LO19 : De l'expression du besoin à la réalisation du logiciel

Introduction : genèse et enjeux du génie logiciel

Cycle de développement

Walter SCHÖN

Jean-Louis BOULANGER

Enjeux du génie logiciel : Sommaire

- Introduction et concepts
- Cycle de développement
- Exemple de catastrophe
- Attention aux coûts !!

Introduction et concepts

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 malheurs logiciels... (1)

- Perte de la sonde Mariner vers Vénus
- Perte en de 72 ballons météo (1971)
- Abandon compilateur PL1 (Control Data)
- Retard 1er lancement navette spatiale (1981)
- Indisponibilité réseau téléphonique ATT (1990)
- Avion F16 déclaré sur le dos au passage de l'équateur : non prise en compte du référentiel hémisphère sud.

Quelques malheurs logiciels... (2)

- Mission Venus : passage à 500.000 km au lieu de 5.000 km : remplacement d'une virgule par un point
- Système Socrate de la SNCF
- Echec du vol Ariane 501 (1996)

Quelques malheurs logiciels... (3)

- 2001: 1 million de portable Sony ont été rappelés au Japon
  - Mai 2001, Sony a rappelé pour problème logiciel
    - 420 000 S0503i et
    - 126 000 C101S
  - Juillet 2001 560 000 C406 S sont rappelés pour problème de batterie
  - Surcoût estimé : 20 Milliards de yens

Quelques malheurs logiciels... (4)

- Octobre 2004 : Importantes perturbations du réseau France Telecom pendant 48h
- Novembre 2004 : paralysie totale du réseau Bouygues Telecom pendant une vingtaine d'heures

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)

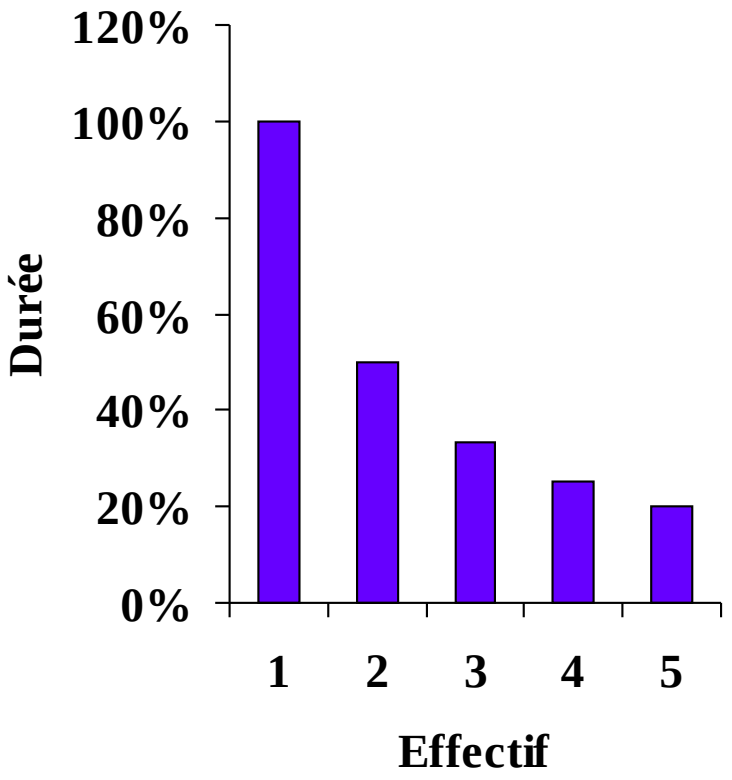
  => logiciels de plus en plus complexes (plusieurs millions de lignes de code).

  => Course permanente contre l'obsolescence.

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

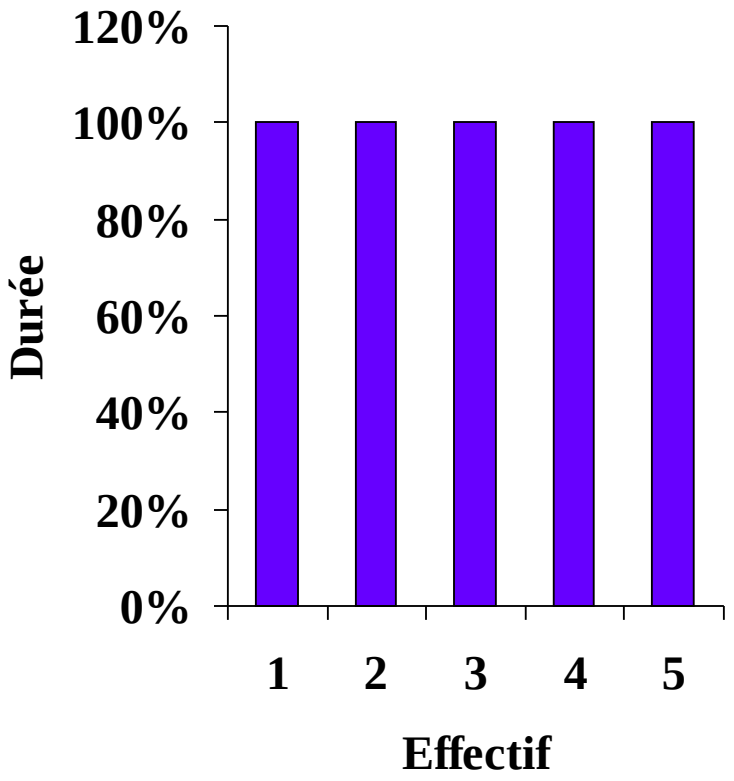
Les enjeux du génie logiciel : De la division du travail...

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



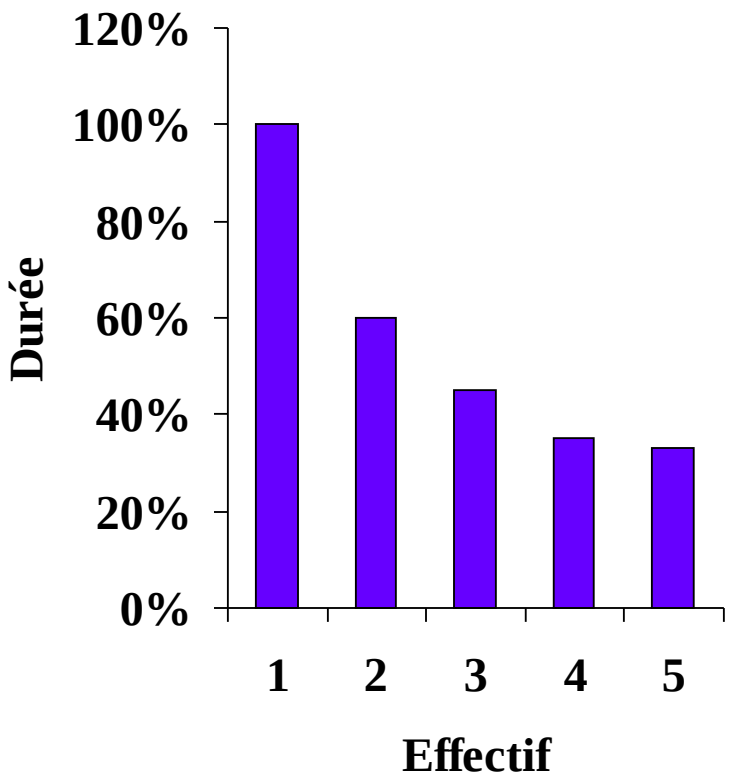
Les enjeux du génie logiciel : De la division du travail...

