Modélisation des Systèmes d'Information

Jean-Yves Antoine

http://www.info.univ-tours.fr/~antoine

Processus de développement logiciel

Jean-Yves Antoine

U. Bretagne Sud - UFR SSI - IUP Vannes

année 2001-2002

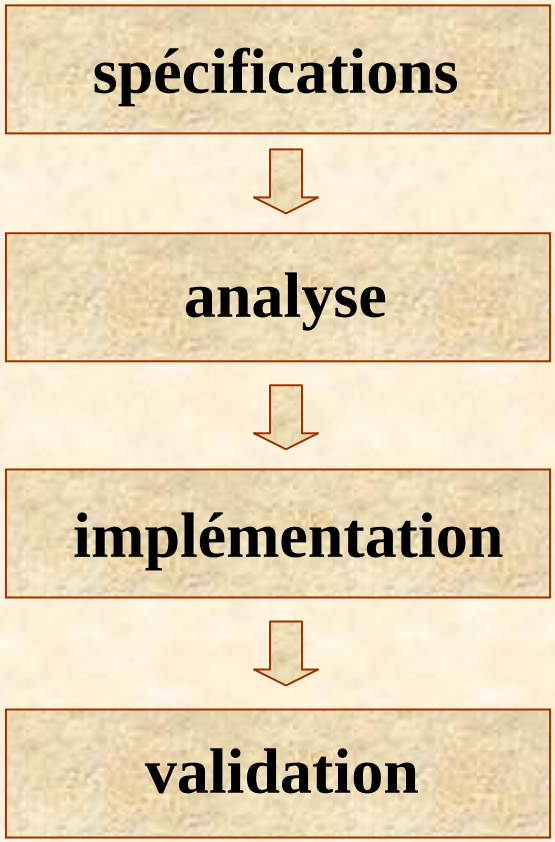
Plan du chapitre

1. Les étapes du processus de développement

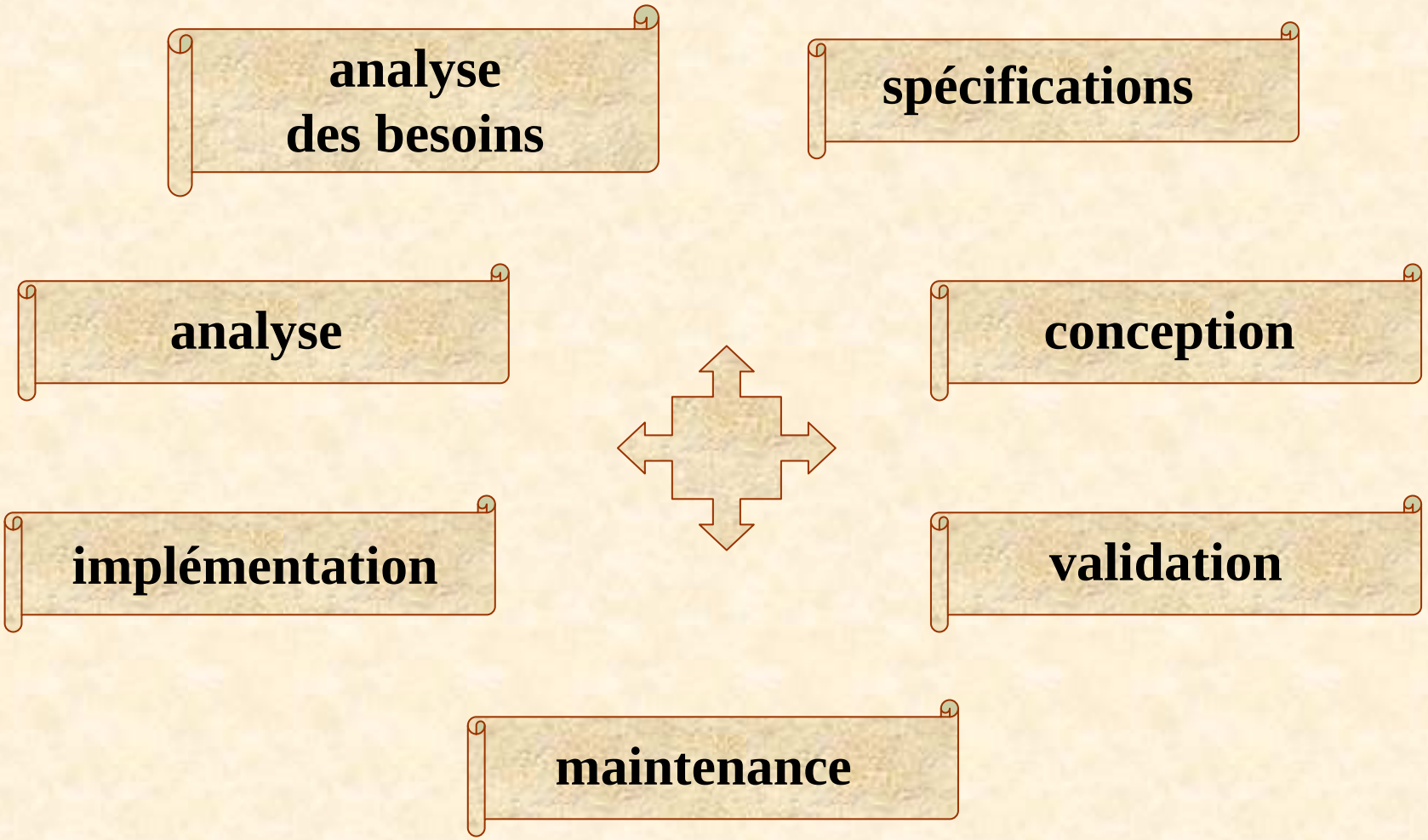
2. Les différents cycles de vie du logiciel

