Génie Logiciel

MODULE 2 - SPÉCIFICATION ET ARCHITECTURE TECHNIQUE DE LA SOLUTION

VENDREDI 13/11

Découpage de ce cours de Génie Logiciel

Cours + TD1

Overview d'un projet informatique

Cours + TD2

Spécification et architecture technique de la solution

- Spécification fonctionnelle
- Architecture technique

Cours + TD3

Construction de la solution

Cours + TD4

Tests de la solution

01

Spécification fonctionnelle de la solution

Quels sont les entrants et les livrables lorsque l'on rédige une spécification fonctionnelle ?

ENTRANTS

- **Quoi ?**
  - Cahier Des Charges (CDC)
  - Expression de Besoin (EB)
- **Qui ?**
  - MOA – Expert métier
- **Pourquoi ?**
  - Exprimer le besoin des utilisateurs et leurs taches à informatiser
  - Exprimer les contraintes non fonctionnelles de la future solution (système)

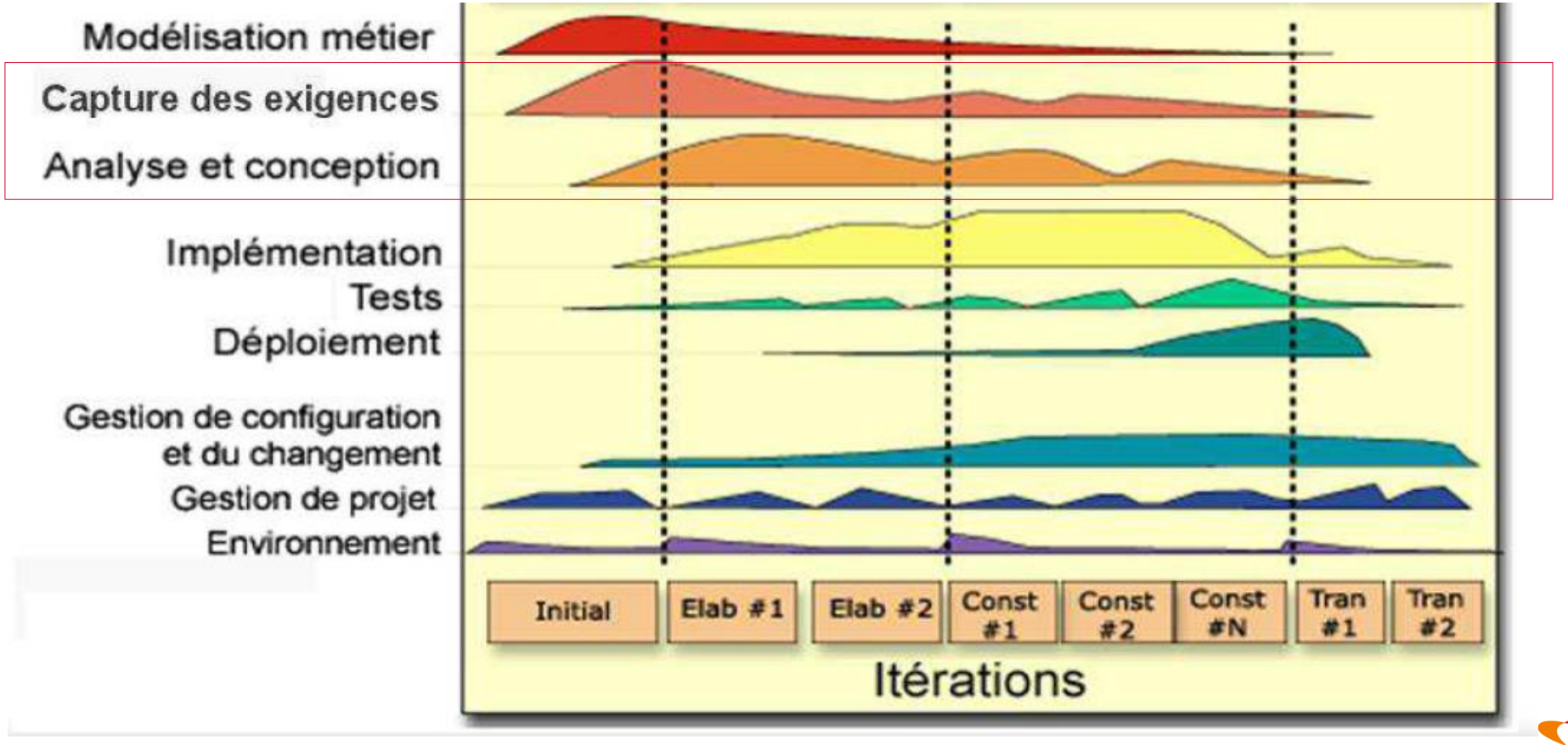
LIVRABLES

- **Quoi ?**
  - Spécification Fonctionnelle Générale (SFG)
  - Spécification Fonctionnelle Détaillée (SFD)
  - Spécification Techniques Détaillées (STD) – cf deuxième partie du cours
- **Qui ?**
  - Rôle fonctionnel (Business Analyst)
- **Pourquoi ?**
  - Retranscrire le besoin dans la solution informatique
    - Pour que le client comprenne ce que l'on va faire
    - Pour que le développeur comprenne ce qu'il doit implémenter

Comment définiriez-vous une exigence informatique ?

Quels seraient les critères de qualité d'une exigence ?

klaxoon



Qu’est-ce qu’une exigence ?

CMMI

Une exigence définit une caractéristique (propriété ou condition) que doit posséder la solution pour répondre aux besoins de ses utilisateurs ou se conformer à une contrainte imposée (norme, standard, spécification ou autre)

Sommerville & Sawyer 1997

Les exigences sont la spécification de ce qui va être implémenté.
Elles correspondent à la description du comportement du système ou à des propriétés ou attributs du système.

