Cours Atelier de génie logiciel

Préparé par Dr.Hiba Chaher

Chapitre 1 : **Le Génie Logiciel**

Introduction au génie logiciel (GL)

- Le terme de « Génie logiciel » a été introduit à la fin des années soixante lors d'une conférence tenue pour discuter de ce que l'on appelait alors (la crise du logiciel « software crisis ») …
- Le développement de logiciel était en crise.
- Les coûts du matériel chutaient alors que ceux du logiciel grimpaient en flèche.
- Il fallait de nouvelles techniques et de nouvelles méthodes pour contrôler la complexité inhérente aux grands systèmes logiciels.

Introduction au génie logiciel (GL)

- La crise du logiciel était perçue à travers ces symptômes :
- La qualité du logiciel livré était souvent déficiente : Le produit ne satisfaisait pas les besoins de l'utilisateur, il consommait plus de ressources que prévu et il était à l'origine de pannes.
- Les performances étaient très souvent médiocres (temps de réponse trop lents)

Introduction au GL

- Le non-respect des délais prévus pour le développement de logiciels ne satisfaisant pas leurs cahiers des charges.
- Les coûts de développement d'un logiciel étaient presque impossible à prévoir et étaient généralement prohibitifs (excessifs)

Introduction au GL

- L'invisibilité du logiciel, ce qui veut dire qu'on s'apercevait souvent que le logiciel développé ne correspondait pas à la demande (on ne pouvait l'observer qu'en l'utilisant « trop tard »).
- La maintenance du logiciel était difficile, coûteuse et souvent à l'origine de nouvelles erreurs

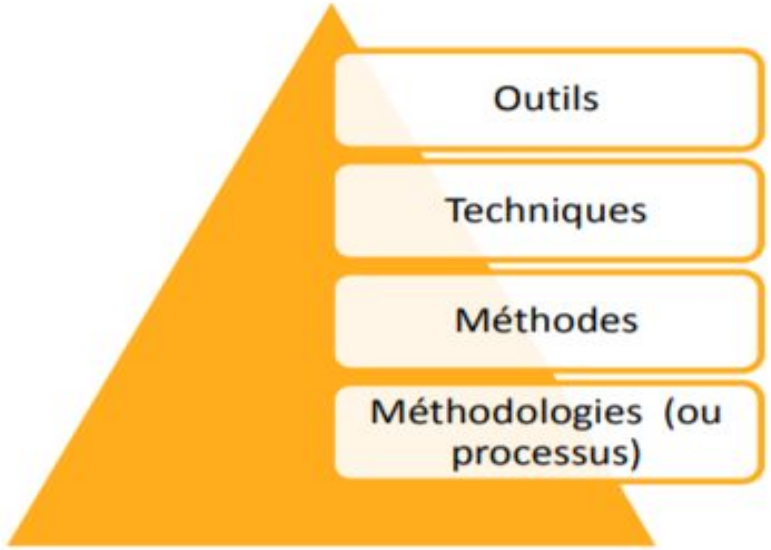
Introduction au GL

- Mais en pratique, il est indispensable d'adapter les logiciels car leurs environnements d'utilisation changent et les besoins des utilisateurs évoluent. Il est rare qu'on puisse réutiliser un logiciel existant ou un de ses composants pour confectionner un nouveau système, même si celui-ci comporte des fonctions similaires. Tous ces problèmes ont alors mené à l'émergence d'une discipline appelée « le génie logiciel »

- Le génie logiciel (software engineering) représente l'application de principes d'ingénierie au domaine de la création de logiciels. Il consiste à identifier et à utiliser des méthodes, des pratiques et des outils permettant de maximiser les chances de réussite d'un projet logiciel

Qu'est-ce que le génie logiciel ?

- **Définition 1**: « Le génie logiciel est une discipline d'ingénierie qui s'occupe de tous les aspects de la production de logiciels ». Le génie logiciel est intéressé par les théories, les méthodes et les outils de développement de logiciels professionnels.
- **Définition 2**: selon l'arrêté du 30 décembre 1983: « ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et de son suivi »
- **Définition 3**: « Le génie logiciel est un domaine des sciences de l'ingénieur dont l'objet d'étude est la conception, la fabrication, et la maintenance des systèmes informatiques complexes».



UML, plateformes, …

Interview, analyse grammaticale, brainstorming, …

OO, Merise, …

UP, RUP, XP, Scrum, …

Définitions des concepts clés

Qu'est-ce qu'un système ?

- **Définition 1**: ensemble d'éléments en interaction dynamique, dont les éléments sont organisés et coordonnés en vue d'atteindre un objectif, qui évolue dans un environnement.
- **Definition 2**: Un système est un ensemble d'éléments interagissant entre eux suivant un certains nombres de principes et de règles dans le but de réaliser un objectif.
- L'environnement est la partie du monde extérieure au système. Un système est souvent hiérarchisé à l'aide de sous-systèmes.
- Un système complexe se caractérise par :
  - sa dimension, qui nécessite la collaboration de plusieurs personnes;
  - son évolutivité.
- **Exemples** : Une fourmilière, l'économie mondiale, le noyau Linux, . . .

  ☞ **De plus en plus, les systèmes sont pilotées par des logiciels**

Qu'est-ce qu'un logiciel ?

- **Définition 1**: Les programmes informatiques et la documentation associée. Les produits logiciels peuvent être développés pour un client particulier ou peuvent être développés pour un marché général.

- **Définition 2**: « Un logiciel est un ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information».
  - Parmi ces entités, on trouve par exemple :
    - des programmes (en format *code source* ou exécutables);
    - des documentations d'utilisation;
    - des informations de configuration.

- **Définition 3**: selon l'arrêté du 22 décembre 1981: ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitements de l'information.

$$\text{Logiciel} = \text{programme} + \text{utilisation}$$

FAQ sur le génie logiciel

Question	Réponse
Qu'est ce qu'un logiciel?	Les programmes informatiques et la documentation associée. Les produits logiciels peuvent être développés pour un client particulier ou peuvent être développés pour un marché général.
Quelles sont les caractéristiques d'un bon logiciel?	Un bon logiciel doit offrir la fonctionnalité et les performances requises pour l'utilisateur et doit être maintenable, fiable et utilisable.
Qu'est-ce que le génie logiciel?	Le génie logiciel est une discipline d'ingénierie qui s'occupe de tous les aspects de la production de logiciels.
Quelles sont les activités fondamentales de génie logiciel?	Spécification des logiciels, développement des logiciels, validation des logiciel et l'évolution des logiciels.
Quelle est la différence entre le génie logiciel et de l'informatique?	L'informatique focalise sur la théorie et les principes fondamentaux; génie logiciel est concerné par les pratiques de développement et de la réalisation des logiciels utiles.
Quelle est la différence entre le génie logiciel et de l'ingénierie de système?	Ingénierie des systèmes s'intéresse à tous les aspects du développement des systèmes à base d'ordinateur, y compris le matériel, les logiciels et l'ingénierie des processus. Génie logiciel fait partie de ce processus plus général.

Définitions des concepts clés

- Un modèle : est une représentation schématique de la réalité.
- Une base de Données: ensemble des données (de l'organisation) structurées et liées entre elles : stocké sur support à accès direct (disque magnétique) ; géré par un SGBD (Système de Gestion de Bases de Données), et accessible par un ensemble d'applications.