Programmation «in-the-small»

- Spécifications : données, complètes et précises
- analyse descendante : décomposition fonctionnelle
- implémentation : un seul langage de programmation
- validation : jeux d'essais



Etapes du processus de développement



Analyse des besoins

Etape préalable si le client n'a qu'une idée peu précise du système à réaliser

- Etude informelle des fonctionnalités (externes) du système sans considération technique : point de vue métier / utilisateur
- Dialogue fournisseur / client en termes intelligibles pour ce dernier : l'aider à formaliser le problème à résoudre
- Produit un document textuel avec schémas…
- Conduit généralement à la définition du cahier des charges

Analyse des besoins : exemple

Contrôle d ’accès à l’IUP de Vannes

(adapté de P-A Muller et N. Gaertner, 2000; Modélisation objet avec UML, Eyrolles)

Le bâtiment de l’IUP se divise en 4 zones (enseignement TD, enseignement TP, administration et enseignants) correspondant à des droits d ’accès différents. ***200*** *personnes ont quotidiennement accès au site. En dehors des visiteurs, tous les autres catégories de personnes disposeront d ’une carte d ’accès : enseignants (20), personnels administratifs (15) et étudiants (150). Les visiteurs n ’auront accès au bâtiment qu’aux heures de libre ouverture (****voir planning associé****). En dehors de ce horaires, l’autorisation d’accès dans chaque zone sera limitée au porteurs de badge et dépendra de leur statut (****enseignant****,* ***administratif****,* ***étudiant****) et de l ’horaire. Chaque zone est contrôlée par une porte d ’accès munie d’un lecteur de badge spécifique.*

Les droits d’accès ainsi que le planning des horaires d ’ouverture seront définis par un ***superviseur*** *….*

Spécifications

Ce que doit faire le système (côté client)

- Document précis spécifiant les fonctionnalités attendues
- Base du contrat commercial avec le client
- Document facile à comprendre par le client / utilisateur
- **Exemple** : *définition de la frontière du système, description des fonctionnalités du système avec scénarios, interactions : enchaînements d'écrans* ⇨ *cas d'utilisation de Jacobson*

Les spécifications ne sont jamais complètes et définitives
évolution du domaine, besoins supplémentaires ...

Spécifications : exemple

- Cas d ’utilisation de l’application « accès IUP »
  - *use case* : configuration système (acteur : superviseur)
  - *use case* : demande d’accès (acteur : porteur de badge)
  - …
- Use case : Configuration système
  - 1) **identification**
    - a) saisie du **code personnel**
    - b)vérification du code personnel
    - c) autorisation
  - 2) Modification de configuration
    - a) choix d ’une zone
    - ….

Analyse du système

« Quoi faire » : comprendre et modéliser le métier

- Réflexion métier hors de toute considération technique
- Reste un support de discussion avec le client/utilisateur
- **Premier modèle du système (niveau métier)**

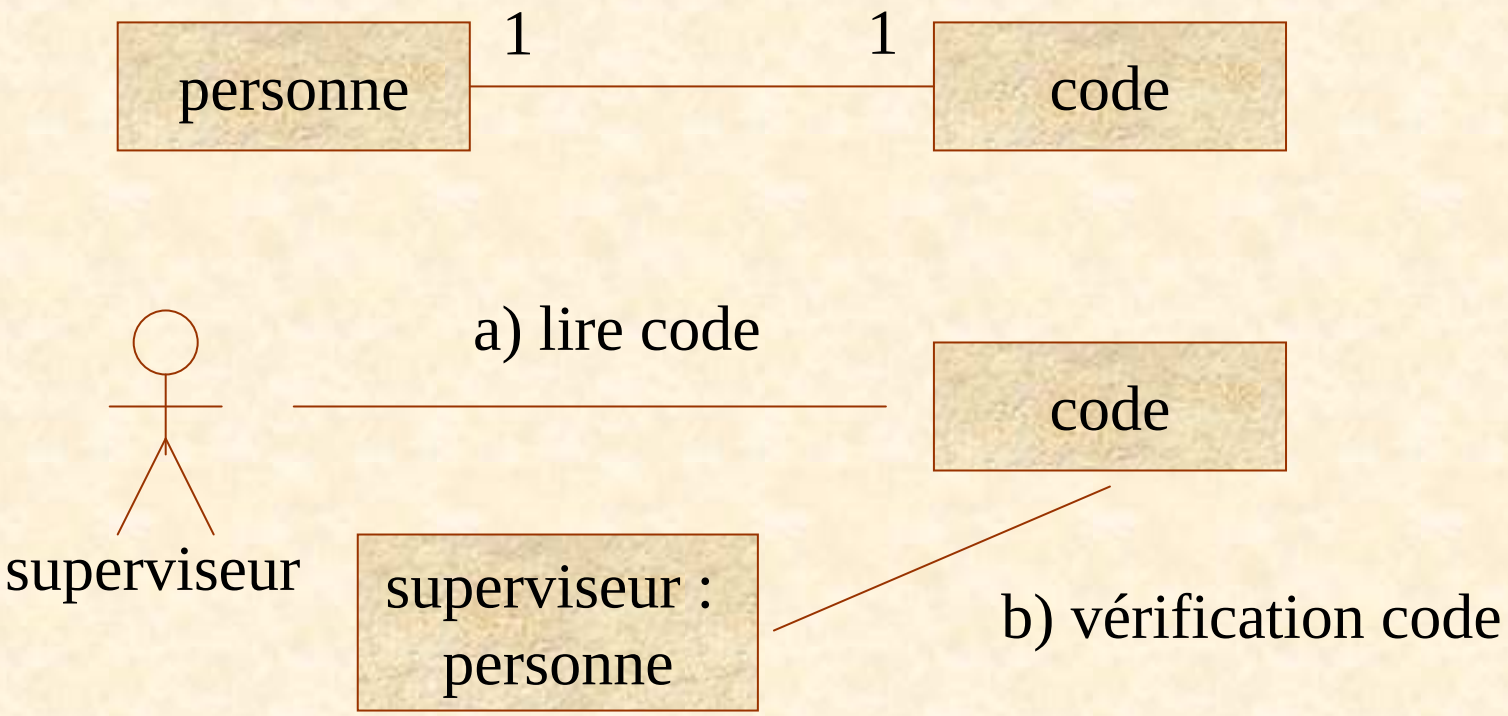
  Identifier les éléments intervenants hors (acteurs) et dans le système : fonctionnalités, structure et relations, états par lesquels ils passent suivant certains événements.