- 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...

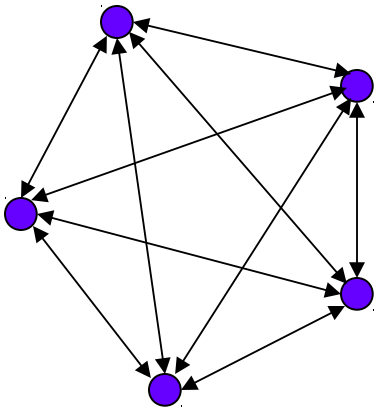
- La réalité du génie logiciel est entre ces deux extrêmes (séquentialité de certaines tâches)
- La durée de développement ne peut tendre vers zéro avec une armée de programmeurs...



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

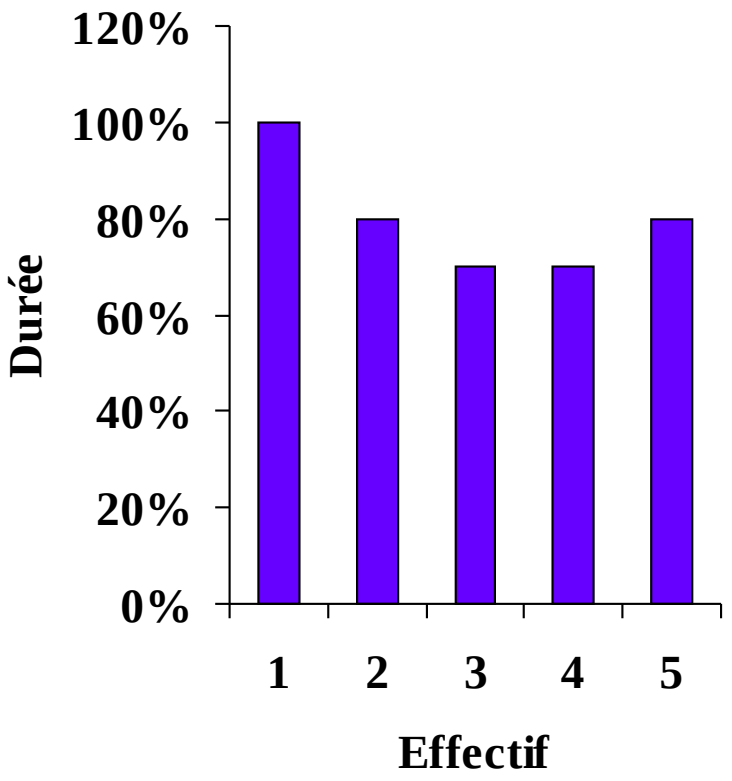
Les considérations qui précèdent ne prennent pas en compte :

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

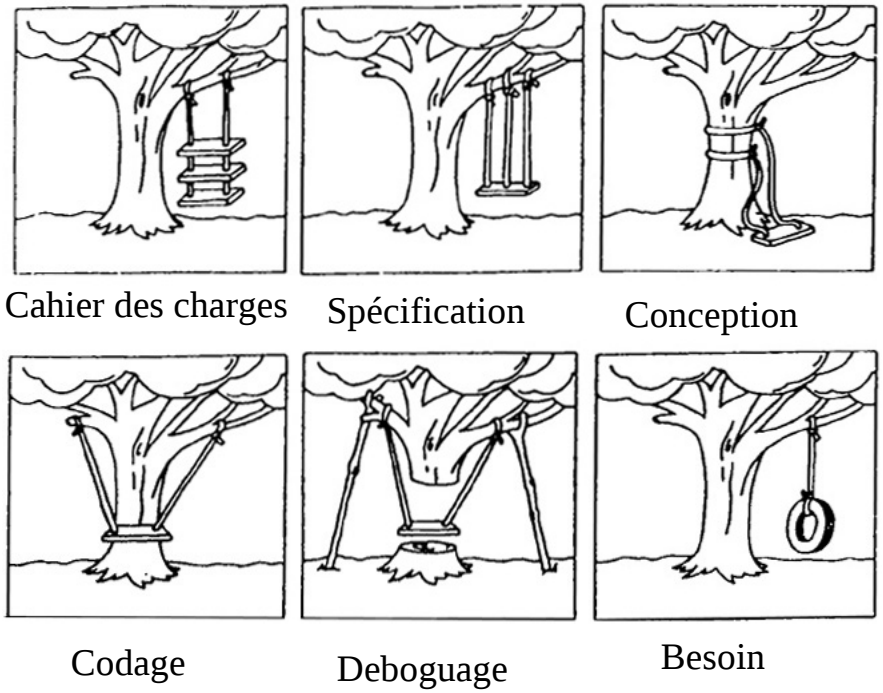


Les enjeux du génie logiciel : De la division du travail...