Principales caractéristiques d'une exigence

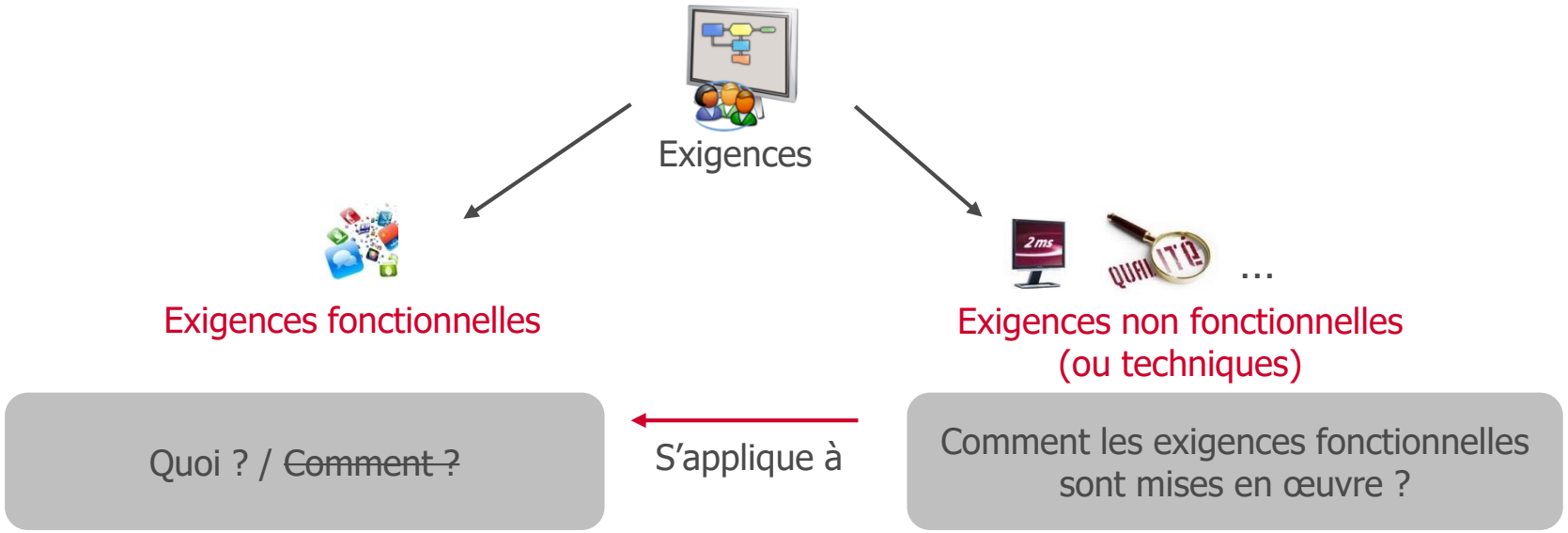
Il n'y a pas de définition détaillée partagée par tous, mais il existe, dans la littérature, un consensus sur certaines caractéristiques clés.

Caractéristique	Description
Unitaire	Adresse un seul sujet
Complète	Sa description est suffisamment détaillée pour permettre d'assurer les travaux de réalisation
Faisable	Réalisable en vue des contraintes du projet
Cohérente	Elle ne doit pas contredire d'autres exigences établies, ni être contredite par d'autres exigences
Non ambigüe	Sa description doit être claire et précise
Vérifiable	Testable

Exigences fonctionnelles / Non fonctionnelles

Les exigences sont à classer en deux catégories :

- Exigences fonctionnelles : présentent un objectif métier ou une fonction rendue à l'utilisateur final. Ce que doit faire la solution.
- Exigences non fonctionnelles (ou techniques) : matérialisent les contraintes et conditions que la Solution doit respecter.



Exigences fonctionnelles / Non fonctionnelles

Exemples

Le système calcule (doit calculer) le taux d'endettement

Taux d'endettement = Somme des charges / Sommes des revenus

Le temps de réponse ne doit pas dépasser n secondes.

Le résultat du calcul est affiché en rouge, en gras sur 4 caractères

Exigences fonctionnelles / Non fonctionnelles

Exemples

Le système calcule (doit calculer) le taux d'endettement

Exigence fonctionnelle

Taux d'endettement = Somme des charges / Sommes des revenus

Exigence fonctionnelle

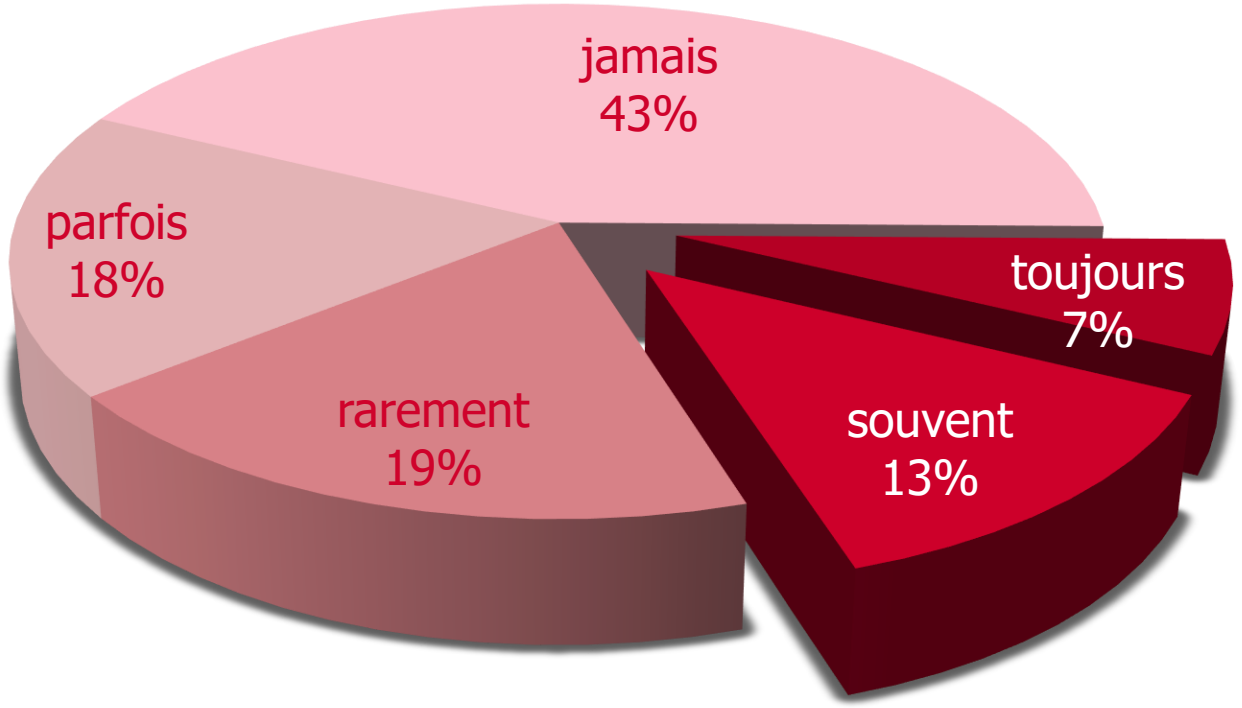
Le temps de réponse ne doit pas dépasser n secondes.