  Exemples : *diagrammes de classes, de collaborations...*

L'analyse n'est jamais complète mais elle doit être juste

Analyse : exemple

- « Accès IUP » : identification du superviseur



Conception

Comment faire le système : choix techniques

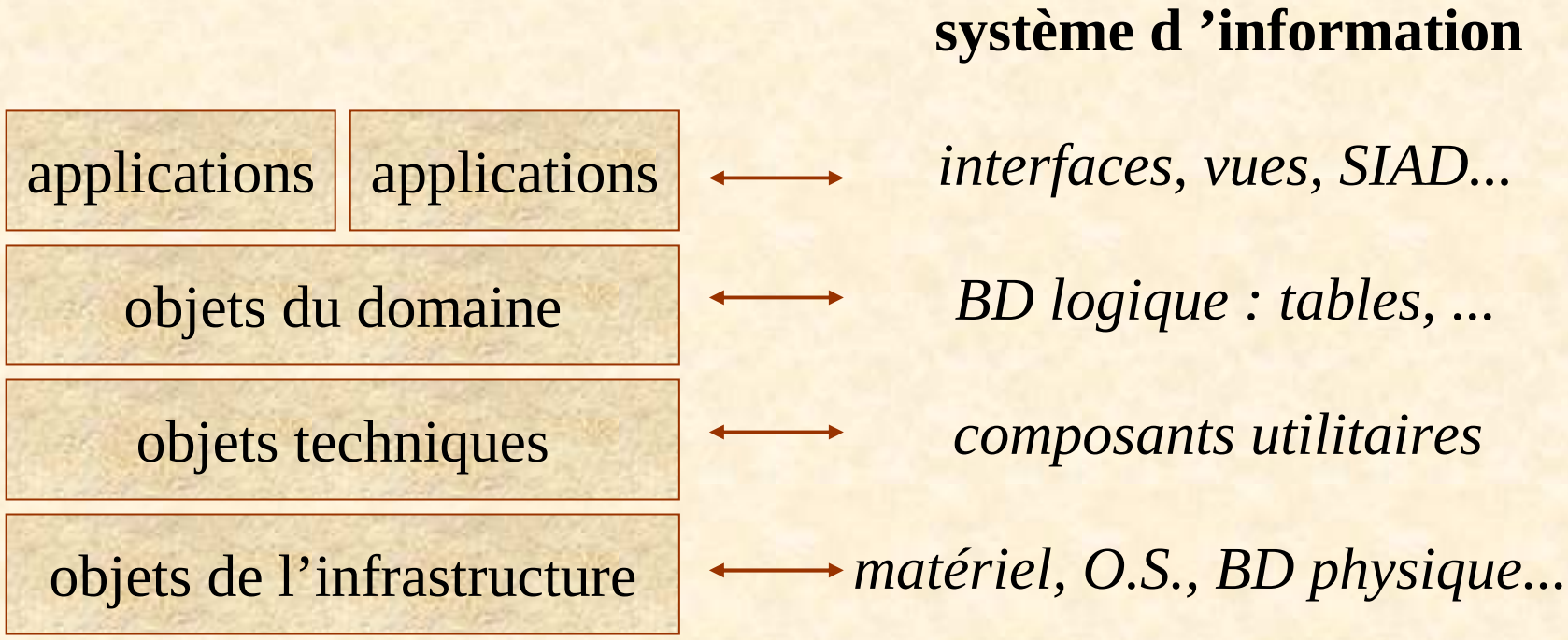
- Choix d'une architecture technique (matériel, logiciel) suivant certaines priorités (facteurs qualités : robustesse, efficacité, portabilité…)
- Expertise informatique : hors compréhension du client
- **Modèle de l'architecture logicielle du système**

  Décomposition en sous-systèmes : application (interfaces), domaine (métier) et infrastructure (implémentation).

