VÉRIFICATION ET VALIDATION

Introduction

VV010
v220b

2014-09-03

Luc LAVOIE
Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Luc.Lavoie@USherbrooke.ca
http://info.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*

PRÉSENTATION

- Rappel
- Positionnement
- Exemple
- Relation avec le cycle de vie
- Nécessité des tests

Présentation
Rappel - Définitions

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

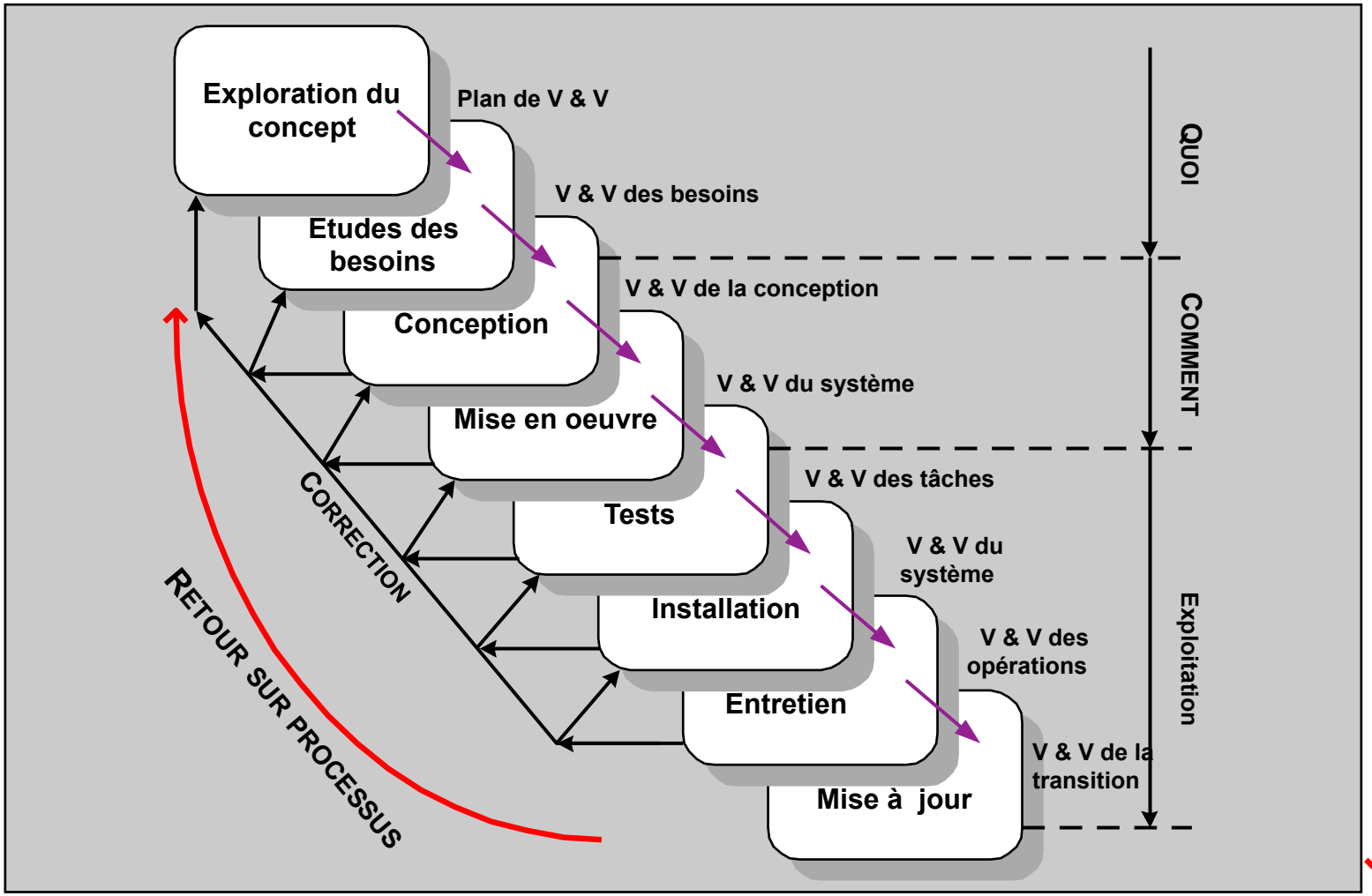
- Vérification
  - Construisons-nous le produit correctement?
  - Traduisons-nous correctement les propriétés?