Exigence non fonctionnelle (performance)

Le résultat du calcul est affiché en rouge, en gras sur 4 caractères

Exigence non fonctionnelle (ergonomie)

La difficulté à capturer les « bonnes » exigences



Etude réalisée vers 2000 sur des milliers de projets dans le monde anglo saxon

➔ Il est essentiel de prioriser les exigences

La difficulté à capturer les « bonnes » exigences

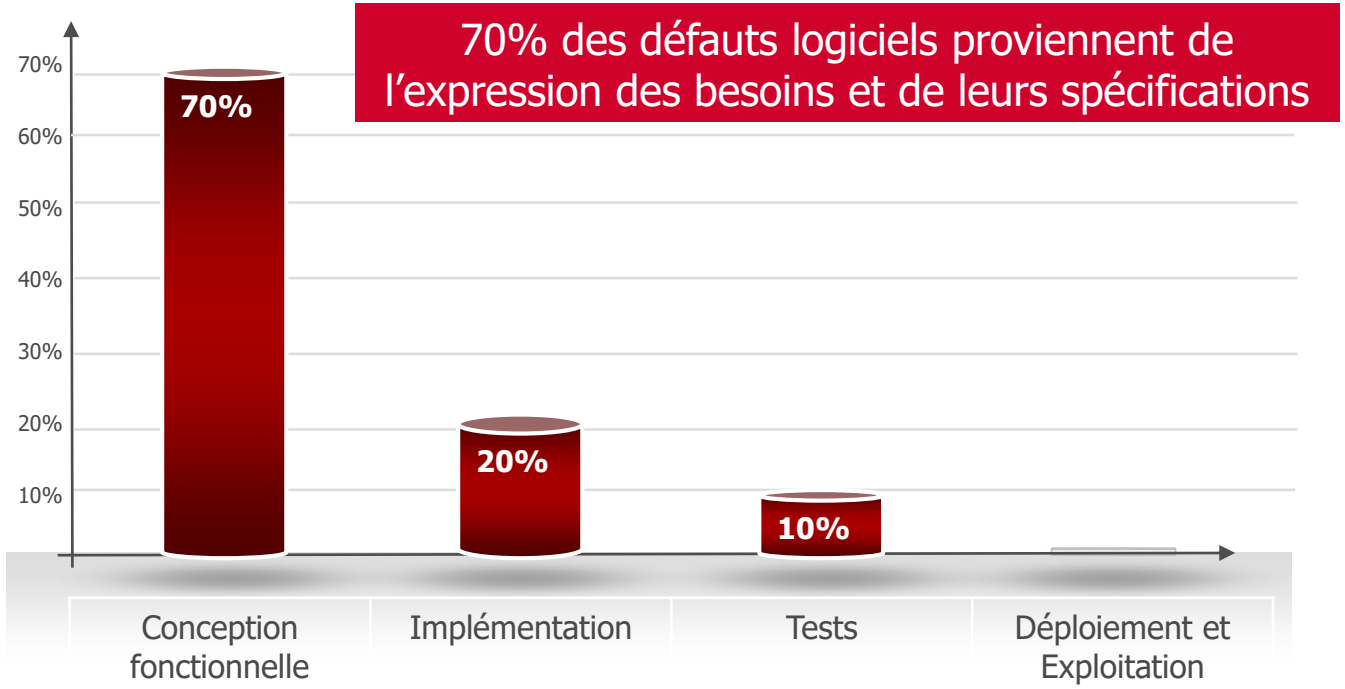
Coût de la détection d'une exigence

Une exigence détectée en phase de conception coûte 1, en phase d'implémentation 10, en test 100, en déploiement >100

Plus la détection d'une exigence se trouve en aval du cycle de vie d'un projet, plus elle s'avère coûteuse à implémenter

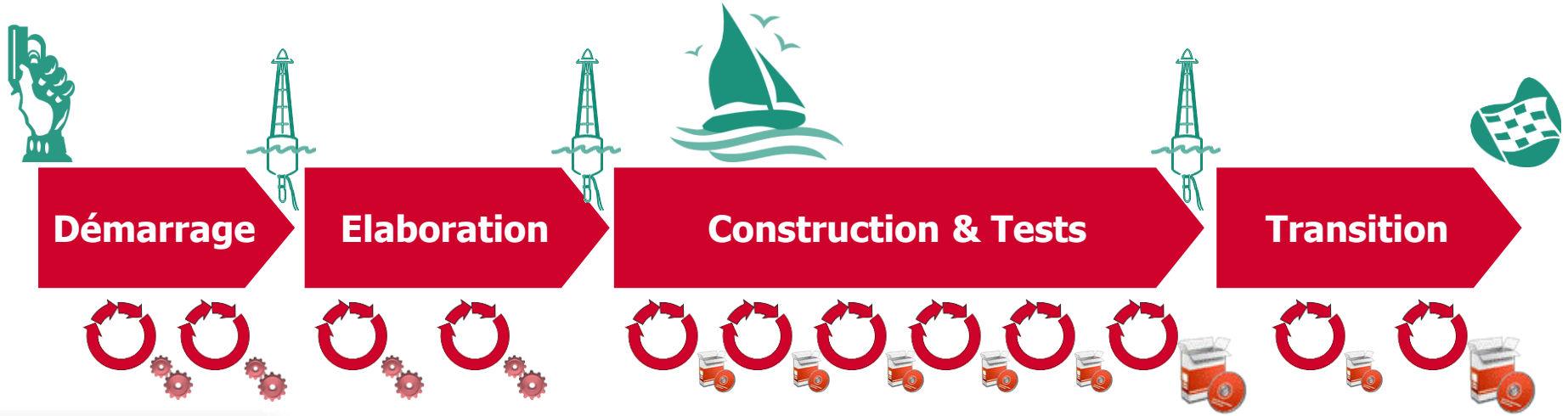
Répartition des introductions de défaillances

70% des défauts logiciels proviennent de l'expression des besoins et de leurs spécifications



Source « National Institute of Standards & Technologies »

La capture des exigences dans le delivery process



Définie la vision (scope), permet de parvenir à une compréhension approfondie du problème et du contexte métier

- Formalise les exigences fonctionnelles et techniques
- Décrit les cas d'utilisation

80 %

- Les processus de validation systématiques des versions intermédiaires permettent de mettre en lumière au plus tôt les défauts fonctionnels et techniques
- La version complète de la release est disponible

