Présentation des UVs

LO22 : ingénierie des logiciels sûrs de fonctionnement

AI20 : sûreté de fonctionnement des systèmes informatiques

Walter SCHÖN
Mohamed SALLAK

Sujets essentiels traités dans l'UV

Trois sujets dans LO22 dont deux dans AI20

- Sûreté de Fonctionnement des systèmes informatiques : LO22 et AI20
- Techniques de Vérification et Validation (V&V) : analyse statique (métriques…) et dynamique (tests…) : LO22 seulement
- Génie logiciel basé sur le formel (méthode B) : machines abstraites, invariants, preuves : LO22 et AI20

Sujets liés à l'ingéniérie des logiciels traités dans d'autres UVs

- UML (LO21)
- Gestion d'un projet logiciel (LO23)

Organisation

- Les cours sont communs LO22/AI20, les TDs et TPs sont spécifiques LO22 et AI20 (bien qu'identiques lors des périodes communes).
- La V&V (enseignée en AI03 pour les apprentis) est traitée en milieu de semestre lors de la période entreprise des apprentis.
- 2h de cours et 2h de TD par semaine, 2h de TP une semaine sur 2 (alternance semaine A / semaine B pour LO22, planning spécifique pour les apprentis)
- Début des TDs dès la deuxième semaine, TPs selon planning transmis
- AI20 est donc à 5 crédits
- Moodles sur l'espace LO22 (les AI20 y auront accès)

Evaluation

LO22

- 1 Median à la mi-Avril 30%
- 1 Note de TPs 20%
- 1 Examen final 50%

AI20

- Une note de TPs 30%
- Un examen final 70%

Même examen final pour AI20 et LO22 (mais une partie en moins et donc 40mn de moins pour AI20)

Introduction : sûreté de fonctionnement et informatique

Sûreté de fonctionnement (SdF) : propriété qui permet à ses utilisateurs de placer une *confiance* justifiée dans le service délivré par un système.

Informatique : de plus en plus présente dans la vie quotidienne (de plus en plus de « smarts bidules » contenant un processeur, « nous sommes passés des puces aux pucerons » selon l'académicien Gérard Berry) => source de problèmes de plus en plus fréquents qui peuvent nuire à la confiance qu'on leur accorde.

La sûreté de fonctionnement des systèmes informatiques est devenu un enjeu stratégique essentiel de nombreux secteurs d'activité.

Introduction : sûreté de fonctionnement et informatique

Sûreté de fonctionnement informatique : deux volets complémentaires :

Traitement des défaillances aléatoires du support matériel : semblable à la SdF de tout système technique avec quelques spécificités informatiques (possibilité de redondances informationnelles…)

Problèmes liés aux défaillances systématiques liés à des erreurs de conception : particulièrement cruciales en informatique où le bug logiciel est une plaie des temps modernes ! => Le génie logiciel est donc très lié à la sûreté de fonctionnement informatique !

Enjeux du génie logiciel

Génie logiciel

- **Génie logiciel = Software Engineering /= Software Genius**

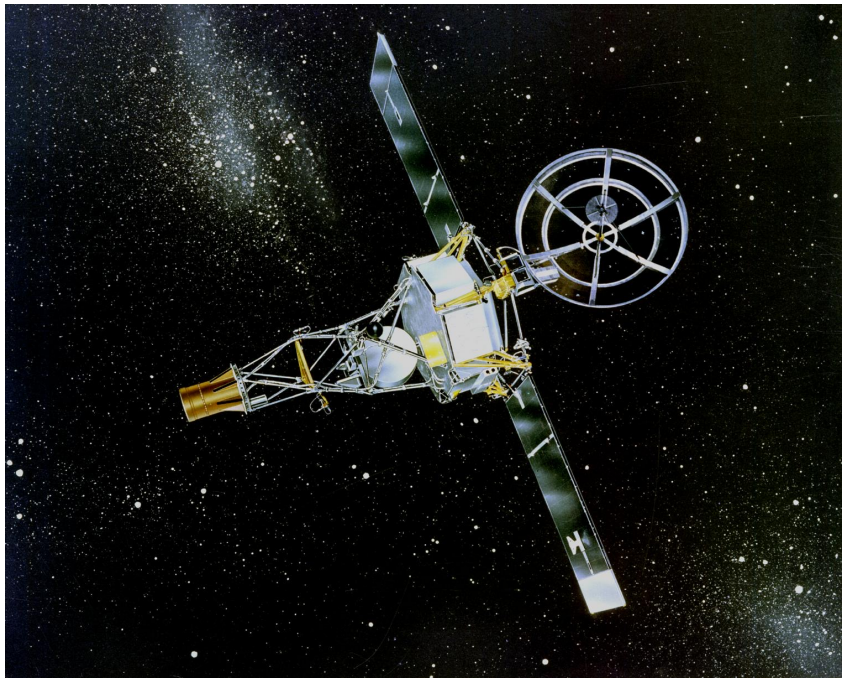
There is no need to be a software genius to do software engineering !

Genèse du génie logiciel

- Avant 1968 : mythe (encore répandu…) du «Sofware genius» : Programmation = activité créative et ludique où l'intelligence est le facteur déterminant.
- =>Crise du logiciel : non maîtrise des coûts et délais de développement, graves incidents dans certains programmes, abandon pur et simple de certains autres.

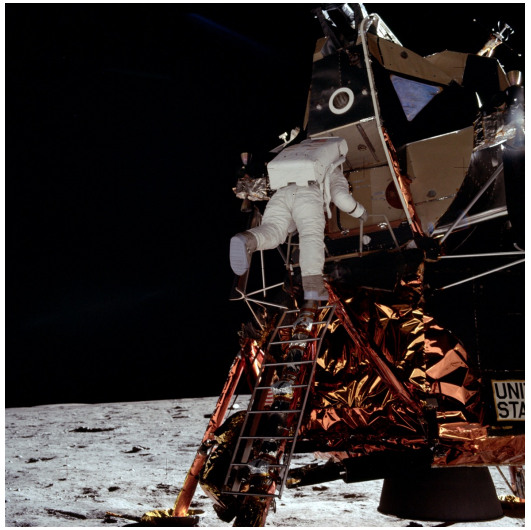
Quelques bugs célèbres

- 22 Juillet 1962 : Le lanceur de la sonde Mariner 2 part en lacet et doit être détruit 5 mn après la mise à feu
- Causes : erreur typographique dans le programme (la légende parle d'un trait d'union ou d'un point, la réalité semble un peu plus complexe)