Définitions des concepts clés

- Une analyse : c'est un processus d'examen de l'existant.
- Une Conception : est un processus de définition de la future application informatique.
- Un système d'Information : ensemble des moyens (humains et matériels) et des méthodes se rapportant au traitement de l'information d'une organisation.

Crise du logiciel

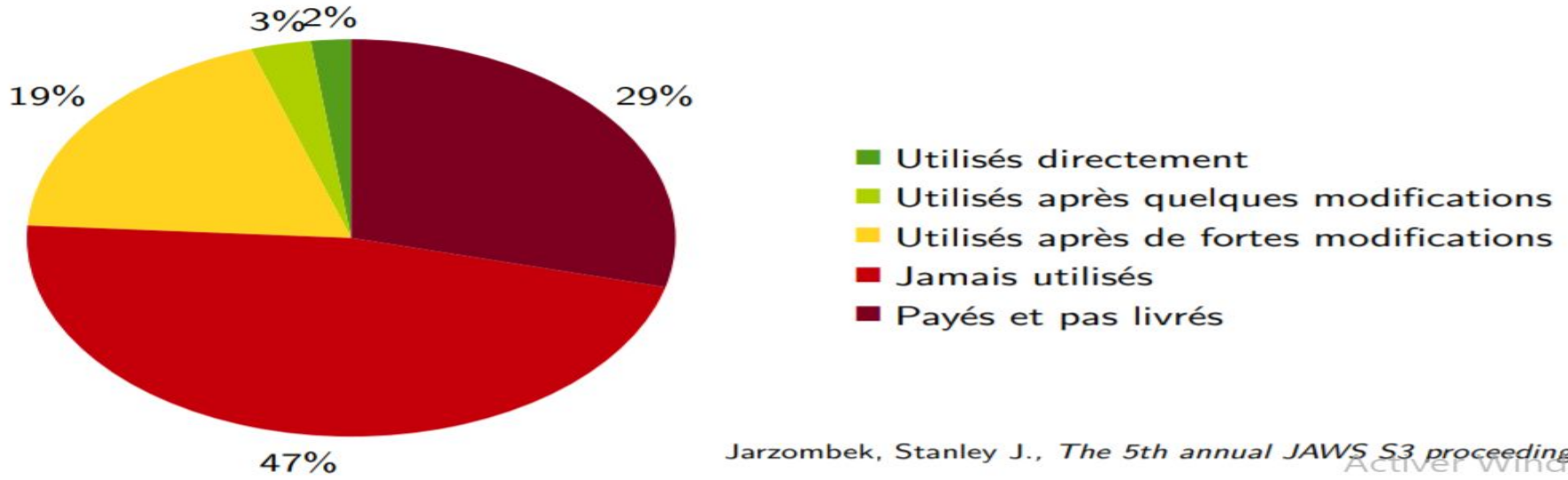
Constat du développement logiciel fin années 60 :

- délais de livraison non respectés
- budgets non respectés
- ne répond pas aux besoins de l'utilisateur ou du client
- difficile à utiliser, maintenir, et faire évoluer

La crise logicielle (Exemples)

Étude du DoD 1995

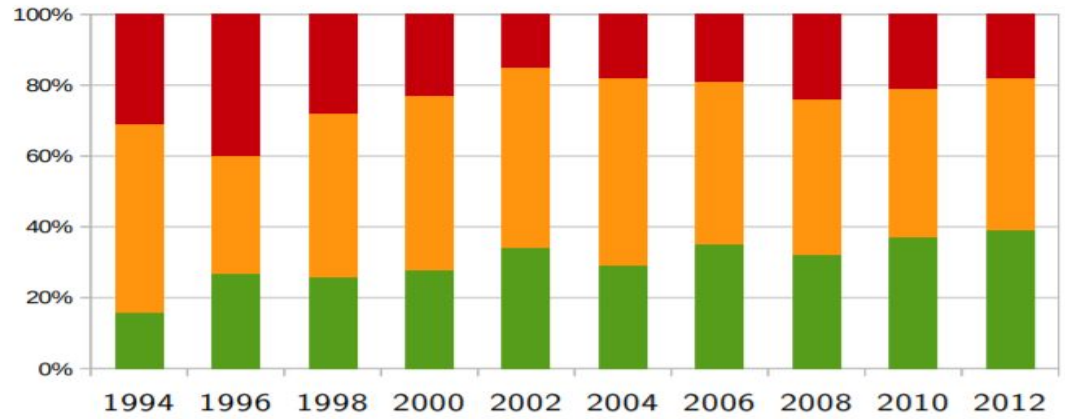
Étude du *Department of Defense* des États-Unis sur les logiciels produits dans le cadre de 9 gros projets militaires



Jarzombek, Stanley J., *The 5th annual JAWS S3 proceedings*, 1999

Étude du Standish group

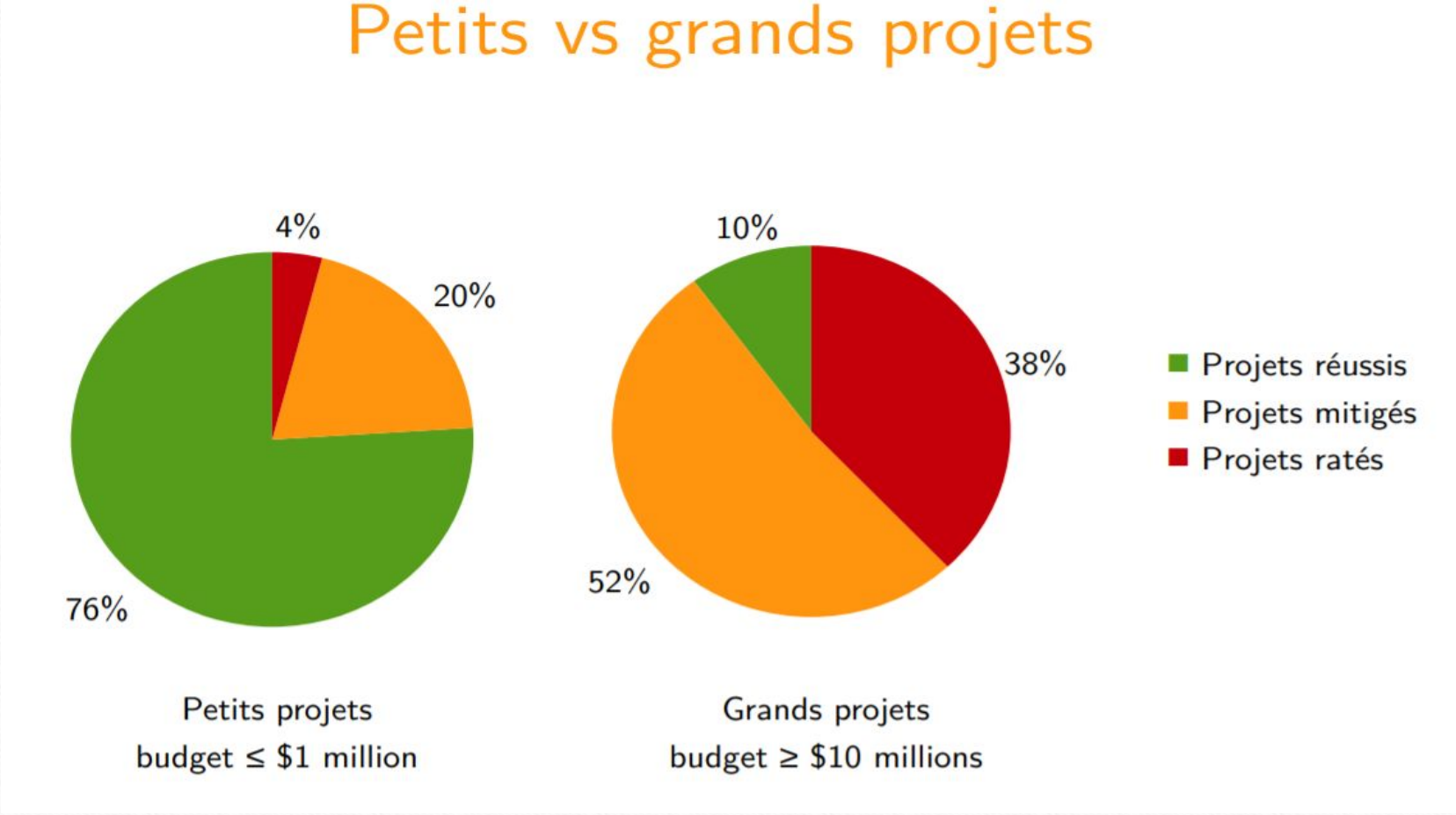
Enquête sur des milliers de projets, de toutes tailles et de tous secteurs