20 %

Cette vision doit être partagée/validée avec les parties prenantes Client

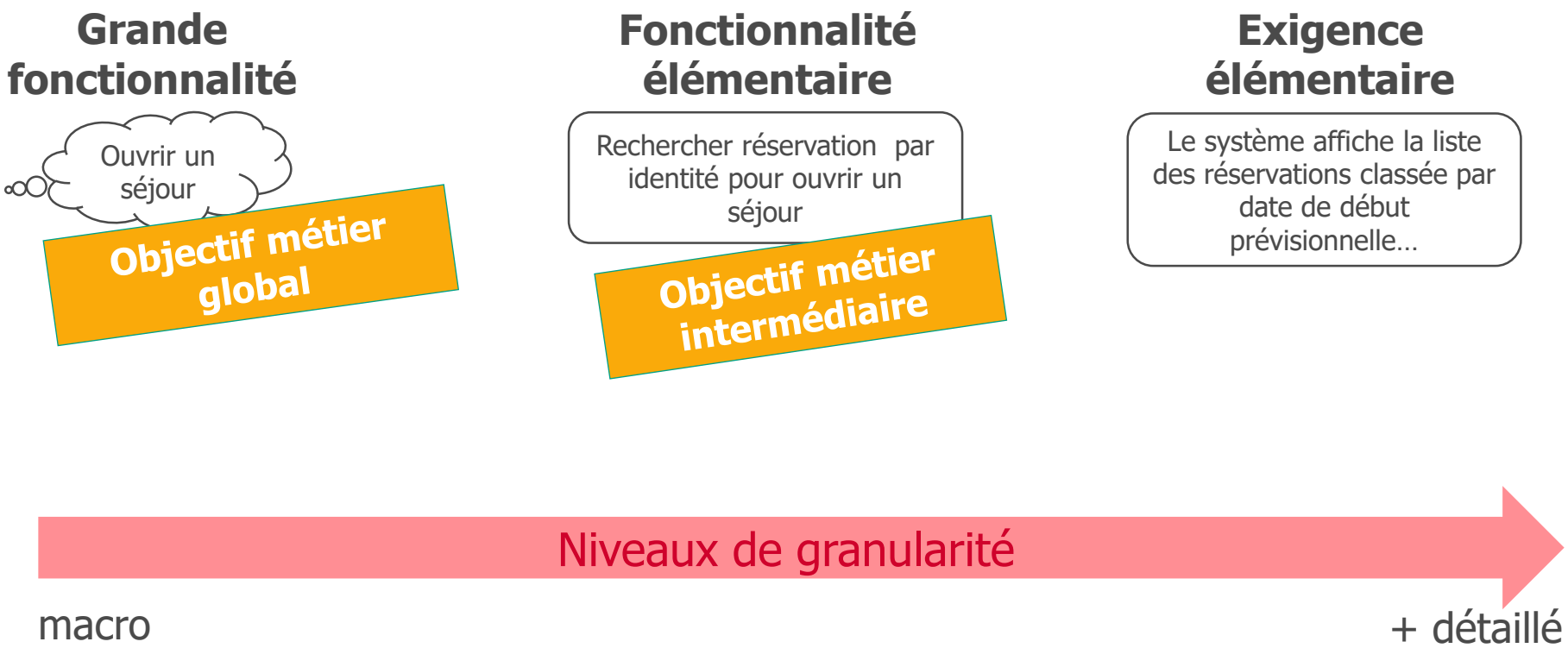
Si je vous exprime un besoin très vaste.

Je suis un constructeur qui doit monter une maison, comment je m'y prend ?

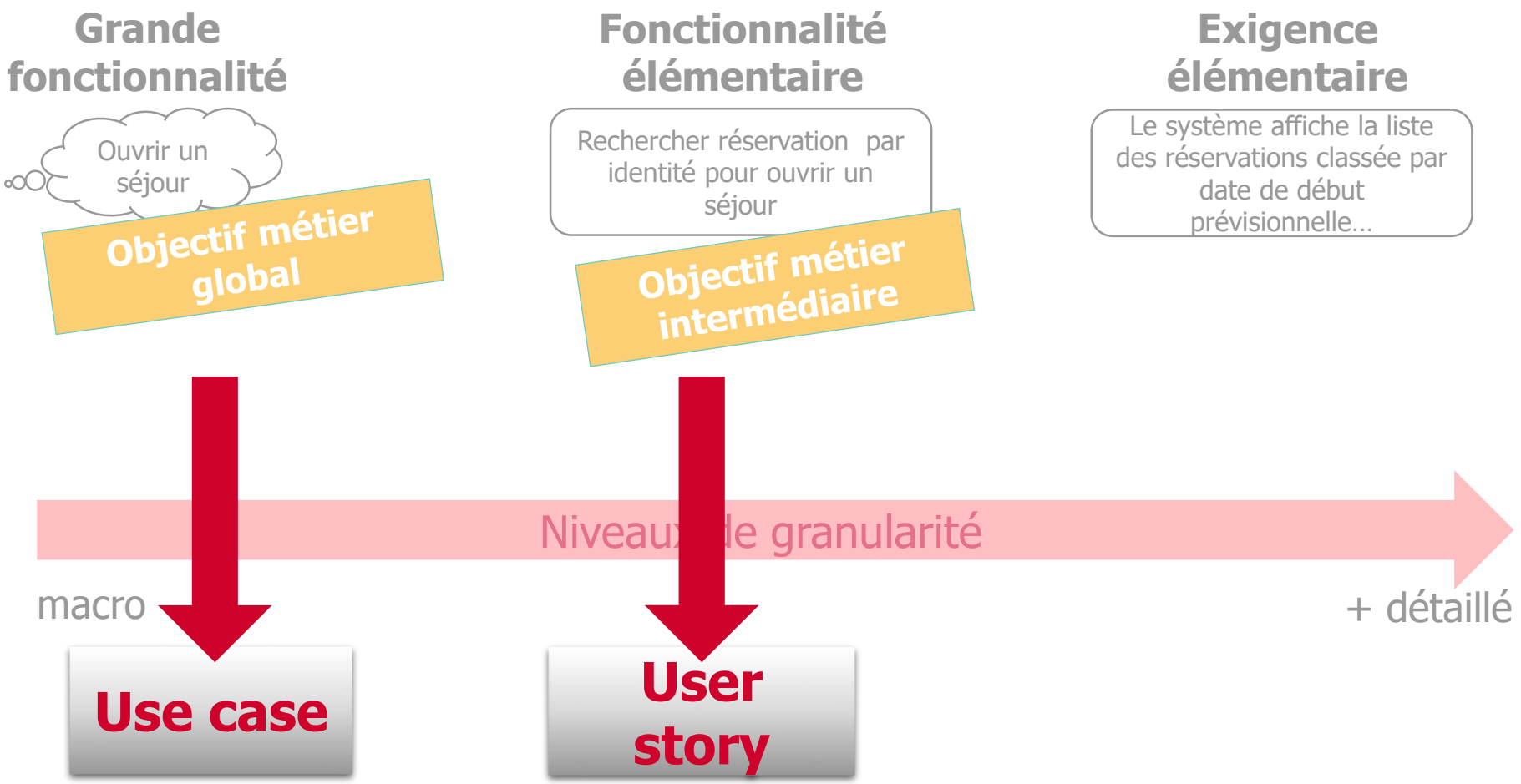
Finalement, j'utilise une méthode qui peut-être tout aussi bien appliquée en informatique…

klaxoon

Exigences fonctionnelles : niveaux de granularité



Exigences fonctionnelles : niveaux de granularité

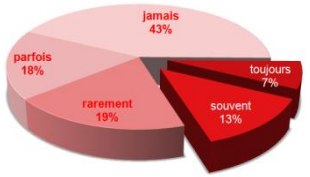


Pourquoi spécifier via la technique des stories ?