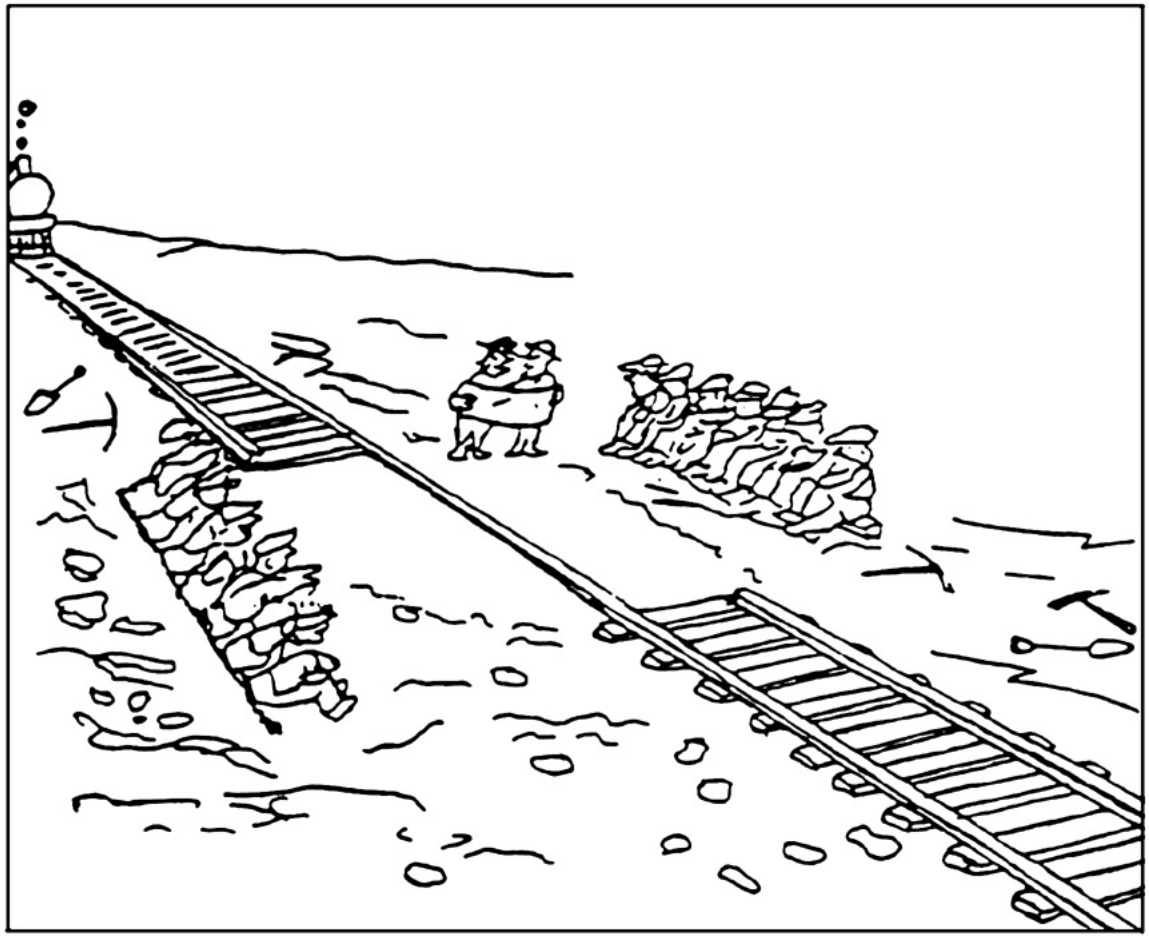
- Quelques problèmes sont alors possibles si les interactions entre tâches sont complexes...
- «Ajouter des gens à un projet logiciel en retard le retarde encore davantage» (Loi de Brooks).



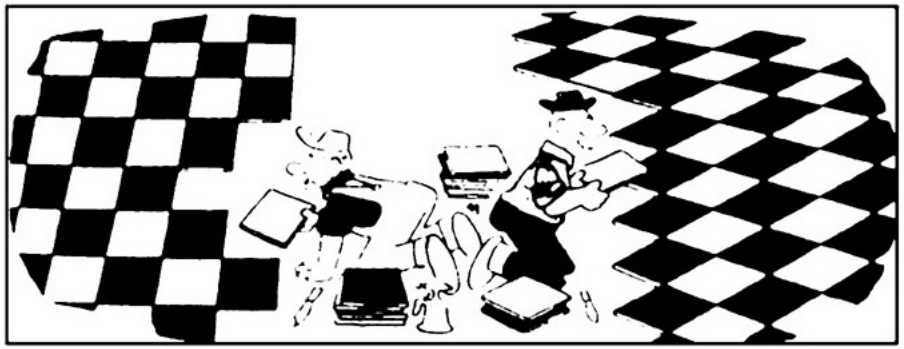
Des vertus d'une communication précise...



Des vertus d'une communication précise...



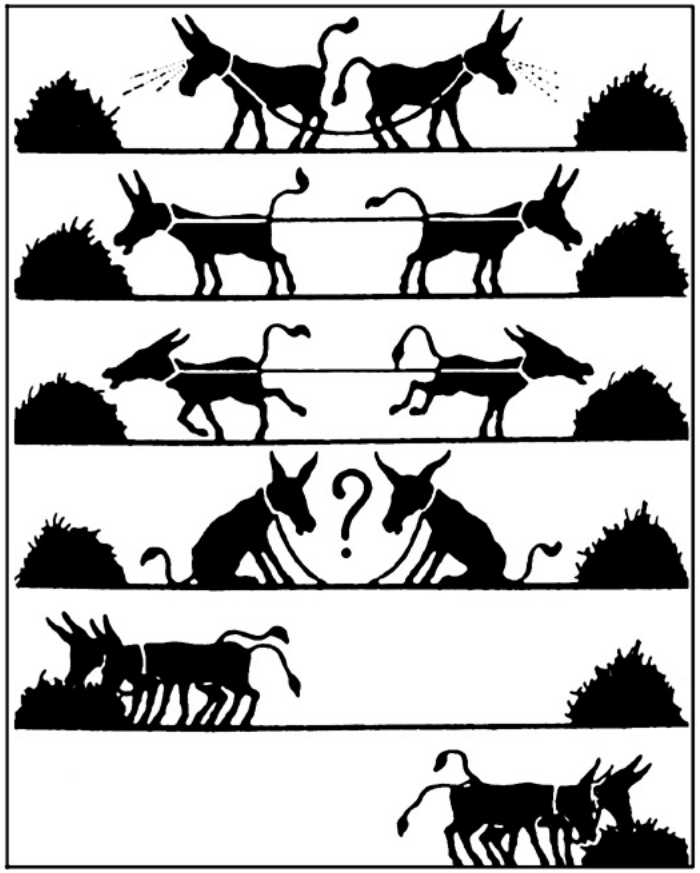
Des vertus d'une communication précise...