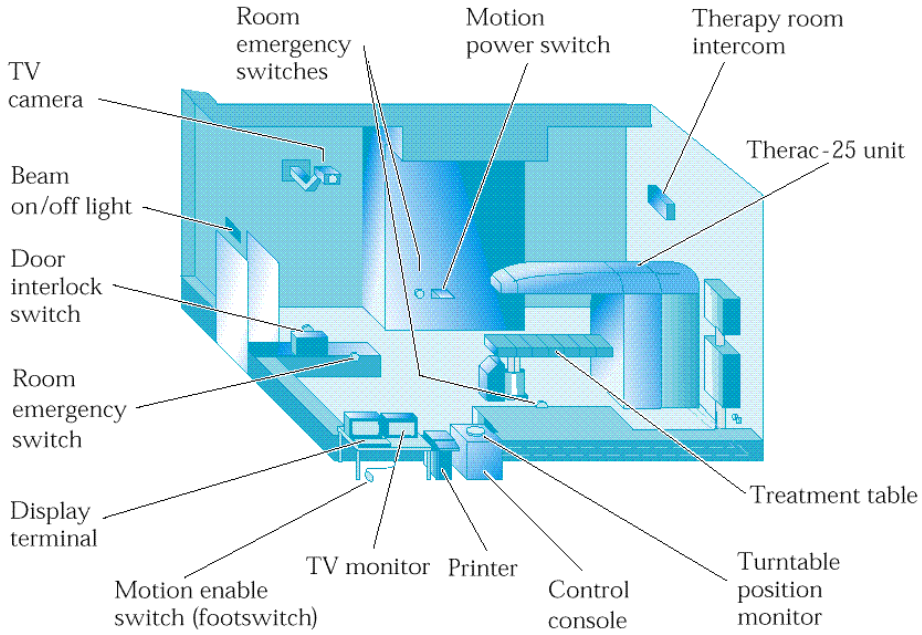
Quelques bugs célèbres

- 20 Juillet 1969 : L'approche finale vers la lune du module d'alunissage d'Appolo 11 ne se déroule pas comme prévu : à 500m du sol lunaire le calculateur de bord « plante » (alarme 1201), Neil Armstrong pose le LEM en manuel…
- Causes : mauvaises gestions d'alarmes répétitives, sur le radar d'approche, ayant conduit à la saturation du calculateur…



Quelques bugs célèbres

- 1985 : Au Canada un appareil de radiothérapie, le Therac 25 provoque plusieurs irradiations mortelles.
- Causes : overflow sur un compteur dans certaines conditions particulières (mémorisation d'une séquence de commandes par l'opérateur)



Quelques bugs célèbres

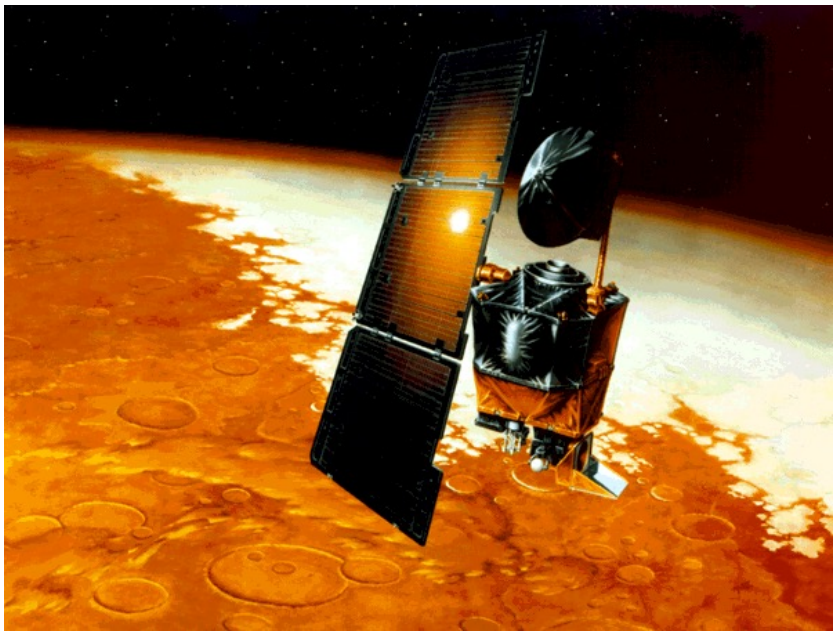
- 25 Février 1991 : un missile anti-missiles US Patriot échoue dans la destruction d'un missile Scud Irakien à Dharan (Arabie Saoudite) : 28 soldats tués
- Cause : bug logiciel provoquant une erreur qui augmentait avec le temps écoulé depuis le dernier reset du système : avait atteint l'équivalent de 600m au bout de 100h de fonctionnement…

Quelques bugs célèbres

- 4 Juin 1996 le vol inaugural d'Ariane 5 (avec des vrais satellites à bord) est un échec. Le lanceur dévie et doit être détruit : le feu d'artifice le plus cher de l'histoire : 500M$
- Cause : overflow de variable (détails à la fin de ce cours…)

Quelques bugs célèbres

- 23 Septembre 1999 la sonde spatiale Mars Climate Orbiter est pulvérisée dans l'atmosphère Martienne alors qu'elle devait se mettre en orbite
- Cause : une partie du logiciel (celui du fournisseur de moteurs) calculait les poussées en livres anglaises, tout le reste étant en système métrique…



Debugging

- As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought.
- Debugging had to be discovered.
- I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.
- Maurice Wilkes, 1949

Genèse du génie logiciel

- Du 7 au 11 Octobre 1968 : conférence de Garmisch-Partenkirchen parrainée par l'OTAN :
- «Software Engineering» : Programmation = activité de production industrielle où la rigueur du processus mis en œuvre est le facteur déterminant (idée choquante pour l'époque !)

Enjeux du génie logiciel
Le nécessaire travail d'équipe

- Progrès des matériels (processeurs, mémoires)

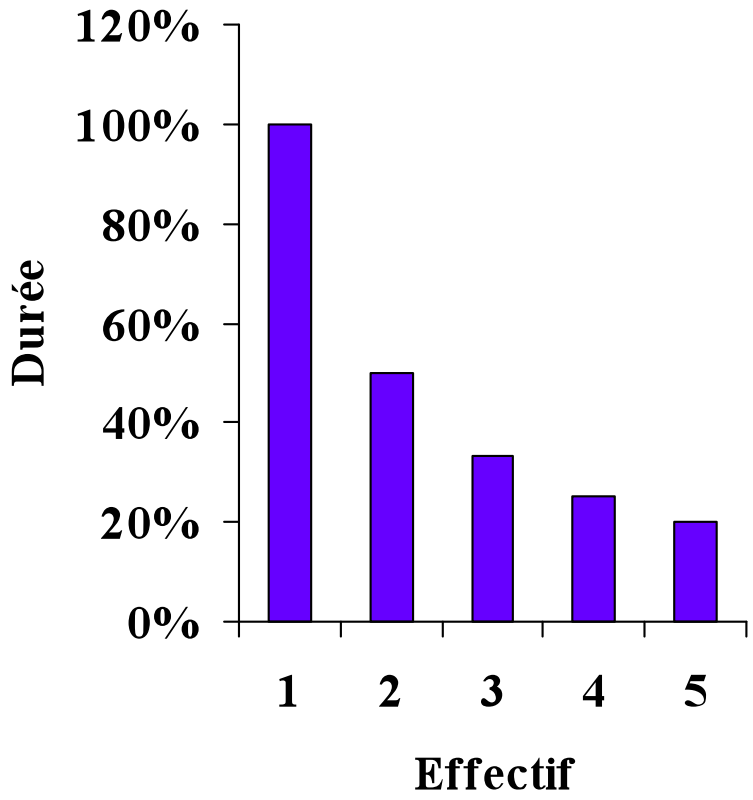
  => logiciels de plus en plus complexes (plusieurs millions de lignes de code).

