Génie Logiciel
SMI
2024-2025

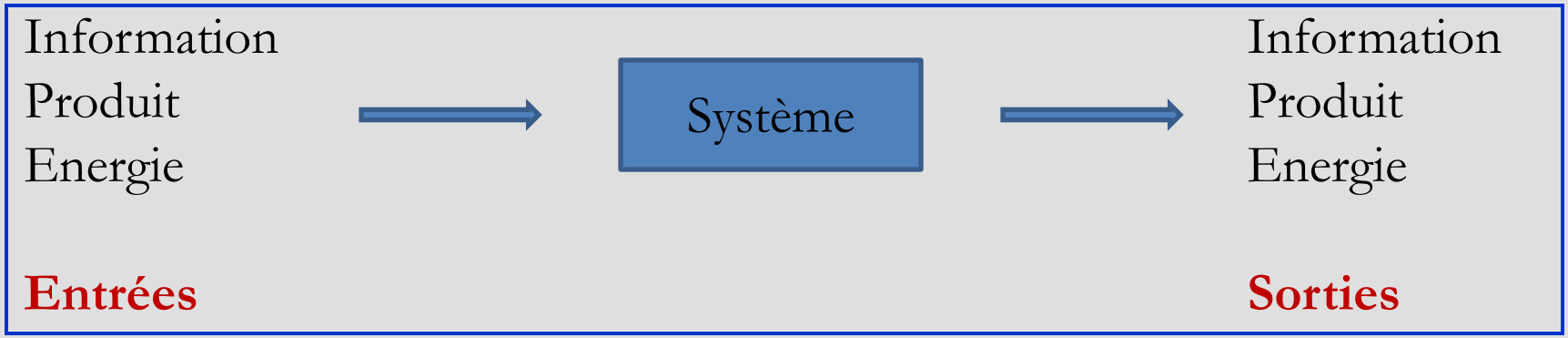
Chapitre I
Généralités

I Conception système

I.1 – Notion de système

Un système est un ensemble d'éléments en interaction dynamique organisés afin de réaliser but précis.

Un système peut être abstrait ou concret, naturel ou artificiel.



- Un système reçoit des données d'entrée d'autres systèmes ou de l'environnement.
- Les entrées subissent des modifications (transformations).
- Les transformations produisent des sorties qu'absorbent d'autres systèmes ou l'environnement.

Chapitre I
Généralités

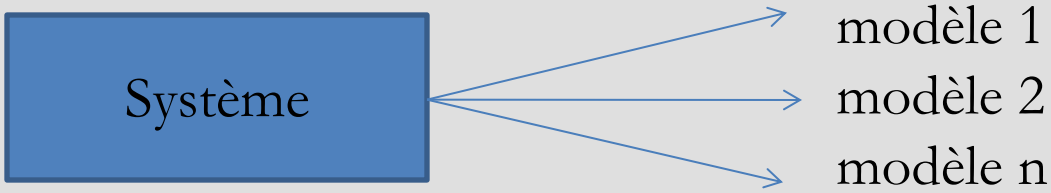
I.2 Caractéristiques d'un système

Un système est caractérisé par :

- Sa structure : les éléments qui le composent.
- Son évolution : les états successifs par lesquels il passe.
- Les fonctions : ce qu'il sait faire.

Un système peut être représenté par un modèle

A un même système, peuvent correspondre plusieurs modèles, selon l'objectif de l'étude, le système de représentation, ……etc.



Chapitre I
Généralités

Le concept de système nous aide à observer la réalité en la considérant comme étant formée d'ensembles dynamiques en relation les uns avec les autres.

Quand on applique le concept de système à la définition et à la réalisation des problèmes, on parle alors d'approche systémique.

Chapitre I
Généralités

II Conception modèle

II.1 Notion de modèle

Un modèle est une représentation abstraite et simplifiée (qui exclut certains détails), d'une entité (phénomène, processus, système, etc.) du monde réel en vue de le décrire, de l'expliquer ou de le prévoir (gérer son comportement).

Concrètement, un modèle permet de réduire la complexité d'un phénomène en éliminant les détails qui n'influencent pas son comportement de manière significative. Il reflète ce que le concepteur croit important pour la compréhension et la prédiction du phénomène modélisé. Les limites du phénomène modélisé dépendant des objectifs du modèle.

Chapitre I
Généralités

II.2 Exemples de modèle

- **Modèle météorologique :** à partir de données d'observation (satellite, …), il permet de prévoir les conditions climatiques pour les jours à venir.

- **Modèle économique:** permet de simuler l'évolution de cours boursiers en fonction d'hypothèses macro-économiques (évolution du chômage, taux de croissance, …).

- **Modèle démographique:** définit la composition d'un panel d'une population et son comportement, dans le but de fiabiliser des études statistiques, d'augmenter l'impact de démarches commerciales, etc.

Chapitre I
Généralités

III Conception logiciel

III.1 Notion de logiciel

Le logiciel est l'ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de l'information.

Autrement dit, un logiciel est un ensemble de programmes informatiques (du code) mais également un certain nombre de documents se rapportant à ces programmes et nécessaires à leur installation, utilisation, développement et maintenance : spécification, schémas conceptuels, jeux de tests, mode d'emploi… etc.

Chapitre I
Généralités

III.2 Crise du logiciel

Le développement de logiciels a longtemps souffert d'un certain nombre de problèmes tels que :

- La qualité moyenne, souvent médiocre, des logiciels produits
- Le nom respect des délais prévus pour le développement de logiciels
- Non satisfactions des cahiers des charges
- Le coût du logiciel dépassait le coût du matériel. **Aujourd'hui** il le dépasse très largement

III.3 Exemples de problèmes liés à la qualité des logiciels

- Octobre 2007, armée sud africaine : un canon anti aérien Oerlikon, quatre cracheurs de balles de 35 millimètres, s'est retourné, tout en tirant, au hasard. Un problème de logiciel serait à l'origine.