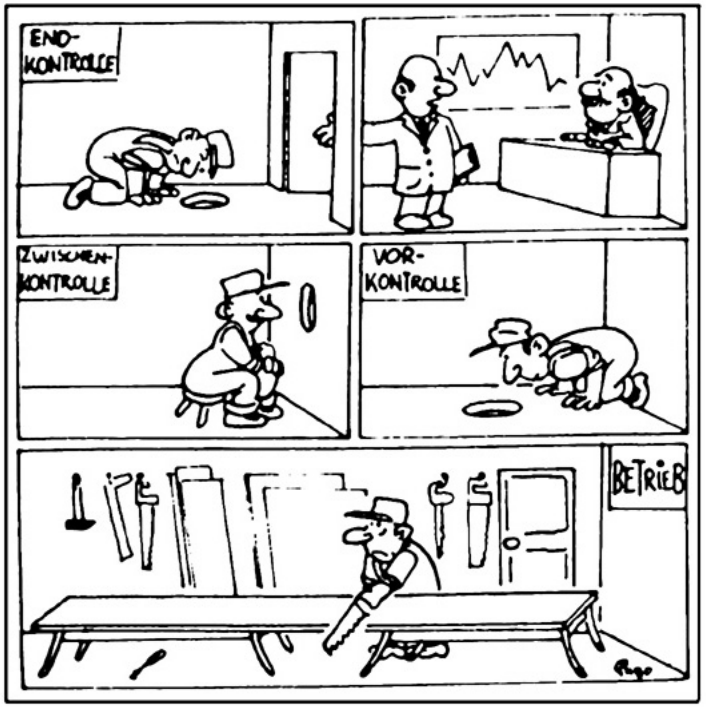
Les enjeux du génie logiciel
Aspects organisationnels

- 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 (bien que peu traité dans le présent cours…).

Des vertus de l'organisation...



Des vertus de l'organisation



- «Et qui gardera les gardiens ?» (Juvénal).
- Organiser n'est pas seulement contrôler, c'est aussi mettre en place les moyens de bien faire du premier coup.

Estimation du coût et de la durée d'un projet logiciel

- De nombreux modèles empiriques existent basés sur des formules paramétriques.
- Le plus connu COCOMO (COnstructive COst MOdel : Barry BOEHM 1981) se décline en trois versions : simple, intermédiaire et détaillé.
- Seule la version simple est décrite ci-après.

Modèle COCOMO simple

- Donne un ordre de grandeur de l'effort nécessaire pour réaliser un logiciel en fonction de sa taille (mesurée en kilo-lignes de code source KLCS).
- Pour les raisons évoquées plus haut, l'effort n'est pas exactement proportionnel à la taille $\text{Effort(Homme*Mois)} = A.(KLCS)^b$ $b>1$.
- Problème : la taille n'est réellement connue qu'après la réalisation du logiciel !

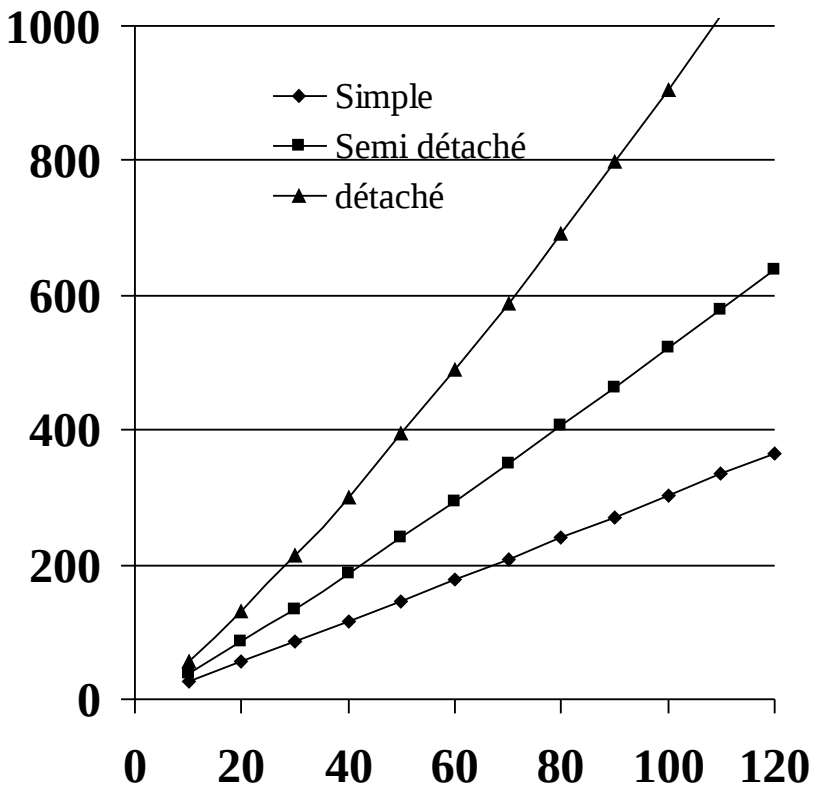
Modèle COCOMO simple : Niveaux de complexité

Distingue trois niveaux de complexité :

- Simple : petites équipes, applications maîtrisées, peu de communications.
- Semi-détaché : équipe moins expérimentée, applications moins courantes.
- Détaché : applications complexes, contraintes importantes (systèmes embarqués…)

Modèle COCOMO simple : Estimation du coût

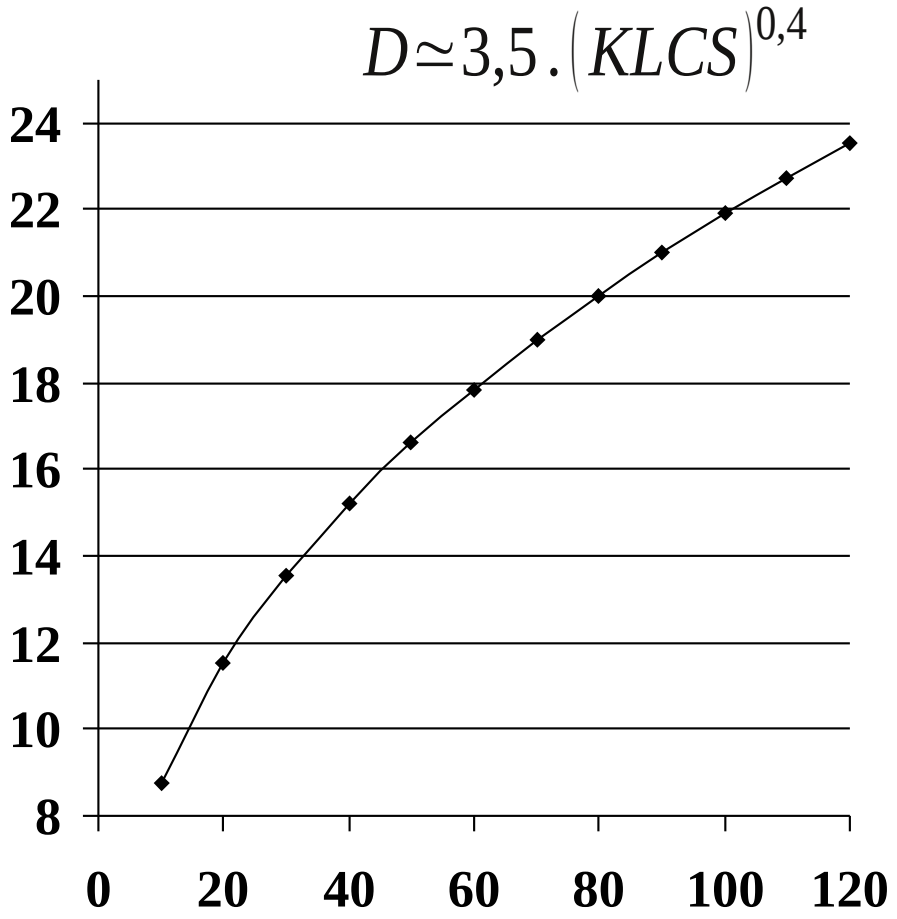
- Simple
  $HM=2,4(KLCS)^{1,05}$
- Semi-détaché
  $HM=3(KLCS)^{1,12}$
- Détaché
  $HM=3,6(KLCS)^{1,20}$