  => Course permanente contre l'obsolescence.

  => Logiciels conçus et développés par des *équipes* de plus en plus importantes.

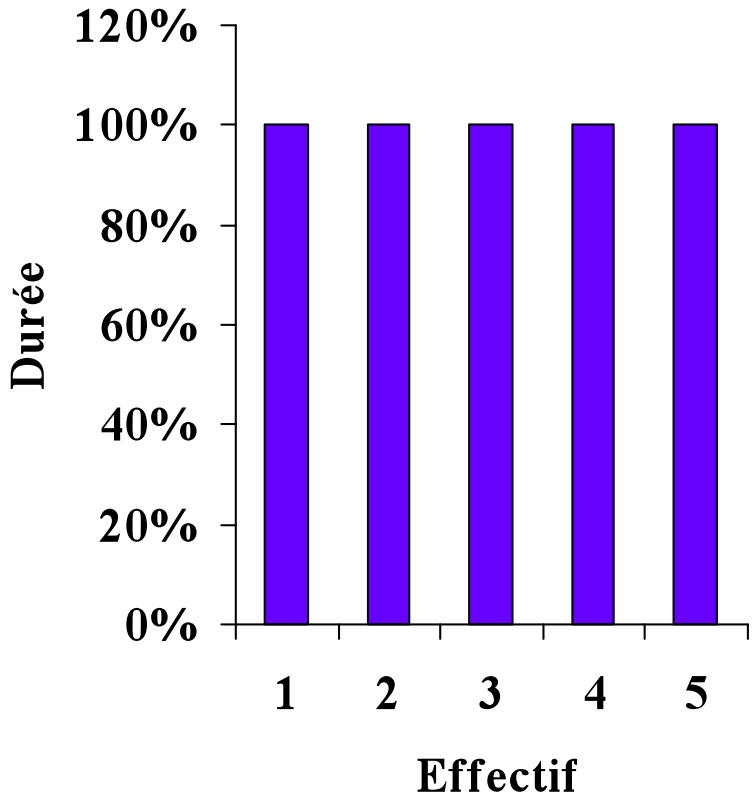
Les enjeux du génie logiciel : De la division du travail...

- Pour une tâche parfaitement divisible (parallélisable), la durée est inversement proportionnelle à l'effectif…
- Cela s'applique probablement à la cueillette du coton...



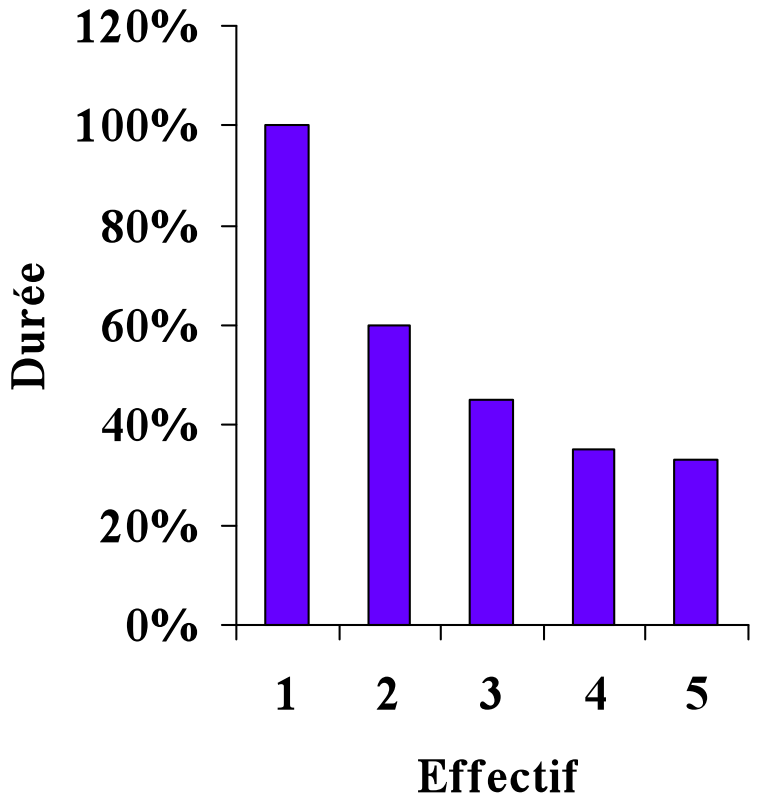
Les enjeux du génie logiciel : De la division du travail...

- Pour une tâche totalement indivisible ou séquentielle, la durée est indépendante de l'effectif…
- On ne fait pas un enfant en trois mois avec trois femmes...



Les enjeux du génie logiciel : De la division du travail...

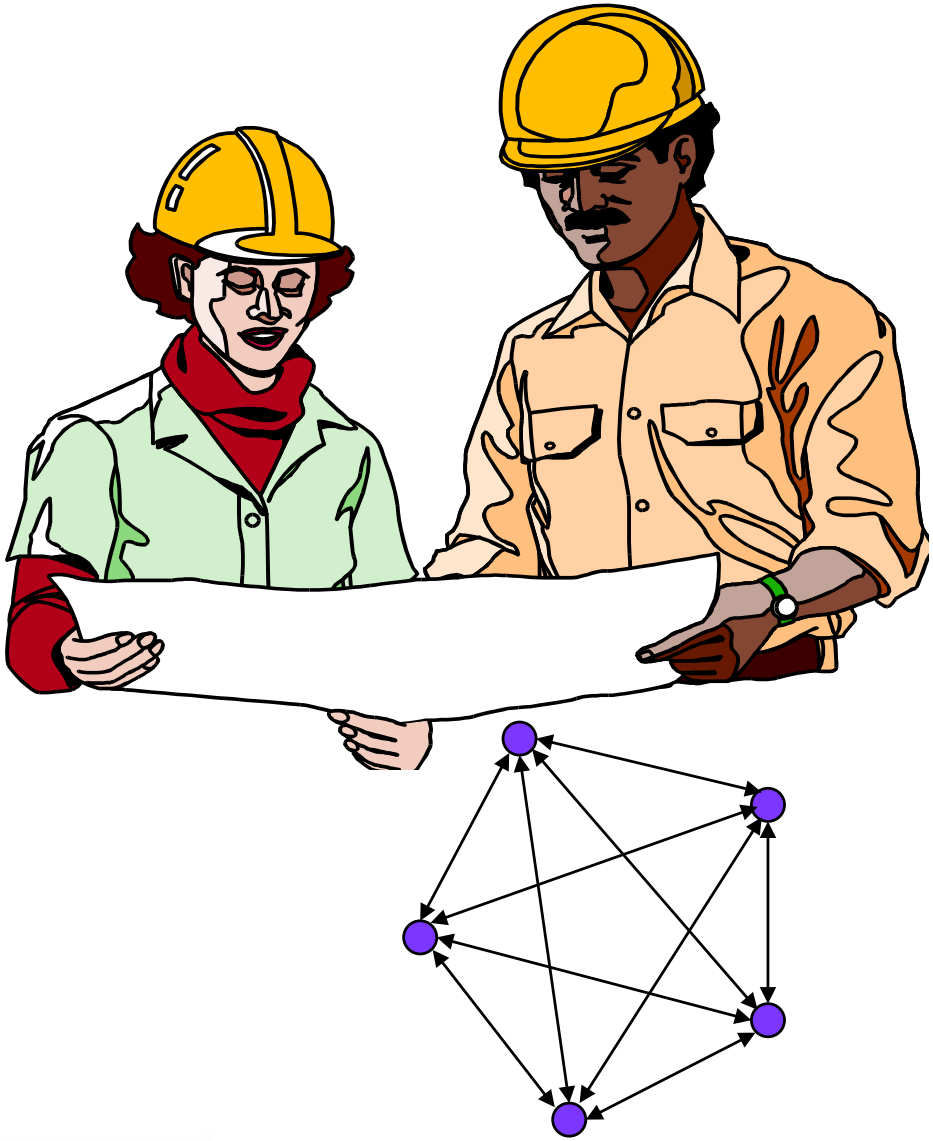
- La réalité du génie logiciel est entre ces deux extrêmes (séquentialité de certaines tâches)
- La durée de développement ne peut tendre vers zéro avec une armée de programmeurs...