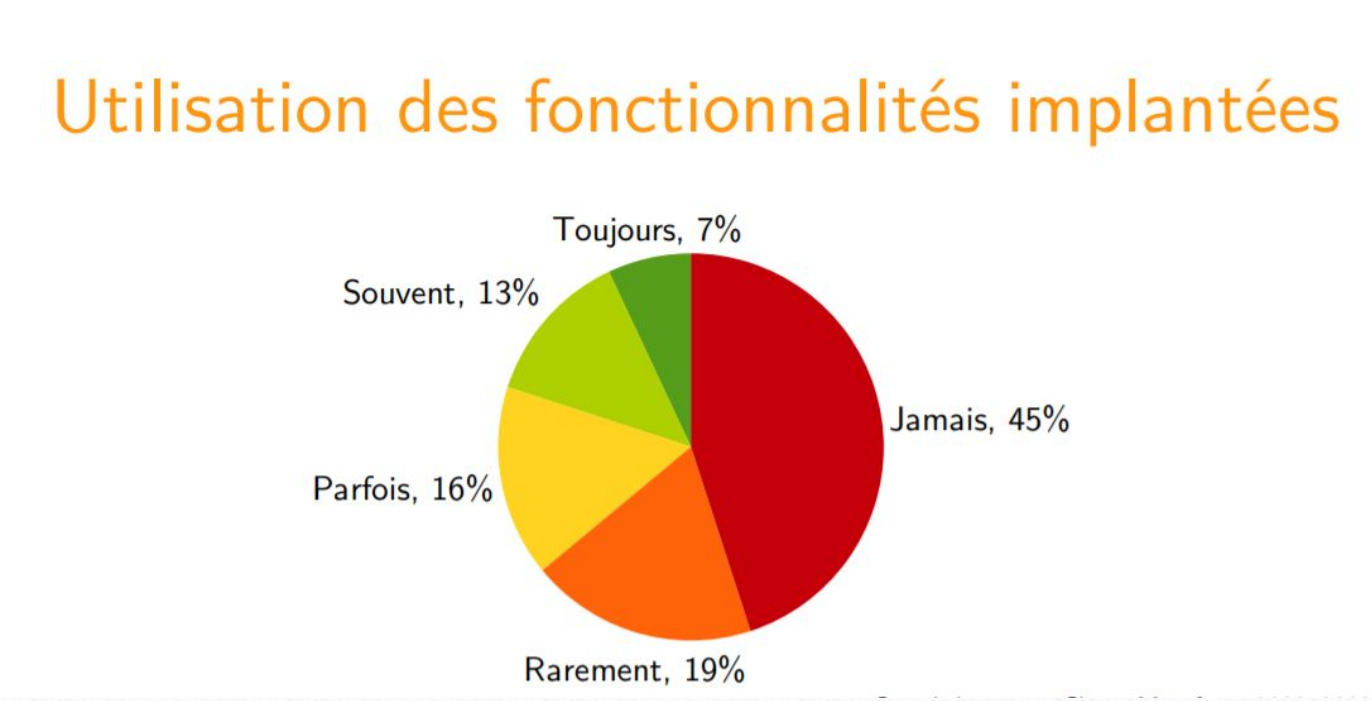
Standish group, *Chaos Manifesto 2013 - Think Big, Act Small*, 2013

- Projets réussis : achevés dans les délais et pour le budget impartis, avec toutes les fonctionnalités demandées
- Projets mitigés : achevés et opérationnels, mais livrés hors délais, hors budget ou sans toutes les fonctionnalités demandées
- Projets ratés : abandonnés avant la fin ou livrés mais jamais utilisés

La crise logicielle (Exemples)



La crise logicielle (Exemples)



La crise logicielle

Raisons de la faible qualité des logiciels

Tâche complexe :

- Taille et complexité des logiciels
- Taille des équipes de conception/développement

Manque de méthodes et de rigueur :

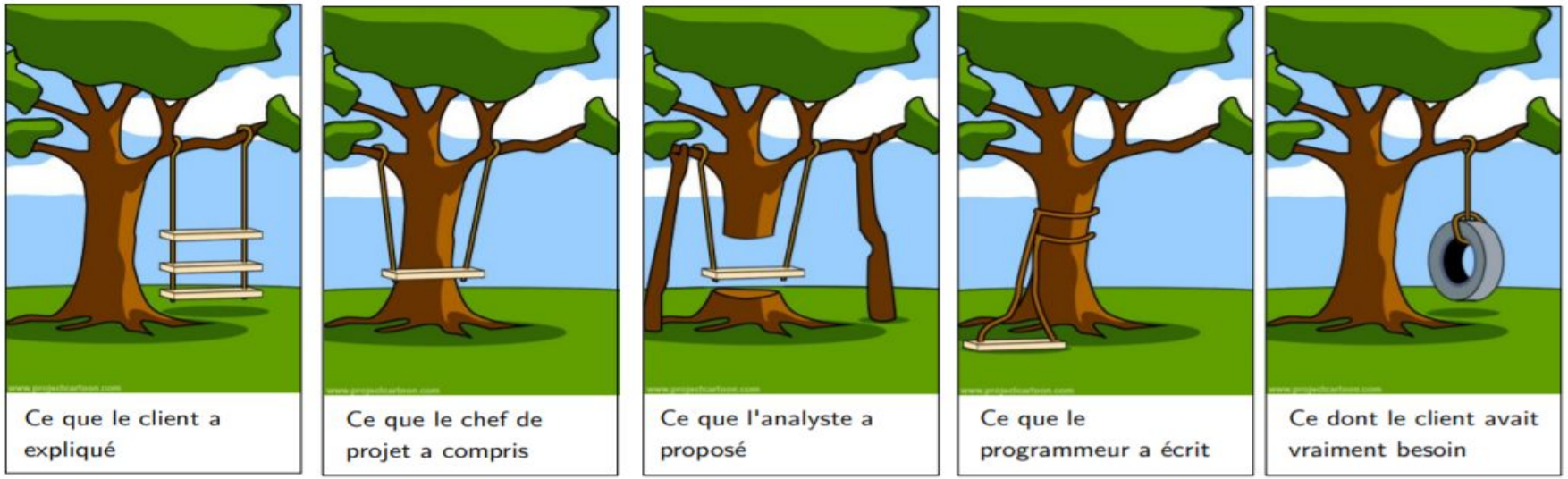
- Manque de méthodes de conception
- Négligence et manque de méthodes et d'outils des phases de validation/vérification