Use Case ou User story

- Lien explicite avec le métier
- Explicite l'enchainement des fonctionnalités
  regroupe toutes les fonctionnalités nécessaires à la réalisation d'un objectif utilisateur
- Se placer du point de vue de l'utilisateur
  raconte, sous forme d'histoire, comment l'acteur utilise le système pour atteindre son objectif

Contribue au développement d'une majorité de fonctionnalités réellement utilisées grâce à un meilleur alignement sur le métier et une priorisation facilitée



Dans le cadre de notre application IterHotel, nous avons identifié un Use Case : « Ouvrir un séjour »

Pour ouvrir un séjour, lorsque le client arrive, le réceptionniste a plusieurs manipulations à réaliser (que l'on verra par la suite).

Sous quelle forme spécifieriez l'enchainement de ces étapes ?

klaxoon

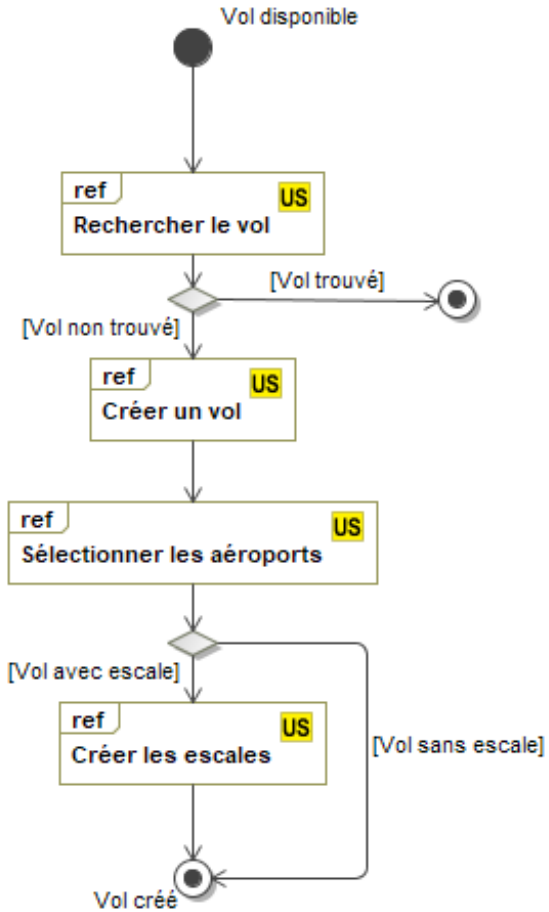
10 min / groupe

Lorsque le client arrive, les étapes sont :

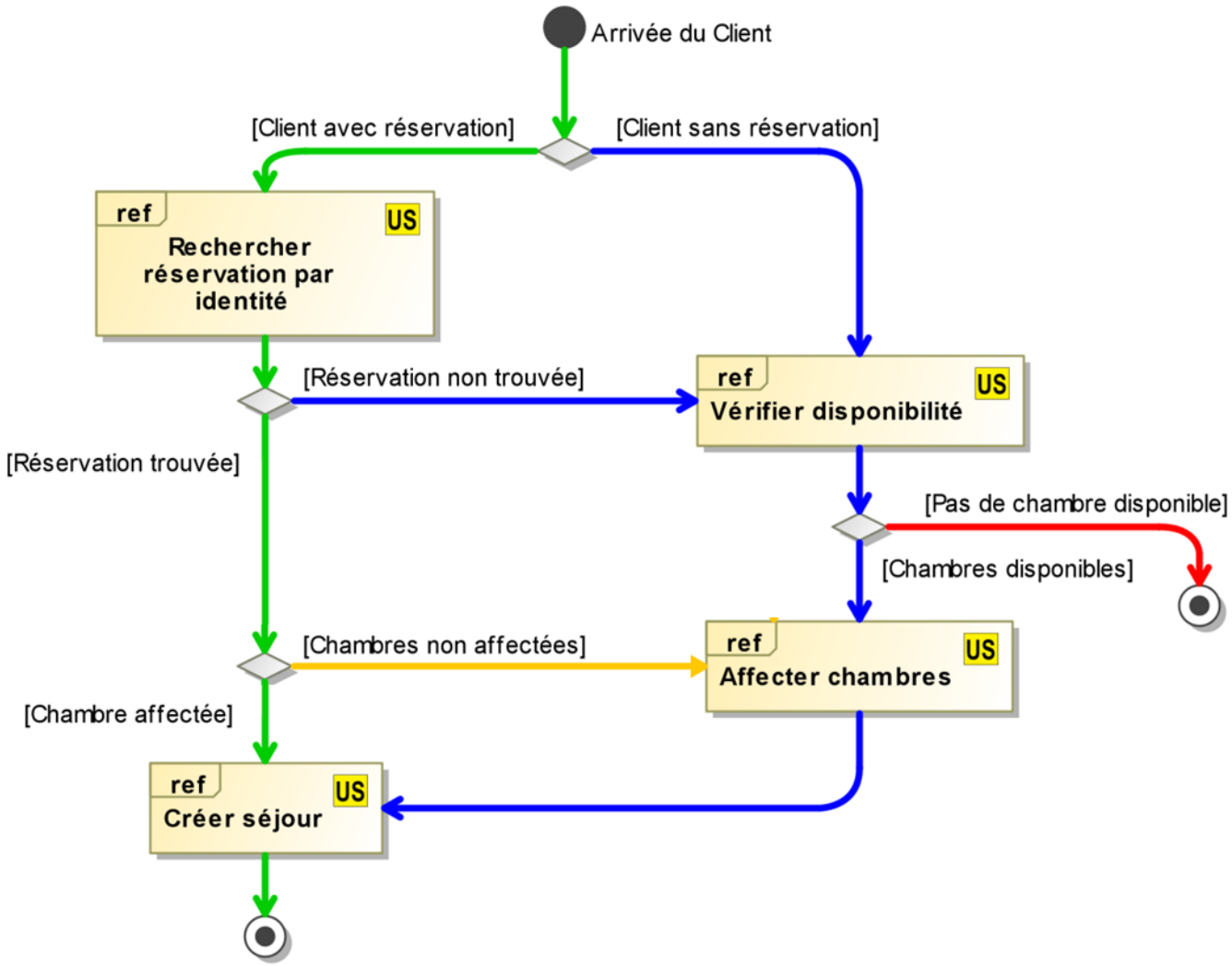
- La recherche de la réservation du client
- La vérification de la disponibilité de la chambre
- Affecter une chambre à la réservation
- Créer le séjour dans le système

Réalisez le diagramme d'activité

Exemple de diagramme :



Correction



- Ce workflow nous permet d'identifier d'identifier des scénarios utilisateurs
- Dans une itération, il est préconisé de développer un scénarios complet (pour l'ordre, pensez à la valeur métier)

Scénario 1 : Ouvrir un séjour avec réservation, chambre affectée et affectation acceptée par le Client

Scénario 2 : Ouvrir un séjour avec vérification de disponibilités

Scénario 3 : Ouvrir un séjour avec affectation de chambres pour un Client avec réservation

Scénario 4 : Ouverture de séjour impossible car pas de chambre disponible

Maintenant que les étapes de la réservation sont claires, comment décririez-vous l'étape « Rechercher une réservation par identité » ?

klaxoon

Décrire les User StorieS

Types de spécification

- Plusieurs formalismes possibles :
  - Description textuelle
  - Diagramme de classe
  - Maquettes (fil de fer, HTML navigables)
  - Cas de tests
  - …

  que l'on combine !