  Permet la définition des phases d'implémentation, de validation et de maintenance

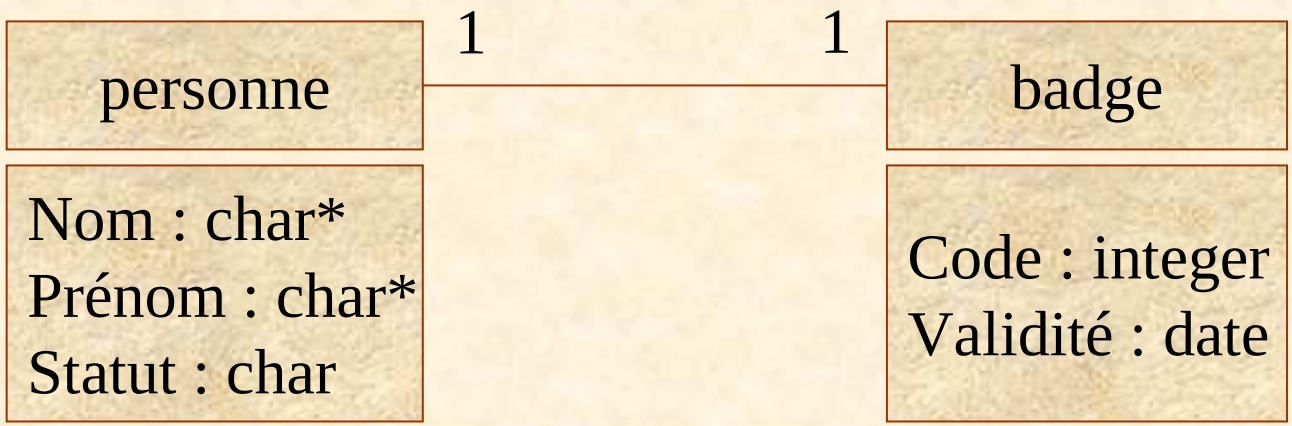
Conception

- Modèle d'architecture en couches



Conception : exemple

- « Accès IUP » : personne et code



Implémentation

- Souvent trop de temps consacré au codage au détriment des phases d'analyse et de conception : mauvaise pratique parfois très coûteuse…
- Savoir user de la réutilisabilité de composants, voire d'outils de génération de code (mise en place automatique du squelette du code à partir du modèle système)

⇨ l'activité de développement sera de plus en plus tournée vers la **réutilisation de composants existants**

Validation

- Tests de vérification
  - Vérification de la robustesse et cohérence du système en particulier dans les cas d ’exceptions
  - Testeur ≠ concepteur ou programmeur
  - Logiciels de test : toute ligne de code doit être testée !
- Recette
  - Validation client : accord avec les besoins
  - Une fois les tests de vérification satisfaisants
  - Planification dès les spécifications : cahier de recettes
- Activité souvent sous-estimée...

Maintenance

- Deux types de maintenance
  - correction des erreurs du système
  - demande d'évolution (modification de l'environnement technique, nouvelle fonctionnalités)
- Facteurs de qualité essentiels
  - corrections : robustesse
  - évolutions : modifiabilité, portabilité
- Etape longue, critique et coûteuse
  - 80 % de l'effort de certaines entreprises (pb de pratiques ?)

Quelques exemples de coûts

1. **Projet typique cité par [Mueller et Gaertner, 2000]**
2. **Projet concernant 5 personnes, d'une durée de 12 mois avec maintenance longue (plusieurs années)**

	❶	❷
✓ analyse des besoins et spécifications	15 %	6%
✓ analyse / conception	25%	5 %
✓ implémentation	30 %	7 %
✓ tests et validation	15%	15 %
✓ maintenance	15 %	67 %

