le cnam
école sciences industrielles & technologies de l'information

RAPPELS SUR LES TESTS ET LA VALIDATION

Développement piloté par les tests – Coût des tests et qualité FURPSE

le cnam
école sciences industrielles & technologies de l'information

VALIDATION VÉRIFICATION & TESTS

Introduction :

Le problème fondamental du génie logiciel : le problème de l'erreur

Le problème de l'erreur

- **COMMENT CONSTRUIRE DES LOGICIELS QUI SOIENT :**
  - ❑ ERGONOMIQUES
  - ❑ FIABLES
  - ❑ EVOLUTIFS
  - ❑ ECONOMIQUES

    ↳ GARANTISSANT LE CONTRAT DE SERVICE REQUIS PAR LES USAGERS (cf. caractéristiques FURPSE ISO/IEC 9126)

    ↳ SATISFAISANT AUX CRITERES Coût/Qualité/Fonctionnalités/Délais de réalisation (**CQFD**)

Le constat et les causes

- **PRESENCE DE NOMBREUX DÉFAUTS DANS LES LOGICIELS**
  - ➢ Cas de la navette spatiale :
    - ❑ logiciel embarqué : 0.11 par KLS (500)
    - ❑ logiciel sol : 0.40 par KLS (1400)
  - ➢ Logiciels grand public : ***plutôt 5-10 voire plus*** par KLS
- **EVOLUTIVITE DES LOGICIELS PROBLEMATIQUE (*Régression en qualité*)**
  - ➢ Cas des grands systèmes

Données statistiques

- **Sources :**
  - Classification et fréquence (Source B. Beizer, 1990)
  - Microsoft (étude M. Cusumano, R. Selby)
  - Productivité (B. Boehm)

Statistique B.Beizer

Statistique portant sur 6.877 millions de LS (avec des commentaires)
Nombre de défaillances découvertes en exploitation : 16 209
Taux d'erreurs : 2.36 Err/KLS

Cause de la défaillance : catégorie et/ou phase	% erreurs découvertes	Description et commentaires
Expression de besoin	8.12	
Spécification fonctionnelle	16.19	
Conception détaillée	25.18	Dont : - 12.82 flot de contrôle, enchaînement - 12.36 algorithmes de traitement
Données	22.44	Dont : - 11.14 valeurs initiales, duplication, etc. - 12.36 typage, accès.
Programmation proprement dite ; codage	9.88	
Intégration	8.98	Interfaces entre les modules
Appels système	1.74	
Tests erronés	2.76	
Divers	4.71	
Total	100	

- **Source : B.Beizer, *Software testing techniques* (1990)**

Microsoft

Testing technique	% of bugs detected	Description
Usage tests	15.0	Bugs détectés par usage quotidien du système
API tests	12.8	Tests de validation des interfaces
Ad hoc, other tests	8.0	Tests faits avec d'autres critères que ceux spécifiés
Apps-16 tests	7.6	
Gorilla tests	6.8	Tests de robustesse, tests aux limites
User interface tests	5.5	
Stress tests	3.8	Scénarios fonctionnels
Apps-32 tests	2.8	
NT verify tests	1.7	Tests de pré-intégration (construction d'incréments)
Applets tests	0.7	
Non NT tests	0.6	Tests faits à l'extérieur du groupe NT
Bug bash tests	0.3	
SGA tests	0.3	
RATS tests	0.2	
Unspecified techniques	33.9	
Total	100	

***Microsoft principle*: Use metric data to determine milestone completion and product release**

Statistique B.Boehm

Source : Improving software productivity, Computer September 1987 et COCOMO

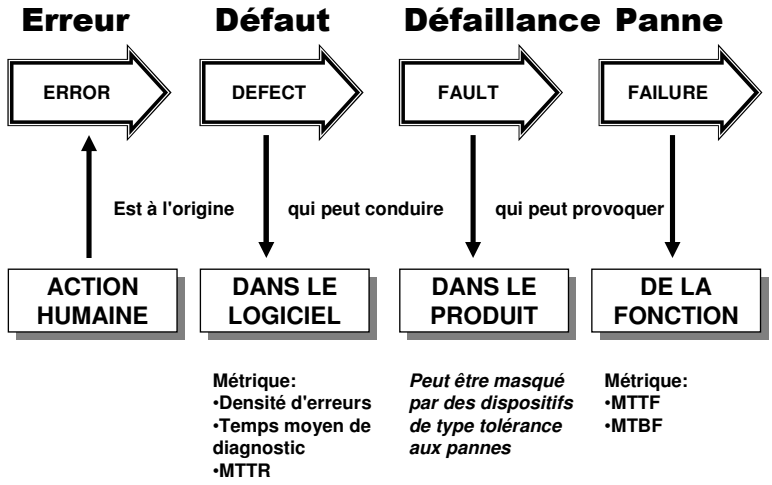
Phase	Poids en % de la phase	Dont refait suite à la découverte d'erreurs
Expression de besoin	7	1.5
Conception générale et Spécification fonctionnelle	16	4.5
Conception détaillée	24	7
Programmation proprement dite ; codage et tests unitaires	24	12
Intégration	29	19
Total	100	44

144

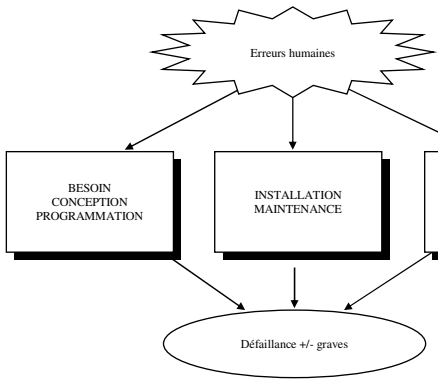
Terminologie

- **QUELQUES DEFINITIONS (IEEE STD 982 et 1044)**
  - ERROR (***Erreur***) : Human action that results in software containing a fault. E.g. omission or misinterpretation of user requirements in a software specification, and incorrect translation or omission of a requirement in the design specification.
  - DEFECT (***Défaut***) : A product anomaly; e.g. (1) omissions and imperfections found during early life cycle phases and (2) faults contained in software sufficiently mature for test or operation.
  - FAULT (***Défaillance***) : (1) An accidental condition that causes a functional unit to fail to perform its required function. (2) A manifestation of an error in software. A fault if encountered may cause a failure.
  - FAILURE (***Panne***) : The termination of the ability of a functional unit to perform its required function. A failure may be produced when a fault is encountered.

La chaîne de l'erreur (1/3)



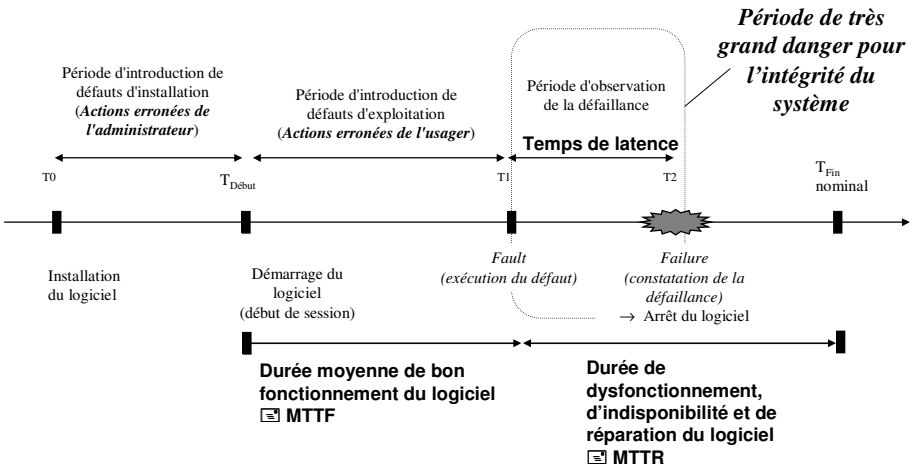
La chaîne de l'erreur (2/3)



Introduction des défauts dans les différentes phases du cycle

Erreur humaine → **Défaut dans le logiciel** → **Manifestation d'une défaillance** → **Éventualité d'une panne ± grave**

La chaîne de l'erreur (3/3)



Durée moyenne de bon fonctionnement du logiciel
MTTF

Durée de dysfonctionnement, d'indisponibilité et de réparation du logiciel
MTTR

le cnam
école sciences industrielles & technologies de l'information

VALIDATION VÉRIFICATION & TESTS

Terminologie

Quelques définitions (1/4)

- **Vérification :** confirmation par l'examen et la fourniture de preuves objectives que des exigences spécifiées ont été remplies [ISO 9000].
- **Validation :** confirmation par l'examen et la fourniture de preuves objectives que les exigences, pour un usage ou une application voulue, ont été remplies [ISO 9000].

Quelques définitions (2/4)

- **Processus de test (*testing*) :** processus consistant en toutes les activités du cycle de vie, statiques et dynamiques, concernant la planification et l'évaluation de produits logiciels, pour déterminer s'ils satisfont aux exigences, pour démontrer qu'ils sont aptes aux objectifs et pour en détecter des anomalies [CFTL].

- **Test :** un ensemble d'un ou plusieurs cas de tests [IEEE 829].

Quelques définitions (3/4)

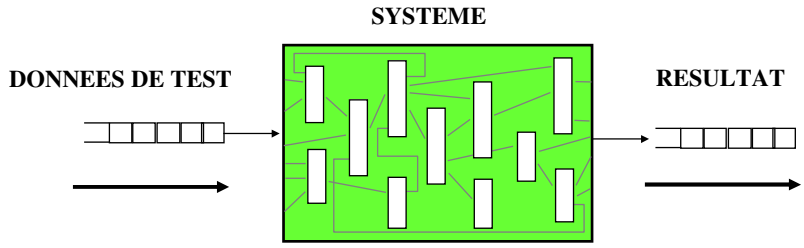
- **Cas de test :** un ensemble de valeurs d'entrée (données de tests), de pré conditions d'exécution, de résultats attendus et de post conditions d'exécution, développées pour un objectif ou une condition de tests particulière, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique [IEEE 610].
- **Objectif de test :** une raison ou un but pour la conception et l'exécution d'un test [CFTL].

Quelques définitions (4/4)

- **Condition de test :** un article ou événement d'un composant ou système qui pourrait être vérifié par un ou plusieurs cas de tests; par exemple une fonction, une transaction, un attribut qualité ou un élément de structure [CFTL].
- **Stratégie de test :** un document de haut niveau définissant, pour un programme, les niveaux de tests à exécuter et les tests dans chacun de ces niveaux (pour un ou plusieurs projets).

Tests et systèmes

- **Un système**
  - Est constitué d'éléments / composants échangeant de l'information par des **interfaces** (cf. découpage architectural)
  - Est exécutable à partir de stimuli / données provenant de l'environnement
- **Test d'un système**



Activité de tests

- **UN TEST IMPLIQUE UNE EXECUTION SUR MACHINE**
  - Du système ou d'un composant du système (nœud/feuille de l'arbre produit)
  - ❑ C'est un protocole expérimental au sens fort du terme
  - ➢ TESTS PREPARÉS A L'AVANCE
    - Cas de tests construits puis exécutés permettant de vérifier ou valider une hypothèse de bon ou mauvais fonctionnement
  - ➢ TESTS INOPINÉS
    - Exécutions défectueuses non planifiées qui révèlent un défaut de fonctionnement
- **L'ACTIVITÉ DE TEST S'INSCRIT DANS CELLE PLUS GENERALE DE RECHERCHE DES DÉFAUTS**
  - ❑ Qui inclut relectures et raisonnements sur les différents textes issus du cycle de développement au moyen de **revues** et **d'inspections**

Du point de vue intuitif

- **Activité aussi vieille que le développement**
- **Souvent négligée et peu formalisée**
- **Considérée (à tort) comme moins « noble » que le développement**
- **Coût souvent > 50% du coût total d'un logiciel**
- **Très peu enseignée**

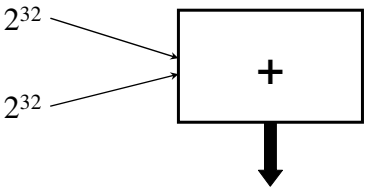
Idées fausses sur les tests (1/2)

- **Il faut éliminer tous les défauts**
  - Doit-on tester le cas où l'automobile roule à 500 km/h ?
- **L'amélioration de la fiabilité est proportionnelle au nombre de défauts corrigés**
  - Il reste un défaut dans une fonction toujours utilisée …
- **Je programme sans erreur, ce n'est pas la peine de tester …**

Idées fausses sur les tests (2/2)

- **L'amélioration de la fiabilité est proportionnelle au taux de couverture du code par les tests**
- **Évaluer la qualité d'un logiciel, c'est estimer le nombre de défauts résiduels**
- **Croire que c'est simple et facile**
- **Croire que cela n'exige ni expérience, ni savoir-faire, ni méthodes**

Juste un exemple



Faire tous les tests ➔ 0,5 milliard d'années !