Attention cependant à limiter les redondances entre formalismes pour ne pas introduire d'incohérences et faciliter la maintenance

- Différentes combinaisons de formes de spécification peuvent être utilisées pour les spécifications des user stories d'une même solution

Rechercher une réservation – spécification détaillée

Exemple

1. Le réceptionniste choisit la recherche de réservation par identité et renseigne le nom (obligatoire) et le prénom (facultatif)
2. Le système recherche les réservations correspondant à l'identité renseignée (cf BR01) et affiche la liste des réservations correspondantes en indiquant le nom, le prénom, la date de début, la durée du séjour et le nombre de nuitées

3a. [Réservation non trouvée]
    3a1. La user story se termine

3b. [Réservation trouvée]
    3b1. Le réceptionniste choisit la réservation du client

4. Le système affiche le détail de la réservation (en plus des informations précédemment affichées) : nombre de personnes, nombre de petit déjeuners, prix par nuitée, prix du petit déjeuner, montant total

Le système répond à une action du réceptionniste

Rechercher une réservation – spécification détaillée

Exemple

1. Le réceptionniste choisit la recherche de réservation par identité
2. Le système recherche les réservations correspondant à l'identité renseignée (cf BR01) et affiche la liste des réservations

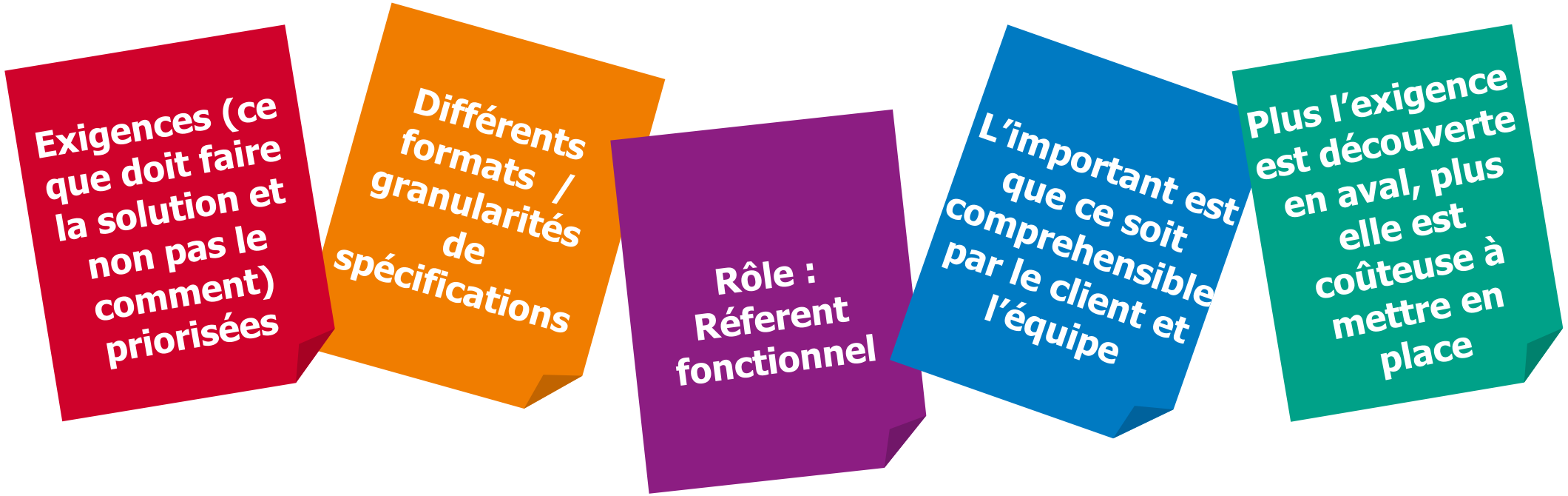
...

3b. [Réservation trouvée]
    3b1. Le réceptionniste choisit la réservation du client
4. Le système affiche le détail de la réservation

En tant que réceptionniste, je veux rechercher la réservation du client pour ouvrir son séjour

La maquette « fil de fer » se concentre sur l'enchaînement logique des écrans et leur description logique
Permet d'éviter de faire des choix de conception IHM