Mauvaise compréhension des besoins :

- Négligence de la phase d'analyse des besoins du client
- Manque d'implication du client dans le processus

La crise logicielle

Raisons de la faible qualité des logiciels



La crise logicielle

Raisons de la faible qualité des logiciels

Difficultés spécifiques du logiciel :

- Produit invisible et immatériel
- Difficile de mesurer la qualité
- Conséquences critiques causées par modifications infimes
- Mises à jour et maintenance dues à l'évolution rapide de la technologie
- Difficile de raisonner sur des programmes
- Défaillances logicielles principalement humaines

La crise logicielle

Qualité du logiciel

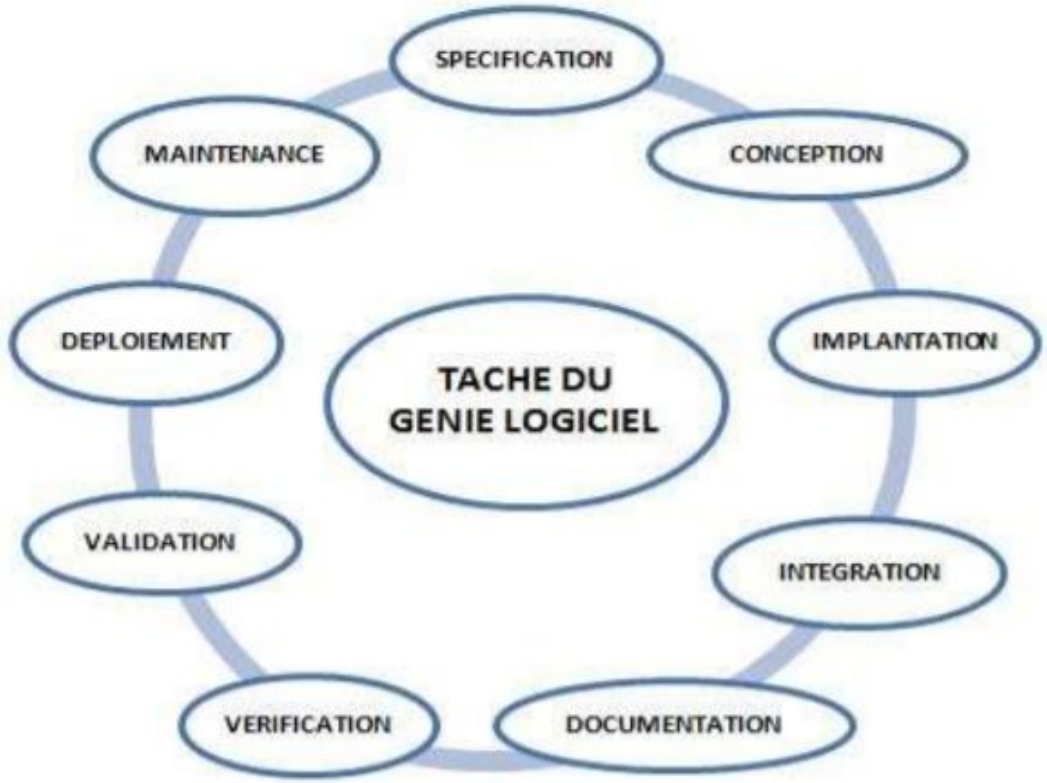
Critères de qualité

- Validité : réponse aux besoins des utilisateurs
- Facilité d'utilisation : prise en main et robustesse
- Performance : temps de réponse, débit, fluidité...
- Fiabilité : tolérance aux pannes
- Sécurité : intégrité des données et protection des accès
- Maintenabilité : facilité à corriger ou transformer le logiciel
- Portabilité : changement d'environnement matériel ou logiciel

- *Le logiciel* est plus qu'un code de programme. En faite, un programme est un code exécutable, qui sert à des fins de calcul, par contre, le logiciel, est considéré comme une collection de code de programmation exécutable, des bibliothèques associées et de documentations. Ainsi, lorsque le logiciel, est conçu pour une exigence spécifique, est appelé un « *produit logiciel* ».

- D'autre part, *l'ingénierie* (*génie*) consiste à développer des produits, en utilisant des principes et méthodes scientifiques bien définis.

Le génie logiciel englobe les tâches suivantes :



Pour chaque activité : Utilisation et production de documents

Activités du développement logiciel

Analyse des besoins : Comprendre les besoins du client

- Objectifs généraux, environnement du futur système, ressources disponibles, contraintes de performance...
- Fournie par le client (expert du domaine d'application, futur utilisateur...)

Spécification :

- Établir une description claire de ce que doit faire le logiciel (fonctionnalités détaillées, exigences de qualité, interface...)
- Clarifier le cahier des charges (ambiguïtés, contradictions) en listant les exigences fonctionnelles et non fonctionnelles

Conception : Élaborer une solution concrète réalisant la spécification

- Description architecturale en composants (avec interface et fonctionnalités)
- Réalisation des fonctionnalités par les composants (algorithmes, organisation des données)
- Réalisation des exigences non fonctionnelles (performance, sécurité...)

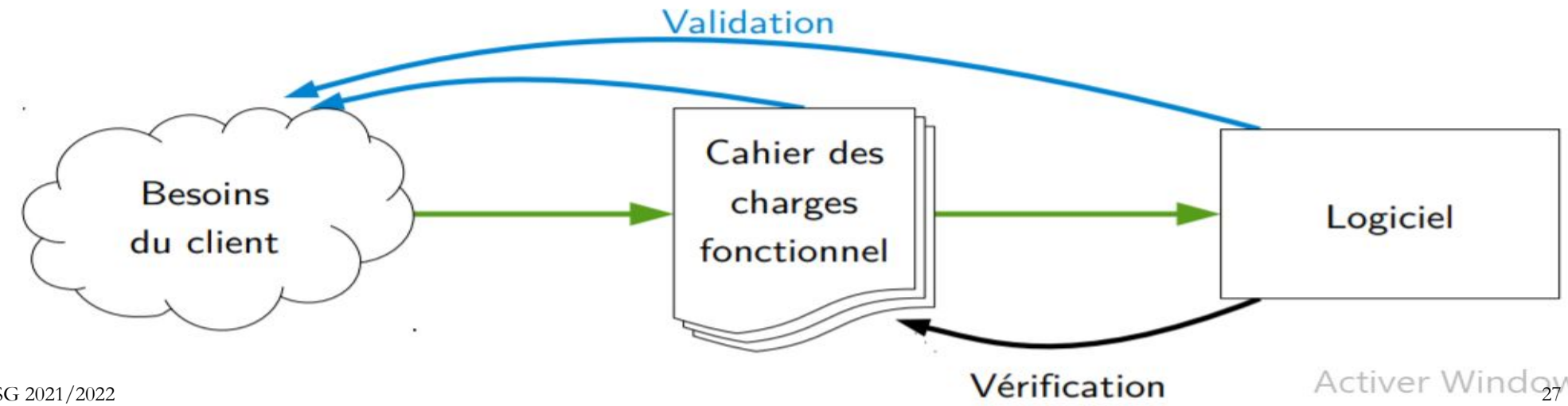
Programmation : Implantation de la solution conçue

- Choix de l'environnement de développement, du/des langage(s) de programmation, de normes de développement...