Les enjeux du génie logiciel : De la division du travail...

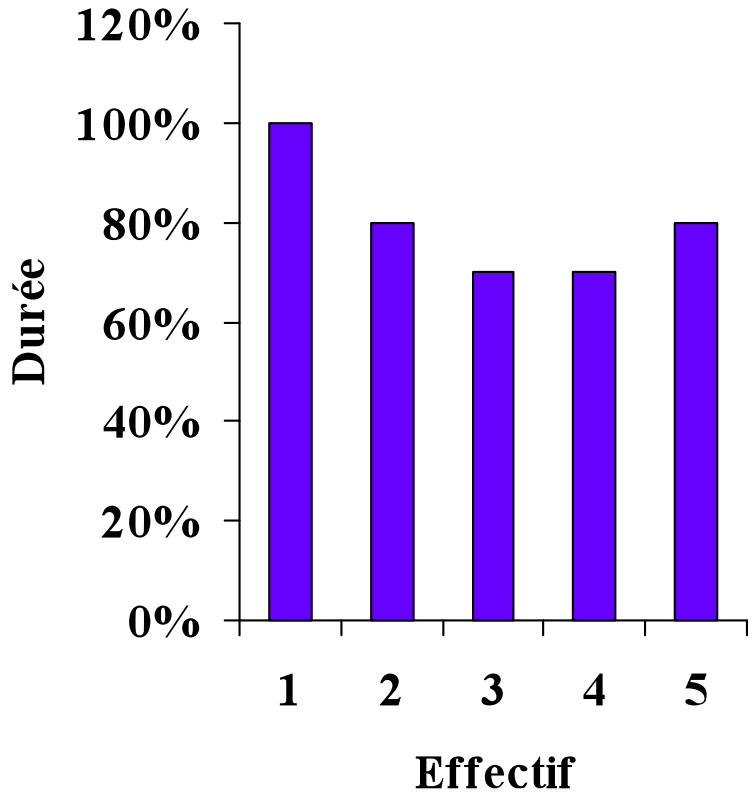
Les considérations qui précèdent ne prennent pas en compte :

- La formation, proportionnelle à l'effectif N.
- La communication (proportionnelle à N(N-1)/2.

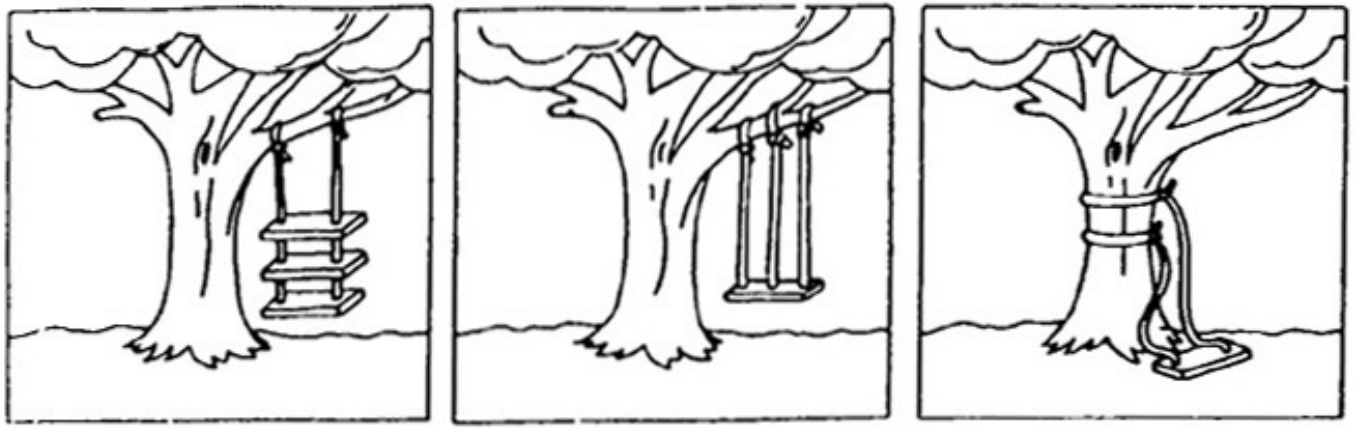


Les enjeux du génie logiciel : De la division du travail...

- Quelques problèmes sont alors possibles si les interactions entre tâches sont complexes...
- «Ajouter des gens à un projet logiciel en retard le retarde encore davantage» (Loi de Brooks).



