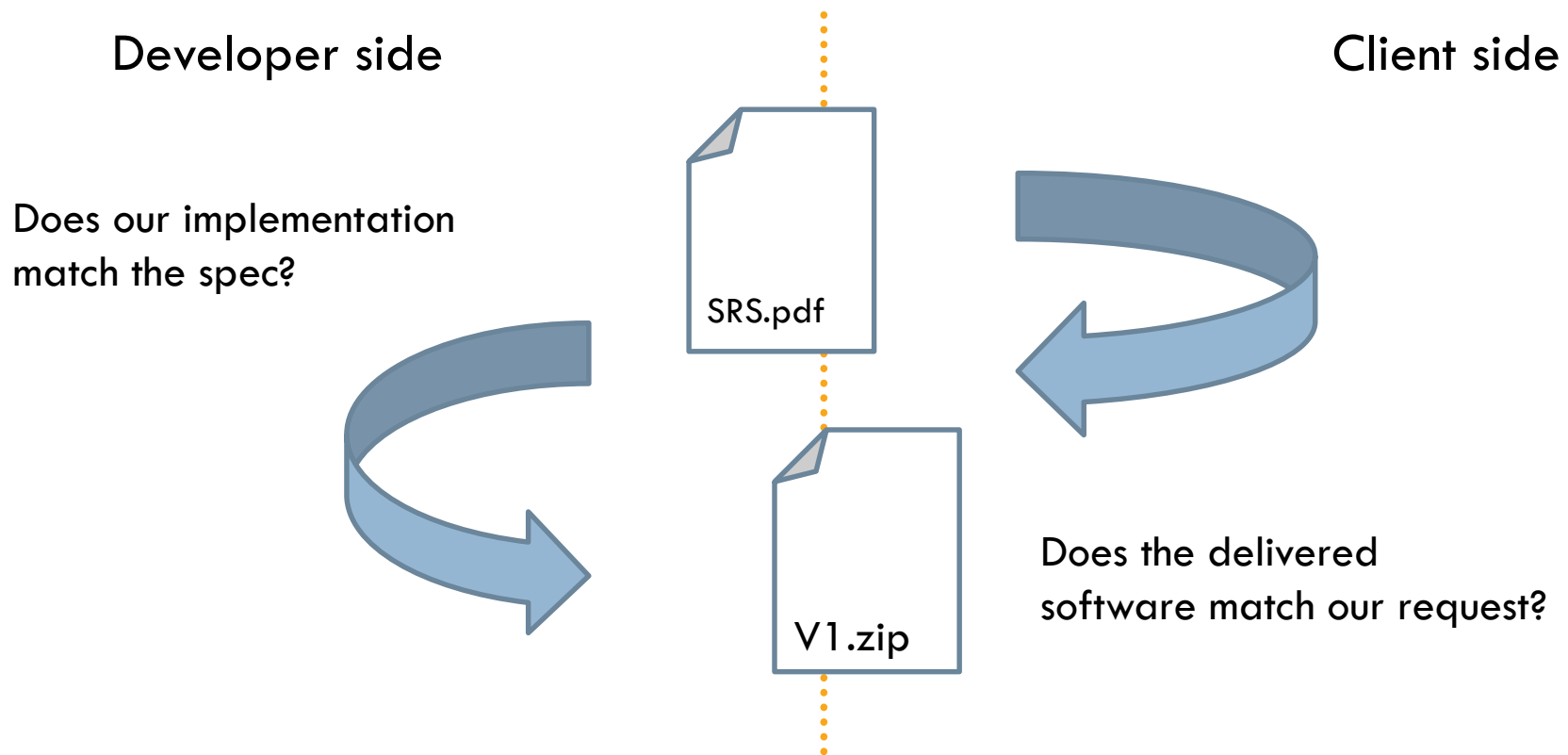
# High order and Pure function

- High order function : function that takes function(s) as argument
- Pure function
  - Properties
    - Stateless: the function doesn't rely or produce side effect on the program or execution environment
    - Determinist: given the same input argument, the same result is always computed
- → Good example: `filter()`, `map()`, `reduce()`
- → Counter example: `time()`, `random()`

# Software project and test

18

- Software requirements specification as a « boundary »



➔ Tests as a central tool to check and show the conformity with the specifications

# Unit test – Jasmine

## □ Framework de test unitaires

### ▣ Structuration : 1\* Test cases within Test suite

- Test case (a suite of assertions)
- Test suite = gather several test cases
- Stored a part from the software code (ie, proper directory)

### ▣ Assertions test for

- Being true
- Being of a special value (ie, Nan, undefined...)
- Strict equality (===)
  - Type identity, make a difference between undefined and null
- Deep equality
  - Value identity for structured data (ie, object)