Chapitre I
Généralités

- 12/10/2006 La "mauvaise utilisation" d'un logiciel à l'origine d'accidents de radiothérapie à Epinal - Entre mai 2004 et août 2005, des patients traités aux rayons pour des cancers de la prostate ont subi des surdosages dus à des erreurs de paramétrage d'un logiciel. Conséquences actuelles : 4 décès et des complications chez 721 patients (au 7/9/07)... Cause : "erreur humaine". Cause réelle : "mauvaise ergonomie d'un logiciel obsolète".
- 01/11/2005 gros bug à la bourse de Tokyo, la plus importante d‘ Asie, et ce sont toutes les cotations qui sont bloquées toute la journée.
- Echec du premier lancement d'Ariane V: au premier lancement de la fusée Ariane V, celle ci a explosé en vol. Un problème de logiciel serait à l’origine.

Chapitre I
Généralités

Apparition du génie logiciel (GL) dans les années 70 pour répondre à la crise du logiciel.

Chapitre II
Génie logiciel-Introduction

II.1 Définition du Génie Logiciel (GL)

Domaine des 'sciences de l'ingénieur' dont la finalité est la conception, la fabrication et la maintenance de systèmes logiciels complexes, sûrs et de qualité ('Software Engineering' en anglais).

Il s'agit de la fabrication collective d'un système complexe, concrétisée par un ensemble de documents de conception, de programmes et de jeux de tests avec souvent de multiples versions

Chapitre II
Génie logiciel-Introduction

II.2 Les principales branches du GL

Le GL est en forte relation avec presque tous les autres domaines de l'informatique et d'autres disciplines de l'ingénieur :

- langages de programmation (modularité, orientation objet…)
- bases de données (modélisation des données, de leur dynamique ...),
- informatique théorique (automates, réseaux de Petri ...)
- ingénierie des systèmes et gestion de projets
- sûreté et fiabilité des systèmes, etc

Les principales branches du GL couvrent :

- la conception
- la validation/vérification
- la gestion de projet et l'assurance qualité
- les aspects socio-économiques.

Chapitre II
Génie logiciel-Introduction

II.3 Objectifs du génie logiciel

Le GL se préoccupe des procédés de fabrication des logiciels de façon à satisfaire les critères suivants:

II.3.1 Le système fabriqué doit répondre aux besoins des utilisateurs.

II.3.2 Les coûts restent dans les limites prévues au départ.

II.3.3 Les délais restent dans les limites prévues au départ.

II.3.4 Produire un logiciel de qualité:

Notion de qualité pour un logiciel

La qualité du logiciel est une notion qui recouvre plusieurs aspects:

- **Validité :** aptitude d'un produit logiciel à remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.
- **Fiabilité:** aptitude d'un logiciel à assurer de manière continue le service attendu.

Chapitre II
Génie logiciel-Introduction

- **robustesse :** aptitude d'un produit logiciel à fonctionner même dans des conditions anormales.
- **Extensibilité (maintenance) :** facilité d'adaptation d'un logiciel aux modifications ou aux extensions de ses fonctions.
- **Réutilisabilité :** aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.
- **Compatibilité :** facilité avec laquelle un logiciel peut être combiné avec d'autres logiciels.
- **Efficacité :** Utilisation optimales des ressources matérielles.

Chapitre II
Génie logiciel-Introduction

- **Portabilité :** facilité avec laquelle un logiciel peut être transféré sur de nouveaux environnements matériels et/ou logiciels.
- **Vérifiabilité :** facilité de préparation des procédures de test.
- **la traçabilité :** capacité à identifier et/ou suivre un élément du cahier des charges lié à un composant d'un logiciel,
- **Intégrité** : aptitude d'un logiciel à protéger ses différents composants conte des accès ou des modifications non autorisés
- **Facilité d'emploi :** facilité d'apprentissage, d'utilisation, de préparation des données, d'interprétation des erreurs et de rattrapage en cas d'erreur d'utilisation.

Ces qualités sont parfois contradictoires, il faut les pondérer en fonction du contexte.

Chapitre II
Génie logiciel-Introduction

II.4 Règles à respecter lors de la production de logiciel

Cette partie liste sept principes fondamentaux (proposés par Carlo Ghezzi):

- Rigueur
- séparation des problèmes
- modularité
- Abstraction
- anticipation du changement
- Généricité
- construction incrémentale

Chapitre II
Génie logiciel-Introduction

II.4.1 Rigueur

La production de logiciel doit ce faire avec une certaine rigueur. Le niveau maximum de rigueur est la formalité, c'est-à-dire que les descriptions et les validations se basent sur des notations et lois mathématiques mais il n'est pas possible d'être formel tout le temps.

II.4.2 Séparation des problèmes

C'est une règle qui consiste à considérer séparément différents aspects d'un problème afin d'en maîtriser la complexité. « diviser pour régner ».

Chapitre II
Génie logiciel-Introduction

II.4.3 Modularité

Un système modulaire est un système qui est composé de sous-systèmes plus simples, ou modules. La modularité est une propriété importante de tous les procédés et produits industriels (cf. l'industrie automobile ou le produit et le procédé sont très structurés et modulaires).

II.4.4 Abstraction

L'abstraction consiste à ne considérer que les aspects jugés importants d'un système à un moment donné, en faisant abstraction des autres aspects (c'est encore un exemple de séparation des problèmes).

Chapitre II
Génie logiciel-Introduction

II.4.5 Anticipation du changement

Les logiciels sont presque toujours soumis à des changements continuels (corrections d'imperfections et évolutions en fonctions des besoins qui changent). Ceci requiert des efforts particuliers lors de la conception du logiciel pour prévoir et gérer ces évolutions.

II.4.6 Généricité

Au lieu de résoudre un problème spécifique à part. Il est parfois avantageux de traiter la résolution du problème d'une manière plus général. Cette solution générique (paramétrable ou adaptable) pourra être réutilisée plus facilement.

Chapitre II
Génie logiciel-Introduction

II.4.7 Construction incrémentale

Un procédé incrémental se fait par étapes c'est-à-dire que chaque résultat est construit en étendant le précédent. Par exemple réaliser d'abord un noyau des fonctions essentielles et ajouter progressivement les aspects plus secondaires.