Présentation
Rappel - Positionnement

- Présents partout
- Présents tout le temps

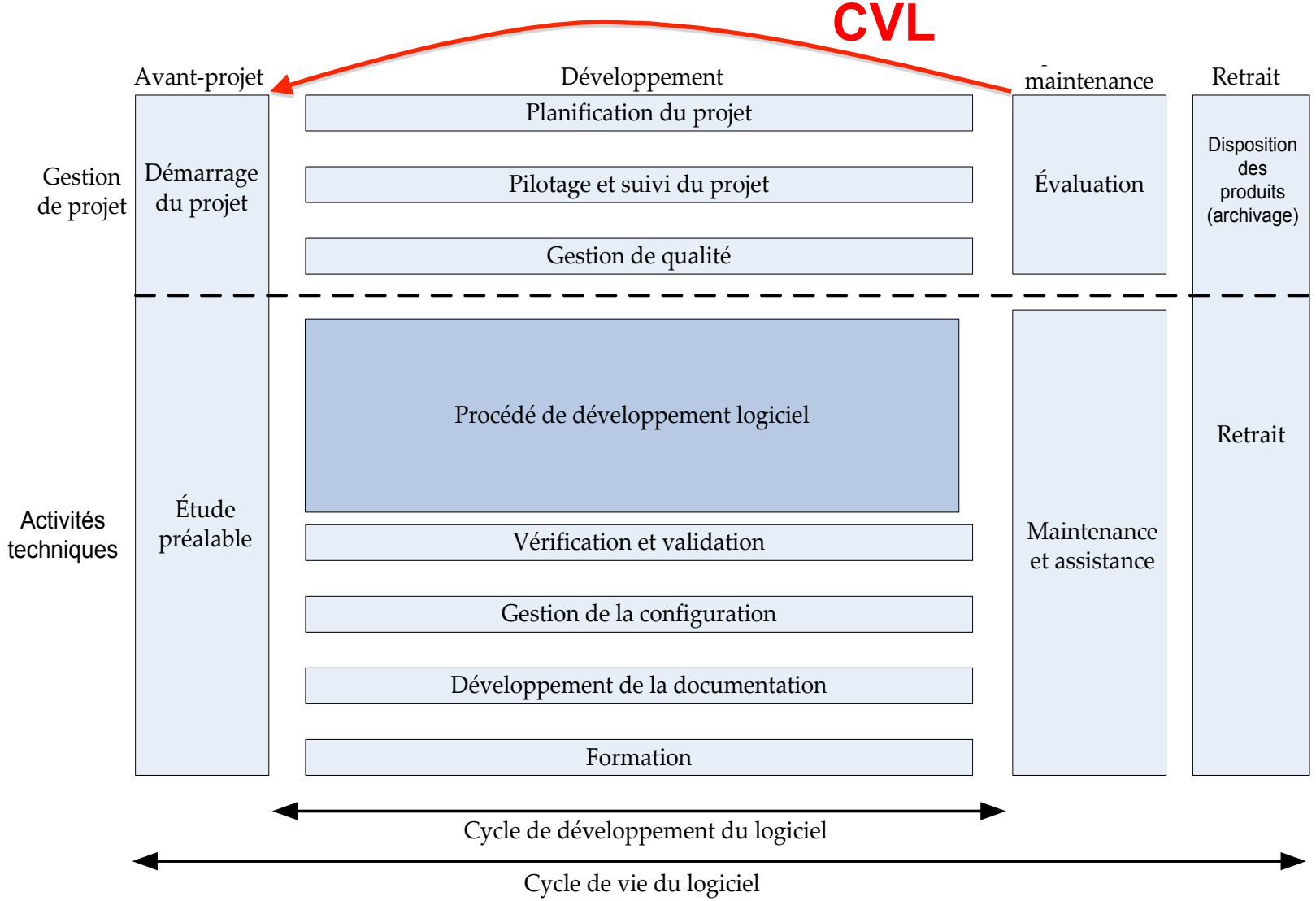
- En amont du développement
  - pour valider le modèle
- En aval du développement
  - pour valider la mise en oeuvre
- Pendant le développement
  - pour vérifier chaque artéfact

Présentation
Rappel - Exemple



Plus une erreur est découverte tardivement, plus elle coute cher!

Présentation
Rappel - Relation au cycle de vie



PRÉSENTATION
CONCLUSION - 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).

PRÉSENTATION
CONCLUSION - 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).

Propriétés et critères

- Origine des propriétés
- Nécessité des critères
- Exemples
  - Validité
  - Fiabilité
  - Robustesse et tolérance aux pannes
  - Disponibilité
  - … encore des propriétés

Propriétés et critères
Origine et nécessité

- 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 contraintes de l'environnement
  - *validité*
    conformité aux exigences du domaine d'application

- Montrer c'est bien, mesurer c'est souvent mieux et parfois nécessaire!
  - ... mais comment mesurer?
  - ... il faut des critères!

Propriétés et critères
Exemples (1/5)

- 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).

