VÉRIFICATION, VALIDATION, REVUES, ESSAIS ET TESTS

Introduction

VV010

2013-03-13

Luc LAVOIE
Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Luc.Lavoie@USherbrooke.ca
http://pages.usherbrooke.ca/llavoie

Plan général

- **Introduction**
  - **Présentation**
  - **Propriétés et critères**
  - **Difficultés**
  - **Besoins**
  - **Définitions**
  - **Classification**
  - **Proposition pragmatique**
- *Vérification et validation*
  - *Principes*
  - *Inadéquation*
  - *Couverture*
- *Stratégies d'essai*
  - *Stratégies unitaires*
  - *Stratégies d'intégration*
  - *Stratégies de système*
  - *Stratégies de non-régression*
  - *Stratégies globales*
- *Techniques de tests*
  - *Partitions*
  - *Techniques fonctionnelles dynamiques*
  - *Techniques structurelles dynamiques*
  - *Techniques de conception*
- *Gestion des essais*
  - *Interaction entre tests et revues*
  - *Plan d'essai*
  - *Placement des activités*
  - *Gestion des équipes*
  - *Gestion des résultats*
  - *Gestion des suites à donner*
  - *Automatisation*
  - *Documentation*

Introduction
Présentation

- Validation
  - Construisons-nous le bon produit?
  - Choisissons-nous les bonnes propriétés?

- Vérification
  - Construisons-nous le produit correctement?
  - Traduisons-nous correctement les propriétés?

Introduction
Présentation – contexte

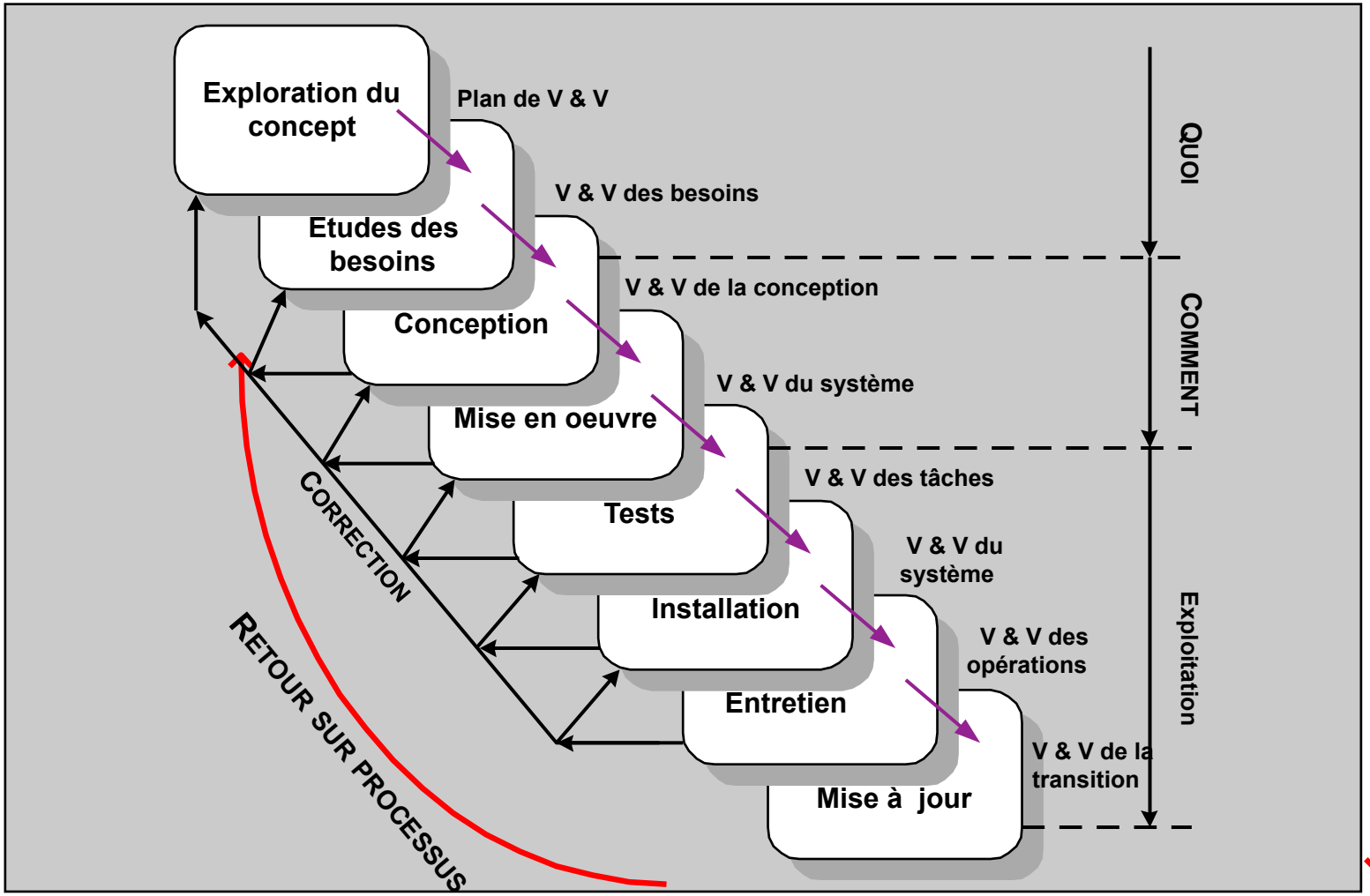
- L'industrie logicielle grandit
  - le nombre d'anomalies augmente
- Les systèmes sont plus complexes
  - les causes des anomalies sont mieux dissimulées
- La variété des configurations augmente
  - les risques aussi
- La portée des systèmes s'étend
  - la faisabilité d'une validation en exploitation diminue
- La durée de vie est souvent longue
  - le cout et l'impact d'une erreur grandit