Chapitre III
Génie logiciel-Cycle de vie

III.1 Définition du cycle de vie

Le cycle de vie d'un logiciel (en anglais software lifecycle), correspond à toutes les étapes du développement d'un logiciel depuis la conception jusqu'à la maintenance.

Il faut souligner la différence entre étapes (découpage temporel) et activités (découpage selon la nature du travail). Il y a des activités qui se déroulent dans plusieurs étapes (ex : la spécification, la validation et la vérification), voire dans toutes les étapes (ex : la documentation).

III.2 Étapes du cycle de vie

Le cycle de vie du logiciel comprend plusieurs étapes:

Chapitre III
Génie logiciel-Cycle de vie

Analyse des besoins et faisabilité

c'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes, puis l'estimation de la faisabilité de ces besoins

Spécifications ou conception générale

Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel. (définir ce que le logiciel doit faire (le quoi)).

La conception architecturale

Une description de l'architecture du logiciel.

Chapitre III
Génie logiciel-Cycle de vie

Conception détaillée

Cette étape consiste à définir précisément chaque sous-ensemble du logiciel (élaboration des algorithmes).

La programmation (le codage)

c'est la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

Les tests unitaires du logiciel

Ces tests permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.

Chapitre III
Génie logiciel-Cycle de vie

L'intégration/tests d'intégration

Il s'agit d'assembler les parties du logiciel pour obtenir un système exécutable. Cela se fait en respectant le plan d'intégration du logiciel (défini dans l'étape de la conception détaillée). En plus, les jeux d'essais définis précédemment sont exécutés.

La vérification

Il s'agit de vérifier la conformité du logiciel aux spécifications globales.

Maintenance

Elle comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

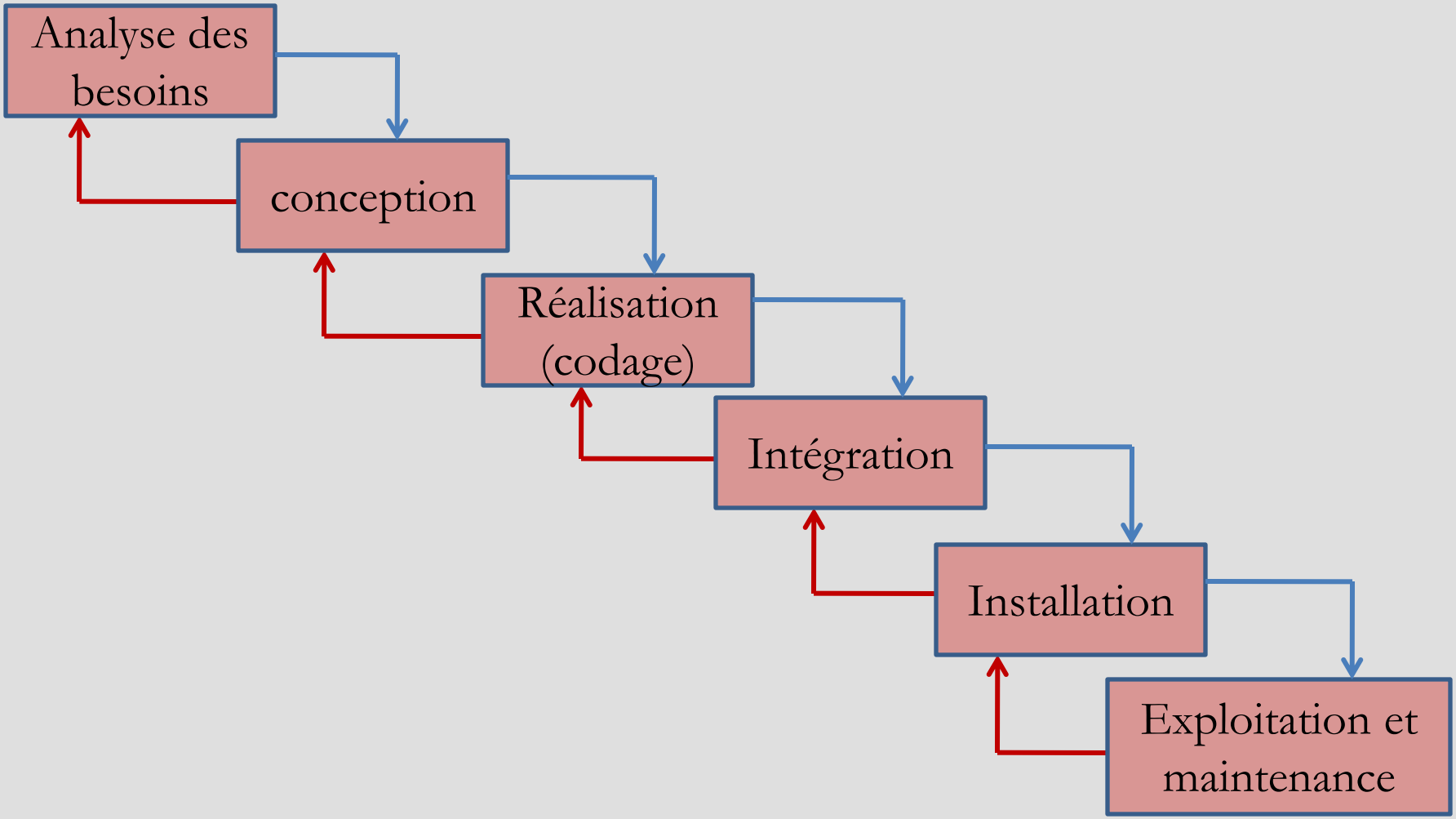
Chapitre III
Génie logiciel-Cycle de vie

La séquence et la présence de chacune de ces activités dans le cycle de vie d'un logiciel dépend du choix du modèle de cycle de vie . En plus des aspects techniques, le cycle de vie permet de prendre en compte l'organisation et les aspects humains.