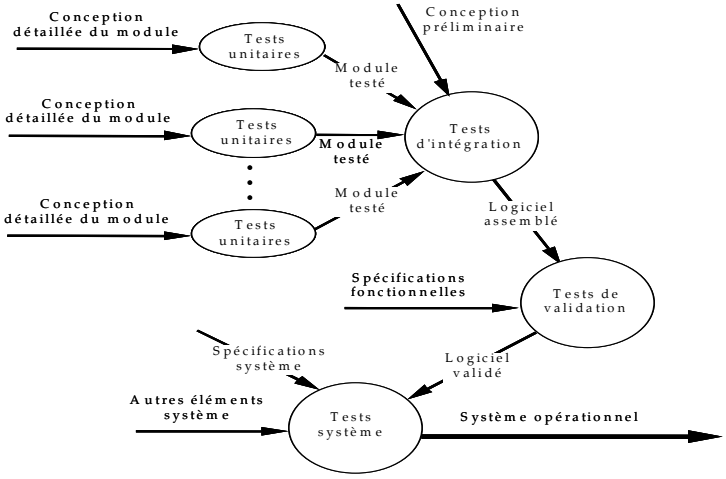
(pour toutes les valeurs)

En phase de test, nous sommes toujours face à l'explosion combinatoire

Les différentes sortes de tests

- **Tests unitaires**
  - On teste chaque fonction de chaque module (tests « boîte blanche » sur les instructions)
- **Tests d'intégration**
  - On teste les interfaces entre modules (tests « boîte blanche » sur les interfaces, tests « boîte noire » sur les instructions)
- **Tests de validation**
  - On teste les fonctionnalités du logiciel (tests « boîte noire »)

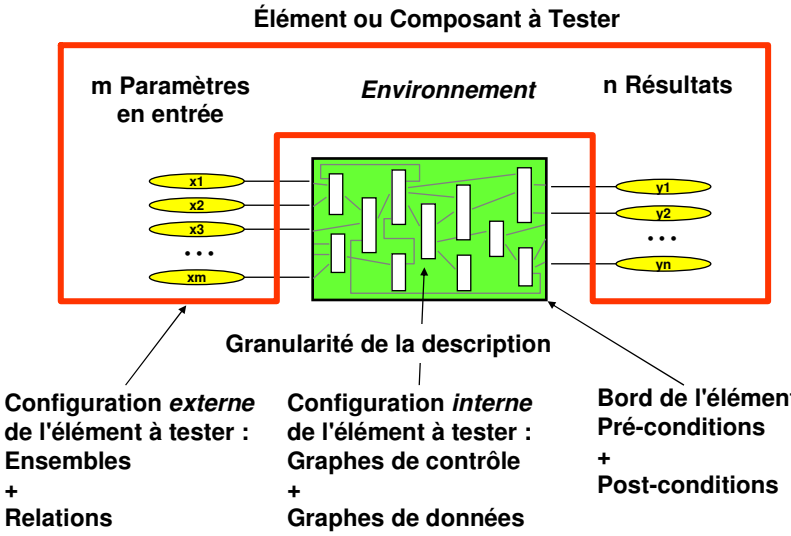
Positionnement des types de tests



Quelques termes courants

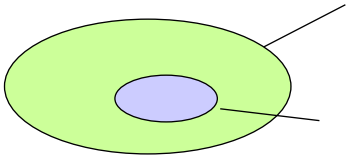
- Tester
- Preuve
- Vérification
- Validation
- Qualification
- Certification
- Mise au point (« Debugging »)
- Tests unitaires
- Tests de composants/éléments (fonction externe visible)
- Tests de produits (ensemble de fonctions)
- Tests de systèmes (ensemble de fonctions + un environnement réel)
- Tests d'intégration (pour produits et/ou systèmes)
- Tests d'acceptance ou de recette
- Tests d'installation
- Tests avec simulation
- « Field » tests (tests sur sites clients)

Nature de l'information utilisée pour les tests



Objectif de tests et ensemble générateur

➘ **ON RECHERCHE L'EXHAUSTIVITÉ DANS UNE CLASSE DE TEST DONNÉE QUI CONSTITUE L'OBJECTIF DE TEST**



Espace des cas possibles : Ecp
- ***TOUS LES CHEMINS*** **possibles induits par la combinatoire des paramètres d'entrée et le mode de construction du système**

Espace générateur : Eg
- ***CERTAINS CHEMINS*** **convenablement sélectionnés**

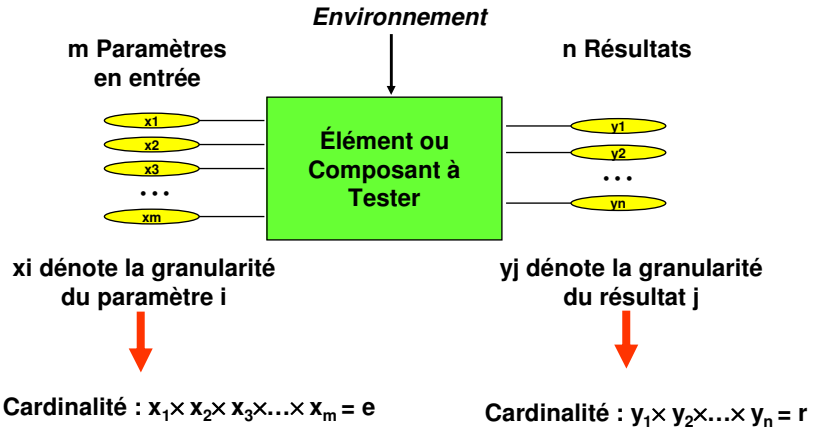
Propriété recherchée : SI Eg est couvert ALORS la probabilité d'une défaillance dans Ecp (mesurée par un MTTF) est < à une limite fixée à l'avance.

Difficulté : Faire que Eg soit à la fois :
- **Pertinent → Identification d'une classe de tests «intéressante»**
- **Consistant et Complet → par rapport à la réalité (sémantique)**

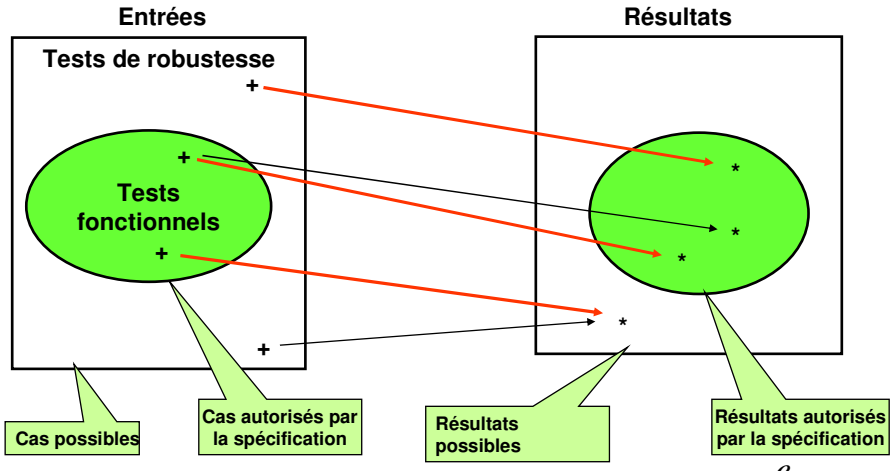
Construction de l'ensemble générateur

- **Critères de construction**
  - Différents niveaux de couvertures selon la fréquence d'emploi et/ou la criticité de l'élément
  - Conditions de «bord» (i.e. contraintes) sur les données des domaines d'entrées et/ou de sorties
    - Notion de contraintes pertinentes permettant de déterminer l'ensemble des données qui sont au voisinage du bord
  - Conditions d'observation du comportement de l'élément
    - Résultats intermédiaires intéressants
    - Consommation des ressources critiques (temps, mémoire, I/O,…)

Aperçu sur la combinatoire (1/2)



Aperçu sur la combinatoire (2/2)



Chaque → est un test, soit : (Πy_i) ** (Πx_i) tests possibles, i.e. exponentiel : r^e

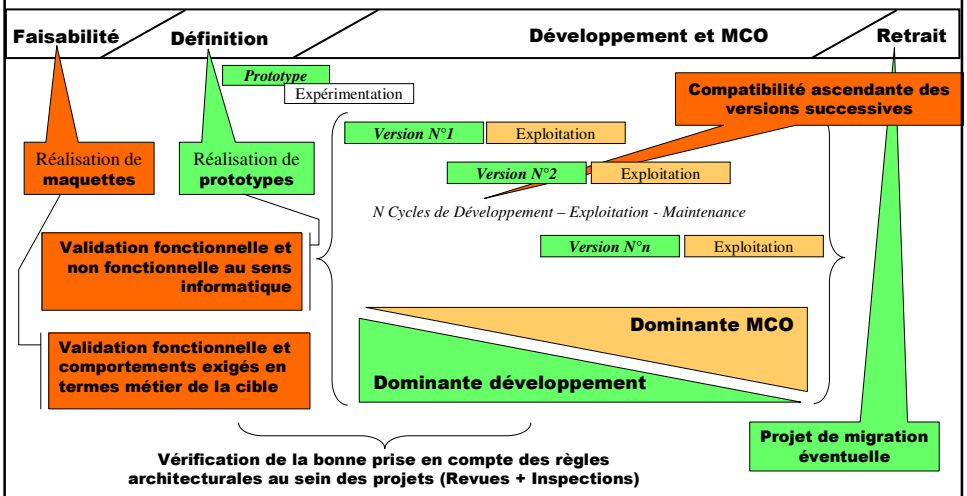
le cnam
école sciences industrielles & technologies de l'information

VALIDATION VÉRIFICATION & TESTS

Place de la VVT dans le cycle système

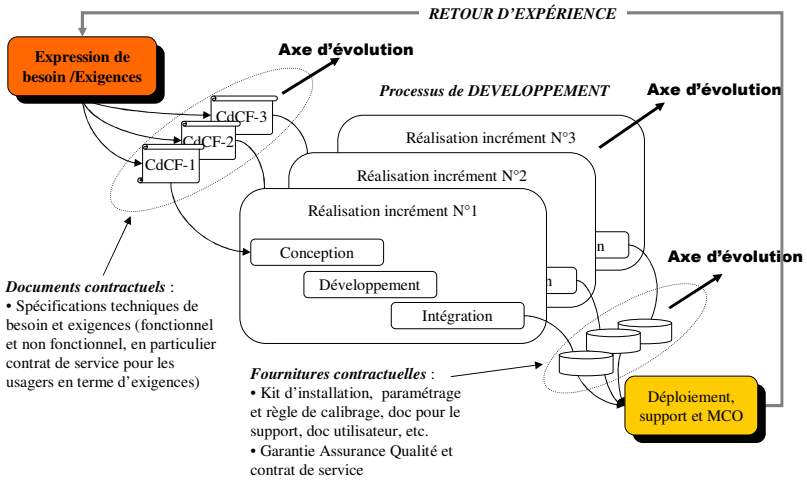
Le cycle système et le cycle de développement