Modèle COCOMO simple : Estimation de la durée

Durée optimale (en mois) déduite :

- Simple $D=2,5(HM)^{0,38}$
- Semi-détaché $D=2,5(HM)^{0,35}$
- Détaché $D=2,5(HM)^{0,32}$



Modèle COCOMO simple : Estimation de la taille de l'équipe

- L'estimation de la taille de l'équipe nécessaire par Taille=Effort/Durée est très grossière (les besoins en personnel ne sont pas uniformes dans le temps).
- Des modèles de "plans de charge type" (variations en fonction du temps du personnel nécessaire) permettent d'affiner cet ordre de grandeur.

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*
- ...

Cycle de développement

Modèles de développement du logiciel

- Modélisation : nécessaire pour décrire dépendances et enchaînements entre activités.
- Doit prendre en compte l'aspect raffinement progressif du développement ainsi que de possibles itérations.
- Souvent appelé cycle de développement ou cycle de vie du logiciel.

Principales activités de développement du logiciel

- Analyse des besoins : adéquation attentes client/faisabilité technologique.
- Spécification : décrire ce que le logiciel doit faire (et non comment il le fera).
- Conception architecturale ou préliminaire : décomposition hiérarchique et interfaces.
- Conception détaillée : description de chaque composant (données, algorithmes…)

Principales activités de développement du logiciel (suite)

- Codage : programmation proprement dite.
- Validation (conformité aux attentes client) : revues, inspections, prototypage...
- Vérification (conformité aux spécifications et à la conception) : revues et inspections plus tests (développement classique) ou preuves (développement formel).

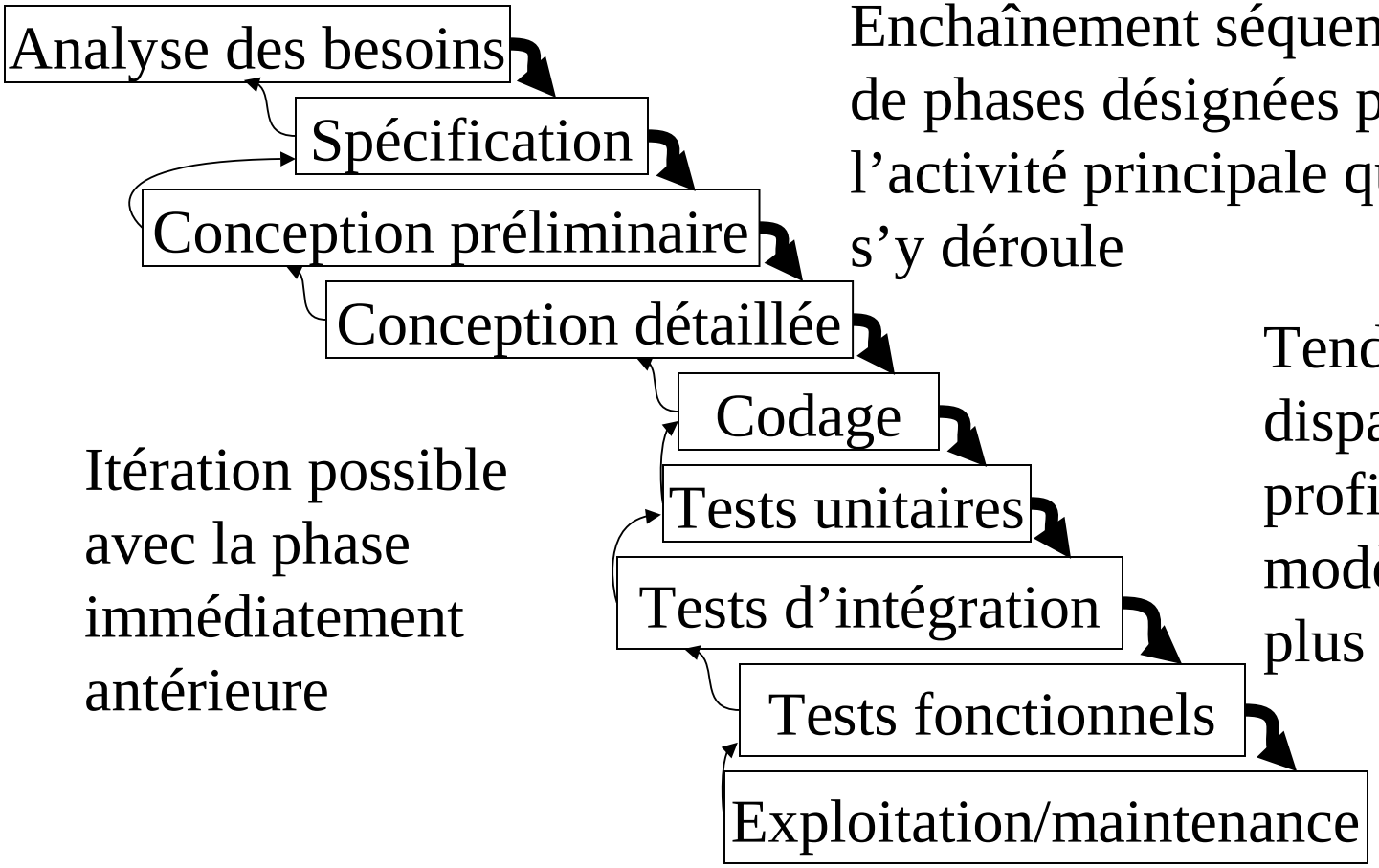
Activités de développement du logiciel : tests

- Tests unitaires : composants isolés
- Tests d'intégration : tests progressifs de l'assemblage des composants
- Tests système ou fonctionnels : logiciel complet sur machine de développement et/ou environnement réel
- En développement formel : remplacé en partie par des preuves.