Propriétés et critères
Exemples (2/5)

- 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**).
  ???

Propriétés et critères
Exemples (3/5)

- 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.

Propriétés et critères
Exemples (4/5)

- 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 totaux de rélève (Mean Time To Repair).

Propriétés et critères
Exemples (5/5) ... encore!

- Performance
  - temps UCT
  - temps de réponse
  - mémoire interne
  - mémoire externe
  - débit réseau
  - latence transactionnelle
- Cout de traitement
- Sécurité
- Sureté
- ...

- voir IEEE 830, 1012, 1233

Difficultés

À la recherche de ce qu'est un test

- Énumération
- Discussion
- Conclusion

Difficultés énumération

- Formalisation (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 compétences et ne représentent pas un défini intellectuel intéressant.
  - Un bon travail en amont élimine les erreurs en aval.

Difficultés
Discussion

- 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.

Difficultés
Conclusion

- 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...

Besoins

Toujours à la recherche de ce qu'est un test

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

Besoins
Ingénierie des exigences

- Exploration et compréhension du domaine d'application
- Vérification d'hypothèses
- Validation de l'adéquation fonctionnelle

Besoins
Développement

- Validation de l'atteinte des critères de qualité au niveau prescrit :
  - fiabilité
  - validité
  - performance
  - disponibilité
  - robustesse
  - tolérance aux pannes
  - etc.
- Vérification
  - sous tous ses aspects

Besoins
Gestion de projet

- Gestion de l'intégration
  - Gestion des modifications
  - Suivi
- Gestion de la portée
  - Compréhension
  - Surveillance
- Gestion du temps
  - Jalons
  - Taux de réalisation
- Gestion des risques
  - Évaluation des risques
- Gestion de la qualité
  - Contrôle de la qualité
- Gestion des approvisionnements
  - Impartition

Besoins
Gestion de l'évolution

- Détection et correction des anomalies
- Protection contre la résurgence des anomalies
- Validation de la portée et de l'impact des changements et des modifications
- Vérification des procédures d'exploitation

Besoins
Gestion des parties prenantes

- Contrôle de qualité
- Audit
- Qualification
- Certification

VOCABULAIRE

Enfin, aurions-nous trouvé ce qu'est un test?

- Essai
- Anomalie
- Défaillance
- Défaut
- Erreur
- Bogue
- Test?
- Attente
- Besoin
- Spécification
- Exigence
- Contrainte
- Critère

VOCABULAIRE
POINT DE DÉPART

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

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

DÉFINITIONS
LA BASE

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

Définitions
La suite

- Essai ( )
  - 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!

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, réalisés 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.

- ... mais il y a des dizaines, d'autres définitions!

DÉFINITIONS
(THE ART OF SOFTWARE TESTING - G.J. MYERS)

- Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts

DÉFINITIONS

(NORME EXPÉRIMENTALE DE TERMINOLOGIE — AFCIQ)

- Le test est une technique de contrôle consistant à s'assurer, au moyen de son exécution, que le comportement d'un programme est conforme à des données préétablies.

DÉFINITIONS
(LE TEST DES LOGICIELS — XANTHAKIS ET AL)

- Partie du processus de développement
- S'intéresse aussi bien au code source qu'au comportement du logiciel
- Consiste à minimiser les chances d'apparition d'une anomalie
- Vise aussi à détecter les divers défauts responsables

DÉFINITIONS
(NORME ISO 9000-3)

- « Processus d'évaluation du logiciel pour s'assurer qu'il satisfait aux exigences spécifiées. La validation d'un produit cherche à s'assurer qu'on a construit le bon produit. »
- D'un point de vue externe ou interne

CLASSIFICATION

Divisons pour mieux comprendre

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

CLASSIFICATION DES DÉFAUTS (1/2)

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

CLASSIFICATION DES DÉFAUTS (2/2)

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

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

CLASSIFICATION DES TESTS (1/2)

- Selon leur appartenance à un essai
  - Test unitaire
  - Test d'intégration
  - Test de système
  - Test de non-régression
  - ...
- Peu utile… pourquoi?
  - et la réponse est....

Classification des tests (1/2)

- Selon leur appartenance à un essai
  - Test unitaire
  - Test d'intégration
  - Test de système
  - Test de non-régression
  - ...
- Peu utile… pourquoi?
  - *Un même test peut être utilisés dans plus d'un essai.*

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?