Cycle de vie système

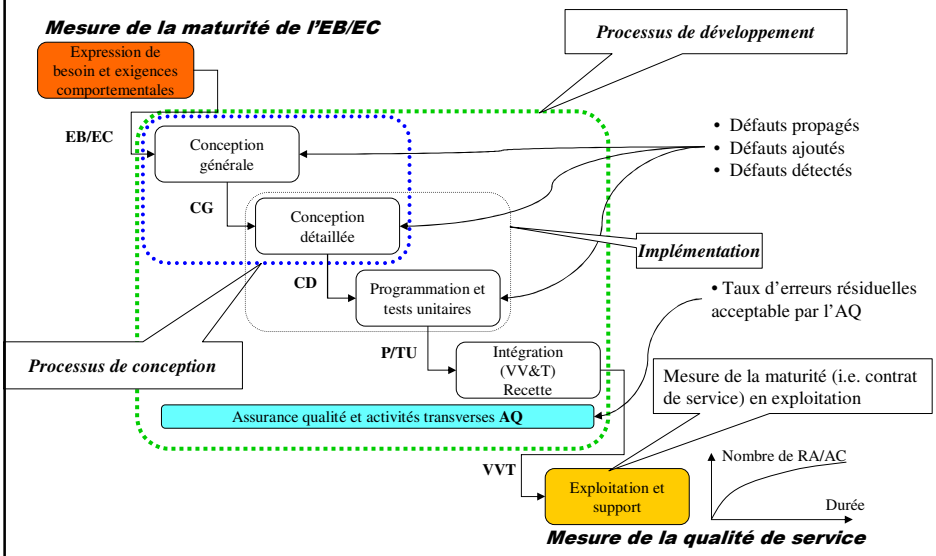


- Grande variété de types de projets selon la nature des activités et « l'age » du système
- Durée d'un cycle : > 15-20 ans, mais > 30 pour les grands systèmes technologiques

Détail du cycle de développement (1/3)

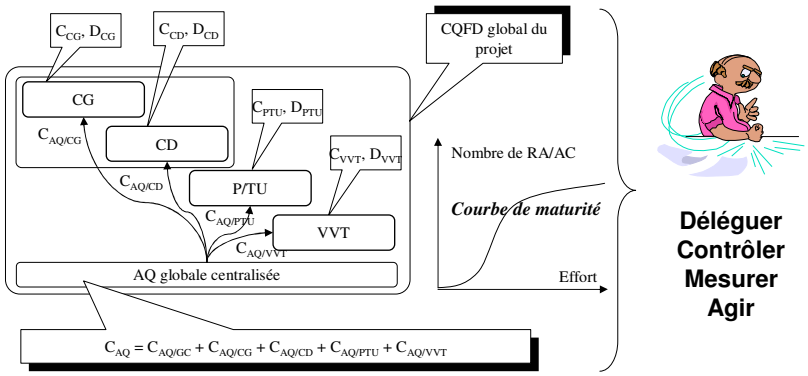


Détail du cycle de développement (2/3)



Détail du cycle de développement (3/3)

Système qualité - Assurance qualité



Déléguer
Contrôler
Mesurer
Agir

↳La recherche systématique des défauts se fait **préventivement** dans toutes les phases du cycle de développement

Des définitions complémentaires (1/2)

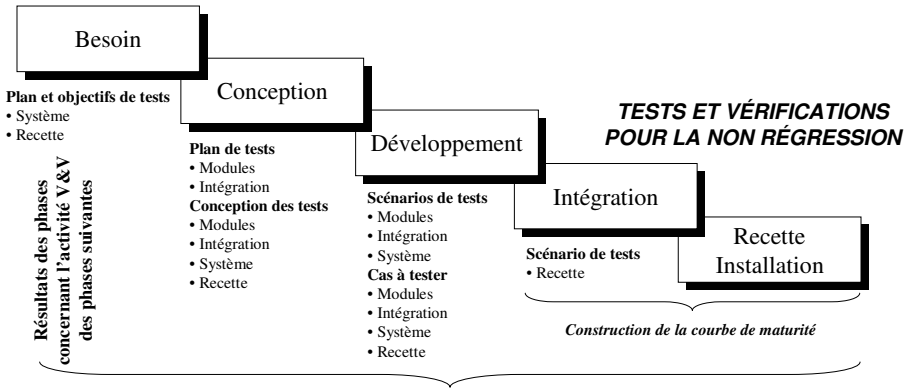
- **Plan de tests :** un document décrivant l'étendue, l'approche, les ressources et le planning des activités de test prévues. Il identifie entre autres les éléments et caractéristiques à tester, qui fera chaque tâche, le degré d'indépendance des testeurs, l'environnement de test, les techniques de conception des tests et les techniques de mesure des tests à utiliser, et tout risque nécessitant des plans de contingence. C'est un document reprenant les processus de planification des tests [IEEE 829].

Des définitions complémentaires (2/2)

- **Scénarios de tests :** une technique de conception de tests boîte noire dans laquelle les cas de tests sont conçus pour exécuter des scénarios de cas d'utilisation.
- **Conception de tests :** Conditions de tests pour l'approche détaillée d'un test et l'identification des cas de tests de haut niveau associés [d'après IEEE 829]
- **Condition de test :** un article ou événement d'un composant ou système qui pourrait être vérifié par un ou plusieurs cas de tests; p.ex. une fonction, une transaction, un attribut qualité ou un élément de structure.

Cycle de vie VVT

- **L'activité de VVT débute dès la phase EB/EC**



Pour toutes les phases : collecte des Rapports d'Anomalies (**RA**) et des Actions Correctrices (**AC**) ; traçabilité

Cf. ANSI/IEEE Std 1012 Software verification and validation plans ; Std 1059 Guide VVT

Les erreurs humaines et les sources des défauts logiciel

Origine des erreurs humaines

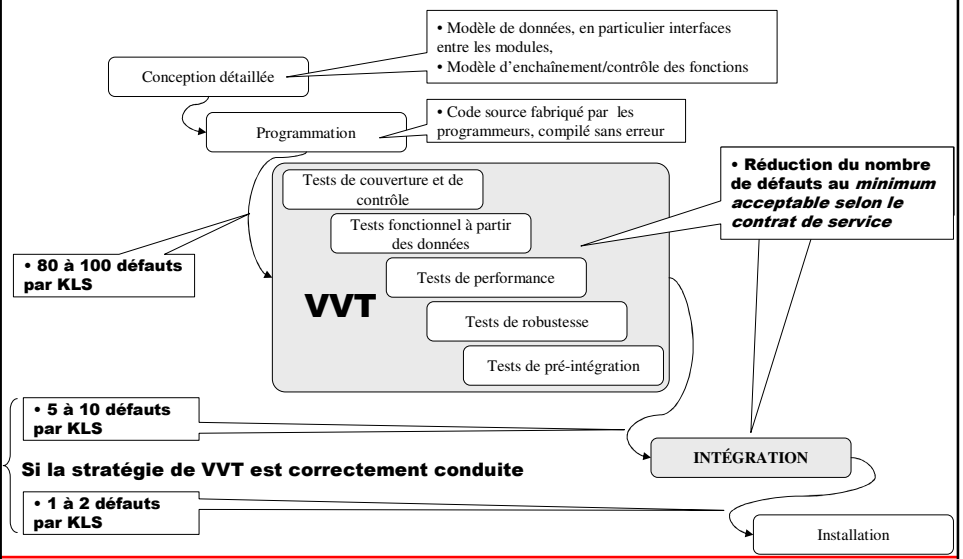
- **Incompréhension du besoin et des exigences de l'organisation cible**
  - En particulier les **caractéristiques non fonctionnelles**
- **Incompréhension de l'environnement de développement et d'exécution**
  - **Complexité des plates-formes**
- **Erreurs inhérentes à l'activité psycho cognitive**
  - Capacité intrinsèque des personnels
  - Expérience et savoir faire

cf. J. Printz, *Puissance et limites des systèmes informatisés*, Chapitre 3, chez Hermès

Erreurs humaines - Psychologie de la programmation

	Les défaillances du processus psychocognitif
D1	Erreur de perception, manque de discrimination, distinction fond/forme et/ou sémantique/syntaxe
D2	Erreur de codage/décodage
D3	Erreur de représentation et/ou symbologie non adaptée (interopérabilité entre systèmes)
D4	Représentation, compréhension et interprétation erronées de phénomènes dynamiques et/ou combinatoires (plusieurs flots d'exécution en parallèle qui interfèrent, perception du temps vrai versus temps psychologique, synchronisation) ; modèle mental erroné du phénomène.
D5	Impasse, non-exhaustivité, oubli pur et simple
D6	Illusions (dans notre cas ce sont surtout les paradoxes de type logique ; non prédicativité)
D7	Acceptation comme vraie d'une hypothèse fausse
D8	Acceptation comme fausse d'une hypothèse vraie
D9	Attribution de propriétés inutiles et/ou erronées
D10	Hypothèse superflue et/ou non appropriée (Exp. : tel événement se produit rarement alors qu'il est fréquent)
D11	Erreur de communication homme - homme, de traduction lors d'un changement de code (contre sens, faux sens, etc.)
D12	Non respect d'une procédure ou d'une règle
D13	Non prise de décision en temps voulu (logiques temporelles)
D14	Action non adaptée au contexte, action contradictoire et/ou antinomique vis à vis d'autres actions
D15	Itération, répétition inutile d'une action (i.e. propriété d'idempotence des actions)
D16	Absence d'information qui entraîne une action par défaut non adaptée (non perception d'un manque ou d'une absence de quelque chose)
D17	Erreur de raisonnement, raisonnement circulaire
D18	Défaut ou excès de généralité, abstraction mal construite, définition ambiguë (non prédicative)
D19	Confusion langage / métalangage, concept / méta-concept (mélange de types au sens logique, ou équations de dimensions incohérentes en physique)
D20	Saturation de la bande passante (Exp. : trop de décisions simultanées, interruptions continuelles)
D21	Saturation de la mémoire de travail
D22	Analogies, associations erronées et/ou non adaptées au contexte
D23	Confusion/Interférence proactive et/ou rétroactive
D24	Changement de tâche fréquent, « papillonnage » , toute perturbation qui augmente la probabilité de confusion et d'interférence

Taux moyen de défauts



Espace méthodologique et maturité de l'activité VVT

Modèle qualité FURPSE – Coûts de mise en oeuvre

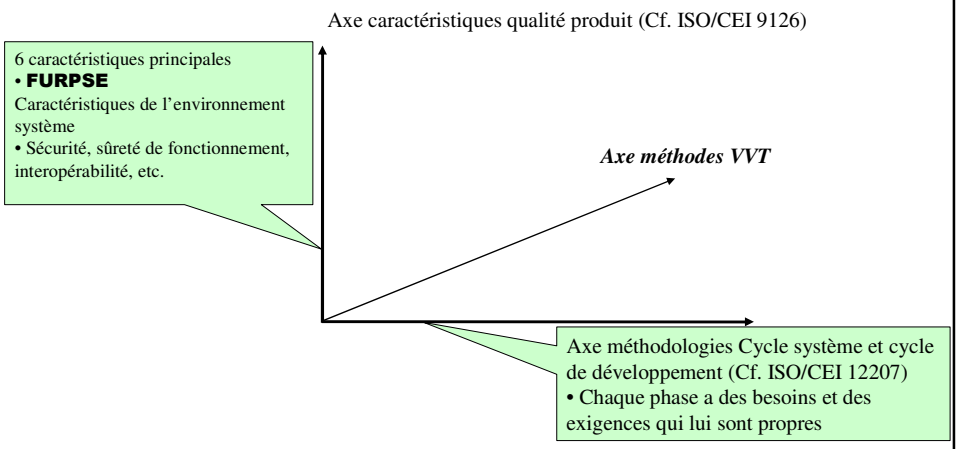
Caractéristiques qualité Internes et Externes