Les mots clés du cours

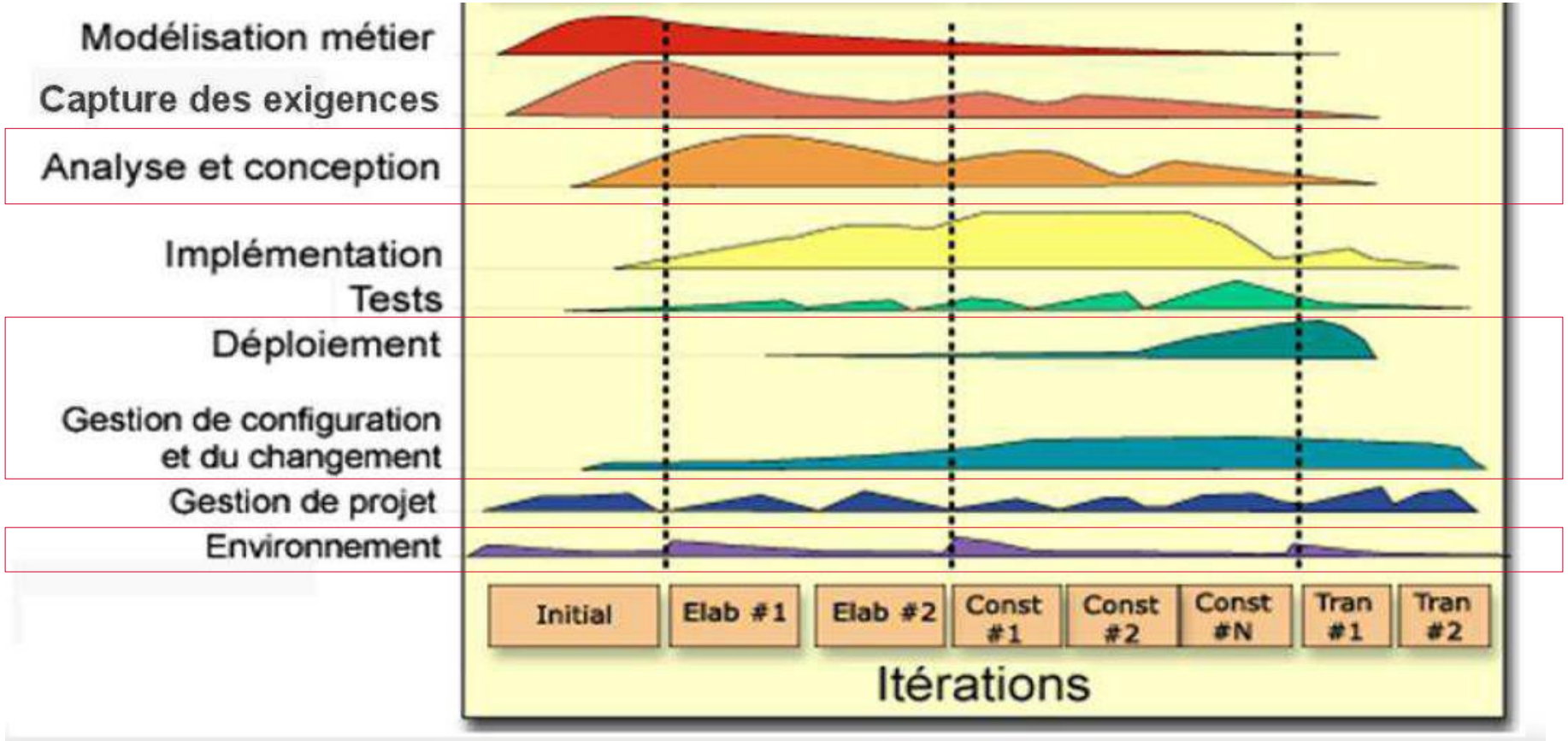


02

Architecture technique de la solution

De quoi est composée l'architecture technique d'un projet ?

klaxoon



Architecture technique

2 points de vue

2 points de vue architecturaux essentiels

Architecture applicative

On s'intéresse aux **applications**

- Leur organisation interne
- Leurs interactions

cf cours n°3

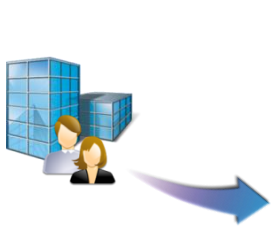
Architecture technique

On s'intéresse aux **composantes techniques accueillant les applications**

- Leurs interactions

Architecture technique
Objectifs

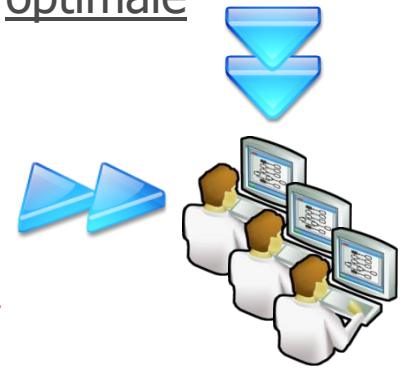
- **Choisir et Mettre en place les « briques » d'infrastructure technique**
  - Adaptées au contexte du projet
  - Répond aux besoins exprimés
  - Opérationnelles




Fait par l'Architecte et/ou le Référent technique

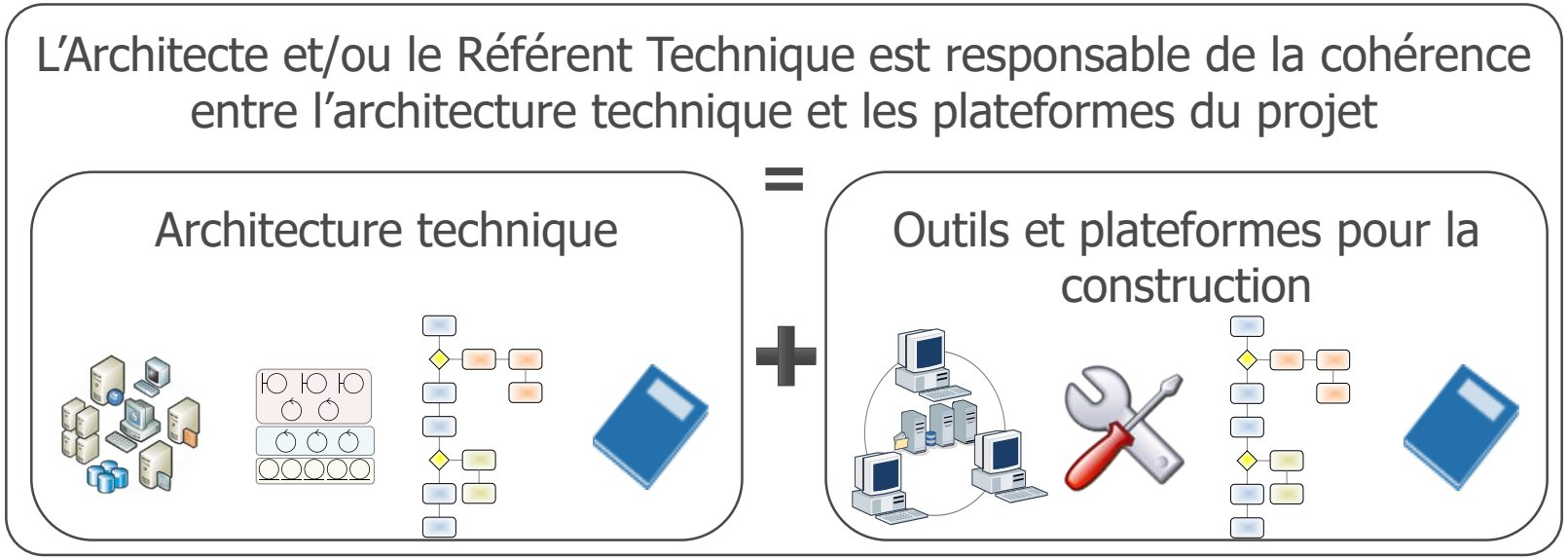
- **Expliquer à l'équipe les moyens de s'en servir**
  - Pour produire de manière optimale