### ▣ Module and test cases titles as well as assertion return messages should be consistent and clear

# Jasmine – Assertions

20

- Assertions are using Matchers with Jasmine

<https://jasmine.github.io/api/edge/matchers.html>

- Expect() : Write something that is true about your program

```
describe("Checking the rocket is working", function(){  
    // plenty of other test cases...  
  
    it("has a Radio that makes bip..bip", function(){  
        // assertions  
        expect(this.crashTestRocket.radio).toBeDefined();  
        expect(this.crashTestRocket.radio).toBeTruthy();  
        // negation  
        expect(this.crashTestRocket.radio).not().toBeNaN();  
        expect(this.crashTestRocket.radio).not().toBeFalsy();  
        expect(this.crashTestRocket.radio).not().toBeNull();  
        expect(this.crashTestRocket.radio).not().toBeUndefined();  
    });  
  
    // there can be more it() and nested describe() as well  
});
```

# Maintenance and evolution

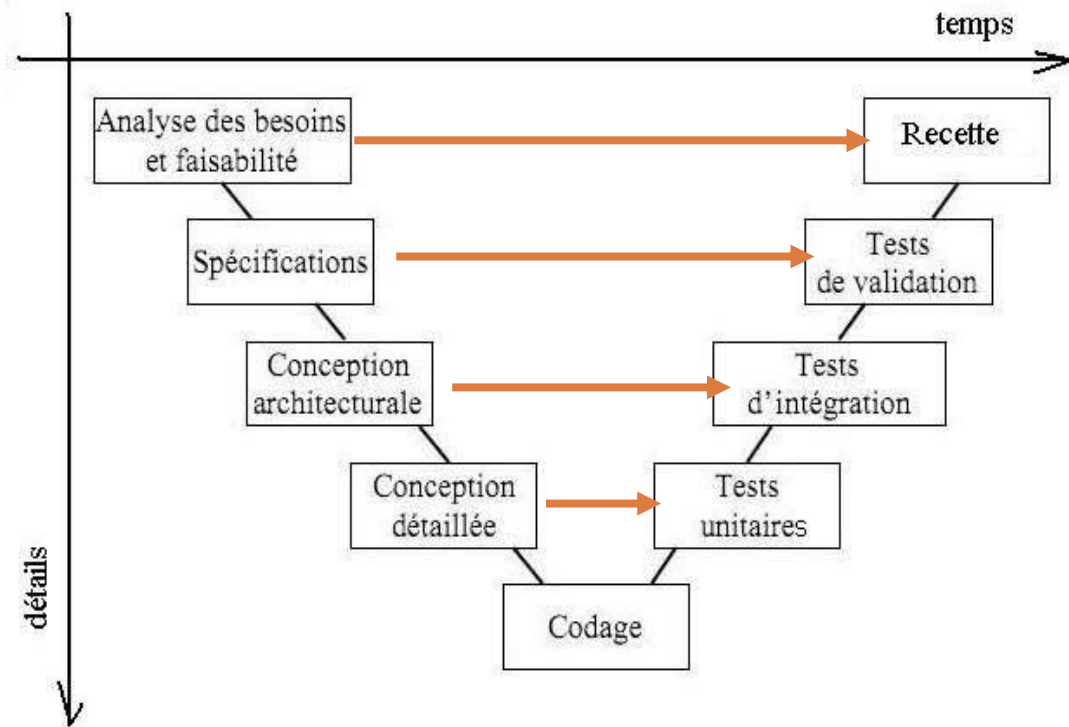


- Code reuse at stake
  - ▣ Open source Licences
- Test plan
  - ▣ Unit test
  - ▣ Acceptance test plan (« Tests de recette »)

# Tests

## □ Acceptance tests

- End user standpoint
- General test review before delivery



Cycle en V

# Maintenance and evolution

- Documentation
  - ▣ Process (of making the software project)
  - ▣ Product
    - System documentation
    - User documentation
  - ▣ Technical writing guidelines
- Refactoring - « Réusinage de code »
  - ▣ The software decay model
  - ▣ Code review
  - ▣ Bug and enhancement reports

# Perspectives

## □ Software projects success and failures...

**50% of all new app development requests end in failure.**



**15%**

never started



**20%**

delivered, but doesn't  
meet business needs



**15%**

started, never completed


International Data Group (IDG), 2018

<https://www.developpez.com/actu/228268/Etude-50-pourcent-des-projets-de-developpement-d-applications-se-soldent-par-un-echec-cela-est-il-du-a-la-lenteur-des-codeurs-et-la-dette-technique/>



# Perspectives

- A set of professional concerns and activities

- 
- **Analyse (Analysis)** IF14
  - **Conception (Design)** IF02
  - **Implementation** NF16 LO02
  - **Tests** IF05
  - **Déploiement/Evaluation** IF10
  - **Evolution**

- Different technological and application context
  - Web (LO07), Web services (LO10), Mobile applications (IF26), Platforms for sharing economy (IF31), Data Analytics (IF29), Data visualization (IF36) ...
- Orientation filière (ATN | | VDC | | IPL)



- Thanks for your attention

- Question(s) ?

- I. Sommerville, *Software Engineering*. Pearson Education, 2009.