Validation et vérification

Objectifs :

- Validation : assurer que les besoins du client sont satisfaits (au niveau de la spécification, du produit fini...)
- Vérification : assurer que le logiciel satisfait sa spécification

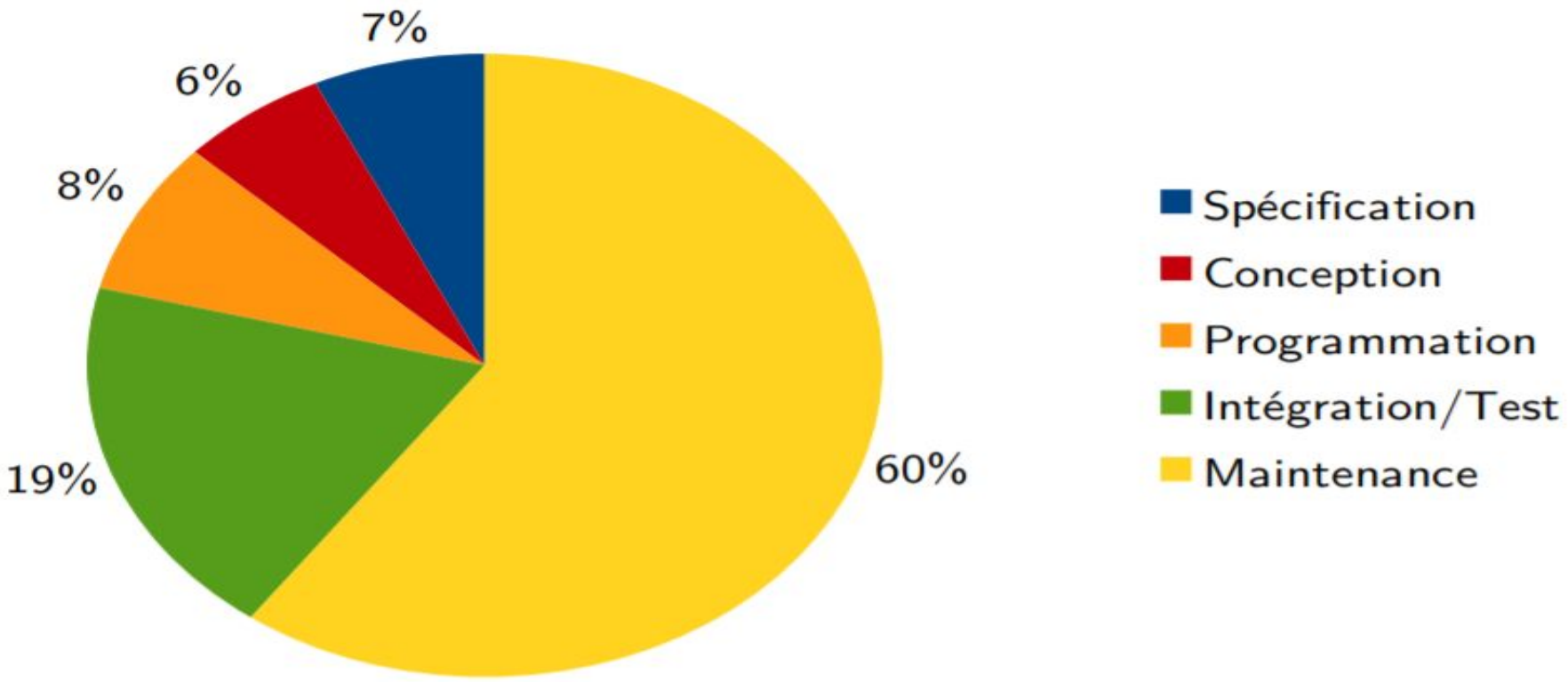


Maintenance

Types de maintenance :

- Correction : identifier et corriger des erreurs trouvées après la livraison
- Adaptation : adapter le logiciel aux changements dans l'environnement (format des données, environnement d'exécution...)
- Perfection : améliorer la performance, ajouter des fonctionnalités, améliorer la maintenabilité du logiciel

Répartition de l'effort



MODELISATION DE PROCESSUS DE DEVELOPPEMENT LOGICIEL

- La modélisation de développement logiciel aide les développeurs à sélectionner la stratégie pour développer le produit logiciel. Ainsi, chaque modélisation de développement possède ses outils propres, ses méthodes et ses procédures qui permettent d'exprimer clairement et définissent le cycle de vie de développement du logiciel.

MODELISATION DE PROCESSUS DE DEVELOPPEMENT LOGICIEL

- Les modèles du cycle de vie du logiciel sont des « plans de travail » qui permettent de planifier le développement. Plus le logiciel à développer est complexe (taille, algorithmes) et critique, plus il est important de bien contrôler le processus de développement et plus les documents qui accompagnent le logiciel doivent être précis et détaillés.

- Il s'avère de nos jours, que nous ne pouvons plus avoir une démarche unique dans le développement de projets informatiques, mais qu'il faut construire le découpage temporel en fonction des caractéristiques de l'entreprise et du projet.

MODELISATION DE PROCESSUS DE DEVELOPPEMENT LOGICIEL

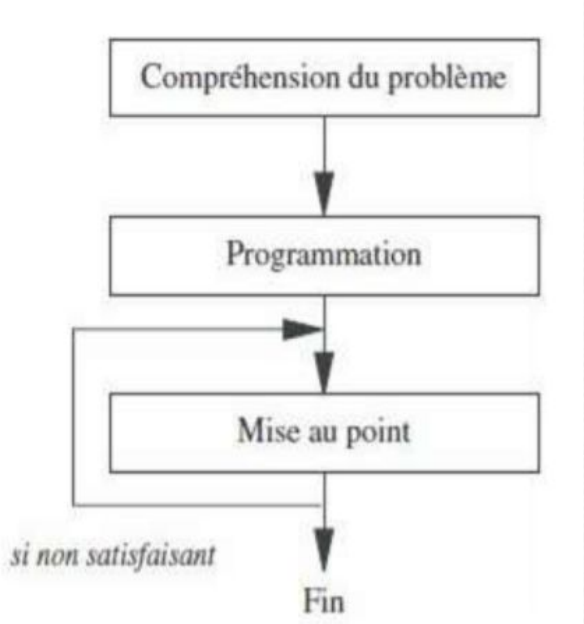
- On s'appuie pour cela sur des découpages temporels génériques, appelés modèles de développement (process models) ou modèles de cycle de vie d'un projet informatique. Les principaux modèles sont :

- le modèle du code-and-fix
- le modèle de la transformation automatique
- le modèle de la cascade
- le modèle en V
- le modèle par incrément
- le modèle par prototypage
- le modèle de la spirale.

LE MODELE DU CODE-AND-FIX

- C’est un modèle qui repose sur la possibilité d’une détermination facile des besoins : une première phase de Compréhension du problème est suivie d’une phase de Programmation ; puis une phase de Mise au point, parfois en collaboration avec l'utilisateur du futur système, est répétée jusqu’à l’atteinte du résultat visé.