Activités de développement du logiciel (fin)

- Installation :livraison du système au client et vérification de son fonctionnement
- Exploitation/maintenance : vie opérationnelle et évolutions éventuelles. Débute à la recette par le client.
- Gestion de configurations : gestion des différents composants, de leur version et de la cohérence globale des versions, maîtrise des évolutions : se déroule tout au long du processus de développement.

Modèle en cascade

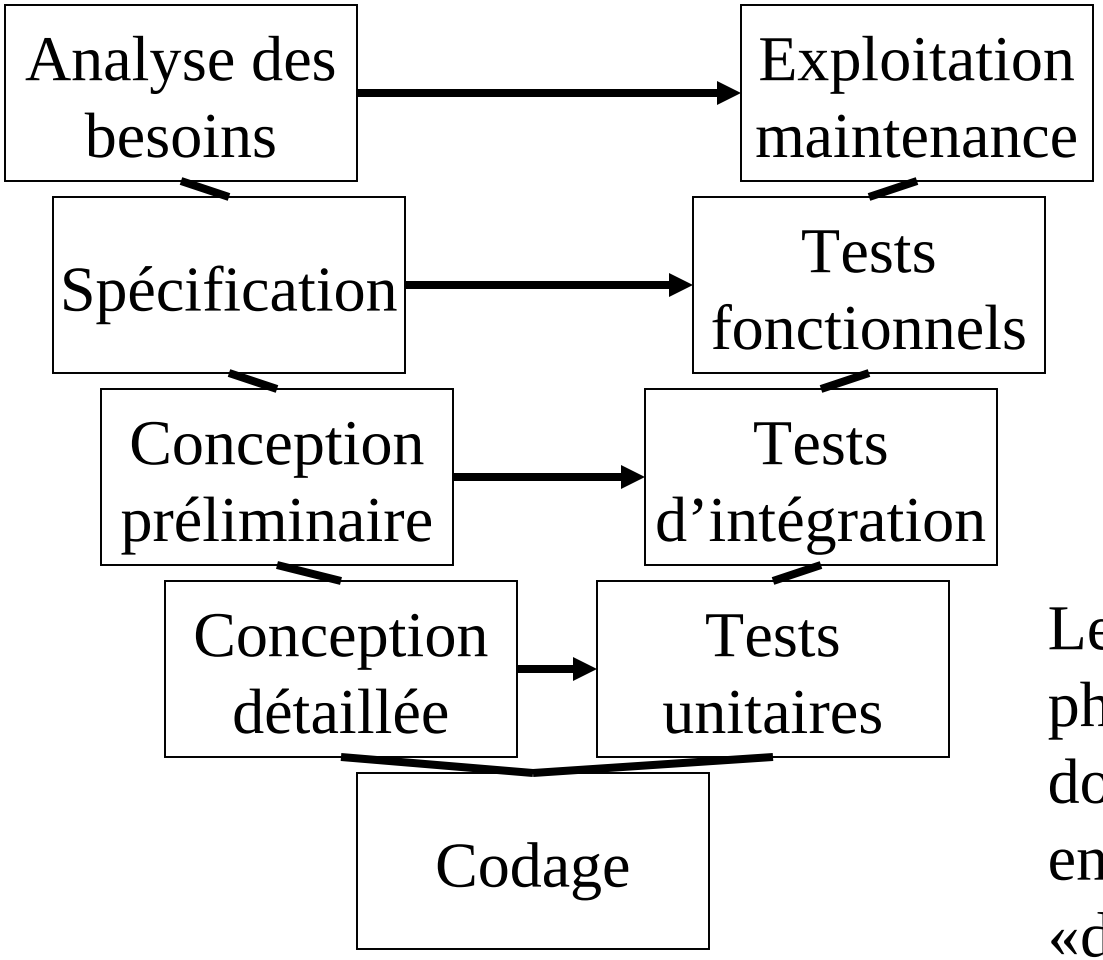


Enchaînement séquentiel de phases désignées par l'activité principale qui s'y déroule