III.3 Modèles de cycles de vie

III.3 .1 modèle en cascade

Chapitre III
Génie logiciel-Cycle de vie



Chapitre III
Génie logiciel-Cycle de vie

Ce modèle de développement est simple il est constitué d'une série d'étapes. Chaque étape se termine à une date précise par la production de certains documents ou logiciels. Le passage à l'étape suivante ne se fait que si les résultats de l'étape en cours sont jugés satisfaisants.

Il faut signaler que le modèle original ne comportait pas de possibilité de retour en arrière (l'inconvénient de ce modèle) : les erreurs de spécifications sont généralement détectées au moment des tests, voire au moment de la livraison du logiciel à l'utilisateur. Leur correction nécessite alors de reprendre toutes les phases du processus. La possibilité de retour en arrière a été rajoutée par la suite.

Chapitre III
Génie logiciel-Cycle de vie

Même avec ces possibilités de retour en arrière, le développement reste fondamentalement linéaire. il se base sur l'hypothèse souvent irréaliste que l'on peut dès le départ définir complètement et en détail ce qu'on veut réaliser. La pratique montre que c'est rarement le cas.

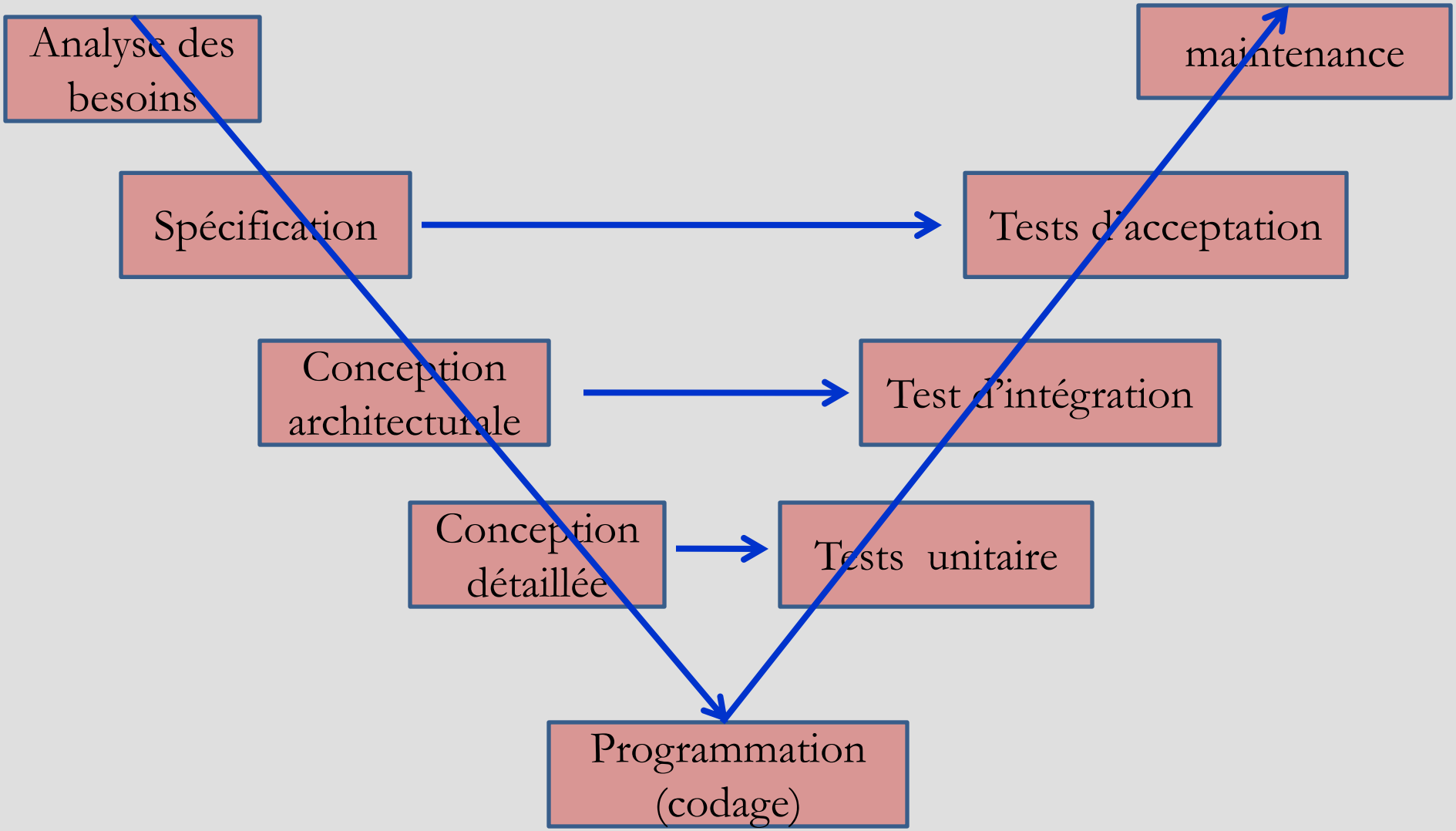
Chapitre III
Génie logiciel-Cycle de vie

III.3 .2 modèle par prototypage

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.
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.