F	U	R	P	S	E
Functionality (Fonctionnalité)	**Usability (Facilités d'emploi)**	**Reliability (Fiabilité – Sûreté)**	**Efficiency (Performance)**	**Maintenability, Serviceability (Garantie de service, MCO)**	**Portability (Évolutivité)**
• Adéquation des fonctions • Précision et fidélité des résultats • Interopérabilité • Sécurité + • **Conformité** aux exigences fonctionnelles	Capacité et facilité de : • Compréhension • Apprentissage • Exploitation • Ergonomie IHM du point de vue métier	• Maturité • Tolérance aux pannes • Remise en état de marche	• Temps de réponse et comportement dynamique • Utilisation des ressources (mémoire, débit en transactionnel, etc.)	Capacité et facilité de : • Analyse des défaillances • Modification • Stabilité (confinement des défaillances) • Test (automaticité, non régression, etc.)	Capacité et facilité de : • Adaptation et évolution • Installation des modifications • Remplacement • Cohabitation

+ **Conformité** aux exigences non fonctionnelles (écart mesuré entre le besoin et ce qui est réalisé)

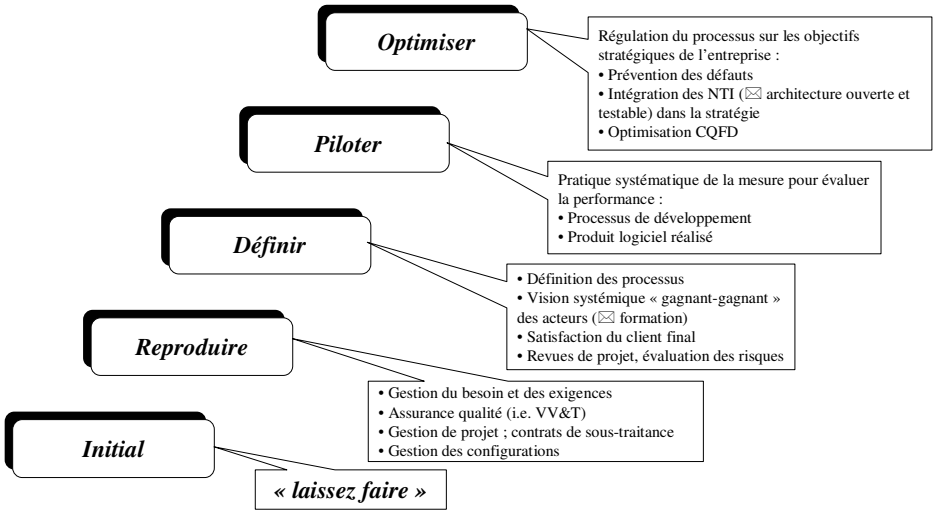
↳ La prise en compte de chacune de ces caractéristiques **implique** du code à développer ou à acquérir **et/ou** des tests (vérification et validation) à effectuer

Espace méthodologique VV&T



↳Espace de ***de choix possibles très grand*** donc risque d'inconsistance et d'incomplétude si la maturité de l'équipe est faible (cf. CMM)

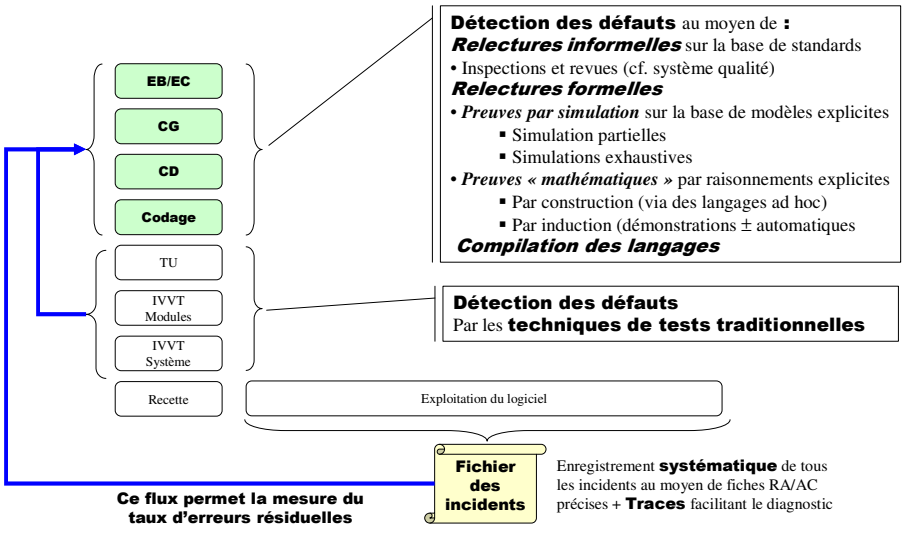
Rappel CMMI : les 5 niveaux



Les acteurs de la VV&T

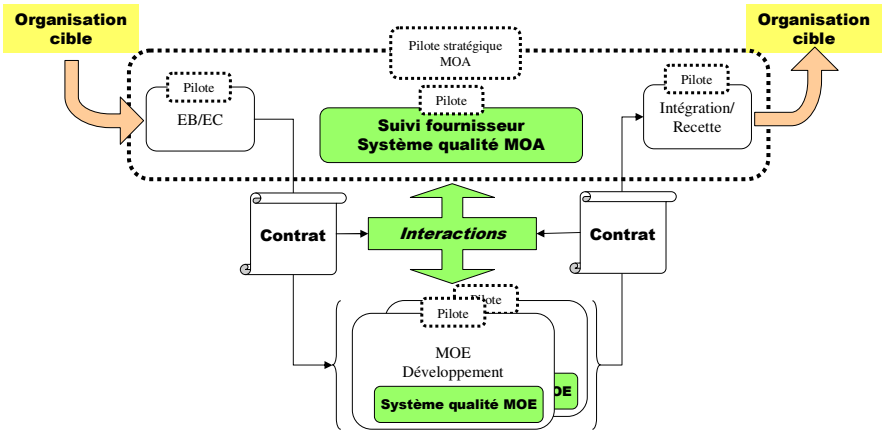
- **Parmi les acteurs, il faut distinguer ceux qui introduisent les défauts, et ceux qui recherchent les défauts pour les corriger**
  - Le chef de projet
    - Planification des tâches et assurance qualité (système qualité) – Organisation de l'équipe – Mise en œuvre de la stratégie et des méthodes
  - L'architecte du projet (au sens large = expression de besoin, spécification fonctionnelle, conception – implique la MOA et/ou le client)
    - Architecture testable
  - Les programmeurs
    - Conception détaillée et programmation – Composants logiciel {Données+Algorithmes +Contrôles} intégrables (i.e. documentés et testés)
  - Le responsable de l'intégration et son équipe
  - Le responsable de la qualification indépendante et son équipe (Assurance Qualité – Inspections et revues – Recette)
  - Le support et/ou la maintenance de 1er niveau

Place des méthodes de VVT



Stratégie coopérative MOA – MOE

Nécessité de mise en cohérence des systèmes qualité MOA ET MOE



- **La mise en œuvre d'une stratégie gagnant-gagnant dépend de la qualité de la spécification de réalisation *ET* de la méthodologie d'accompagnement**
- **Plus les référentiels sont rigoureux et explicites, meilleures sont les chances de détecter les erreurs et de corriger les défauts**

Méthodes agiles cycle de vie XP

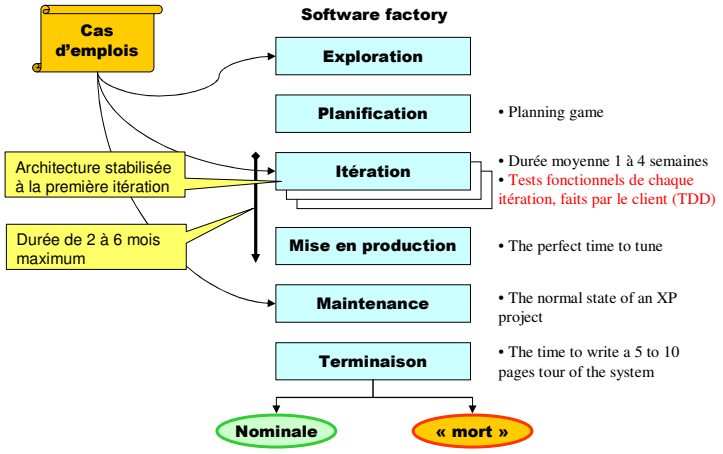
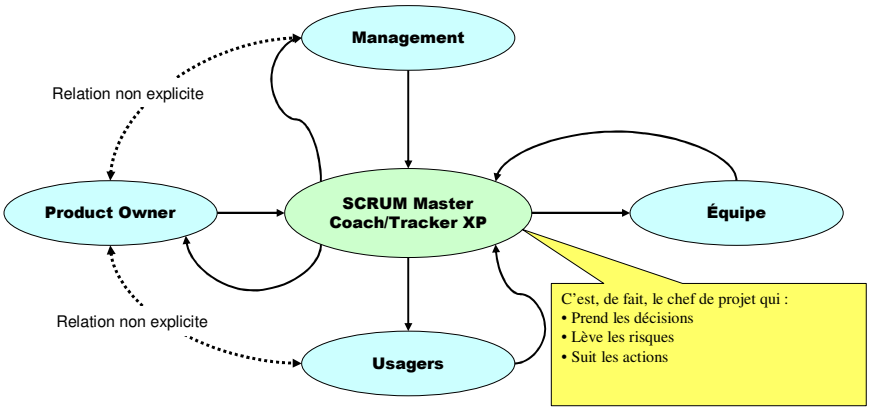


Schéma de communication entre les acteurs SCRUM / XP



Les interactions correspondent aux processus de livraison, ainsi que de fourniture des besoins et de validation (cf. TDD : test driven development)
Forte cohésion entre les équipes

La place des tests dans les 12 règles de l'XP-programming

- **R1 : Whole Team - All the contributors to an XP project – developers , business analysts, testers, etc. – work together in an open space, members of one team. The walls of this space are littered with big visible charts and other evidences of their progress.**
- **R2 : Planning Game - Planning is continuous and progressive. Every two weeks, for the next two weeks, developers estimate the cost of the candidate features, and customers select those features to be implemented based upon cost and business value.**
- **R3 : Customer Tests - As part of selecting each desired feature, the customers define automated acceptance tests to show that the feature is working.**
- **R4 : Simple Design - The team keeps the design exactly suited for the current functionality of the system. It passes all the tests, contains no duplication, expresses every thing the authors want expressed, and contains as little code as possible.**
- **R5 : Pair Programming - All production software is built by two programmers, sitting side by side, at the same machine. – 1 testeur 1 développeur, les roles sont interchangeables.**
- **R6 : Test-Driven Development - The programmers work in very short cycles, adding a failing test, then making it work.**
- **R7 : Design Improvement - Don't let the sun set on bad code. Keep the code as clean and expressive as possible.**
- **R8 : Continuous Integration - The team keeps the system fully integrated at all times.**
- **R9 : Collective Code Ownership - Any pair of programmers can improve any code at any time.**
- **R10 : Coding Standard - All the code in the system looks as if it was written by a single–very competent–individual.**
- **R11 : Metaphor - The team develops a common vision of how the program works.**
- **R12 : Sustainable Pace - The team is in it for the long term. They work hard, at a pace that can be sustained indefinitely. They conserve their energy, treating the project as a marathon rather than a sprint.**

Stratégie de tests

Stratégie de test : les objectifs (1/2)