Itération possible avec la phase immédiatement antérieure

Tend à disparaître au profit du modèle en V plus complet

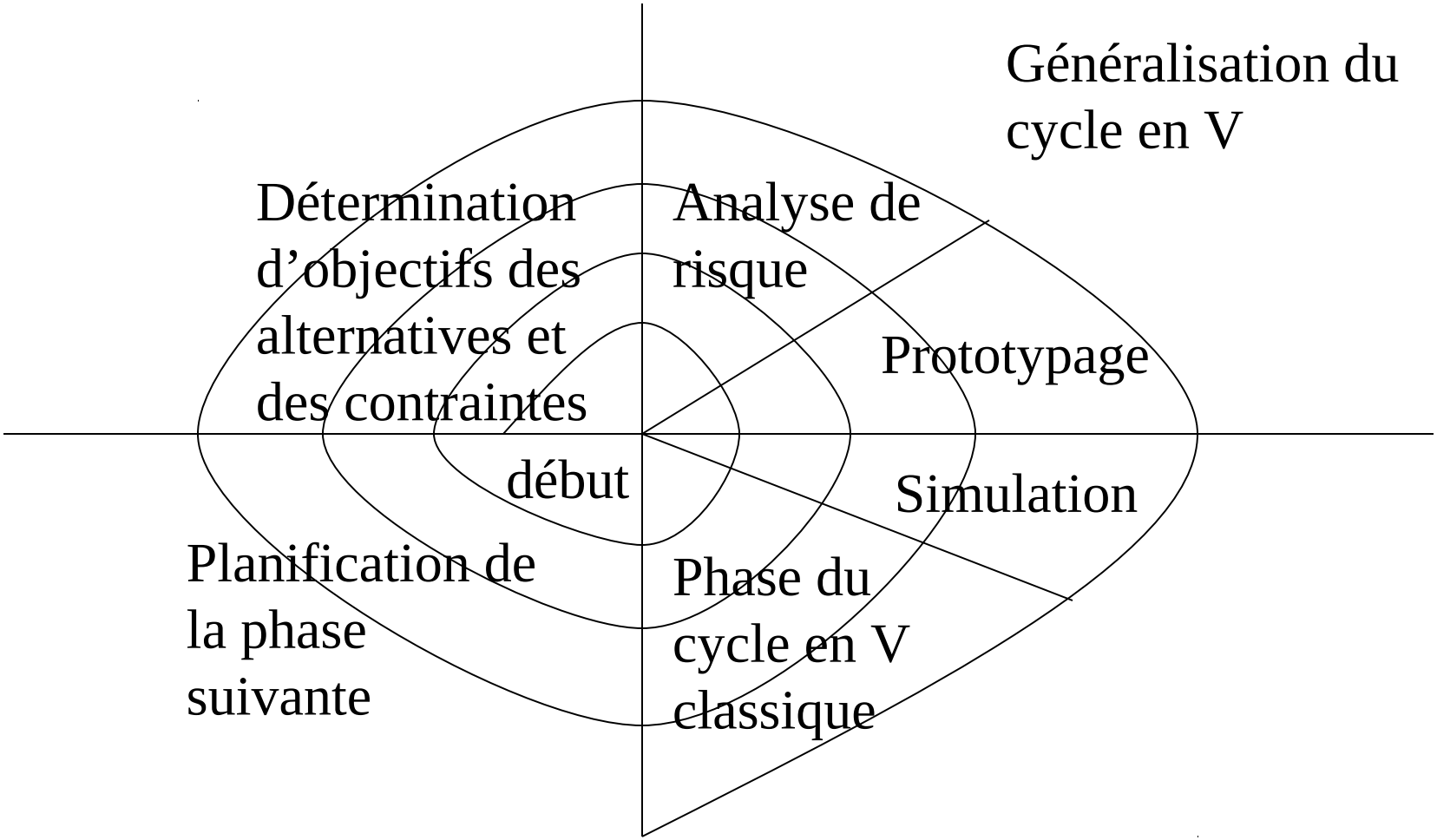
Modèle en V



Correspondance horizontale entre spécification conception et activités de vérification

Les activités des phases «montantes» doivent être préparées en phases «descendantes»

Modèle en spirale (Boehm 1988)



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.

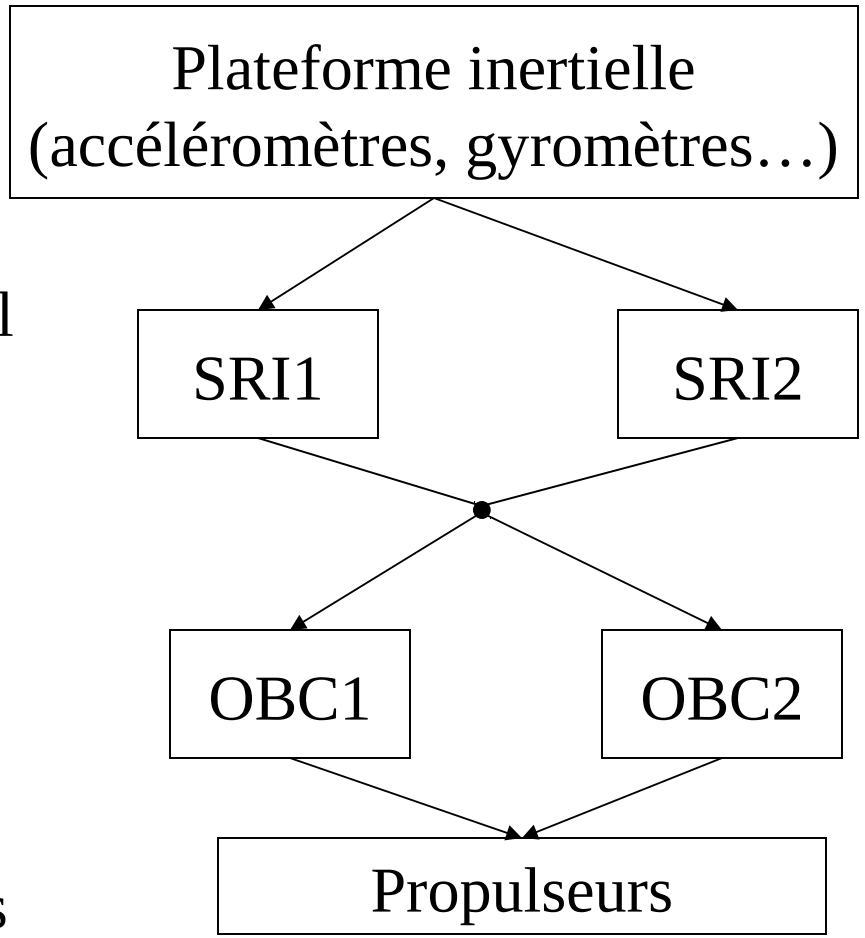
Autopsie d'une catastrophe : Ariane 501 : Analyse

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

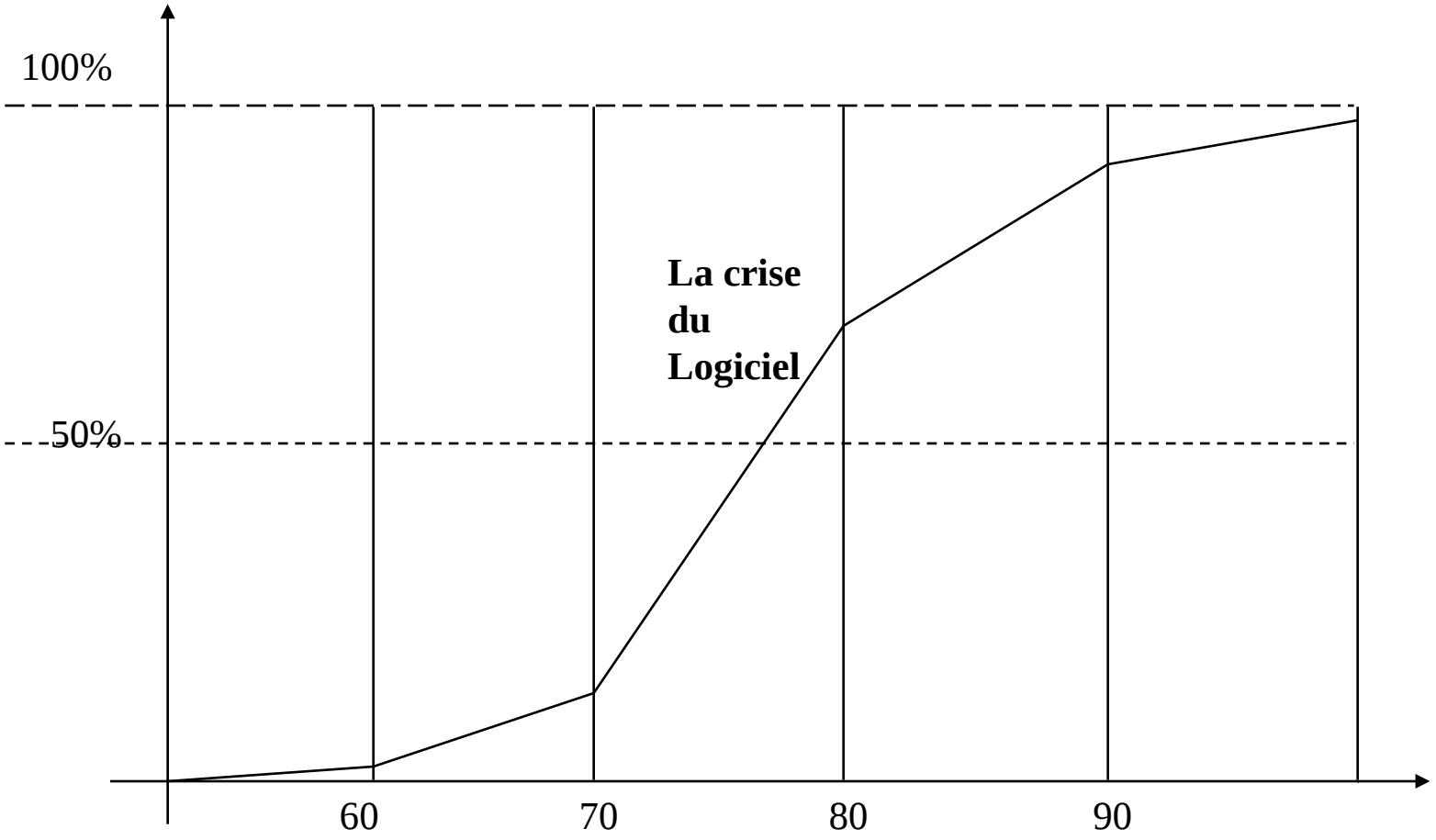
- La vraie cause de la catastrophe réside dans le choix d'arrêt total des SRI en cas d'exception.
- C'est pourtant ce que demandait la spécification avec l'idée que la redondance couvrait les défaillances matérielles.
- On voit que ce type de redondance est sans effet en cas de bogue de conception.