III.3 .3 modèle en V

Chapitre III
Génie logiciel-Cycle de vie



Chapitre III
Génie logiciel-Cycle de vie

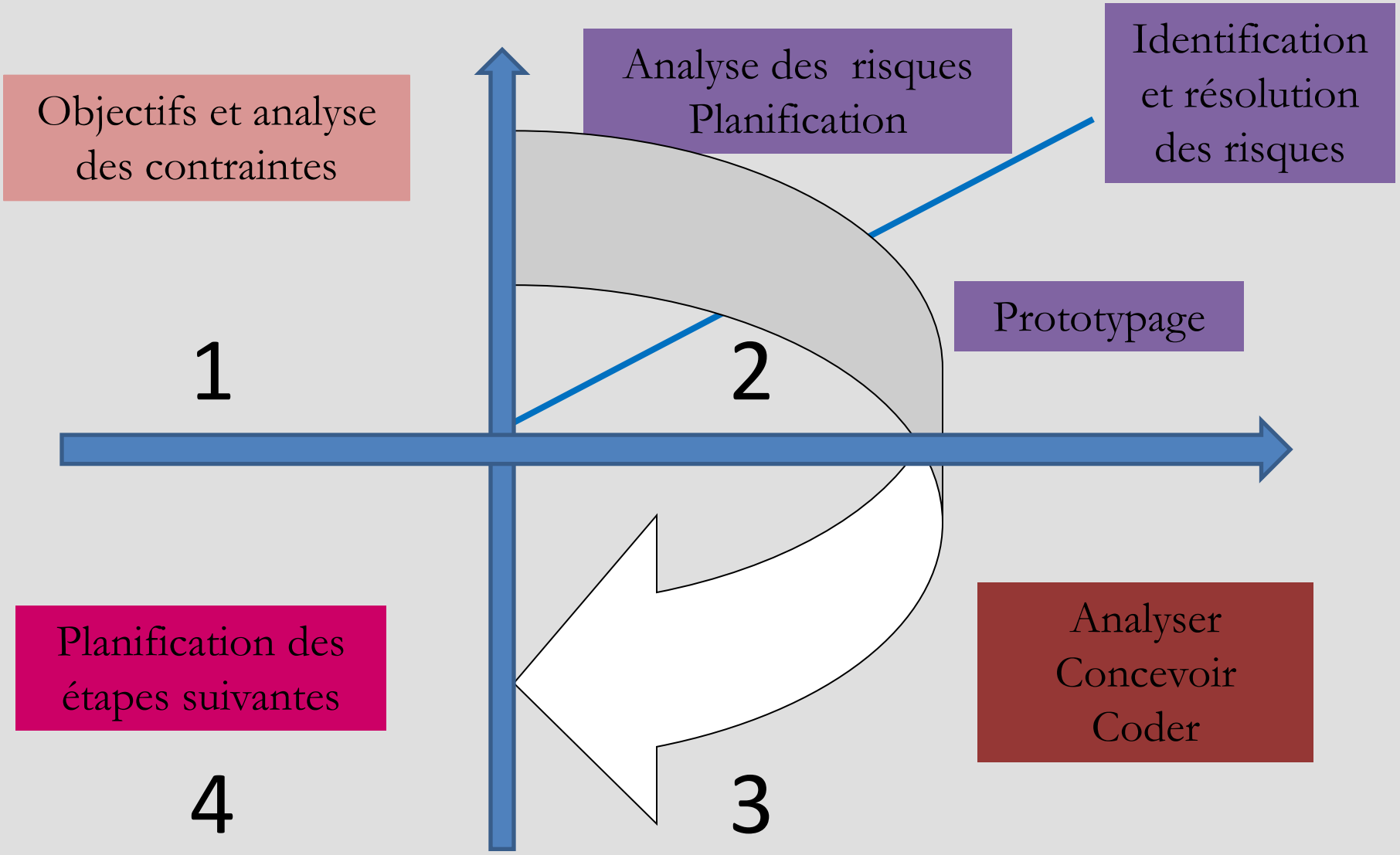
Le model en V présente deux branches. La première branche correspond à un modèle en cascade classique mais avec des sorties correspondant à des descriptions et des tests plus détaillés (vérification + validation). La seconde branche correspond à des tests effectifs effectués sur des composants réalisés.

- L'avantage d'un tel modèle est d'éviter d'énoncer une propriété qu'il est impossible de vérifier objectivement une fois le logiciel réalisé.

- Avec les jeux de tests préparés dans la première branche, les étapes de la deuxième branche peuvent être mieux préparées et planifiées.

III.3 .4 modèle en spirale

Chapitre III
Génie logiciel-Cycle de vie



Chapitre III
Génie logiciel-Cycle de vie

Le modèle en spirale (proposé en 1988) insiste particulièrement sur l'analyse des risques tels que les exigences démesurées par rapport à la technologie, la réutilisation de composants, calendriers et budgets irréalistes, la défaillance du personnel ….etc.
Chaque cycle se déroule en quatre étapes :

1. Enquête et examen de la situation existante (objectifs, alternatives, contraintes) dans l'entreprise, en prenant soin de bien prendre en compte les systèmes existants.

La situation doit être mesurée de façon à pouvoir évaluer les risques existants.

2. Si le poids des risques positifs est plus important, la phase suivante consiste alors à établir (planifier) ce qui doit être fait. Il peut y avoir une étape de prototypage pour éventuellement choisir entre plusieurs solutions.