LE MODELE DU CODE-AND-FIX

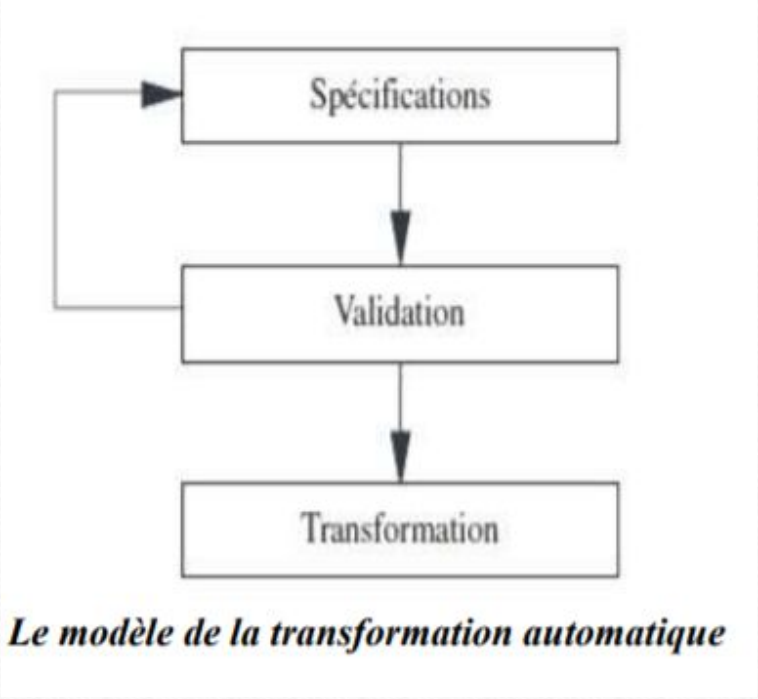


Le modèle du code-and-fix.

LE MODELE DE TRANSFORMATION AUTOMATIQUE

- C'est un modèle basé sur la possibilité de transformer automatiquement des spécifications en programmes. L'essentiel de l'effort va donc porter sur une description exhaustive des spécifications qui devront être complètement validées. Une succession de cycles Phase de Spécifications – Phase de Validation s'achève par la phase de Transformation, où s'effectue la génération du code.

LE MODELE DE TRANSFORMATION AUTOMATIQUE



Le modèle de la transformation automatique

LE MODELE EN CASCADE (WATERFALL MODEL)

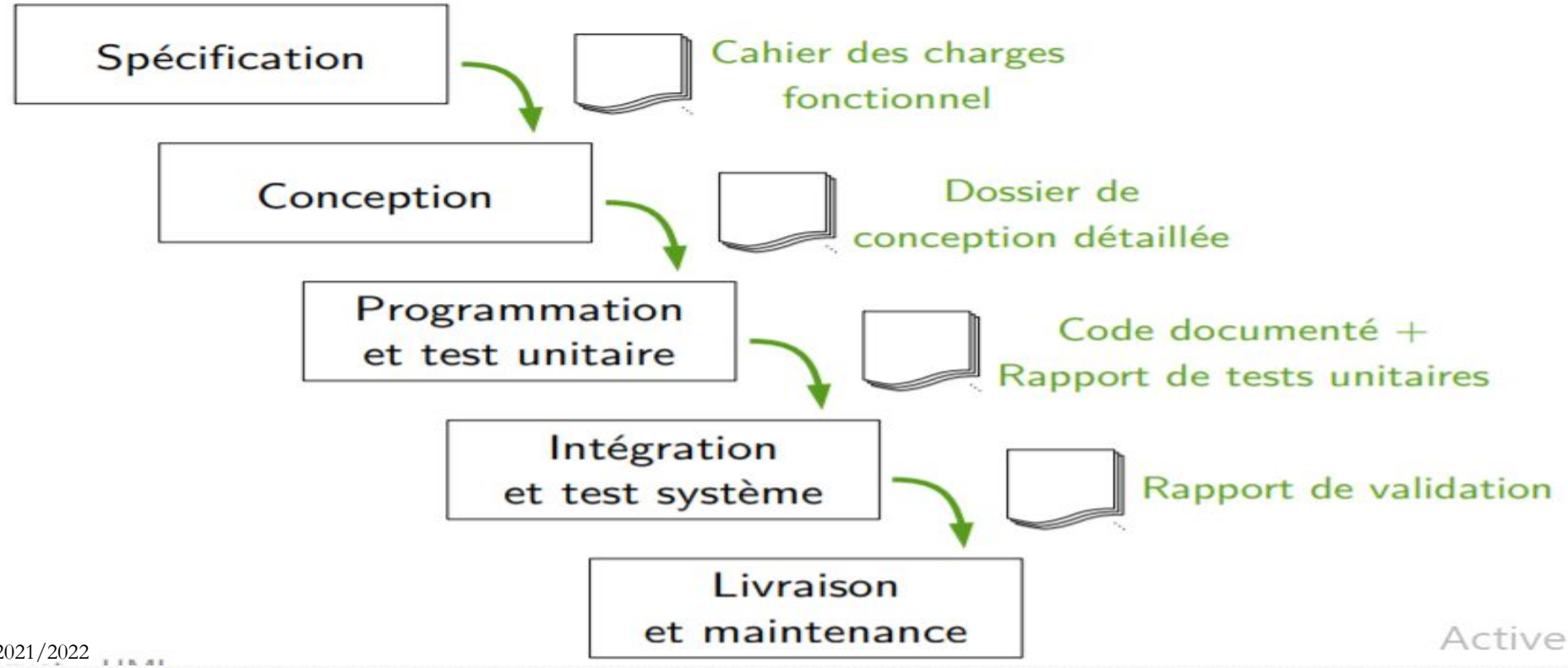
- C'est un modèle qui a comme objectif majeur de jalonner (présenter sobrement) rigoureusement le processus de développement et de définir de façon précise les rôles respectifs du fournisseur qui produit un livrable et du client qui accepte ou refuse le résultat.

LE MODELE EN CASCADE (WATERFALL MODEL)

- Le découpage temporel se présente comme une succession de phases affinant celles du découpage classique : Étude de faisabilité, Définition des besoins, Conception générale, Conception détaillée, Programmation, Intégration, Mise en œuvre.
- Chaque phase donne lieu à une validation officielle. Si le résultat du contrôle n'est pas satisfaisant, on modifie le livrable. En revanche, il n'y a pas de retour possible sur les options validées à l'issue de phases antérieures

Processus en cascade

Chaque étape doit être terminée avant que ne commence la suivante
À chaque étape, production d'un document base de l'étape suivante



Processus en cascade

Caractéristiques :