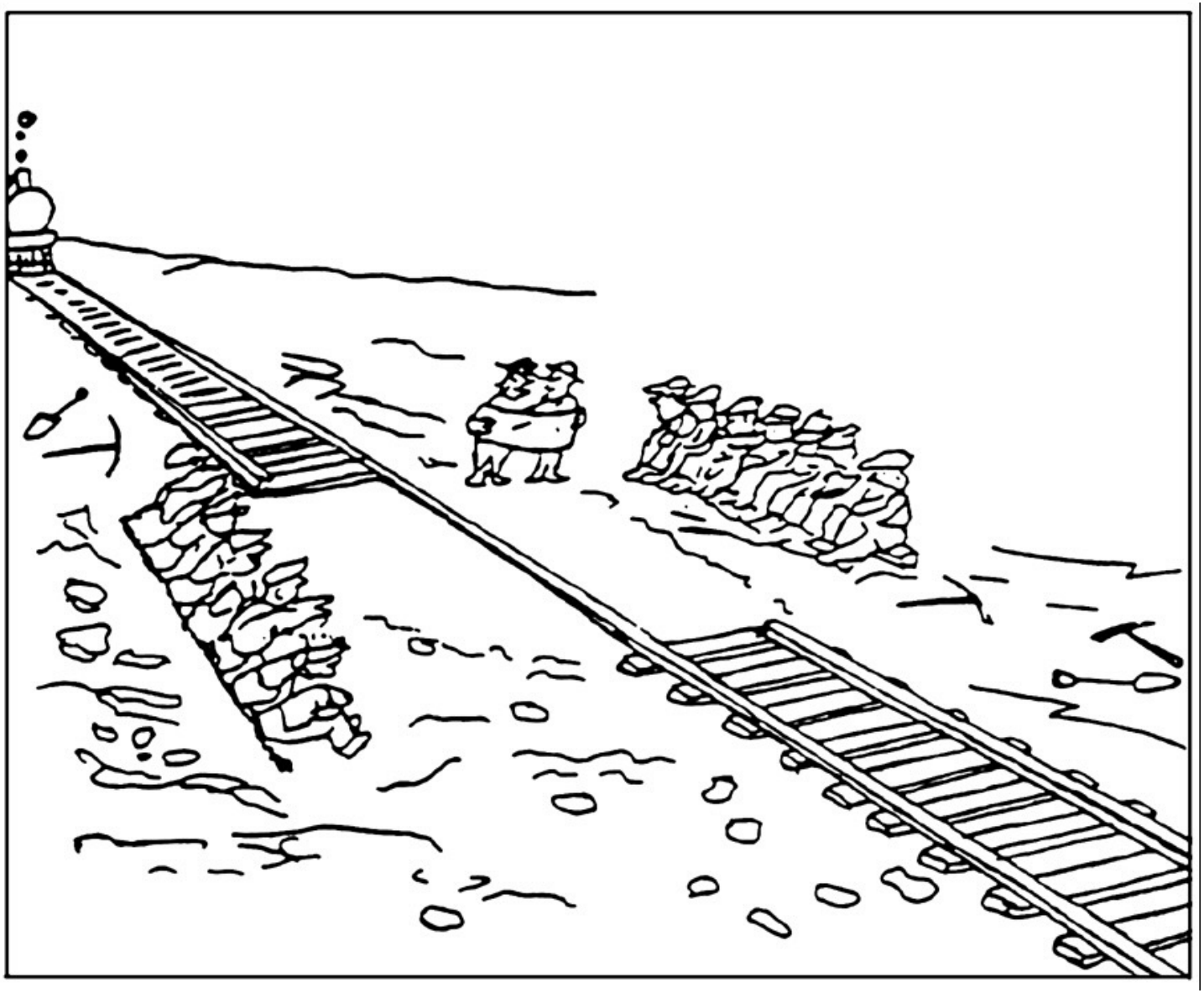
Des vertus d'une communication précise...



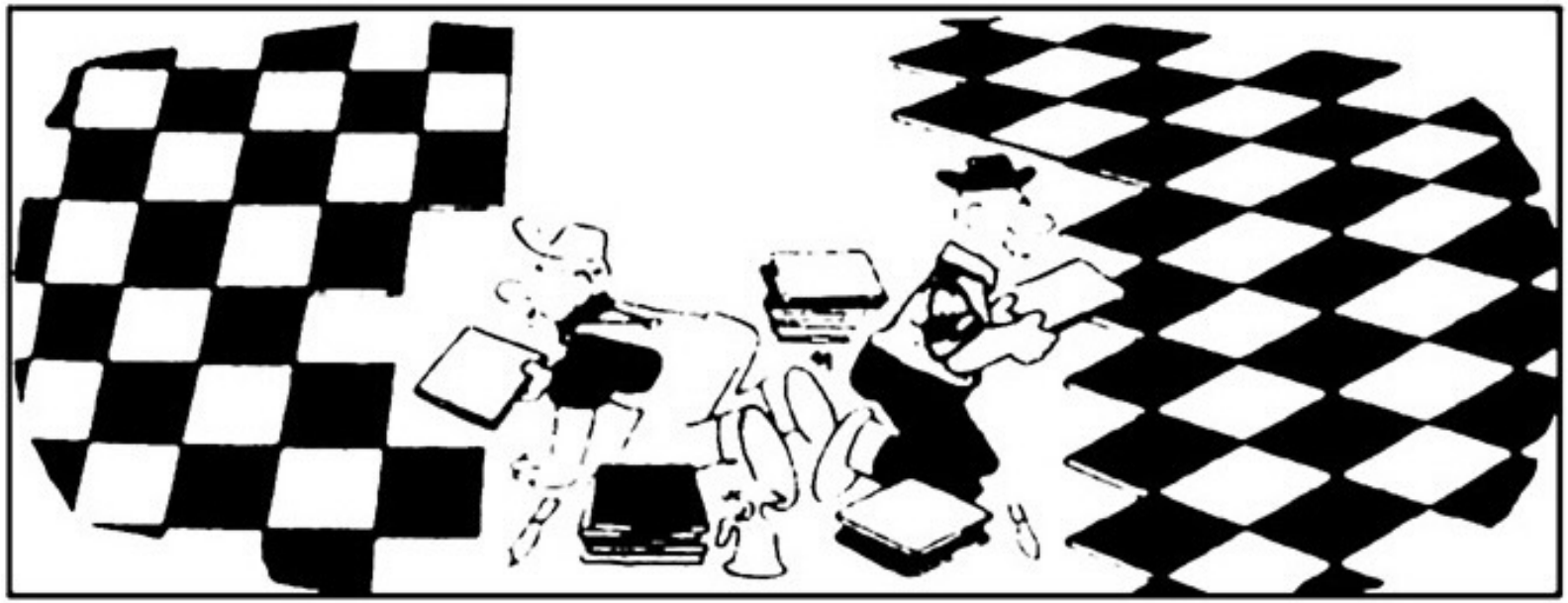
Cahier des charges | Spécification | Conception

Codage | Deboguage | Besoin

Des vertus d'une communication précise...