Introduction
Présentation – positionnement

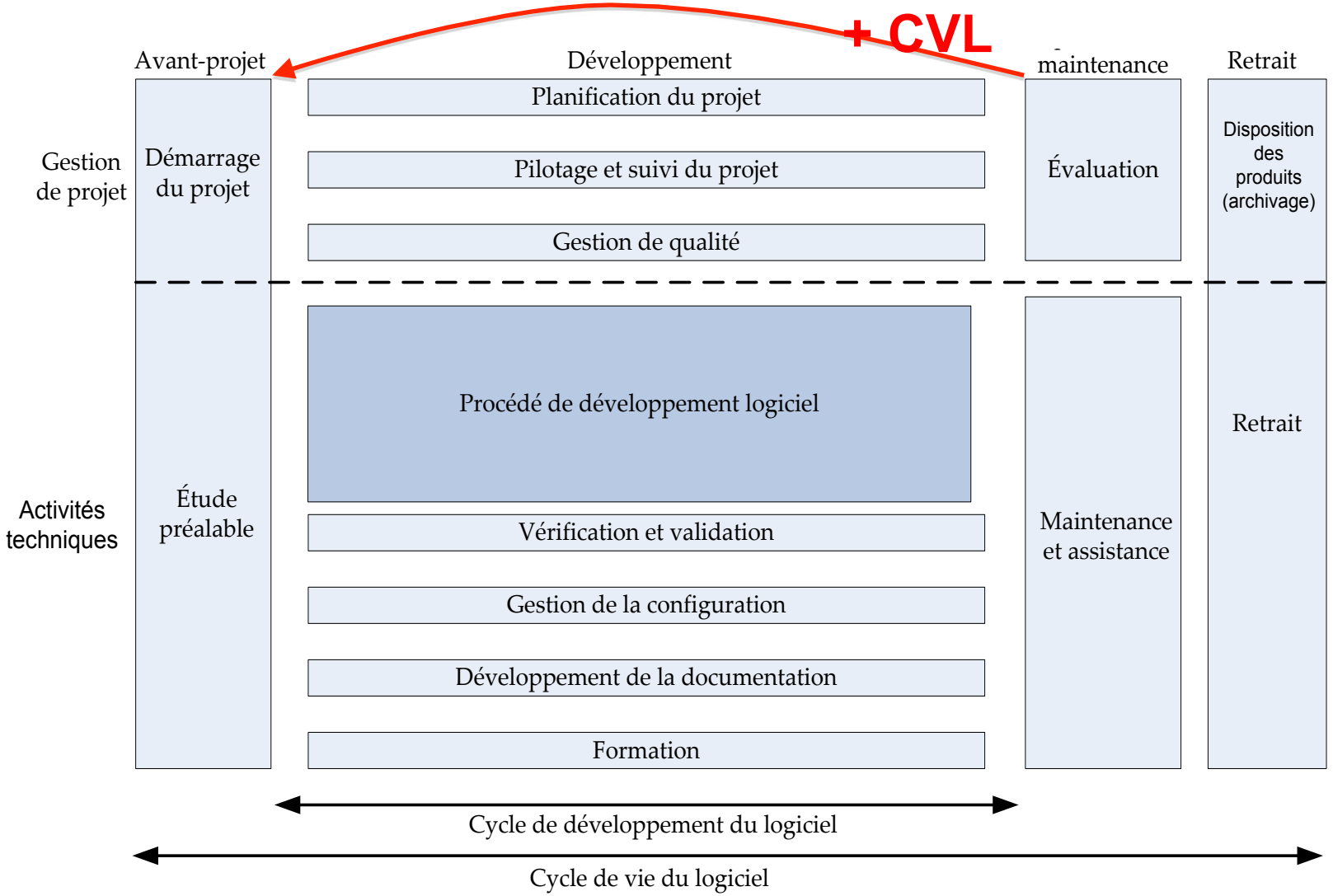
- Présents partout
- Présents tout le temps