Chapitre III
Génie logiciel-Cycle de vie

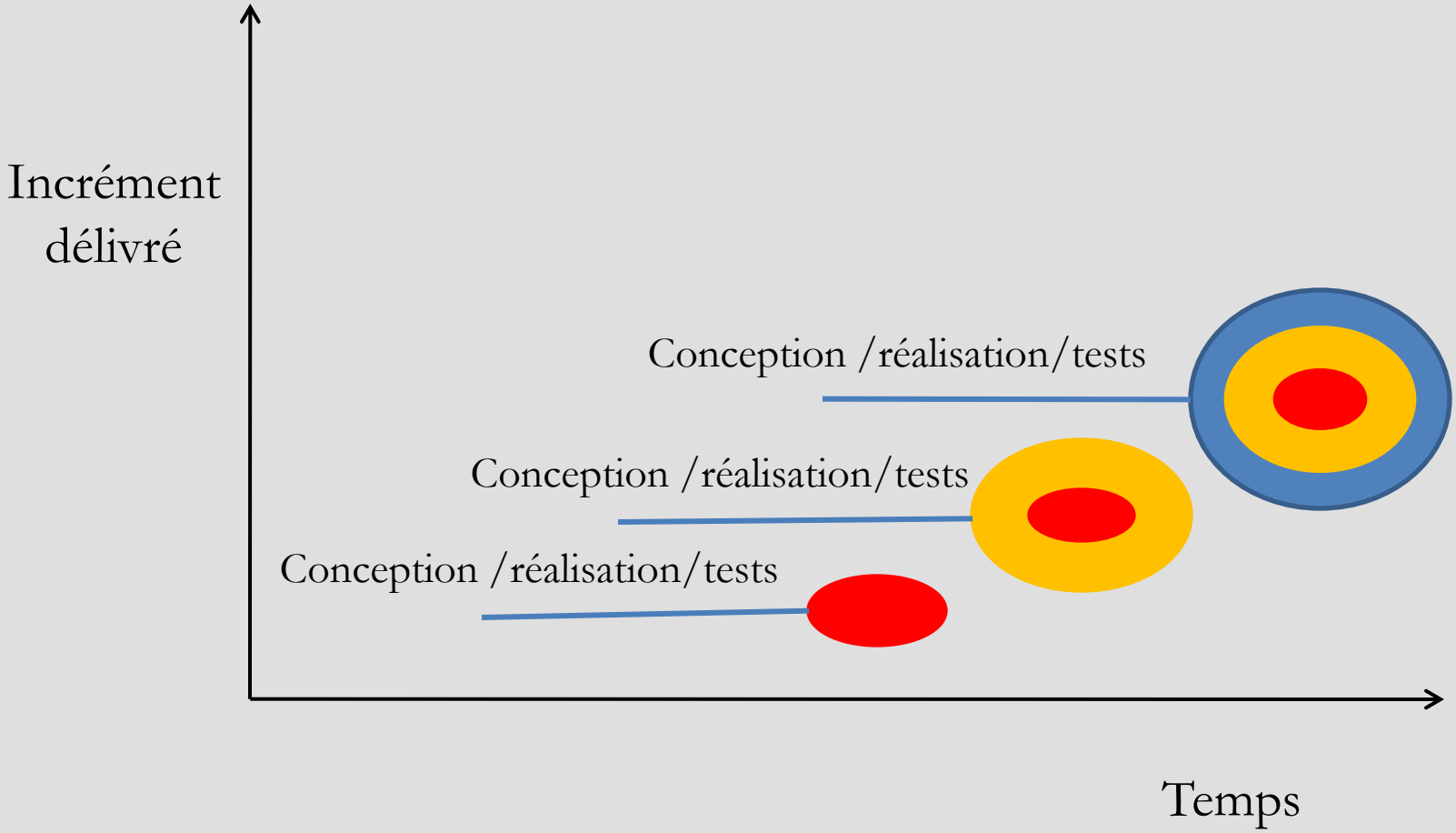
3. La réalisation de la solution nécessite la mise en œuvre d'un mini processus en cascade.

4. La situation est ensuite réévaluée pour déterminer si un développement supplémentaire est nécessaire, puis planifier l étape suivante.

Ce modèle est utilisé pour des projets dont les risques sont importants.

III.3 .5 modèle incrémental

Chapitre III
Génie logiciel-Cycle de vie



Chapitre III
Génie logiciel-Cycle de vie

Contrairement aux autres modèles où un logiciel est séparé en composants (développés séparément) et intégrés à la fin du processus, dans le modèle incrémental le produit est délivré en plusieurs fois, de manière incrémentale, c'est à dire en le complétant au fur et à mesure et en profitant de l'expérimentation opérationnelle des incréments précédents.

Chaque incrément peut donner lieu à un cycle de vie classique plus ou moins complet. Les premiers incréments peuvent être des maquettes (jetables s'il s'agit juste de comprendre les besoins des utilisateurs) ou des prototypes (réutilisables pour passer au prochain incrément en les complétant et/ou en optimisant leur implantation).

Chapitre III
Génie logiciel-Cycle de vie

- Le problème majeur dans ce modèle est la possibilité de remise en cause du noyau et des incréments précédents.
- De même, la possibilité de ne pas pouvoir intégrer d'autres incréments.

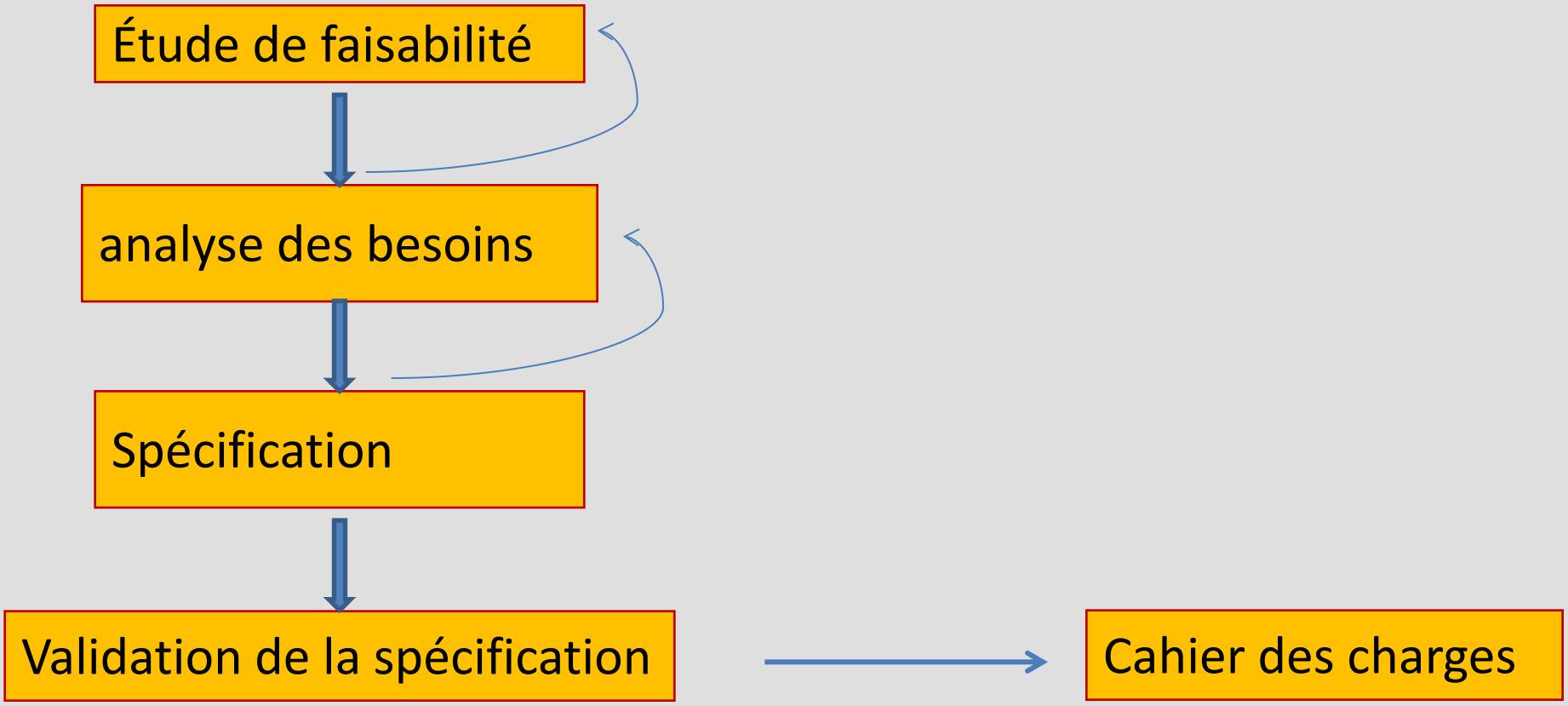
Afin de réduire ces risques, d'une part il est nécessaire de définir les incréments au début du projet et ce de façon globale. D'autre part, les incréments devront être le plus indépendant possible, aussi bien fonctionnellement qu'au niveau de la séquence de développement.