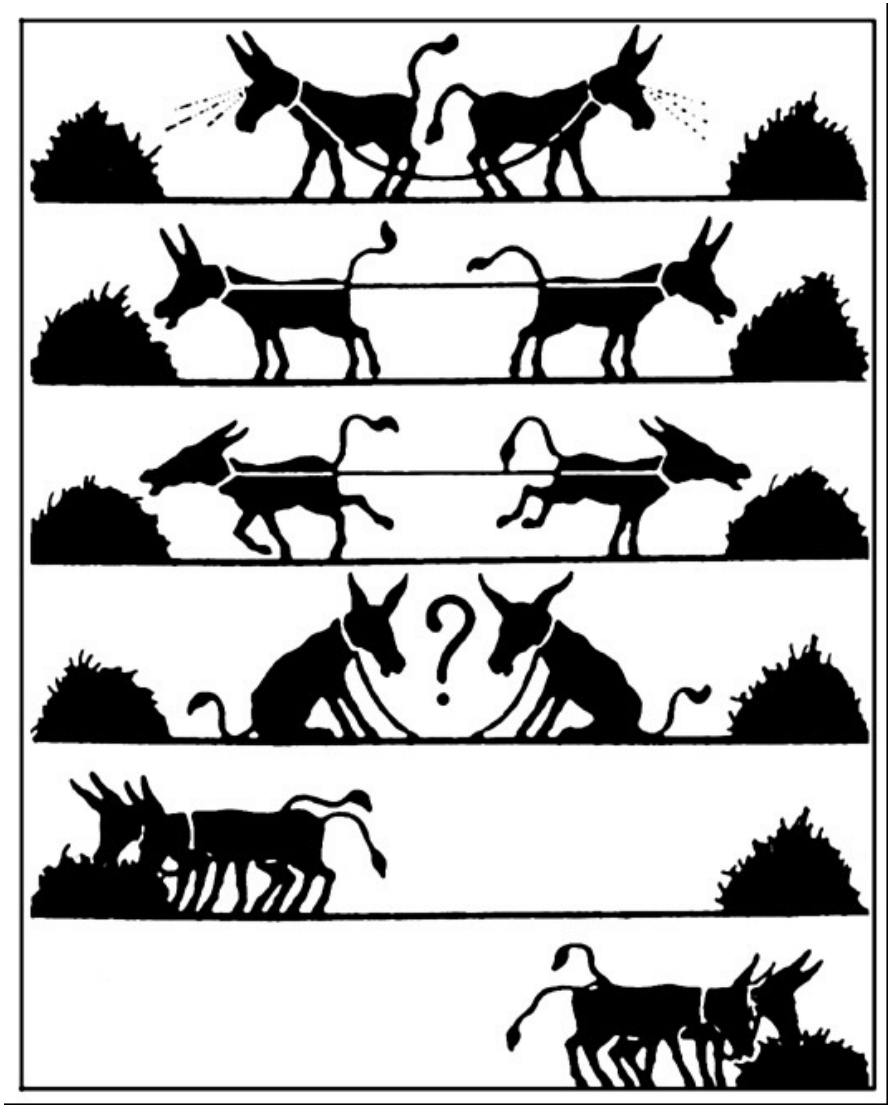
Des vertus d'une communication précise...



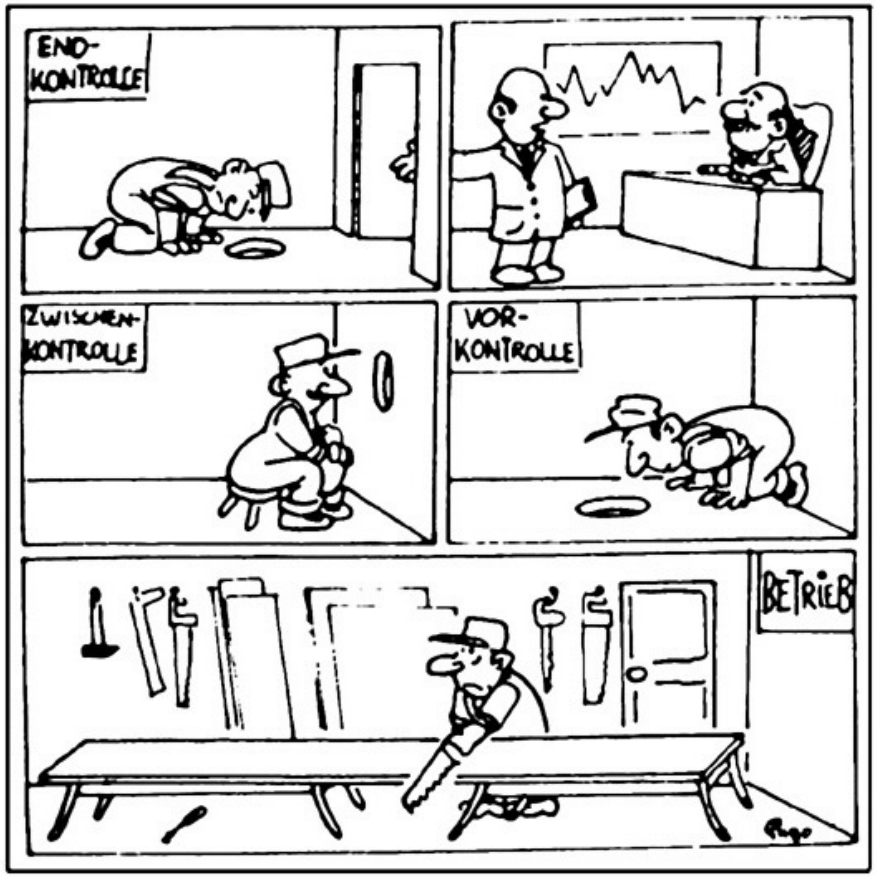
Les enjeux du génie logiciel
Aspects organisationnels

- On ne parvient à rien en laissant libre cours à la créativité débridée de chacun.
- Organiser est également nécessaire pour parvenir à une unité de conception.
- Le management fait donc également partie du génie logiciel.

Des vertus de l'organisation...



Des vertus de l'organisation



- N'oublions pas ce célèbre mot du poète satirique latin Juvénal (346 après J.C.)
- Sed quis custodiet ipsos custodes ?(Mais qui gardera les gardiens ?)
- Organiser n'est pas seulement contrôler, c'est aussi mettre en place les moyens de bien faire du premier coup.

Les enjeux du génie logiciel
Aspects qualité

- Le génie logiciel c'est également parvenir à un produit fini de qualité (i.e. satisfaisant aux attentes de son client).
- Cette tâche est compliquée par l'aspect immatériel du logiciel => Problèmes d'observabilité du processus de développement

Qualité(s) d'un logiciel (1)

- Fiabilité : faire ce que l'utilisateur attend (et non pas rien ou tout autre chose !)
- Maintenabilité : possibilité de le faire évoluer dans des coûts/délais maîtrisés
- Sécurité : aptitude à éviter de provoquer des événements catastrophiques (pour les logiciels dits «critiques»)

➵ Très lié à la sûreté de fonctionnement

Qualité(s) d'un logiciel

- Validité : conformité aux attentes initiales du client.
- Efficacité : temps de réponse (peut être critique pour le temps réel).
- Convivialité : importance de l'interface homme/logiciel.

Le logiciel : des propriétés curieuses !

- il est visible *mais* intangible
- il vieillit *mais* ne s'use pas
- il *ne* se détériore *pas* sous l'effet des tests
- il est *encore* et *toujours* fabriqué artisanalement
- il est (trop ?) *facilement* reproductible
- il est (trop ?) *facile* à modifier
- il est d'une grande complexité : coût très (trop ?) *élevé*
- ...

Exemple de catastrophe

Ariane 501

Quand:

le 4 juin 1996

Coût :

500 millions de dollars de pertes

Autopsie d'une catastrophe : Ariane 501 : Rappel des faits

- 4 Juin 1996 Kourou H0=9h33'59'' : Lancement du vol inaugural Ariane 5.
- Jusqu'à H0+36s : Comportement nominal
- H0 36,7s : défaillance quasi-simultanée des deux SRI (dispositifs redondants de calcul de la position et de l'attitude de la fusée).
- Commande de déviation maximale des tuyères de tous les propulseurs

Autopsie d'une catastrophe : Ariane 501 : Rappel des faits

- H0+39s : Angle d'attaque de la fusée d'environ 20 degrés.
- Charges aérodynamiques trop importantes => les boosters se détachent => autodestruction du lanceur à 3700m d'altitude.
- Parmi les débris éparpillés sur 12km2 dans la mangrove, on retrouve les deux SRI.