- En amont du développement
  - pour valider le modèle
- En aval du développement
  - pour valider la mise en oeuvre
- Pendant le développement
  - pour vérifier chaque artéfact

INTRODUCTION
PRÉSENTATION – EXEMPLE



Introduction
Présentation – Relation au cycle de vie



INTRODUCTION
PRÉSENTATION – NÉCESSITÉ DES TESTS (1/2)

- En pratique, il n'y a pas de vérification ni de validation sans tests...
- Pourquoi?
  - Dans la très grande majorité des cas, les exigences du logiciel ne sont pas entièrement formalisées.
  - Dans la très grande majorité des cas, le logiciel est produit par un processus imparfait (donc faillible).

INTRODUCTION
PRÉSENTATION – NÉCESSITÉ DES TESTS (2/2)

- En théorie, il n'y a pas de vérification et ni de validation sans tests...
- Pourquoi?
  - Pas d'autres moyens de valider l'atteinte des besoins en fin de parcours
    (car le passage initial des besoins aux exigences ne peut être formalisé).
  - Pas d'autres moyens de qualifier le produit dans son contexte d'exploitation
    (non formalisable, lui non plus).

Introduction
Propriétés et critères (1/5)

- Idéalement, un test devrait permettre de montrer la présence ou l'absence de certaines propriétés

- Exemples de classes de propriétés
  - utilité [usefulness]
    conformité aux besoins
  - sécurité de fonctionnement [dependability]
    conformité aux exigences
  - la validité
    conformité aux lois du domaine d'application

- Montrer c'est bien, mesurer c'est (parfois) mieux!
  - ... mais comment mesurer?
  - ... il faut des critères!

Introduction
Propriétés et critères (2/5)

- Validité (**propriété**) [validity].
  Lorsque le logiciel livre un résultat, celui-ci est juste.

- Validité (**critère**).
  La proportion des résultats justes sur le nombre de résultats livrés est supérieure à une cible déterminée (par exemple 99,9998 % sur un échantillon aléatoire uniforme d'au moins 10 millions de cas).

Introduction
Propriétés et critères (3/5)

- Fiabilité (**propriété.v1**) [*reliability*].
  Le logiciel livre toujours un résultat dans un temps prescrit, généralement considéré comme « raisonnable » (donc, le logiciel ne plante pas, ne boucle pas). Par exemple, le logiciel X est fiable.
- Fiabilité (**critère.v1**).
  La proportion des opérations réussies sur le nombre d'opérations tentées. Par exemple, la fiabilité du logiciel X est de y %.

- Fiabilité (**propriété.v2**).
  Propriété d'un système informatique capable d'assurer ses fonctions sans défaillance, dans des conditions préalablement définies et sur une période déterminée.
- Fiabilité (**critère.v2**).
  ???

Introduction
Propriétés et critères (4/5)

- Robustesse (**propriété**) [*robustness*].
  Capacité d'un composant à continuer de fonctionner même quand il reçoit de mauvaises données en entrée ou dans des conditions environnementales anormales. Robustesse par rapport à une IPM implique de considérer que les actions de l'un utilisateur ne sont jamais des erreurs (au sens que nous lui donnons).

- Tolérance aux pannes (**propriété**) [*fault tolerance*].
  Capacité d'un composant à continuer de fonctionner malgré la défaillance de sous-composants (matériels ou logiciels). Très complexe non seulement parce qu'il est impossible de prévoir toutes les fautes, mais aussi parce qu'elles peuvent avoir été introduites très tôt dans le CVL.