Autopsie d'une catastrophe : Ariane 501 : Moralités

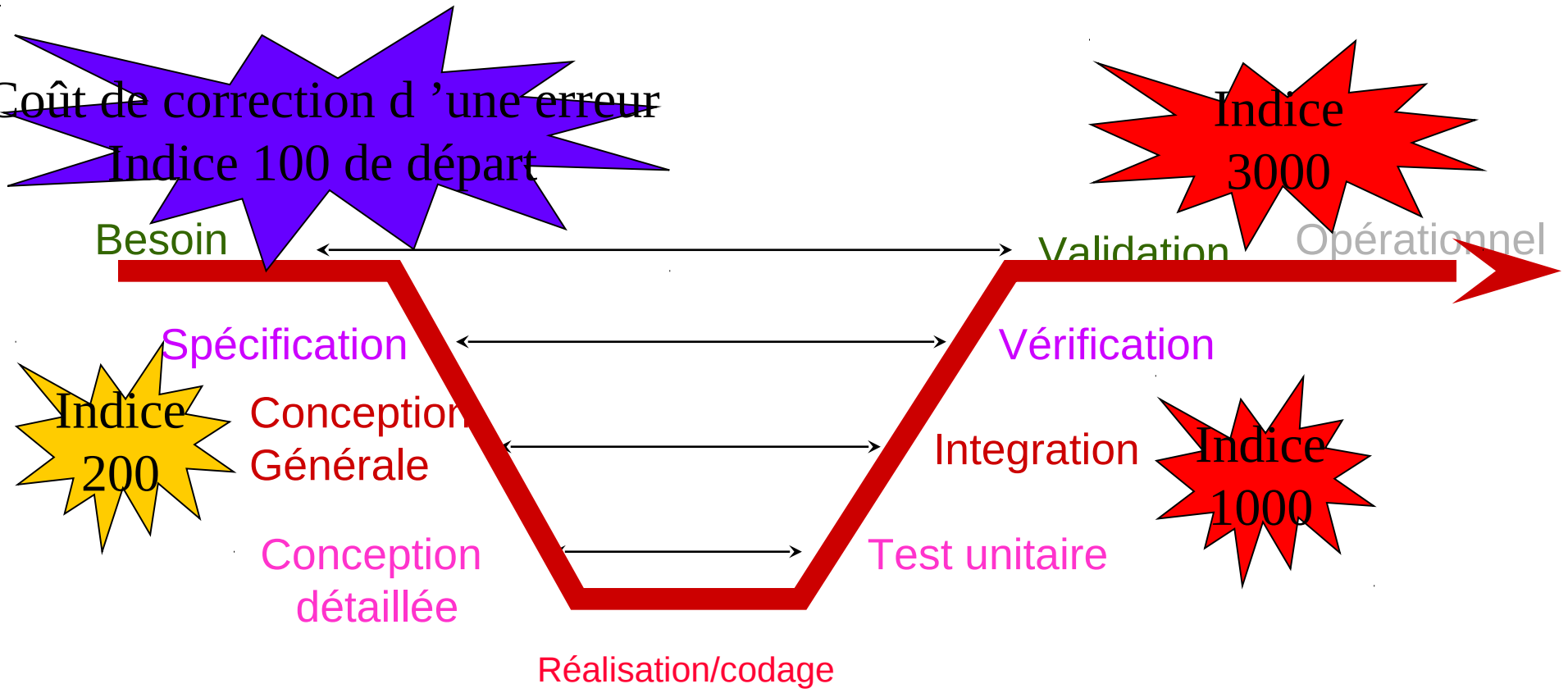
- Se méfier des fonctions qui sont actives quand on n'en a pas besoin.
- Se méfier de l'argument du tout reconduit lorsque des hypothèses changent (conditions d'environnement).

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)

Fiabilité et efficacité

- Fiabilité souvent plus importante qu'efficacité les logiciels non-fiables ne sont pas utilisés
- Le coût d'une erreur peut dépasser largement le coût du système
  - D'après le cabinet de conseil en technologies de l'information Standish Group International, les pannes causées par des problèmes de logiciel ont coûté l'an dernier aux entreprises du monde entier environ 175 milliards de dollars, soit deux fois plus au moins qu'il y a deux ans.

  Le Monde – 23 oct. 2001

Répartition des coûts …

- Répartition des coûts de développement
  - spécification : 6%
  - conception : 5%
  - codage : 7%
  - tests et validation :15%
  - maintenance : 67%
- Coûts de correction des erreurs provenant
  - exigences et spécification : 56%
  - conception : 24%
  - codage : 10%
  - autres : 10%

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).