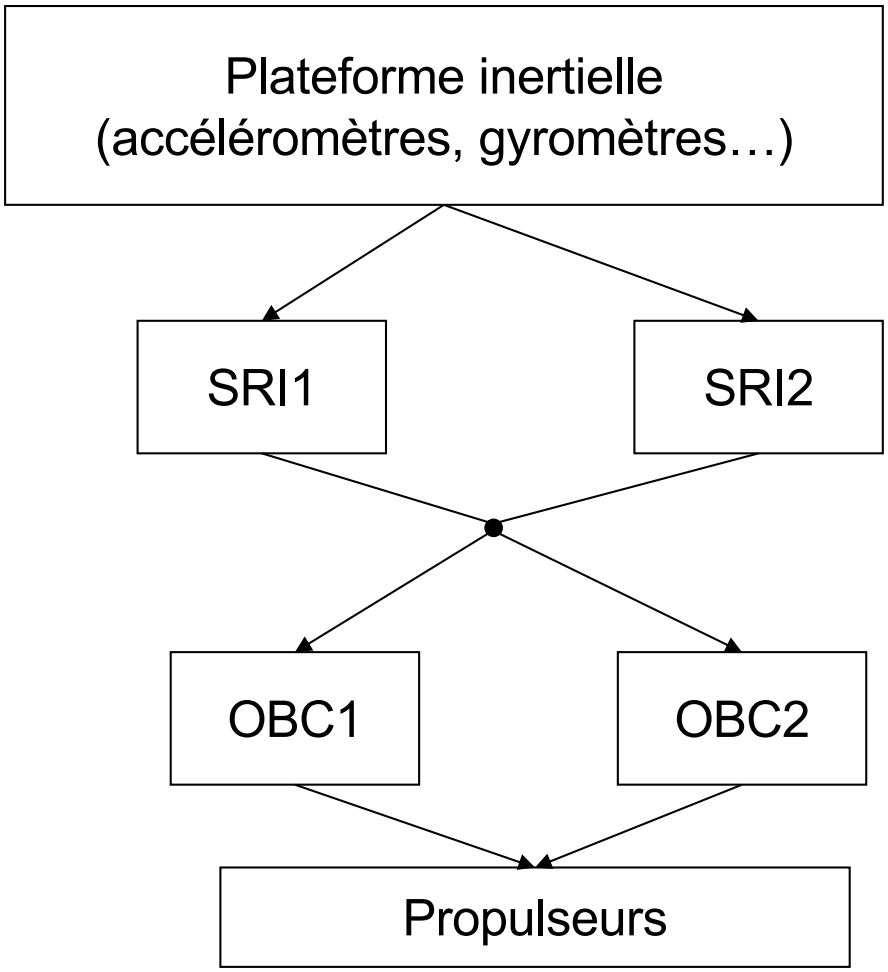
Autopsie d'une catastrophe : Ariane 501 : Analyse

Instruments de mesure

Système de Référence Inertiel (calcul des paramètres)

On Board Computer (commande des propulseurs)

Moteur Vulcain et boosters



Autopsie d'une catastrophe : Ariane 501 : Analyse

- Les deux SRI ont défailli pour la même raison : valeur trop grande d'une variable transmise par la plateforme à inertie.
- Conformément à leur spécification, ils se sont alors arrêtés en ne transmettant plus que des données de diagnostic au lieu des données de vol. L'OBC a commandé les déviations maximales sur la base de ces données erronées.

Autopsie d'une catastrophe : Ariane 501 : Analyse

- La fonction où a eu lieu le problème ne sert qu'à l'étalonnage au sol de la plateforme inertielle. En vol elle est inutile.
- Elle était maintenue active pendant 50s après H0 pour permettre des décalages de lancement de dernière minute sans reprendre tout le ré-étalonnage de la centrale. Cette possibilité ne sert que sur Ariane 4 elle est inutile sur Ariane 5.

Autopsie d'une catastrophe : Ariane 501 : Analyse

- La variable qui a débordé ne pouvait faire de même sur Ariane 4 (performances différentes du lanceur).
- Elle ne faisait pas partie de celles protégées contre les débordements pour des raisons de performances (considérée comme physiquement limitée ce qui au sol est vrai).

Autopsie d'une catastrophe : Ariane 501 : Analyse

- La fonction étalonnage n'a pas été modifiée ni même testée à nouveau pour Ariane 5 car elle était réputée validée par l'expérience.

Autopsie d'une catastrophe : Ariane 501 : Moralités

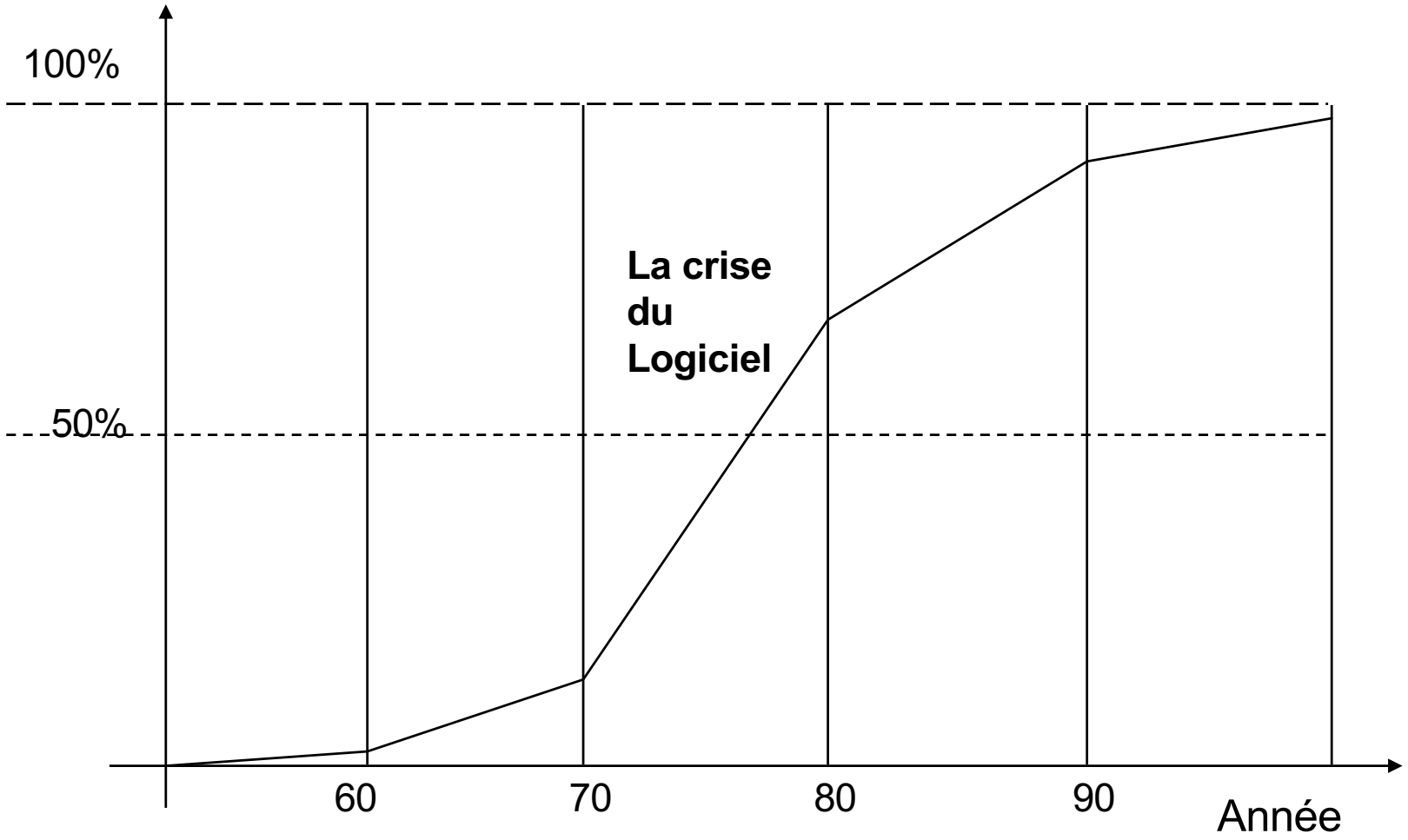
- La vraie cause de la catastrophe réside dans le choix d'arrêt total des SRI en cas d'exception.
- C'est pourtant ce que demandait la spécification avec l'idée que la redondance couvrait les défaillances matérielles.
- On voit que ce type de redondance est sans effet en cas de bogue de conception.