Remarque

Il n'y a pas de modèle idéal car tout dépend des circonstances. Souvent, un même projet peut mêler différentes approches, comme par exemple le prototypage pour les sous-systèmes à haut risque et la cascade pour les sous-systèmes bien connus et à faible risque.

Chapitre IV
Des besoins du client à la spécification du système

Plusieurs étapes sont nécessaires pour aller des besoins du client à la spécification du système



Chapitre IV
Des besoins du client à la spécification du système

1. Étude de faisabilité

L'objectif de l'étude de faisabilité est de donner une vue d'ensemble du rôle du système à développer dans son éventuel environnement d'utilisation pour évaluer sa pertinence.

Cette étude est donc courte et se focalise sur les questions suivantes :

- Est-ce que le système peut être implémenté à l'aide des technologies existantes et dans le cadre budgétaire et les contraintes temporelles fixées ?
- Est-ce que le système contribue aux objectifs essentiels de l'environnement d'utilisation ?
- Est-ce que le système pourra être intégré aux autres systèmes déjà en place ?

Cette étude se termine par l'élaboration d'un rapport et qui sert à décider si l'étude du système doit continuer (approfondir l'étude) ou si elle doit être suspendue.

Chapitre IV
Des besoins du client à la spécification du système

2. analyse des besoins

Une fois que l'étude de faisabilité a reçu une réponse positive on passe à l'analyse des besoins. Il s'agit d'une extraction minutieuse d'information.

Pour ceci il est nécessaire de :

- définir les personnes concernées par le système;
- d'évaluer leurs besoins par ordre d'importance.

Les difficultés de l'interaction avec les personnes affectées par le système sont multiples :

- Elles sont pour la plupart étrangères à l'informatique et décrivent donc leur besoin avec des termes imprécis.
- Elles n'ont pas de notion de ce qui est réalisable techniquement ou non.

Chapitre IV
Des besoins du client à la spécification du système

Pour l'extraction d'informations (qui est un exercice difficile):

- On peut faire des interviews, ce sont des outils efficace pour l'extraction de connaissances.
- On peut préparer un ensemble de questions précises.
- On doit rester ouvert d'esprit et être à l'écoute de son interlocuteur.
- Il faut faire preuve de qualité de communication.

Chapitre IV
Des besoins du client à la spécification du système

3. Spécification

- L'ingénierie de la spécification consiste donc à établir une communication entre les clients et les concepteurs du système.
- La spécification établit ce que le système doit faire (le QUOI) et les contraintes sous lesquelles il doit opérer.
- Définir une spécification, c'est expliciter des besoins de la façon la plus méthodique, complète et cohérente possible.
- Une spécification se développe, se vérifie, et évolue. Elle se concrétise sous la forme d'un cahier des charges qui dirigent la conception du système et sert de contrat entre les différentes parties.

Chapitre IV
Des besoins du client à la spécification du système

3.1 Types de spécifications

Lors de la description d'un système, il est essentiel de distinguer :

- L'aspect externe (celui des utilisateurs non informaticiens, des décideurs, . . .)

  du point de vue externe on définit la spécification des besoins : une description des services que doit rendre le système et les contraintes sous lesquelles il opère avec un haut niveau d'abstraction.

- L'aspect interne (celui des concepteurs, des personnels techniques, . . .)

  du point de vue interne, on définit une spécification du système : une description la plus précise possible du système qui doit être réalisé.

Chapitre IV
Des besoins du client à la spécification du système

3.2 Types de spécifications du système

Lors de la spécification d'un système il faut sépare entre:

- Les spécifications fonctionnelles : on définit les fonctions et services (transformation) que le système doit réaliser en prenant en considération la relation entre les entrées et les sorties. C'est-à-dire que chaque fonction sera décrite en détail, en spécifiant ses entrées et ses sorties.

Exemple: correcteur d'orthographe

- Lorsqu'un mot non identifié est rencontré dans le texte, l'utilisateur doit avoir le choix entre remplacer le mot par un mot de substitution, insérer le mot dans le dictionnaire, rechercher des mots similaires dans le dictionnaire afin d'en déterminer l'orthographe correcte, ou sortir du programme.

Chapitre IV
Des besoins du client à la spécification du système

- L'utilisateur doit avoir la possibilité d'afficher les mots du dictionnaire qui débutent par une chaîne de caractères indiquée.
- Chaque fois que c'est possible, les données entrées par l'utilisateur doivent être vérifiées, et en cas d'erreur, le programme affichera un message pour avertir l'utilisateur.