Classification des tests, et un peu plus

	Objets nécessaires			Exemples de techniques
Classe	**Spéc.**	**Source**	**Exec.**	
0 — TZS				Aucun test possible !
1 — TZD			✔	Certains tests aléatoires?
2 — TSS		✔		Analyse statique du flot de données, revues de code, calcul de complexité
3 — TSD		✔	✔	Analyse dynamique du flot de données
4 — TFS	✔			Analyse de cohérence des spécifications
5 — TFD	✔		✔	Test aux limites, graphes cause-effet, algorithmique qualitative...
6 — TXS	✔	✔		Exécution symbolique, preuve formelle
7 — TXD	✔	✔	✔	Tests mutationnels

CLASSIFICATION DES TESTS TECHNIQUES FONCTIONNELLES

- Ce sont les techniques
  - Classées 4 à 7
  - Nécessitant les spécifications
  - Ne nécessitant pas le code source
  - Ciblant la fonctionnalité
  - Appelées aussi techniques
    - « en boite noire »
    - opaques
    - fermées
    - externes

CLASSIFICATION DES TESTS
TECHNIQUES STRUCTURELLES

- Ce sont les techniques
  - Classées 2, 3, 6 et 7
  - Nécessitant le code source (où ce qui peut en tenir lieu, par rétro-ingénierie)
  - Utilisant la génération des cas de test par examen du code
  - Appelées aussi techniques
    - « en boite blanche »
    - translucides
    - transparentes
    - ouvertes
    - internes

CLASSIFICATION DES TESTS
TECHNIQUES STATIQUES

- Ce sont les techniques
  - Classées 2, 4 et 6
  - Reposant sur la forme « passive » du programme
  - Ne nécessitant pas le code exécutable
  - Demandant souvent des ressources importantes :
    - matérielles (model-checking)
    - humaines (techniques dont la maitrise nécessite un deuxième cycle universitaire dans l'état des formations actuelles)

CLASSIFICATION DES TESTS
TECHNIQUES DYNAMIQUES

- Ce sont les techniques
  - Classées 1, 3, 5 et 7
  - Reposant sur le comportement (l'exécution) du programme
  - Nécessitant le code exécutable

Classification des tests
Portée du cours IGL 601

- Le cours vise principalement
  - les techniques dynamiques (1, 3, 5, 7),
  - mais plus particulièrement les classes 3 et 5
- Les autres classes (statiques) sont couvertes par ailleurs :
  - revues de code (2)
    - introduites en IFT 232 et développées en IFT 785
    - *elles seront néanmoins survolées rapidement*
  - validation des spécifications (4)
    - introduite en IGL 301 et approfondies en IFT 719
  - méthodes formelles (6)
    - couvertes en IGL 501 et approfondies en IFT 734

PROPOSITION PRAGMATIQUE

- Hypothèses
- Définition d'un test
- Définition d'un jeu de tests
- Corolaires
- Principes généraux
- Description d'un essai
- Description d'un test

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 (de refus)
    - les anomalies constatées sont insuffisantes pour motiver le rejet du composant
  - critère d'erreur
    - les anomalies constatées sont des erreurs
  - etc.
- Ces critères se fondent sur les exigences ou sur la comparaison entre les résultats obtenus et les résultats attendus obtenus d'un oracle et désignés comme étalon.

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

- Incidemment, notre proposition s'appuie sur six 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?

- Vos suggestions!

Proposition pragmatique
Que faut-il pour faire un essai?

- Le code exécutable du composant
- Un banc d'essai
- Un jeu (de données) d'essai
- Une procédure d'essai
- Un cahier d'essai
- Un guide d'interprétation des résultats
- … et un ***oracle***!

Proposition pragmatique
Description 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
Description 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)

Exercice

- Énoncé
- Enseignements

Exercice
Énoncé

- On désire procéder aux essais d'un serveur FTP
- Proposer une organisation générale des essais
- Esquisser un essai de validité de téléchargement pour un serveur FTP
- Esquisser un essai de performance de téléchargement pour un serveur FTP
- Adapter ces essais pour le téléversement
- Commentaires

Exercice
Enseignements

- Qu'avons-nous montré avec l'exercice d'essai d'un serveur FTP?
  - Que des exigences précises sont essentielles pour formuler et concevoir un essai.
  - Que des objectifs clairs sont nécessaires pour éviter d'augmenter inutilement l'envergure de l'essai.
  - Qu'un essai de système peut être rédigé avant le développement.
  - Que la rédaction de la SXS contribue à l'amélioration et la vérification de la SES (voir le procédé V).

SYNTHÈSE ET RAPPELS

Pourquoi tester?

- *V et V*
- *CQFD*
- *FURPSE*
- *PESTEL*

SYNTHÈSE ET RAPPELS
V ET V

- 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

Synthèse et rappels
Propriétés et critères de projet

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

- Référence : → PMBoK

Synthèse et rappels
Propriétés et critères d'exploitation

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

- Référence : Printz → ISO 9126

Synthèse et rappels
Propriétés et critères environnementaux

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

- Référence : Printz → INCOSE