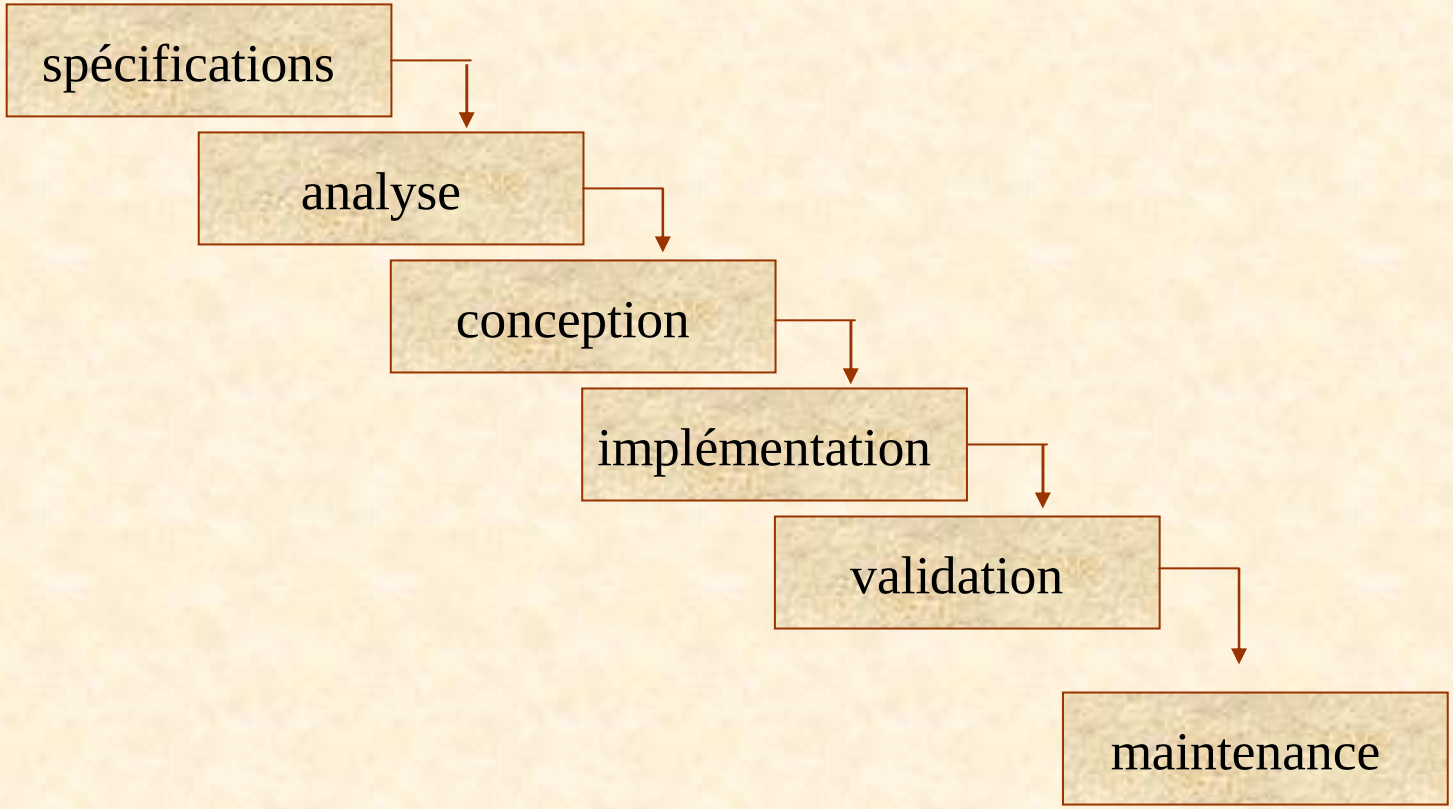
Cycle de vie d'un logiciel

Séquencement des différentes étapes du processus de développement

- Cycles de vie linéaire
  - modèle en cascade
  - modèle en « V »
- Cycle de vie itératif
  - modèle en spirale (ou incrémental) : prototypes

Modèle en cascade

Cycle de vie linéaire sans aucune évaluation entre le début du projet et la validation