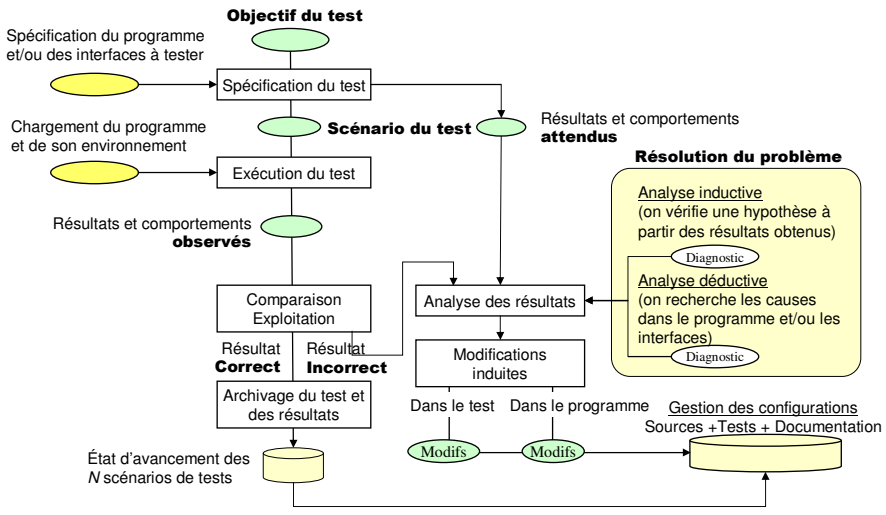
- **Un double objectif :**
  - ***Choix N°1 : Augmenter*** le MTTF
    - Réduire au maximum le taux d'erreurs résiduelles
    - Reconfigurer dynamiquement le système sur des états cohérents malgré le non-déterminisme (c'est une compensation des effets de certaines défaillances connues)
  - ***Choix N°2 : Diminuer*** le MTTR
    - Se donner les moyens d'observation des états déterministe du système (élimination systématique des erreurs reproductibles)
    - Réserver des ressources en quantité suffisante pour les tests « en ligne »

  En respectant les contraintes de Coût-Délai du projet
- **Comment répartir et organiser l'effort de VVT – Comment choisir**

Caractéristiques de l'activité de test

- **Spécifier et développer les « *bons* » tests – élaborer les résultats attendus**
- **Classer, organiser et gérer les tests (gestion de configuration)**
- **Exécuter les tests dans un environnement ad hoc – Localiser les défauts**
- **Analyser les résultats obtenus et diagnostiquer les erreurs commises**
- **Rédaction des rapports d'anomalies RA et envoi des RA aux acteurs impliqués**

Le processus de test



Objectif de l'effort de tests

- **Parmi tous les tests possibles, identifier ceux qui sont véritablement pertinents pour le système à tester**
  - Objectif et critère de test explicite
- **Pour un coût donné, trouver le maximum de défauts (rentabilité, rendement)**
  - NB : Coût = effort humain + outillages + plates-formes
- **Capitaliser ce qui est répétitif et automatiser si le coût de l'automatisation est compatible avec l'économie du système**
  - Coût de l'automatisation << Coût humain (projet et/ou coût complet)

Considération MOA, MOE, Projet

- **Pour le maître d'ouvrage qui représente le client, il faut toujours raisonner en coût complet**
  - Le MOA doit intégrer dans son analyse économique l'exploitation, le support et la maintenance, en plus du développement – **l'investissement test doit être géré comme un livrable du projet**
- **Pour le maître d'œuvre et a fortiori le chef de projet, la tendance sera de raisonner en coût projet sur une version du système**
  - La vision du MOE est bornée à celle de son projet – Son action s'arrête dès que la recette est prononcée
- **Quand faut-il arrêter les tests**
  - Mesure et/ou estimation de la maturité du point de vue de l'usager

La perception de l'usager

- **Défaillances et pannes perturbent plus ou moins fortement le fonctionnement de l'organisation cible**
- **Il faut soigneusement distinguer :**
  - Le **coût de la réparation** du point de vue du fournisseur, tel que vu par le chef de projet
  - Le **coût de l'interruption de service** tel que perçu par l'usager du système
    - Un cas extrême : ARIANE 501
  - Le rapport de ces deux coûts peut être de plusieurs ordres de grandeur
- **Il est prudent de considérer le début de l'exploitation comme la fin de la VVT projet**

Estimation de l'effort de test

- **Le nombre de défauts introduits est une fonction croissante de la taille du référentiel de programmation**
  - Documents de conception et volume de code réellement écrit par les programmeurs
- **L'effort de VVT est une fonction croissante de la taille du référentiel de programmation ET du degré d'organisation de ce référentiel, i.e. l'architecture**
  - Dans le modèle d'estimation COCOMO, la forme de cette fonction est polynomiale si l'architecture est en couche
    - Cf. J.Printz, Productivité des programmeurs, chez Hermès

Équation générale de l'effort

- **Peut-on justifier la forme de cette équation ?**

$$Effort = \underbrace{k}_{\text{Terme linéaire}} \times (KLS)^{\overbrace{1+\alpha}^{\text{Terme NON linéaire}}}$$

Facteurs d'influence

Facteurs de coût

Dépend de la puissance expressive et du « grain » sémantique du langage
+ ***Expérience des acteurs***

Facteurs d'échelle

Dépend de la complexité de l'application et de la maturité du processus de développement
• α est le ***facteur d'intégration***

Interprétation des équations COCOMO avec CQFD

C

Q

F

D

$$Effort = k \times (KLS)^{1+\alpha}$$

$$D = (0{,}5 \pm 0{,}04) \times (E_{\text{en HA}})^{0{,}3 \pm \varepsilon}$$

Rendement de l'effort VVT exprimé en termes de :

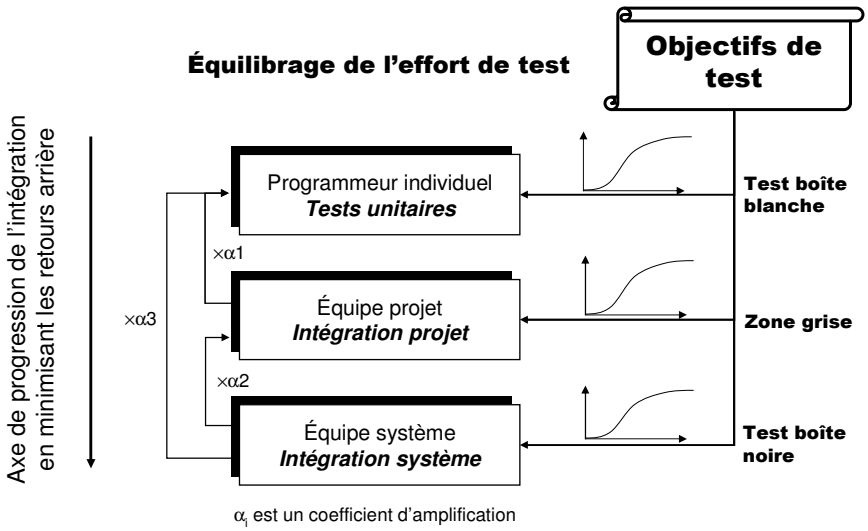
- **Taux de défauts résiduels**
- **Disponibilité de l'application ou du système**

$$Effort[Q] = C \times (k \times (KLS)^{1+\alpha})$$

Stratégie de test : les moyens (2/2)

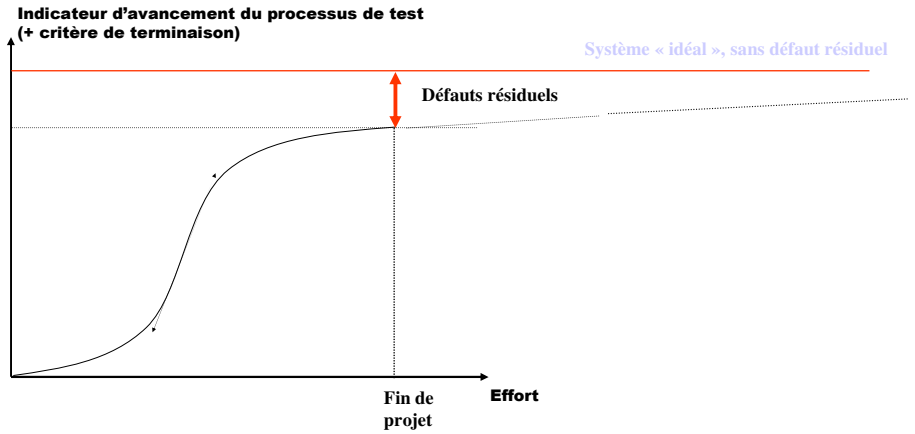
- **Architecture testable**
  - **Créer les conditions d'observation** des états du système que l'on saura interpréter et reproduire pour améliorer le diagnostic
- **Évaluation des caractéristiques non fonctionnelles** (i.e. réduire le facteur d'incertitude)
  - Exemples : Déterminer les performances et la fiabilité véritablement souhaitables ; Valider l'ergonomie avec les usagers ***REELS*** ; etc.
- **Équilibrage entre :**
  - Techniques AQ : revues, inspections, audits,
  - Tests ***Boîte Noire*** et tests ***Boîte Blanche***
  - Tests de ***robustesse*** et tests d'***innocuité/sûreté***

La vision qualité : répartition de l'effort

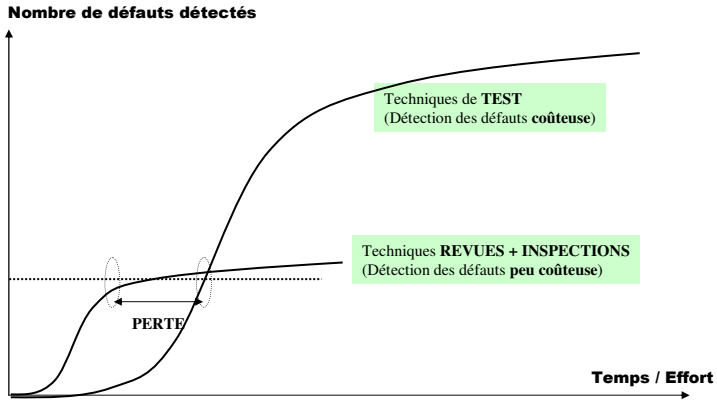


α_i est un coefficient d'amplification

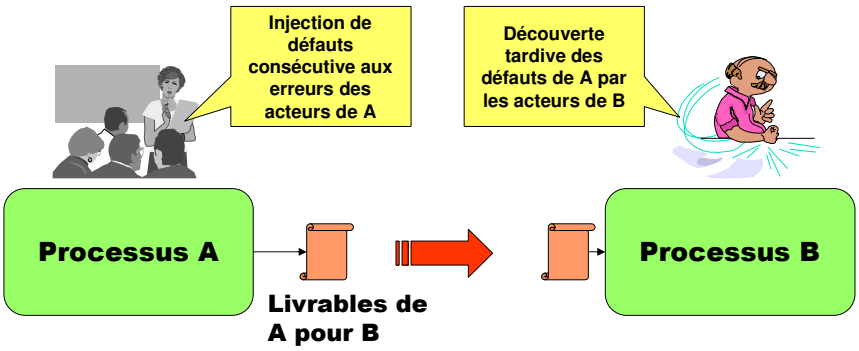
Maturité de développement d'un système



Productivité de l'effort VVT

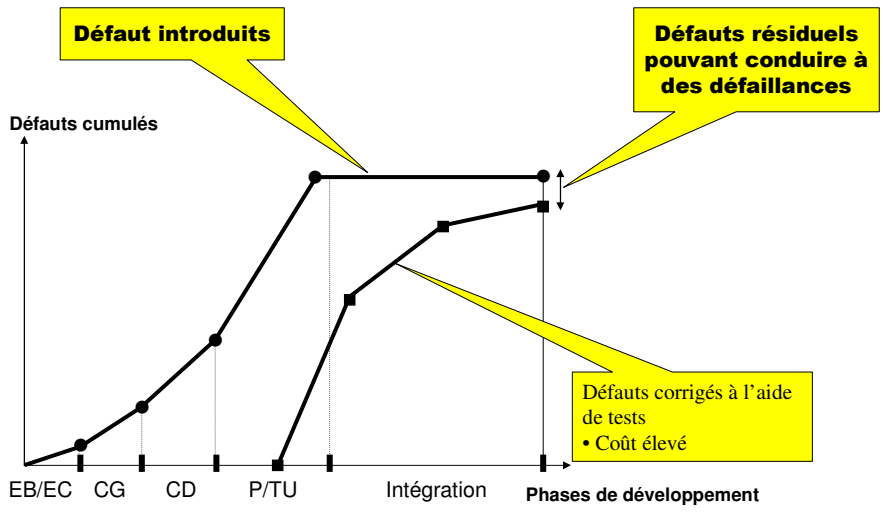


La détection précoce des défauts

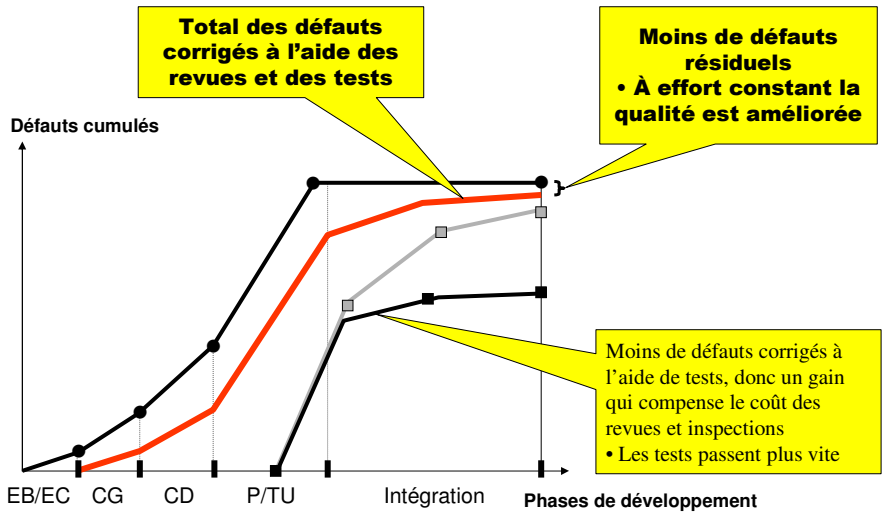


- **Remède : Faire en sorte que les acteurs de A découvrent leurs propres erreurs, ou à tout le moins permettent à B de les découvrir plus vite**
- **Qualité des livrables**