INTRODUCTION
PROPRIÉTÉS ET CRITÈRES (5/5)

- Disponibilité (**propriété**) [*availability*].

  La disponibilité indique le pourcentage de temps pendant lequel le système est disponible

  - disponibilité = MTBF/(MTBF+MTTR) où
  - MTBF est la fiabilité, la moyenne des temps de bon fonctionnement (Mean Time Between Failures), et
  - MTTR est le temps de relève, la moyenne des temps de réparation (Mean Time To Repair).

Introduction
Encore des propriétés...

- Performance
- Coût de traitement
- Sécurité
- Sureté
- ...

- voir IEEE 830, 1012, 1233

Introduction
Difficultés — énumération

- Axiomatisation
- Décidabilité (Gödel et Turing)
- Complexité (NP et NP-complet)
- Déterminisme
- Caractère abstrait
  - de l'objet des tests
  - des activités elles-mêmes
- Perceptions erronées
  - Les tâches liées aux activités de test nécessitent peu de de compétences et ne représentent pas un défini intellectuel intéressant.
  - Un bon travail en amont élimine les erreurs en aval.

Introduction
Difficultés — conclusion

- Un test peut
  - Démontrer la présence d'erreurs
  - Illustrer la disparition d'une anomalie dans un contexte limité
- Un test ne peut pas
  - Montrer l'absence d'erreur
- Il sera donc difficile d'utiliser les tests pour montrer la présence ou l'absence d'une propriété, en général.
- Au mieux, ils pourront être utilisés pour l'illustrer ou le montrer dans un contexte limité.

Introduction
Qu'est-ce qu'un test?!?

- Pour mieux répondre à cette question, tentons de faire le tour (systématique) des besoins auxquels les tests doivent répondre.
- En chemin, nous aurons à définir, ou à rappeler la définition de, plusieurs autres termes, par exemple...

Introduction
Vocabulaire

- Essai
- Anomalie
- Défaillance
- Défaut
- Erreur
- Bogue

- Attente
- Besoin
- Spécification
- Exigence
- Contrainte
- Critère

Introduction
Besoins généraux

- Compréhension du domaine
  — ingénierie des exigences
- Maîtrise du produit
  — développement
- Maîtrise du processus
  — gestion de projet
- Maîtrise de l'exploitation
  — gestion de l'évolution
- Arbitrage contractuel
  — gestion des parties prenantes

Introduction
Définitions

- Les tests servent dans deux contextes différents
  - prestation des essais
    - constatation de propriétés
    - estimation de mesures
  - gestion des anomalies
    - dépistage de la cause de l'anomalie

- … ce qui nous amène à définir
  - anomalie
  - essai

Introduction
Définitions – un point de départ

- Anomalie (*anomaly*)
  - écart constaté entre le comportement effectif et le comportement attendu
- Défaillance (*failure*)
  - incapacité constatée de réaliser une fonction à un moment donné et dans des conditions documentées
- Erreur (*error*)
  - écart constaté entre la spécification et la mise en oeuvre (conception ou mise en exploitation)
- Défaut (*defect*)
  - cause de l'erreur

Introduction
Définitions – la suite

- Essai (*Test suite*)
  - Collection de tests ayant pour but d'évaluer un artéfact ou un processus en regard d'un ensemble de propriétés définies
- … mais on n'a toujours pas défini ce qu'était un test!

INTRODUCTION

DÉFINITIONS

(STANDARD GLOSSARY OF SOFTWARE ENGINEERING TERMINOLOGY)

- Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, réalisée dans le but
  - soit de vérifier qu'il répond à ses spécifications,
  - soit d'identifier les différences entre les résultats attendus et les résultats obtenus.

Introduction
Classification

- Pour nous y retrouver, tentons de catégoriser et de classer
  - les défauts
  - les essais
  - les tests
  - ...

Introduction
Classification des défauts (1/2)

- Calcul (manipulation)
- Logique (formalisation)
- Traitement des données (modèle)
- Définition des données (théorie)

Introduction
Classification des défauts (2/2)

- *Calcul*
- *Logique*
- *Traitement des données*
- *Définition des données*
- Interface
  - Interne
  - Externe
    - PM (personne-machine)
    - MM (machine-machine)

Introduction
Classification des essais

- Selon la portée des propriétés ciblées
  - Essai unitaire
  - Essai d'intégration
  - Essai de système
    - Essai protégé
    - Essai alpha
    - Essai bêta
  - Essai contractuel
    - Essai de livraison (par le fournisseur)
    - Essai d'acceptation (par le commanditaire)
    - Essai de qualification (selon une norme et par les PP)
    - Essai de certification (selon une norme et par un tiers)
  - Essai de mise en exploitation
  - Essai de non-régression

Introduction
Classification des tests (2/2)

- Selon l'objet testé
  - spécification
  - code source
  - code exécutable
- Selon la technique utilisée
  - fonctionnelle vs structurelle
  - statique vs dynamique
  - vérification vs validation

- Plus utile… pourquoi?