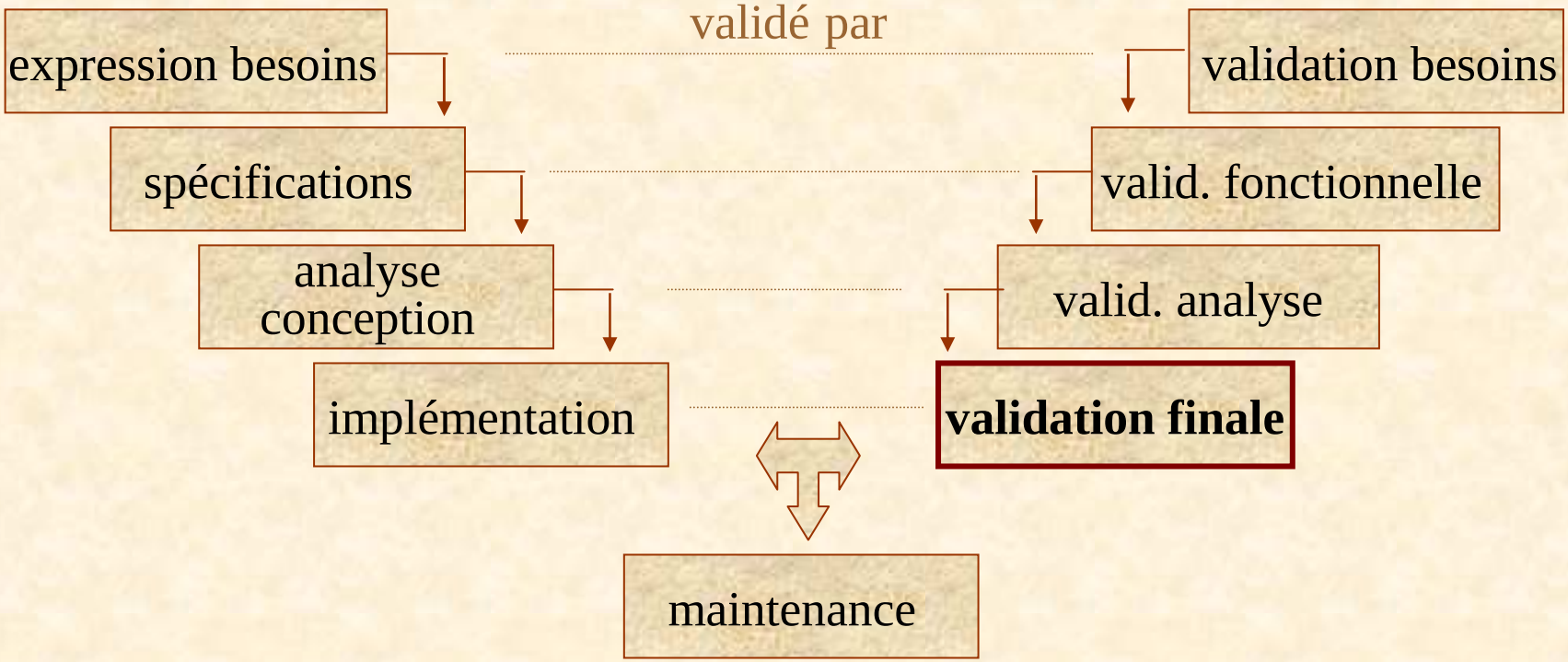
Modèle en cascade : inconvénient

- Aucune validation intermédiaire
  - impossibilité de suivre le déroulement du projet, donc de planifier un travail en équipe
  - augmentation des risques car validation tardive :erreur d ’analyse ou de conception très coûteuse !
- Solution limitée aux petits projets
  - risques bien délimitées dès le début du projet
  - projet court avec peu de participants

Cycle en « V »

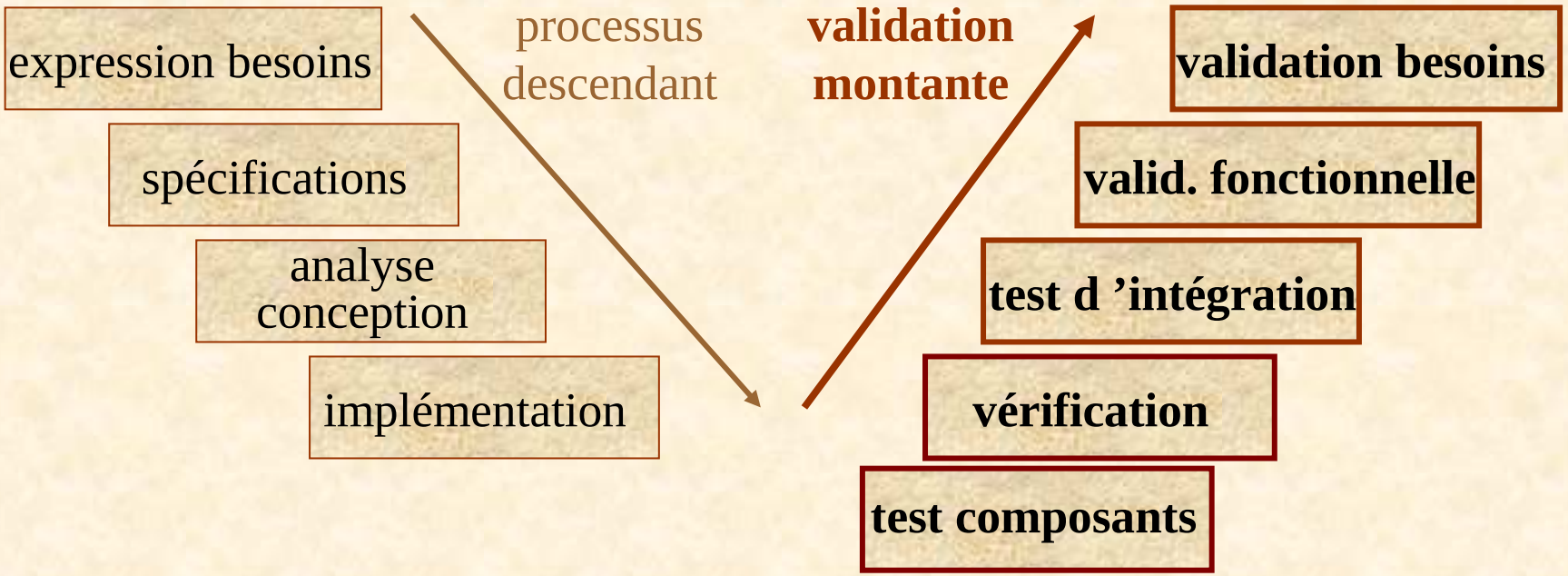
- Objectif
  - Mieux gérer le risque et faciliter la planification
- Principes
  - Processus linéaire
  - Prévention des erreurs : validation des produits à chaque sortie d'étape descendante
  - Préparation des protocoles de validation finaux à chaque étape descendante
  - Validation finale montante : confirmation de la pertinence de l'analyse descendante

Cycle en « V » : analyse descendante



- Validations intermédiaires «papier» : documentation

Cycle en « V » : validation



protocoles de validation définis par l’analyse descendante

Cycle en « V » : intérêts

- Validations intermédiaires
  - bon suivi de projet : points de mesure concrets de l'avancement du travail avec étapes clés
  - favorise la décomposition fonctionnelle de l'activité
  - limitation des risques en cascade par validation de chaque étape
  - existence d'outils support
- Modèle éprouvé très utilisé pour de grands projets