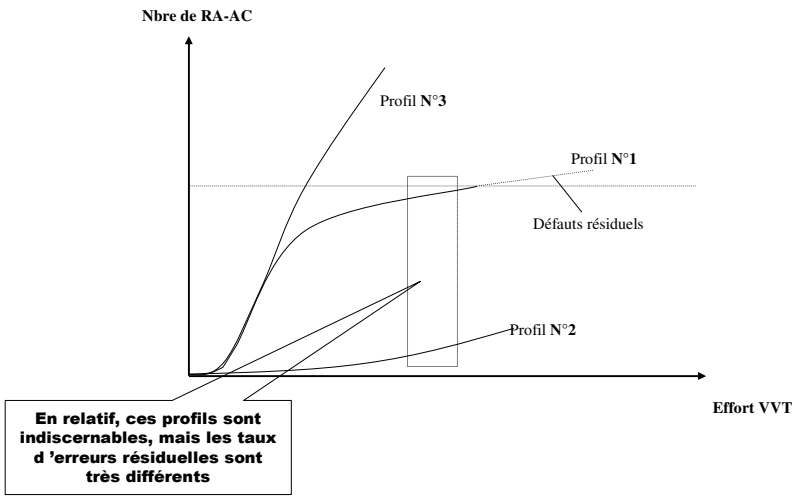
Efficacité des revues et inspections (1/2)



Efficacité des revues et inspections (2/2)



Profils de maturité qualité produit



Principes de la VVT

Quelques principes VV&T (1/3)

- **Principe N°1**
  - Tester **exhaustivement** un logiciel est généralement impossible
    - Phénomènes combinatoires et coûts exponentiels
- **Principe N°2**
  - Tester **correctement** un logiciel est une tâche difficile qui requiert intelligence et créativité
    - Choix de stratégies, critères d'arrêt + Connaissance indispensable du contexte d'emploi réel
  - Pièges :
    - croire que c'est simple et facile par rapport à la programmation jugée plus « noble »
    - croire que cela n'exige ni expérience, ni savoir-faire, ni méthodes et qu'il est inutile de planifier cette activité

Quelques principes VV&T (2/3)

- **Principe N°3**
  - L'essence de l'activité de test est la ***prévention***
    - Elle s'applique à ***toutes*** les phases du cycle
    - Il est futile de concevoir ce que l'on ne saura pas tester
      - Concept de testabilité à tous les niveaux
- **Principe N°4**
  - Le volume et la nature des tests à effectuer (i.e. l'effort VVT en terme de CQFD) doit s'apprécier en terme de ***risques*** que l'emploi du logiciel fait courir à l'organisation cible

Quelques principes VV&T (3/3)

- **Principe N°5**
  - La ***planification sérieuse*** de la VVT est indispensable à la maîtrise du projet
    - Chaque tâche du projet a sa propre VVT afin d'éviter l'effet d'avalanche lors de l'intégration
    - Piège : considérer que l'effort de test est une marge de manœuvre
- **Principe N°6**
  - L'***évaluation honnête*** de la qualité ***exige*** la présence d'un tiers de confiance (AQ logiciel) indépendant du développement
    - Piège : croire que le développement est seul juge

VV&T et QUALITÉ

- **Qualité**
  - ❑ Conformité aux exigences du contrat de service défini par l'organisation cible
  - ❑ La qualité est une notion relative (Appréciation du risque → Notion de qualité de service QoS)
- **VVT**
  - ❑ Le but des tests est de rendre la qualité « visible »
  - ❑ Le ratio de l'effort de VVT est un indicateur de la qualité du logiciel

$$r = \frac{Effort\ de\ test}{Effort\ total} \qquad Efficacité = \frac{Effort\ de\ test}{Nbre\ de\ défauts\ découverts}$$

$$Densité = \frac{Nbre\ de\ défauts\ découverts}{Nbre\ de\ lignes\ source\ (KLS)}$$

Maturité

- ❑ NIVEAU 1 : mise au point et tests ne sont pas différenciés.
- ❑ NIVEAU 2 : le but des tests est de ***montrer que le système fonctionne***.
- ❑ NIVEAU 3 : le but des tests est de ***montrer que le système ne fonctionne pas***.
- ❑ NIVEAU 4 : le but des tests est de ***réduire le risque*** de non fonctionnement tel que perçu par l'usager à une valeur compatible avec la mission du système.
- ❑ NIVEAU 5 : la ***testabilité du système est complètement intégrée*** au processus de conception.
  - ↳ ***Il est futile de « concevoir » ce que l'on ne saura pas tester***

Lois de la testabilité

- LOI 1 : toute méthode de tests laisse un ***résidu d'erreurs*** contre lesquelles la méthode adoptée est inefficace.
  - Corollaire : Le potentiel de détection des défauts d'une suite de test s'épuise → Il faut constamment renouveler les tests en changeant de point de vue (objectif et stratégie de test).
- LOI 2 : ***l'accroissement de complexité*** des systèmes dépasse très facilement le niveau de complexité que l'on sait raisonnablement valider, vérifier et tester.
- LOI 3 : code et données entretiennent des relations de dualité ; le code se transforme facilement en données. Les ***données sont une source d'erreurs*** aussi importante que le code.
- LOI 4 : la ***topologie des défauts*** passe progressivement de l'état « dense » à l'état « diffus » ; l'analyse locale devient globale. Les « *heisenbugs* » deviennent prépondérants.

Influence de la VVT sur la productivité et le rendement de l'organisation de développement

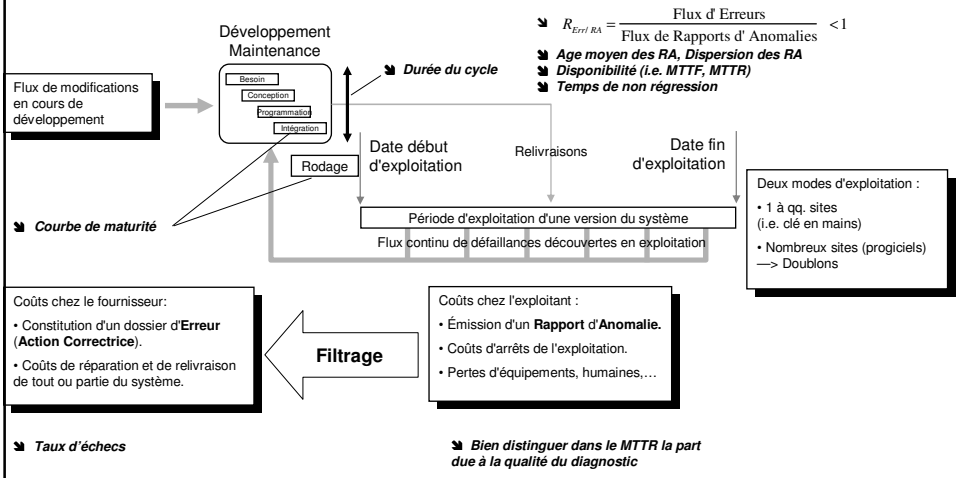
Données économiques

L'analyse des données économiques à l'aide de métriques et d'indicateurs

- **Métriques qualité**
- **Coût des corrections**
- **Courbes de maturité**
- **Facteurs d'amplification des coûts**
- **Coûts pour l'usager**

Management qualité fondée sur la métrologie des flux d'anomalies

INDICATEURS CARACTÉRISTIQUES



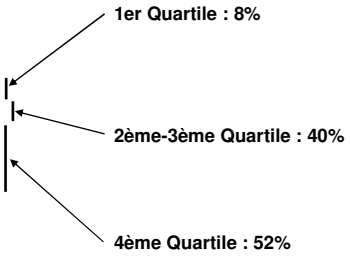
Coût moyen des corrections

	Conception	Programmation	Intégration VVT	
Coût moyen par phase selon vade-mecum	40%	20%	40%	**Taux d'erreurs acceptable ???**
Ce qui est refait selon vade-mecum	30%	50%	70%	
Coût moyen des corrections	12%	10%	28%	≈ **50%**

↘ Domaine de la ***prévention*** (amélioration de la productivité)

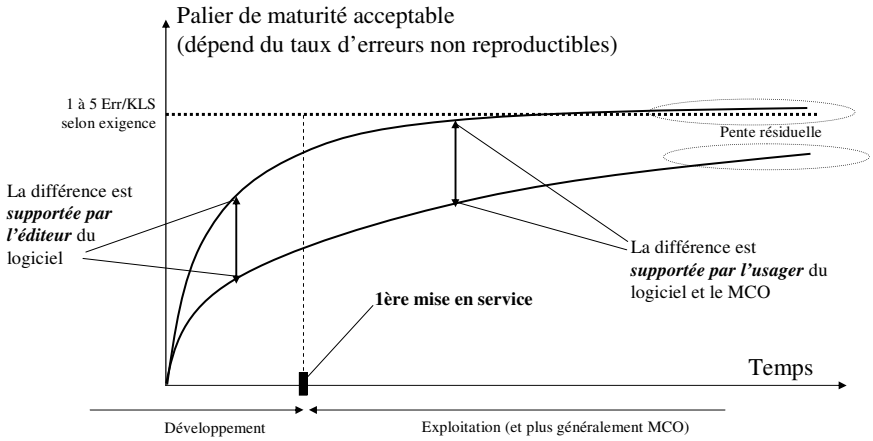
Répartition des temps de correction par défaut

%	Effort moyen par défaut (en heures)	Effort cumulé (en heures)
25%	2h	50h
50%	5h	250h
20%	10h	200h
4%	20h	80h
1%	50h	50h
100 erreurs	6.3h/err	630h



Source : Hewlett-Packard

Courbes de maturité - Transfert de coût entre les processus



Amplification du coût de correction (1/5)

- **Le coût de traitement d'une erreur dépend fortement du temps de latence (Introduction/Découverte) :**