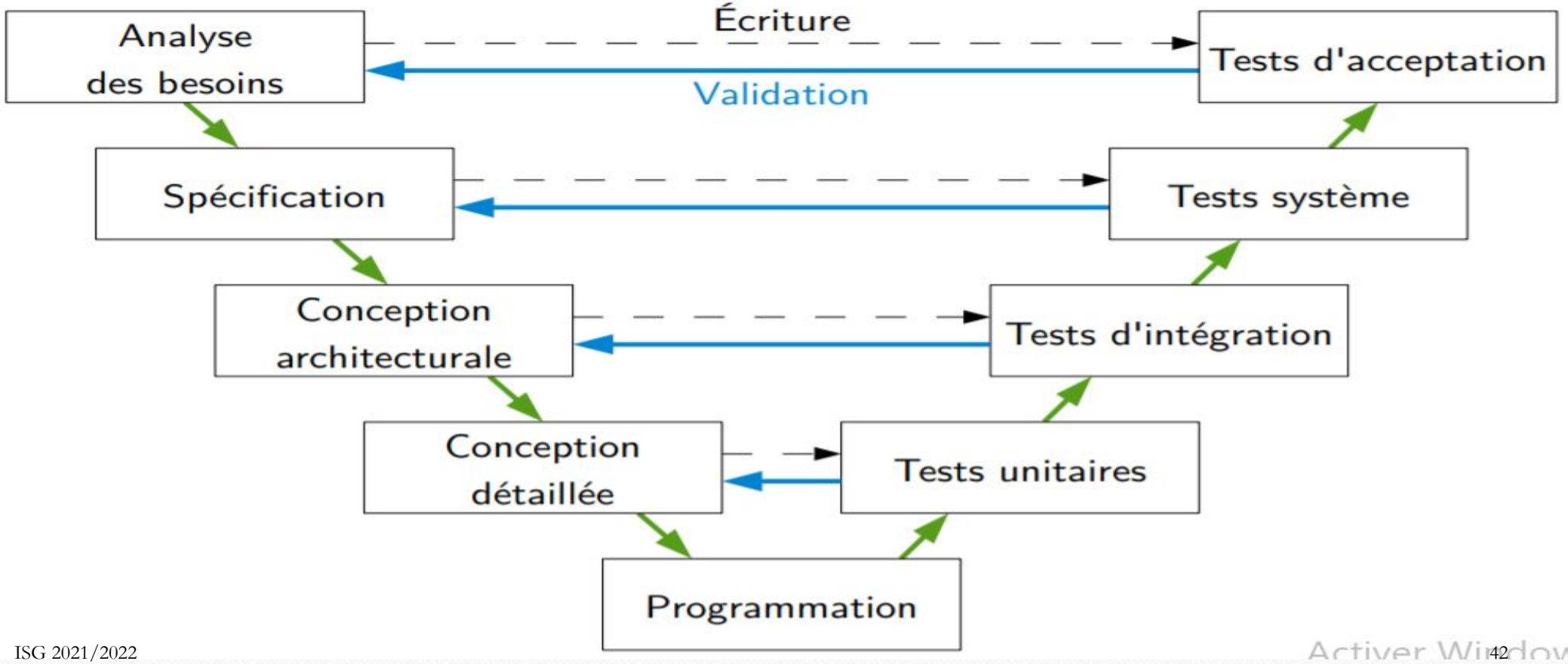
- Hérité des méthodes classiques d'ingénierie
- Découverte d'une erreur entraîne retour à la phase à l'origine de l'erreur et nouvelle cascade, avec de nouveaux documents...
- Coût de modification d'une erreur important, donc choix en amont cruciaux (typique d'une production industrielle)

Pas toujours adapté à une production logicielle, en particulier si besoins du client changeants ou difficiles à spécifier

LE MODELE EN V.

- Le modèle en V du cycle de vie du logiciel précise la conception des tests : (les tests système sont préparés à partir de la spécification; les tests d'intégration sont préparés à partir de la conception architecturale ; les tests unitaires sont préparés à partir de la conception détaillée des composants).
- Dérivé du modèle de la cascade, le modèle en V du cycle de développement montre non seulement l'enchaînement des phases successives, mais aussi les relations logiques entre phases plus éloignées.
- Ce modèle fait apparaître le fait que le début du processus de développement conditionne ses dernières étapes.
- Le modèle du cycle de vie en V est souvent adapté aux projets de taille et de complexité moyenne.

Processus en V



Niveaux de test

Test unitaire : test de chaque unité de programme (méthode, classe, composant), indépendamment du reste du système

Test d'intégration : test des interactions entre composants (interfaces et composants compatibles)

Test système : test du système complet par rapport à son cahier des charges

Test d'acceptation (recette) : fait par le client, validation par rapport aux besoins initiaux

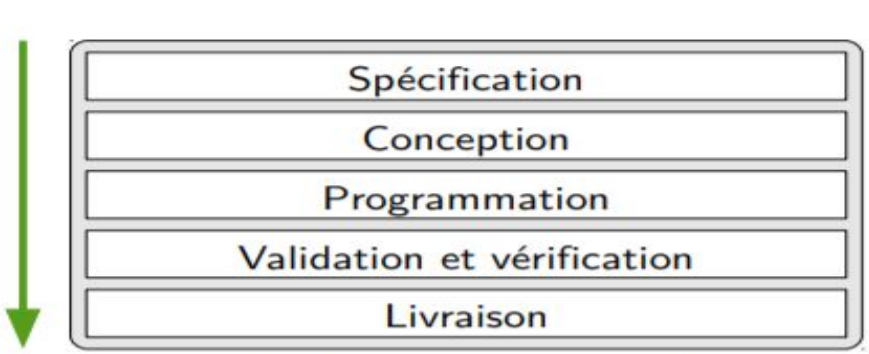
LE MODELE PAR INCREMENT

- Le modèle incrémental est un modèle itératif, qui consiste à sélectionner successivement plusieurs incréments. Un incrément du logiciel est un sous-ensemble du logiciel complet, qui consiste en un petit nombre de fonctionnalités.
- Dans un premier temps un logiciel noyau est développés, puis successivement, les incréments sont développés et intégrés. Chaque incrément est développé selon l'un des modèles précédents.
- Dans ce modèle, les intégrations sont progressives et il peut y avoir des livraisons et des mises en service après chaque intégration d'incrément.

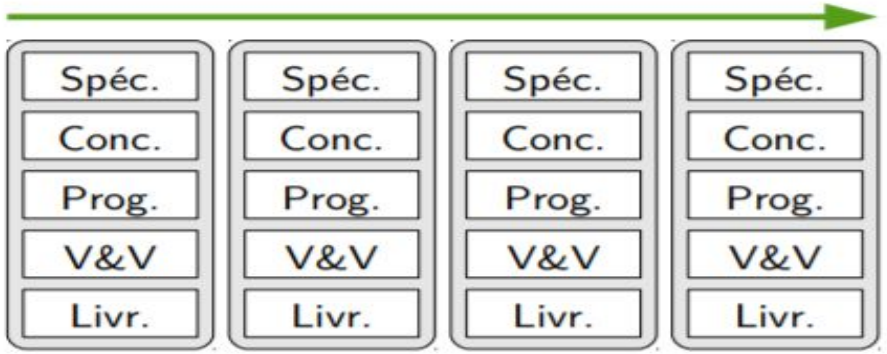
Développement incrémental

Principe :

- Hiérarchiser les besoins du client
- Concevoir et livrer au client un produit implantant un sous-ensemble de fonctionnalités par ordre de priorité



Développement en cascade



Développement incrémental

Avantage : Minimiser le risque d'inadéquation aux besoins

Difficulté : Intégration fonctionnalités secondaires non pensées en amont

LE MODELE PAR INCREMENT

- Le modèle incrémental présente comme avantages :
  - Chaque développement est moins complexe ;
  - Les intégrations sont progressives ;
  - Possibilité de livraisons et de mises en service après chaque incrément ;

LE MODELE PAR INCREMENT

- Meilleur lissage du temps et de l'effort de développement à cause de la possibilité de recouvrement des différentes phases.
- L'effort est constant dans le temps par opposition au pic pour spécifications détaillées pour les modèles en cascade ou en V

LE MODELE PAR INCREMENT

- Le modèle incrémental présente comme Risques :
  - la remise en cause du noyau de départ ;
  - la remise en cause des incréments précédents;
  - l'impossibilité d'intégrer un nouvel incrément

LE MODELE PAR PROTOTYPAGE

- Il est quelquefois difficile de formuler une esquisse des besoins, surtout lorsque l'on connaît peu le domaine. Dans ces cas-là, on ne peut pas espérer de manière réaliste définir les besoins de manière définitive avant le début du développement du logiciel.
- Un modèle de processus basé sur le prototypage se révèle alors plus approprié que le modèle classique de la cascade.
- Le prototypage permet de contourner la difficulté de la validation liée à l'imprécision des besoins et caractéristiques du système à développer.
- Cela veut dire que lorsqu'il est difficile d'établir une spécification détaillée, on a recours au prototypage qui est considéré, dans ce cas, comme un modèle de développement de logiciels.