Chapitre IV
Des besoins du client à la spécification du système

- Les spécifications non fonctionnelles : ce sont les contraintes et les propriétés remplies par le système dans son intégralité.

  Exemple de contraintes : les contraintes imposées par l'environnement logiciel (le programme doit s'exécuter sur tel ou tel système d'exploitation), par l'environnement matériel (le programme doit utiliser les caractéristiques de tel ou tel terminal)

  Exemple de propriétés: l'efficacité, la robustesse, la sécurité, (des qualités qui sont difficile à vérifier et à quantifier).

Chapitre IV
Des besoins du client à la spécification du système

Exemple: correcteur d'orthographe

- Le dictionnaire doit être capable de contenir au moins 40'000 mots.
- Le logiciel doit être capable de traiter 250 mots à la minute.
- Les caractères qui font partie d'un mot sont les lettres majuscules et minuscules (sans distinction entre elles) et les apostrophes.
- Les mots jusqu'à au moins 13 caractères doivent être analysés pour vérifier s'ils sont écrits correctement.
- Les mots plus longs doivent pouvoir être affichés au terminal, afin de permettre à l'utilisateur d'en vérifier l'écriture.

Chapitre IV
Des besoins du client à la spécification du système

Les spécifications liées aux domaines d'activité : ce sont des spécifications, fonctionnelles ou non fonctionnelles, qui définissent des informations ou des contraintes propres à chaque type d'activité.

Chapitre IV
Des besoins du client à la spécification du système

3.3 Langage de rédaction de la spécification du système (notations)

Les langages naturels sont souvent utilisés pour spécifier les systèmes. L'utilisation de la langue naturelle facilite et compréhensible pour le client, mais malheureusement, donne certaines Ambiguïtés :

un même mot ou un même concept peut être expliqué de plusieurs façons différentes selon le lecteur ou le contexte.

Exemple : rapidité

Facilité d'utilisation

ces termes sont difficile à vérifier et à quantifier

Chapitre IV
Des besoins du client à la spécification du système

Des langages spécialisés ont été alors développés pour remplacer l'utilisation du langage naturel.

1. Langage naturel structuré : on fixe des patrons, des modèles de fiches, et des formulations, dont le sens est explicité de la façon la plus clair possible.

2. Description algorithmique : un langage de programmation abstrait est utilisé pour donner une vision opérationnelle du système.

Chapitre IV
Des besoins du client à la spécification du système

3. Notation graphique : des diagrammes, commenté par du texte en langage structuré. Ces notations sont particulièrement pratiques pour fournir une vue d'ensemble, statique ou dynamique, d'un système ou d'un sous-système.

4. Spécification mathématique : des formules ou des objets mathématiques sont utilisés pour définir un système en s'appuyant sur des théories lorsque le contexte le permet.

Chapitre IV
Des besoins du client à la spécification du système

4. Comment écrire un cahier des charges

Le cahier des charges est le document contractuel décrivant la spécification.

Il existe un plan standard de cahier des charges.

- Préface audience, version history, changelog (journal des modifications), . . .
- Introduction objectifs du document, portée du produit, . . .
- Glossaire définitions, acronymes, conventions utilisées, . . .
- Spécification des besoins point de vue externe (fonctionnelle et non fonctionnelle)

Chapitre IV
Des besoins du client à la spécification du système

- Architecture du système description de l'architecture générale du système, répartition de fonctionnalités sur les modules.
- Spécification du système point de vue interne (fonctionnelle et non fonctionnelle)
- Modèles du système relations entre sous-systèmes, composants, et systèmes externes, . . .
- Évolution du système prévisions sur comment le système va évoluer
- Appendices information plus détailles: description hardware et bases de données utilisées, configurations minimales, conseilles pour opérer le système, . . .
- Index plusieurs indexes, pour utiliser le cahier des charges comme référence

Chapitre IV
Des besoins du client à la spécification du système

5. Validation de la spécification

L'objectif de la validation de la spécification est:

1. de montrer que les besoins décrits dans la spécification correspondent à ceux des utilisateurs du système,

2. de détecter d'éventuelles erreurs dans le cahier des charges. Dans le cas général, détecter une erreur dans la spécification une fois que la conception, le développement et la validation du système sont terminés implique une nouvelle passe sur plusieurs phases voir toutes les phases avec le modèle en cascade. **« Tant qu'une erreur est détectée tôt tant qu'il est moins coûteuse ».**

Chapitre IV
Des besoins du client à la spécification du système

Les Points à vérifier sont les suivants:

- Vérification de la validité : il faut s'assurer que l'utilisateur confirme bien les besoins exprimés au début car, il se peut qu'un acteur revienne sur ses déclarations et réexprime ses besoins différemment.
- Vérification de la cohérence : les besoins exprimés dans le cahier des charges ne doivent pas être contradictoires.
- Vérification de la complétude : le cahier des charges doit, tant que possible, contenir la totalité des besoins des acteurs.

Chapitre IV
Des besoins du client à la spécification du système

- Vérification du réalisme : on doit vérifier que les besoins peuvent être effectivement satisfaits à l'aide de la technologie existante et en respectant le budget et les délais.
- Vérification de la vérifiabilité des besoins: on doit s'assurer que les besoins sont exprimés sous une forme vérifiable, avec le moins d'ambiguïtés possibles, de façon à ce que le document puisse faire office de contrat.

Chapitre IV
Des besoins du client à la spécification du système

6. Intervenant et spécification du système

Selon le type d'intervenant, il y a différent type de descriptions de la spécification d'un système :

- La spécification système des utilisateurs : c'est un contrat entre les concepteurs et les utilisateurs qui fixent, en des termes les plus précis possibles, ce que l'on attend du système en vue de satisfaire les besoins (qui ont été spécifiés plus tôt).

- La spécification de l'architecture : c'est un contrat entre les concepteurs et les développeurs qui établit le partitionnement modulaire du système.

- La spécification technique : c'est un contrat entre les développeurs qui établit une interface entre les composants logiciels développés indépendamment.