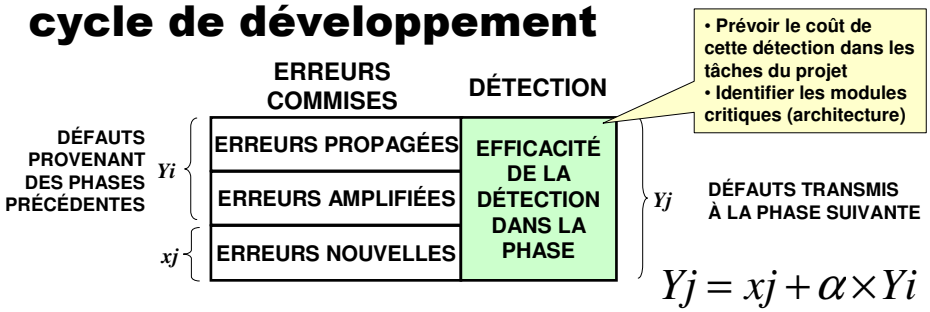
  Erreur humaine/défaut → Défaillance **reproductible**

  - ❑ **Plus le temps de latence est long, plus le coût de la correction est élevé**
  - ❑ **Toute erreur non détectée peut occasionner d'autres erreurs (amplification)**

↘ Avec l'augmentation de complexité, seule une stratégie préventive est gagnante

Amplification du coût de correction (2/5)

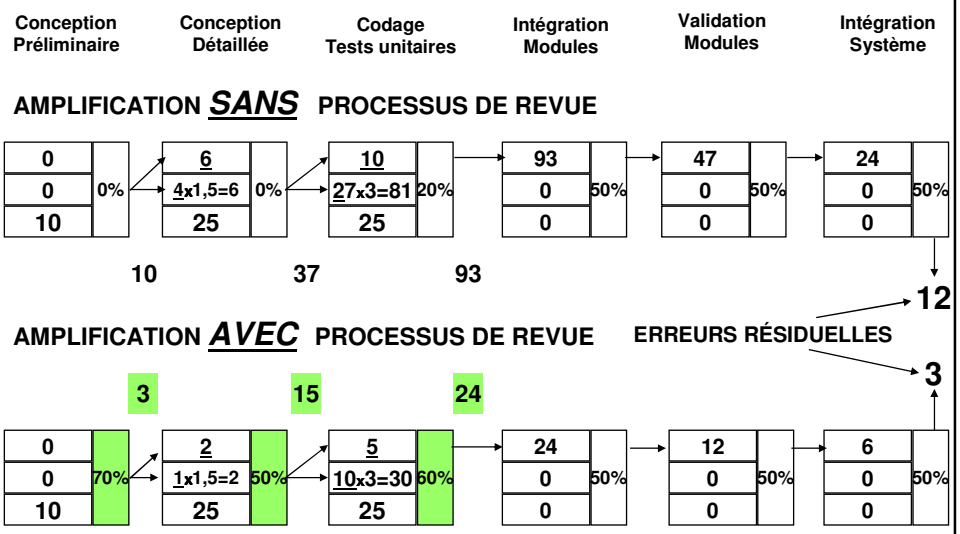
- **Modèle d'amplification par phase du cycle de développement**



COËFFICIENT D'AMPLIFICATION : #ERR ⇨ α

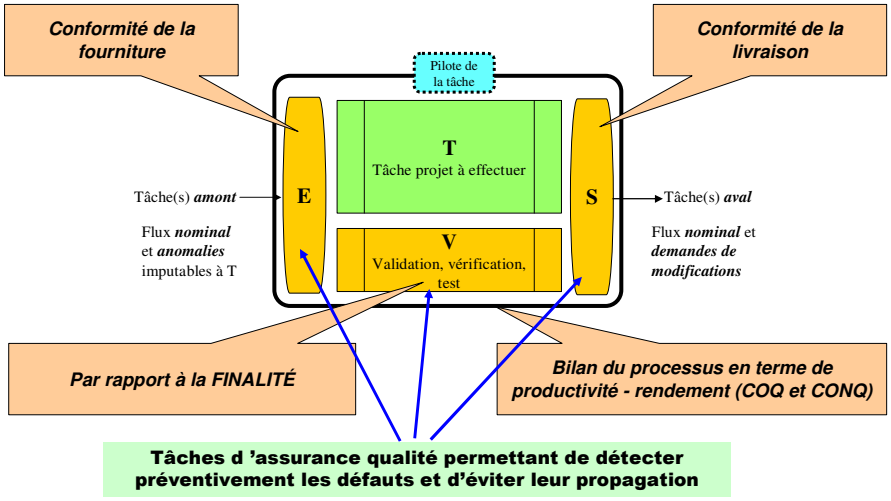
- **α dépend fortement de l'architecture (interfaces, modularité)**
- **l'efficacité de la détection dépend de la documentation, des standards, de l'organisation qualité et de l'expérience de l'équipe de revue (cf. facteurs AEXP et ACAP de COCOMO)**

Amplification du coût de correction (3/6)

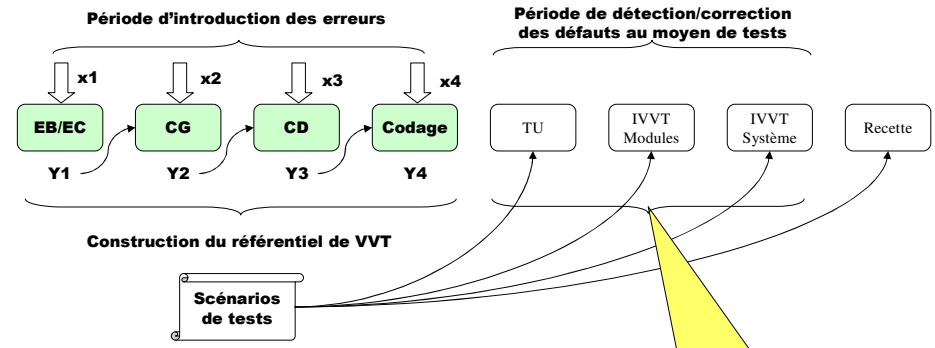


Amplification du coût de correction (3/5)

- **Application du modèle VEST**



Amplification du coût de correction (4/5)

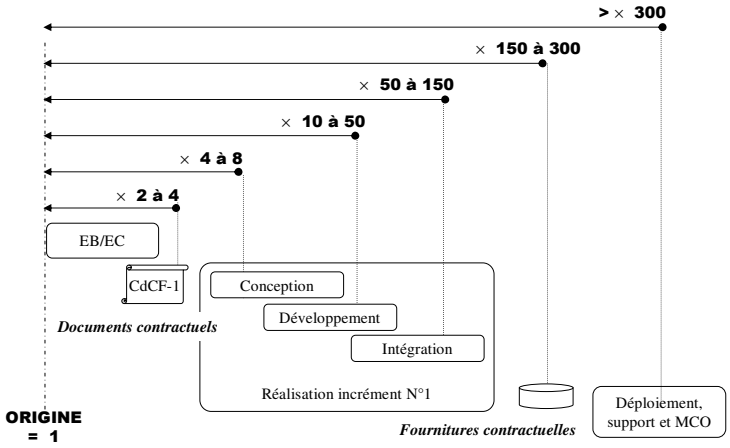


Statistique de répartition des défauts (selon B.Beizer, *Software testing techniques*)

- EB/EC : **9%**
- CG : **26%**
- CD : **52%** (dont 24% sur les données)
- Codage : **10%**
- Tests : **3%**

Potentiel de détection des scénarios de tests fonction du volume de scénarios (fonctionne comme un filtre)
- **Défaillances les + probables**
- **Absence de doublon**

Amplification du coût de correction (5/5)



Coût pour l'usager (1/2)

- **Coûts de gestion**
  - Émission de RA, installation des corrections, re-livraisons, tests de régression, etc.
- **Coûts des interruptions de service**
  - Systèmes « clé en main »
    - Possibilité d'impact « catastrophique » selon la criticité du système
      - Exemples : Infrastructures techniques (contrôle aérien, énergie, communications, réseaux bancaires, défense, etc.)
  - Progiciels
    - Existence de contournement selon le niveau de maturité
      - Systèmes d'exploitation, progiciels système (SGBD, Réseaux, etc.), progiciels applicatifs, etc.

Coût pour l'usager (2/2)

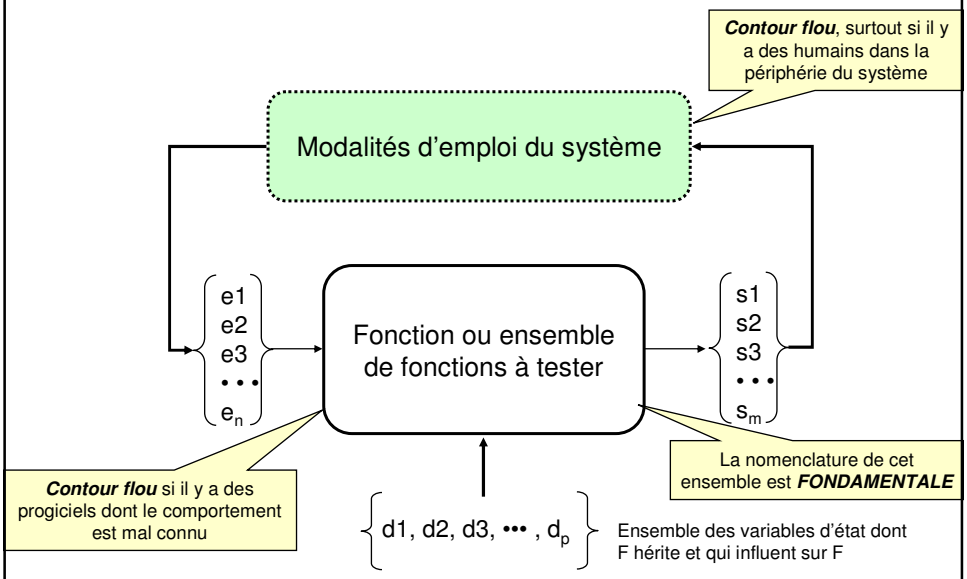
- **Impact des défaillances en terme de coût**
  - Le coût induit par une erreur est fonction :
    - FRÉQUENCE DE LA DÈFAILLANCE : liée au taux d'erreurs résiduelles, effet de parc (variété/nombre des configurations installées)
    - COÛT DE LA RÉPARATION : dépend de l'architecture du système (par exemple : avec ou sans dispositif de journalisation, avec ou sans gestion de configuration, etc.)
    - COÛT DE LA CORRECTION : très dépendant de l'automatisation des tests (cf. caractéristique de maintenabilité du logiciel)
    - COÛT DE L'INSTALLATION : dépend de l'architecture du système (par exemple : avec ou sans édition de liens dynamique, avec ou sans moniteur de machines virtuelles, etc.) + effet de parc
    - COÛT DE L'INTERRUPTION DE SERVICE : la non disponibilité du système peut induire des pertes qui doivent être comptabilisées (dommages et intérêts)

Élaboration et mise en œuvre d'une stratégie de tests

Définition

- **Stratégie de VV&T : le problème**
  - Rentabiliser REVUES et INSPECTIONS
  - Comment produire et ordonnancer les tests de façon à
    - Maximiser la probabilité de découverte d'erreurs intéressantes du point de vue de l'utilisateur
      - Satisfaire au mieux le contrat de service
    - Minimiser la composante CQFD de l'ensemble des activités de VV&T sur l'ensemble du cycle de développement
  - Garantir la qualité de l'assemblage
    - En terme de
      - Disponibilité
      - Capacity Planning et de System Management
        - Rendement du système, COST/EFFECTIVENESS du point de vue du contrat de service
- **Techniques de tests**
  - Boite noire
  - Boite blanche

Le contexte système du test



Les objectifs de tests (1/5)

- **N°1 : Couverture du domaine FONCTION**
  - ❑ Toutes les fonctions sont exécutées au moins une fois
  - ❑ Toutes les régions de code (selon granularité) sont visitées au moins une fois
- **N°2 : Couverture du domaine *données* ENTRÉES**
  - ❑ Toutes les entrées sont sollicitées au moins une fois, y compris les limites
- **N°3 : Couverture du domaine *données* SORTIE**
  - ❑ Toutes les sorties attendues sont produites au moins une fois
- **N°4 : Couverture du domaine CONTRÔLE et des ENCHAÎNEMENTS**
  - ❑ Les comportements les plus fréquents sont sollicités au moins une fois
    - ↳ Implique une excellente connaissance du contexte d'emploi du système
  - ❑ Toutes les exceptions et les messages d'erreurs sont levés au moins une fois