Autopsie d'une catastrophe : Ariane 501 : Moralités

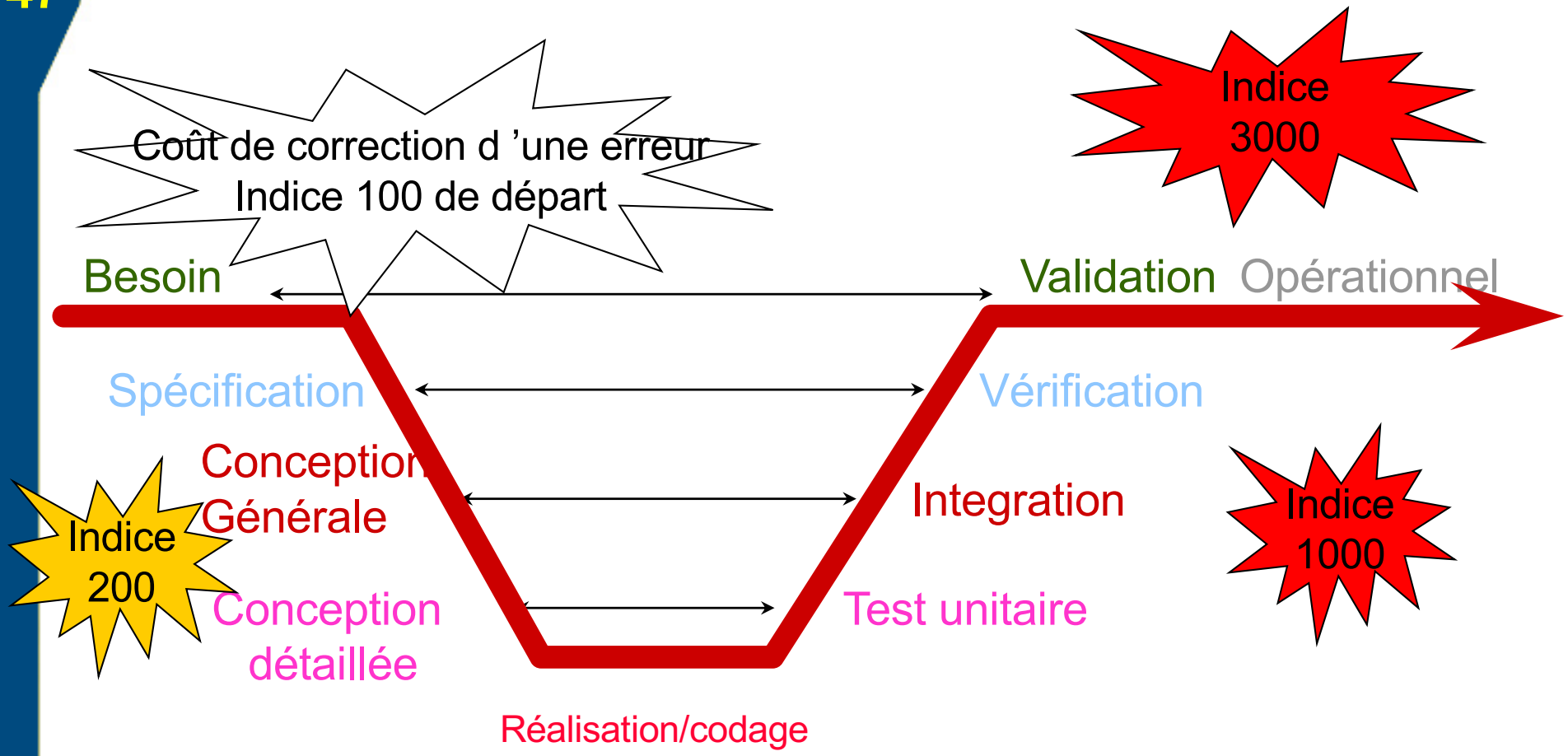
- Se méfier des fonctions qui sont actives quand on n'en a pas besoin.
- Se méfier de l'argument du tout reconduit lorsque des hypothèses changent (conditions d'environnement).

Attention aux coûts !!

Coût du Développement Logiciel / Coût du matériel



Cycle du projet et coûts d'une correction



53% des projets coûtent 189% des estimations initiales (données rapport CHAOS)

Le coût des « bug »

- D'après un rapport du NIST (Assesses Technical Needs of Industry to Improve Software-Testing) daté de 2002 qui porte sur les états unis:
  - Le coût des erreurs et des « bugs » logiciels représente annuellement 59.5 billion de dollars.
- Cette étude indique aussi
  - que tous les « bugs » ne peuvent être corrigés,
  - mais que l'amélioration du processus de vérification permettrait d'en éviter un tiers (soit 22.2 billion de dollars).

Exemple de CdCF : La Citroën 2CV

- Pour finir : l'expression des besoins d'un client est en général l'objet du Cahier des Charges Fonctionnel (CdCF)… souvent assez éloigné de considérations de conception !

- Le CdCF établi par Mr P.J. BOULANGER (responsable de Citroën) à Mr BROGLY (Directeur du Bureau d'études) en 1936 :
- "Faites étudier par vos services, une voiture pouvant transporter deux cultivateurs en sabots, cinquante kilos de pommes de terre ou un tonnelet à une vitesse de 60 km/h, pour une consommation de 3 litres au cent. La voiture pourra passer dans les plus mauvais chemins; elle devra pouvoir être conduite par une conductrice débutante et avoir un confort irréprochable. Son prix devra être inférieur au tiers de la traction avant 11 CV. Le point de vue esthétique n'a aucune importance".