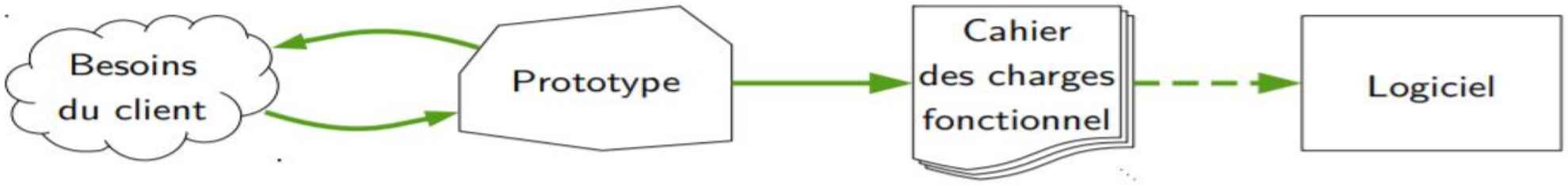
LE MODELE PAR PROTOTYPAGE

- Il s'agit d'écrire une première spécification et de réaliser un sous-ensemble du produit logiciel final. Ce sous ensemble est alors raffiné incrémentalement et évalué jusqu'à obtenir le produit final. On en distinguera deux types de prototypage :
- Le prototypage Jetable : ici, le squelette du logiciel n'est créé que dans un but et dans une phase particulière du développement.
- Le prototypage Evolutif : ici, on conserve tout, au long du cycle de développement. Il est amélioré et complété pour obtenir le logiciel final.

Développement par prototypage

Principe :

- Développement rapide d'un prototype avec le client pour valider ses besoins
- Écriture de la spécification à partir du prototype, puis processus de développement linéaire



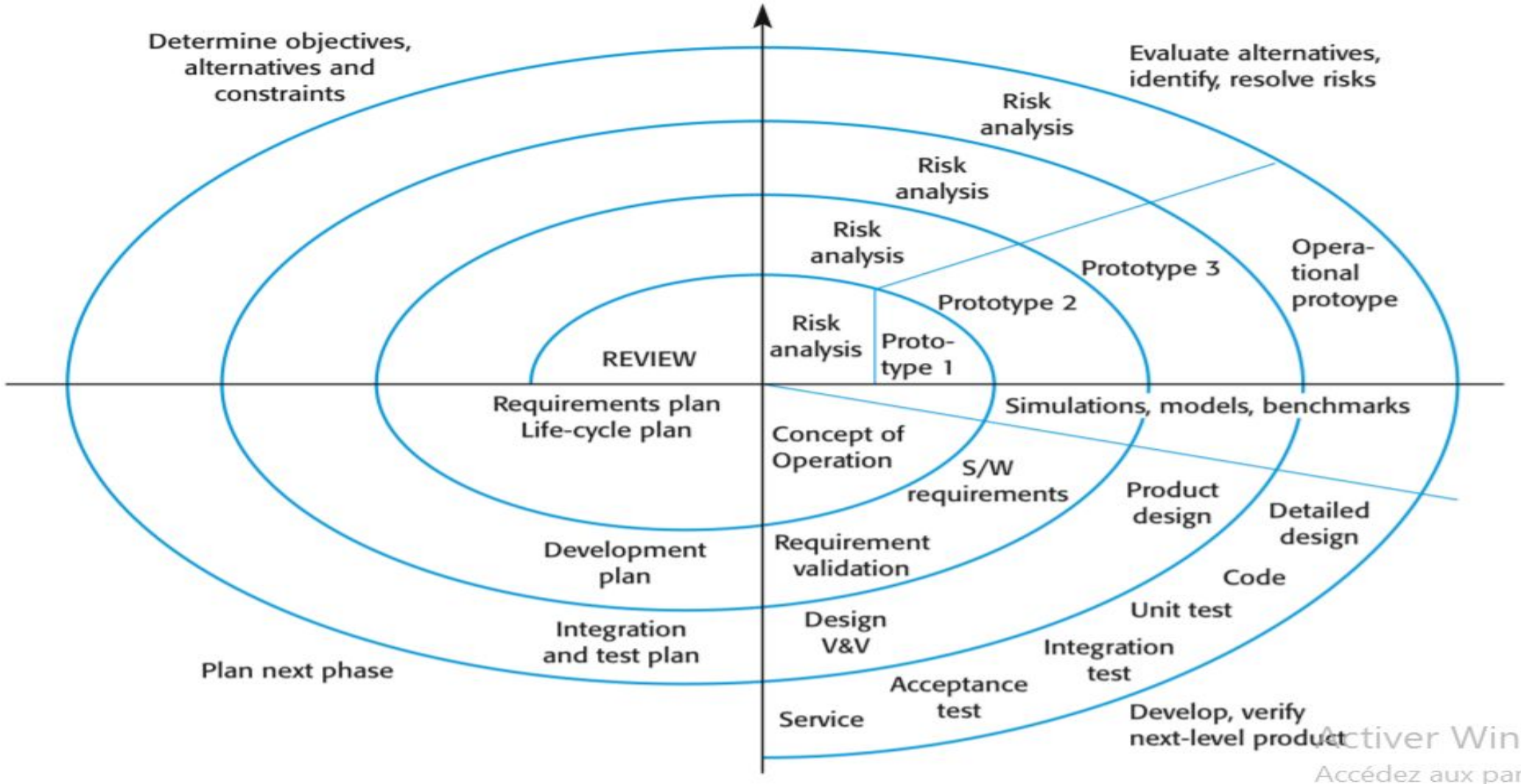
Avantage : Validation concrète des besoins, moins de risques d'erreur de spécification

LE CYCLE DE VIE EN SPIRALE

- Ce modèle repose sur le même principe que le modèle évolutif, mais il s'inscrit dans une relation contractuelle entre le client et le fournisseur.
- De ce fait les engagements et validations présentent un caractère formalisé. Chaque cycle donne lieu à une contractualisation préalable, s'appuyant sur les besoins exprimés lors du cycle précédent.
- Un cycle comporte six phases :
- Analyse du risque
- Développement d'un prototype
- Simulation et essais du prototype
- Détermination des besoins à partir des résultats des essais
- Validation des besoins par un comité de pilotage
- Planification du cycle suivant

LE CYCLE DE VIE EN SPIRALE

- Le processus de développement est représenté par une spirale plutôt qu'une séquence d'activités avec retour arrière éventuels
- Chaque boucle dans la spirale représente une étape du processus de développement
- Les risques sont explicitement adressés et résolus tout au long du processus



LE CYCLE DE VIE EN SPIRALE

- Définition des objectifs :
- Les objectifs spécifiques de l'étape sont identifiés
- Estimation et réduction des risques

Les risques sont évalués et des activités sont mises en place pour réduire les risques clés

- Développement et validation

Un modèle de développement est choisi pour le système

- Planification

Le projet est inspecté et l'étape suivante de la spirale est planifiée