PROPOSITION PRAGMATIQUE HYPOTHÈSES

- Notre proposition repose sur deux hypothèses :
  - L'axiome du programmeur intègre et compétent
  - L'axiome du couplage

Proposition pragmatique
Définition d'un test

- Un test est une procédure permettant d'exécuter un (composant) logiciel dans un contexte déterminé et contrôlé dans le but d'évaluer certains critères tels
  - critère d'acceptation
    - les anomalies constatées sont insuffisantes pour motiver le rejet du composant
  - critère d'erreur
    - les anomalies constatées sont des erreurs
  - etc.

Proposition pragmatique
Composition d'un essai

- Composition d'un essai
  - But
  - Description de la portée
    - Exigences, contraintes, limites, hypothèses
  - Description et motivation de la stratégie retenue
  - Description du banc d'essai
  - Composition (énumération des tests constitutifs)
  - Étapes d'application
  - Conditions
    - démarrage
    - arrêt
    - reprise
    - ...

- On ajoute souvent une liste des jeux de test lorsqu'ils sont utilisés par plus d'un test.

Proposition pragmatique
Composition d'un test

- Composition d'un test
  - Objectifs
    - exigences,
    - critères,
    - mesures
    - ...
  - Sélection et motivation des techniques utilisées
  - Jeux de test
  - Description de la préparation du banc d'essai
  - Étapes d'application
  - Conditions
    - démarrage,
    - arrêt,
    - reprise
    - ...
  - Code (ou spécification détaillée)

Proposition pragmatique
Définition d'un jeu de test

- Un jeu de test est la séquence des données utilisées lors de l'exécution d'une instance d'un test
- Qualités d'un bon jeu de test
  - Représentatif
    - d'une classe de problèmes
    - d'une classe de solutions
    - d'un ensemble de traces
  - Facilement générable et gérable
    - manuellement
    - automatiquement
  - À résultat prévisible
    - calculable,
    - qualifiable ou
    - approximable

Proposition pragmatique
Corolaires

- Notre définition du test impose la capacité de l'exécuter.
- Un test doit être défini en fonction de critères précis, eux-mêmes dérivés d'exigences documentées.
- Le test est distinct de la revue dont il est le complément.
- Le plan de vérification et validation (PVV) a, entre autres objectifs, celui de synchroniser les deux.

Proposition pragmatique
Principes généraux

- Sensibilité (*sensitivity*)
- Redondance (*redundancy*)
- Réduction (*restriction*)
- Partition (*partition*)
- Visibilité (*visibility*)
- Rétroaction (*feedback*)

- Pour plus de détails, voir PY-3

Proposition pragmatique
Que faut-il pour faire un essai?

- Le code exécutable du composant
- Un banc d'essai
- Un jeu d'essai
- Une procédure d'essai
- Un cahier d'essai
- Un guide d'interprétation des résultats
- … et le plus souvent un ***oracle***!

CONCLUSION

- Vérification
  - ensemble des activités visant à éliminer les erreurs de conception et de mise en oeuvre
  - contrôle du passage des exigences au produit
- Validation
  - ensemble des activités visant à éliminer les erreurs d'ingénierie des exigences
  - contrôle du passage des besoins (attentes) aux exigences

Rappels
Propriétés et critères de projet

- CQFD
  - cout (des ressources)
  - qualité (du produit)
  - fonctionnalité (ou portée du produit)
  - durée (ou temps de réalisation)

- Référence : INF 754→ PMBoK

Rappels
Propriétés et critères d'exploitation

- FURPSE
  - *functonality*
  - *usability*
  - *relibility*
  - *performance*
  - *maintenability*
  - *evolutivity*

- Référence : Printz → ISO 9126

Rappels
Propriétés et critères environnementaux

- PESTEL
  - politique
  - économique
  - social
  - technologique
  - écologique
  - légal

- Référence : Printz → INCOSE