Guide developpeur




Utilisé par les Developpeurs

Architecture technique
Eléments clés

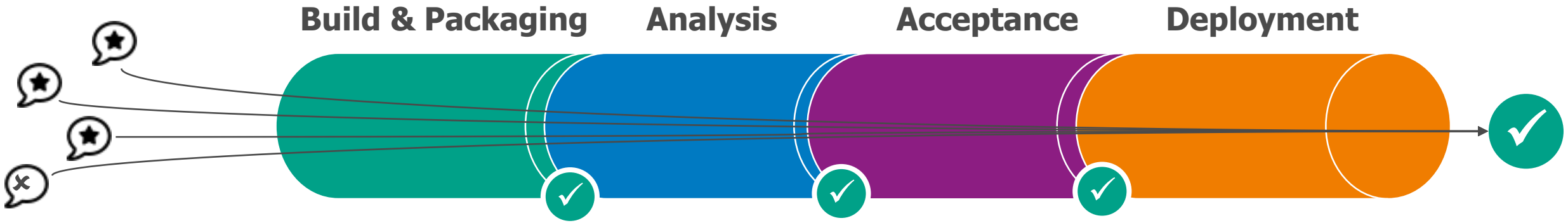


- **Le socle architectural doit être :**
  - Clairement défini
  - Issu des exigences du client (techniques et autres)
  - Opérationnel
  - Expliqué à tous
  - Partagé par tous

Quels outils connaissez-vous (développement, test, analyse de code, gestion de conf) ?

klaxoon

Devops et Pipeline

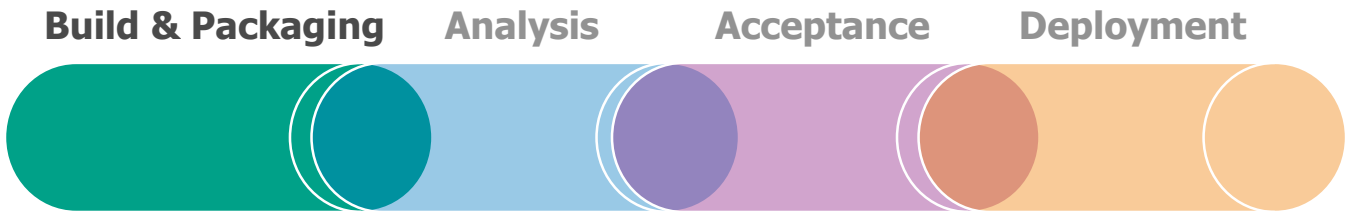


Chaque commit est un candidat pour passer par toutes les étapes du pipeline. Le pipeline peut s'arrêter à n'importe quelle étape si un test échoue: cela permet une correction immédiate, évitant une propagation de mauvaise qualité tout au long du cycle de vie de la livraison.

Elle doit être mise en place et partagée avant de démarrer l'implémentation.

Devops et Pipeline

1 - Build & Packaging



Build automatiquement l'application et exécute les tests unitaires

Compilation	Tests unitaires automatisés	Revue de code	Packaging
maven	JUnit	GitLab	GitLab
Gradle	nunit		docker
nuget	PHPUnit		Nexus
npm			
php			

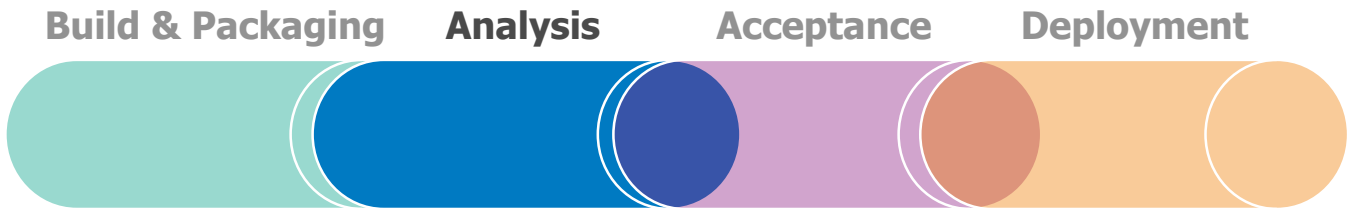
Minimum à mettre en place sur un projet

État d'esprit DevOps: Toutes les activités répétitives doivent être automatisées.

Obligatoire lorsque vous voulez être en mesure de livrer souvent et rapidement.

Devops et Pipeline

2 - Analysis



Assurer la qualité et la sécurité du code

Analyse de code statique	Test statique de sécurité des applications	Test statique de sécurité des conteneurs	Gestion de la dette technique
sonarqube	sonarqube	clair	themis
	DEPENDENCY-CHECK		
	LicenceFinder		
	CHECKMARX *		

Allier Time to Market (délai de mise sur le marché), sécurité et qualité

Automatiser l'analyse, c'est bien, mais vérifier les résultats est essentiel !

** External component*

Devops et Pipeline

3 - Acceptance

Effectuer des tests automatisés

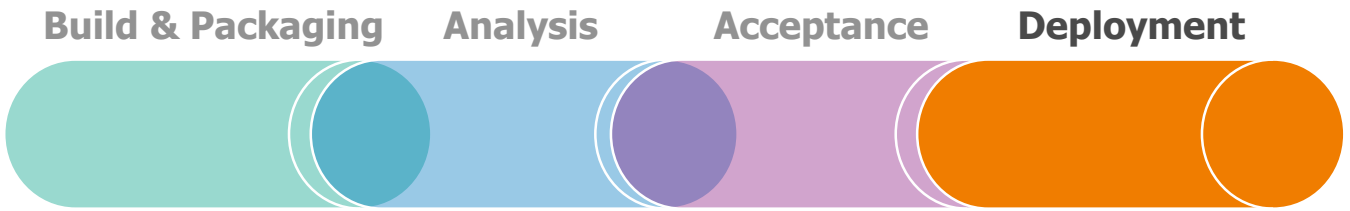
- Tests de non régression
- Tests de performance
- Tests de sécurité dynamique des applications
- Traçabilité

Attention, l'automatisation des tests « dynamiques » à un coût non négligeable sur un projet !
Automatiser d'abord les tests de non régression que vous réalisez à chaque fois, les plus critiques, ceux qui ne sont pas modifiés à chaque fois.

Leur mise en place nécessite des compétences fonctionnelles et techniques.

Devops et Pipeline

4 - Deployment



Automatiser le build de l'infrastructure, installer les configurations et déployer les applications

Gestion d'infrastructure

Terraform

Gestion de configuration

ANSIBLE

Déploiement automatisé

docker

Ces étapes seront nécessaires plusieurs fois car sur les projets nous avons plusieurs environnements.

Automatiser le déploiement permet d'être sûr de pouvoir déployer en toute sérénité chaque fois que l'on en a besoin, y compris les correctifs

D’après-vous combien faut-il d’environnements sur un projet ?

klaxoon

Combien d’environnements ?

- Pas de chiffre exact

- Exemples d’environnements type :
  - Développements
  - Intégration
  - Qualification
  - Recette
  - Pré-production
  - Production

Les mots clés du cours