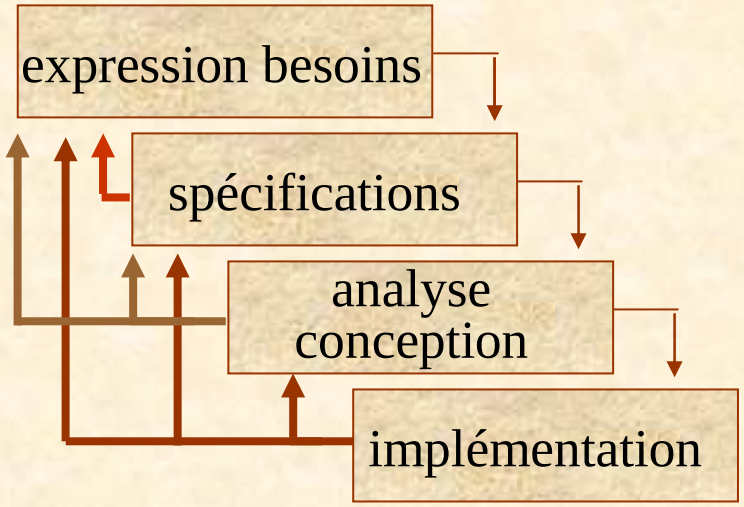
pourtant...

Cycle en « V » : limitations

- Un modèle toujours séquentiel ...
  - Prédominance de la documentation sur l'intégration : validation tardive du système par lui-même
  - Les validations intermédiaires n'empêchent pas la transmission des insuffisances des étapes précédentes
  - Manque d'adaptabilité
  - Maintenance non intégrée : syndrome du logiciel jetable
- … adapté aux problèmes sans zones d'ombre
  - Idéal quand les besoins sont bien connus, quand l'analyse et la conception sont claires

Améliorations du cycle en « V »

- Retours de correction des phases précédentes



- Réponse *ad hoc* pour casser la linéarité du processus
- Fonctionne si besoins de corrections limités

↳ **Cycle de vie itératif**

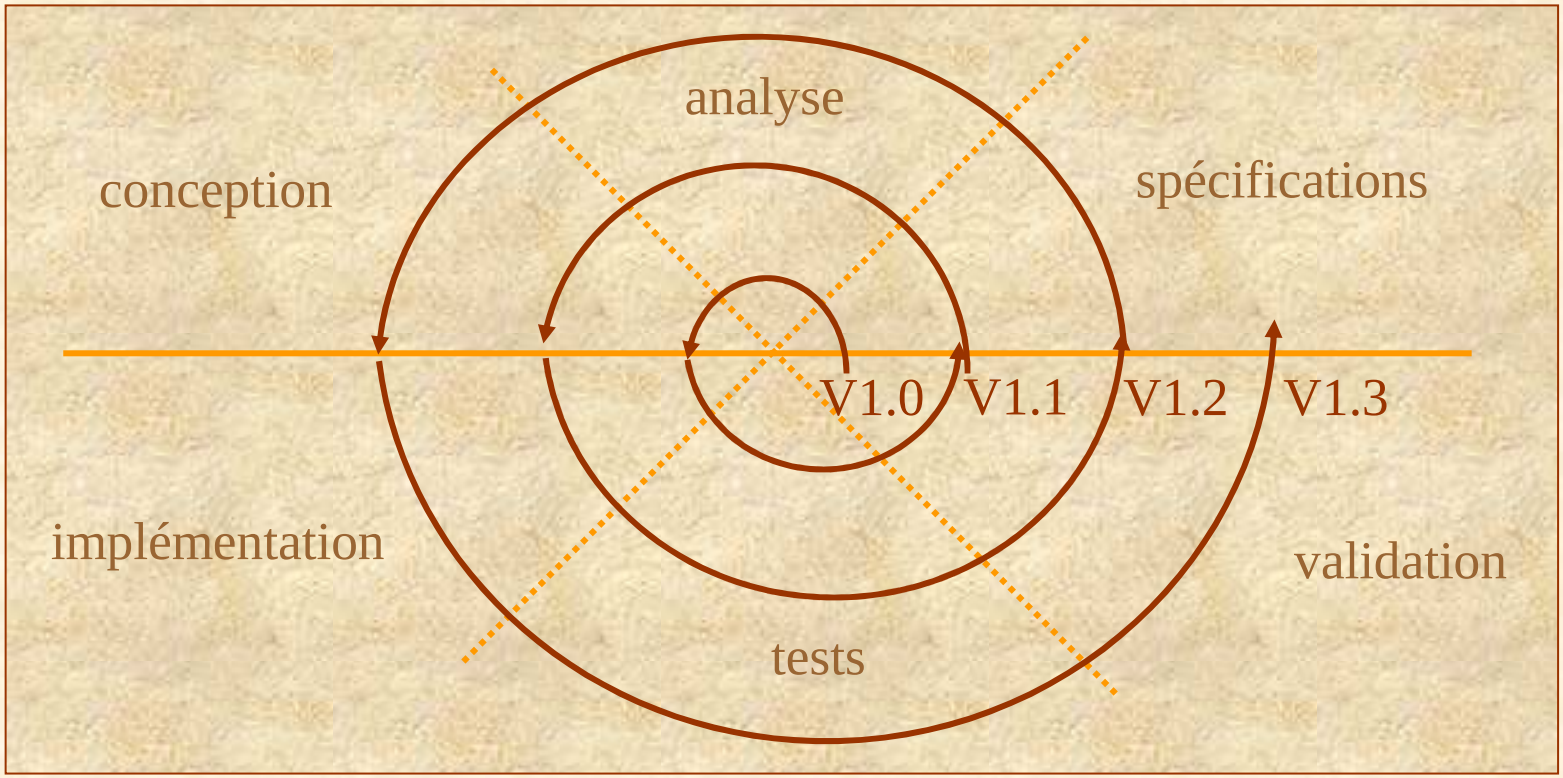
Cycle de vie itératif

- Prototypage

  - **Idée** : fournir le plus rapidement possible un **prototype** exécutable permettant une validation concrète et non sur document
  - Progression du projet par incréments successifs de versions successives du prototype : **itérations**
  - Certains prototypes peuvent être montrés au client et utilisateurs. Par ailleurs, une **maquette** peut être réalisée préalablement au premier prototype (Prolog !)
  - La validation par prototypes ne justifie pas l'absence de recours à la **documentation** !