Les objectifs de tests (2/5)

- **FONCTION**
  - ➢ Échantillonnage raisonnable de la transformation {ei}→{sj}
    - ↳ Tests de couvertures
  - ➢ Influence des variables d'état héritées sur la transformation {ei}→{sj}
    - ↳ Prise en compte du contexte d 'exécution de la fonction
  - ➢ Échantillonnage raisonnable des cas invalides {ei et dk}
  - ➢ Valeurs particulières
  - ➢ Dépendances fonctionnelles et/ou contraintes sur les {ei et dk}
  - ➢ Erreurs spécifiques à la fonction
    - ↳ Erreurs fonctionnelles

Les objectifs de tests (3/5)

- **ENTRÉE**
  - Analyse systématique de tous les types possibles en entrée
    - MCD, états logiques et/ou physiques des données en entrée
  - Analyse systématique de toutes les bornes des domaines de validité des données
    - 3 valeurs par borne : =, > et < (y compris les combinaisons)
  - Analyse systématique de toutes les situations conduisant à un overflow
    - Dépassement de capacité des ressources allouées à la fonction compte tenu des données en entrée
  - Impact des interruptions possibles
    - File d 'attente d 'événements pris en compte par la fonction ; saturation des files, etc.
    - Effet « cache » ; saturation du cache, etc.
  - Robustesse (Données incomplètes et/ou fausses)
    - Innocuité : la réponse fausse ne dégrade pas l'environnement de F

Les objectifs de tests (4/5)

- **SORTIE**
  - Analyse systématique de tous les types possibles en sortie
    - MCD, états logiques et/ou physiques des données en SORTIE
  - Édition de tous les diagnostics, de tous les message d'erreurs
  - Analyse systématique de tous les types possibles de rapports et/ou fichiers créés par la fonction
    - Y compris les états incomplets et/ou faux
  - Non altération des données qui ne font que transiter par la fonction
    - Non propagation des contaminations (confinement et latence)
  - Analyse systématique de tous les modes de terminaison possibles et des cas de reprises qui leur sont associés
    - Cas des arrêts sur événements inopinés et/ou générés par l'opérateur ou l'environnement

Les objectifs de tests (5/5)

- **CONTRÔLE et ENCHAINEMENTS**
  - Analyse systématique de scénarios d'emploi de la fonction jugés significatifs pour le contrat de service
    - Tests de chemins
  - Analyse raisonnable de scénarios catastrophes
    - Prévention des risques décrits dans le contrat de service
    - Non propagation des défaillances
  - Analyse raisonnable de combinatoires {ei et dk} du point de vue contrôle
    - Chemins anormaux devant conduire à une erreur, latence, etc.
    - Erreurs dues aux conditions d'entrée
  - Analyse raisonnable de combinatoires {sj et dk} du point de vue contrôle
    - Erreurs dues à l'environnement (matérialisée par dk)

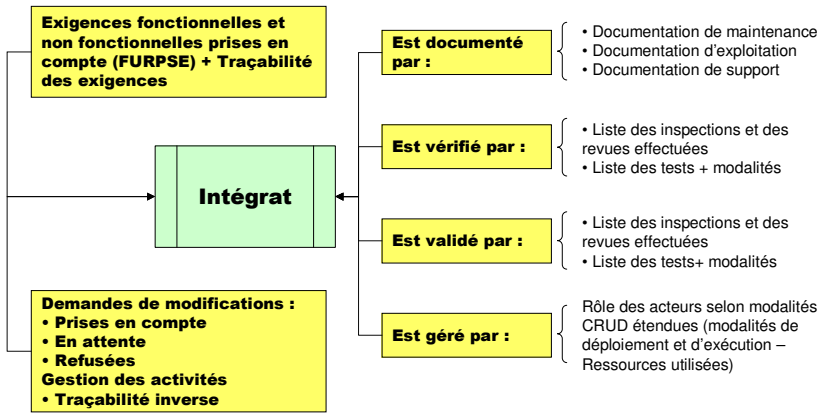
Logique d'intégration et ingénierie système (1/2)

- **Identifier le(s) chemin(s) d'intégration optimum de l'application selon le critère CQFD**
  - Maturité des composants élémentaires et des interfaces critiques (à intégrer le plus tôt possible)
    - Les plus fréquemment utilisés du point de vue du contrat de service
  - Identification des chaînes fonctionnelles longues
    - i.e. ossature du système
  - Croissance incrémentale par ajouts successifs pour les composants basiques des chaînes longues
- **Construire des « intégrats » permettant d'éviter le recours trop systématique à la simulation**
  - équilibrage entre simulation vs contexte et environnement de tests
    - La simulation est remplacée par un contexte spécifique ad hoc (jeté en fin de test)

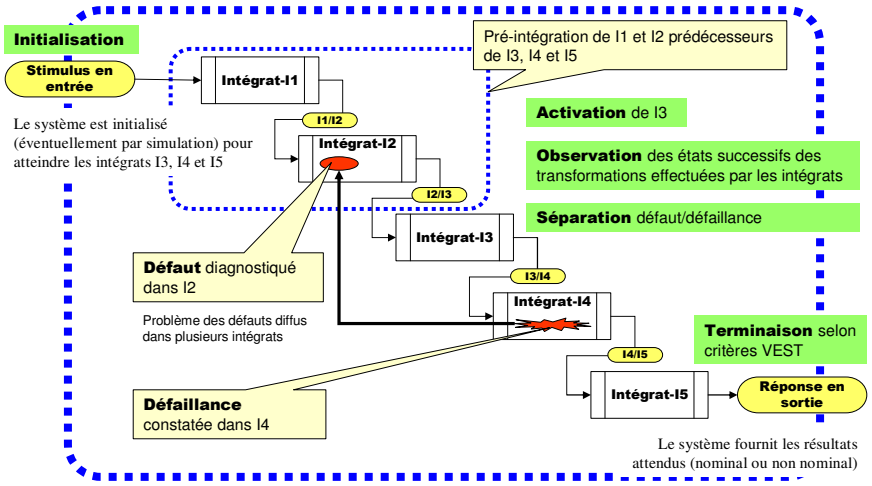
Logique d'intégration et ingénierie système (2/2)

- **Répertorier les dépendances fonctionnelles de l'application vis à vis des COTS et de l'environnement système**
  - ➢Encapsulation systématique et/ou homologation préalable des COTS
  - ➢IHM interactive et/ou disposant d 'un mode commande
    - ↳Prise en compte des coûts de non régression
- **Mise en œuvre des 4 principes d'intégration**
  - ➢*Activation*, *Séparation*, *Observation*, *Terminaison*

Attributs indispensables d'un intégrat



Stratégie d'intégration
Construction d'une chaîne d'intégrats satisfaisant VEST



Stratégie de l'effort de test

- **Construction de la matrice du jeu**

Actions possibles

Situations possibles

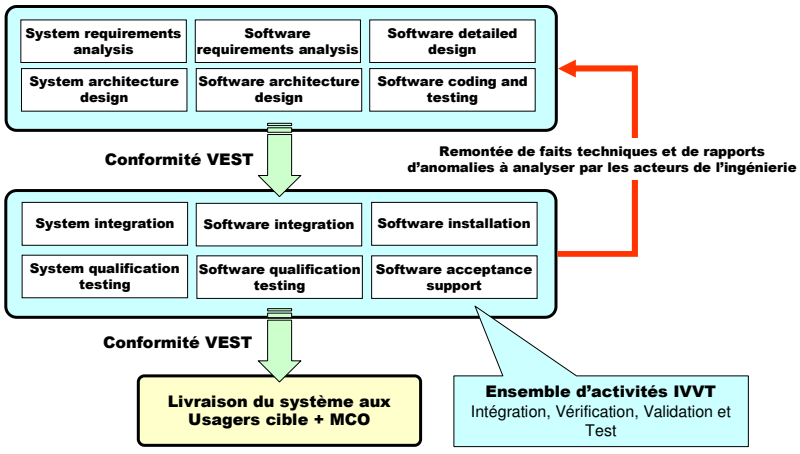
	A1	A2	A3	• • •	A_m
S1	✓		✓		
S2		✓			
• • •				✓	✓
S_n	✓			✓	

➘ L'analyse des situations se fait par rapport au but fixé dans le contrat de service, en fonction de l'évolution des courbes de maturité

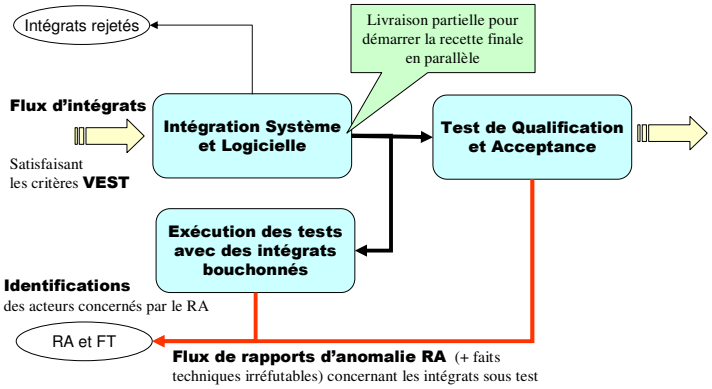
Aperçu sur l'intégration des systèmes informatisés

Aspect logiciel de l'intégration

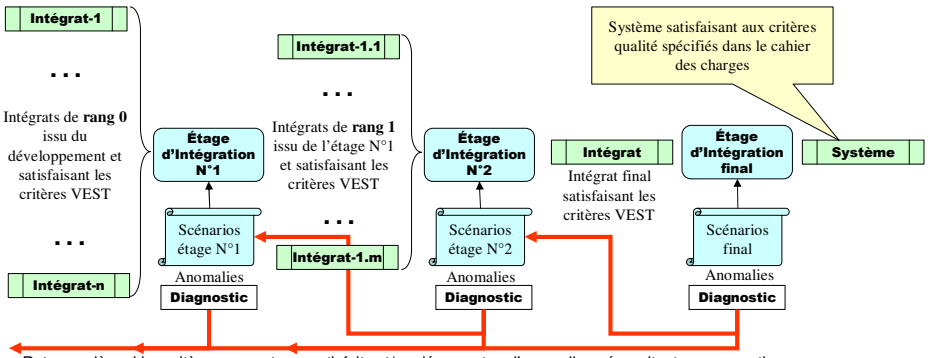
Interactions dans le processus d'intégration (Norme ISO 12207)



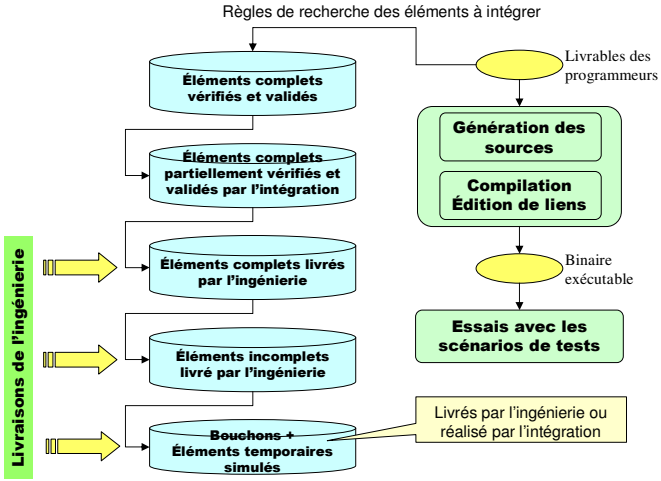
Le procédé de construction



La machine à intégrer (chaîne de montage) – Assemblage des intégrats



Construction progressive de la configuration livrée



L'organisation de la bibliothèque est en relation duale avec l'organisation du projet ; c'est un aspect fondamental de l'ingénierie projet

Bibliographie

- **Glossaire CFTL/ISTQB des termes utilisés en tests de logiciels**
- **IEEE SW standards collection**
- **J. Printz, JF. Peyre, Pratique des tests logiciels, 2009**