Cycle de vie itératif

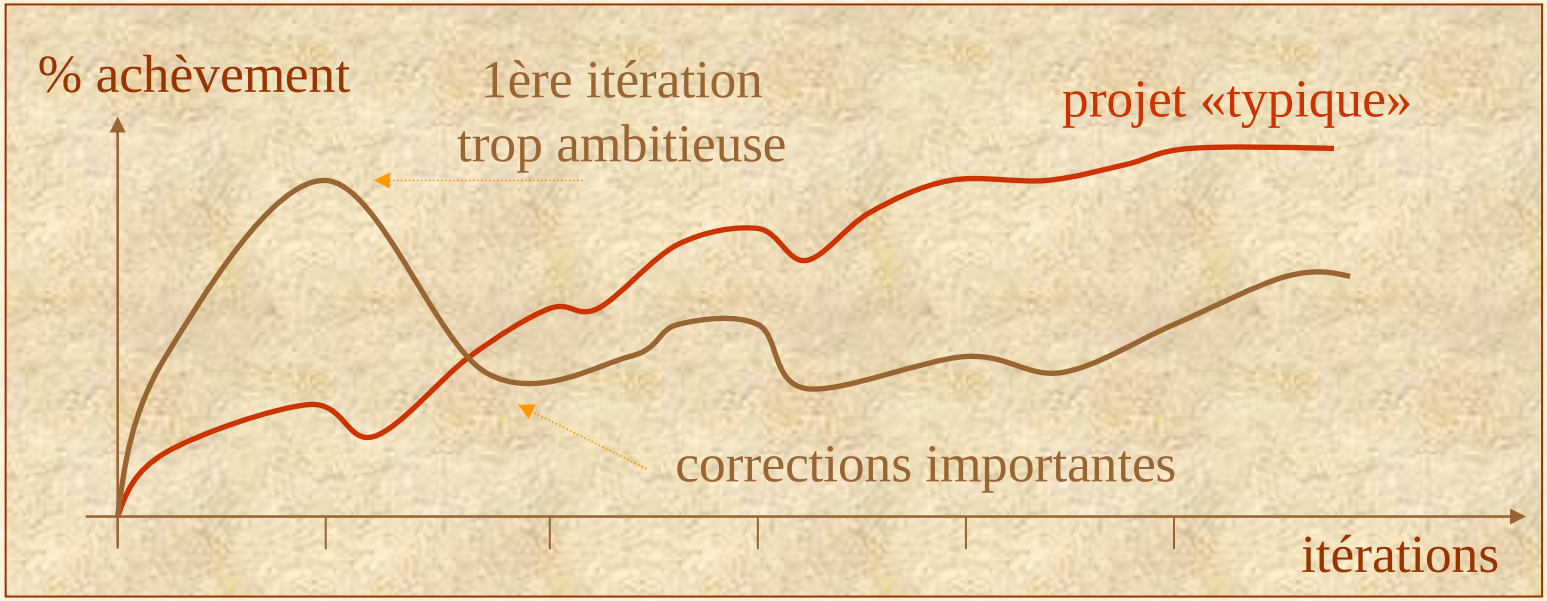
- Itération : mini-cycle de vie en cascade
- La spirale du cycle itératif



Cycle de vie itératif

- Criticité de la première itération

  Ne pas chercher à réaliser le système complet à la première itération : travers d'un processus linéaire



Cycle de vie itératif : intérêts

- Validation concrète et non sur documents
- Limitation du risque à chaque itération
- Client partenaire : retour rapide sur ses attentes
- Progressivité : pas d'explosion des besoins à l'approche de la livraison : pas de «*n'importe quoi pourvu que cela passe*»
- Flexibilité :
  - modification des spécifications = nouvelle itération
  - maintenance = forme d'itération
- Planification renforcée

Cycle itératif : fausses limitations

- Le cycle itératif….
  - N ’encourage pas la bidouille
  - Ne crée pas de problèmes : il les révèle au contraire
  - Ne demande pas de refaire la roue à chaque itération : enrichissement et non recommencement
  - N ’encourage pas la modification constante des besoins : hiérarchise avant tout les besoins

Cycle itératif : limitations

- Pas de processus idéal
  - Cycle itératif : planification très attentive et rigoureuse
  - Cycle en «V» : processus éprouvé le plus répandu, surtout pour les systèmes connus. Cycle itératif : peut dérouter à premier abord
- Processus adapté à la modélisation objet
  - Modèle objet : se prête parfaitement à une démarche incrémentale
  - Le cycle en spirale a cependant une portée générale