IFT2255 – Génie Logiciel
Chapitre 1: Introduction

Amal Ben Abdellah
Ing., Dr., Chargée de cours

Département d'Informatique et de Recherche Opérationnelle,
Université de Montréal

Qu'est ce que le génie logiciel?

Génie Logiciel

- Une branche de l'ingénierie associée au développement de logiciels utilisant des principes, méthodes et procédures scientifiques bien définis.
- Le résultat de l'ingénierie logicielle est un produit logiciel efficace et fiable.

Génie

- Consiste à développer des produits, en utilisant des principes et méthodes scientifiques bien définis.

Logiciel

- Collection de code de programmation exécutable, des bibliothèques associées et de documentations.
- Lorsque le logiciel, est conçu pour une exigence spécifique, est appelé un « produit logiciel ».

Le logiciel

- Le logiciel est omniprésent

- Synonymes: programme, application
- Les personnes qui développent le logiciel
  - Ingénieurs logiciel, développeurs logiciel, analystes, programmeurs
  - Possèdent des talents et utilisent des outils qui permettent de développer et faire évoluer les logiciels
    - Talents: créativité, logique, mathématique, méthodologie, résolveur de problèmes
    - Outils: d'autres logiciels (ex: IDE)

Mauvaise ingénierie donne lieu à...

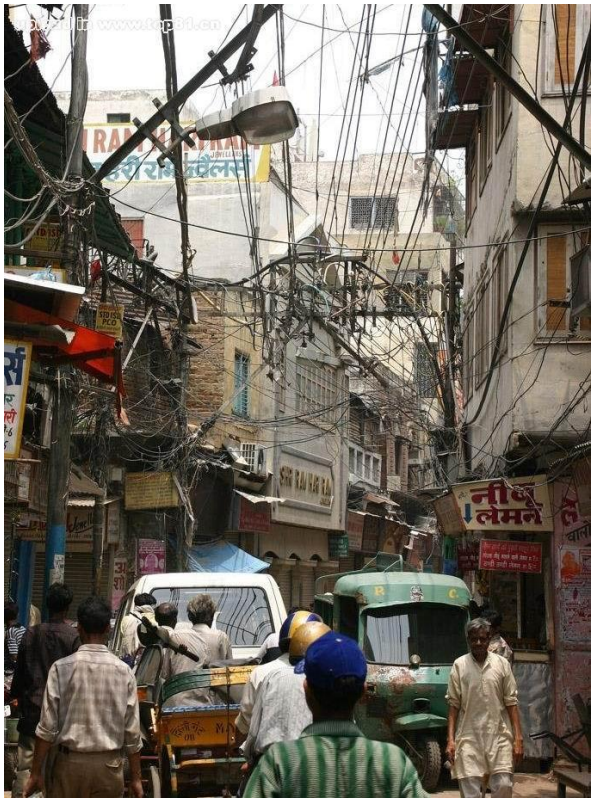
Maison mystère de Winchester
http://www.winchestermysteryhouse.com/

Le résultat de construire continuellement sans avoir pris la peine de concevoir adéquatement.

Résultat:

- Escaliers qui montent au plafond
- Fenêtres au milieu des chambres
- Portes qui s'ouvrent sur un mur

Impossible à maintenir!



- Comment maintenir si quelque chose ne fonctionne plus?
- Comment ajouter de nouvelles connexions ou fonctions?

Même dans le logiciel

- **Simulateur de F16 (1986)**
  - L'avion faisait volte-face à chaque fois qu'il traversait l'équateur
- **Ariane 5 (1996)**
  - Dépassement de capacité dans la conversion d'un entier de 64 bits à 16 bits qui a activé le module d'autodestruction

- **Panne de courant en Amérique du nord (2003)**
  - Problème de concurrence a paralysé le système d'alarme durant plus d'une heure

- **Toyota Prius (2015)**
  - Paramètres du logiciel surchauffent le moteur qui éteint le système hybride durant la conduite

Caractéristiques du logiciel

- Logiciel est **ubiquitaire** dans tous les domaines de métier (affaires, génie, applications scientifiques, musique, médecine…)
- Logiciel s'étend à tous les **aspects de la vie humaine**
  - Simple ou très complexe
  - Pour usage interne ou ciblé pour le grand public
- **But**
  - Dédié à une tâche spécifique, ex: gestion de paie
  - Application qui prend en charge toutes les fonctions d'une organisation
- **Lieu**
  - Exécuté dans un emplacement donné
  - Distribué sur plusieurs machines à travers le monde
- **Exécution**
  - Utilisation unique
  - En lot *(batch)*, sans interaction
  - En temps réel, avec interaction humaine

Difficultés auxquelles font face les programmeurs

Fred Brooks (1987)

- **Complexité accidentelle**: plus facile à résoudre
  - Due aux technologies utilisées
  - Imprévus de l'environnement
  - Problèmes transitoires

No Silver Bullet

Essence and Accidents of Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

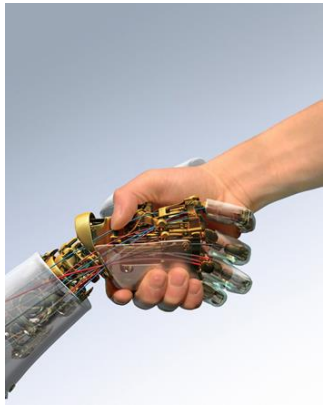
Fashioning complex conceptual constructs is the *essence*; *accidental* tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.

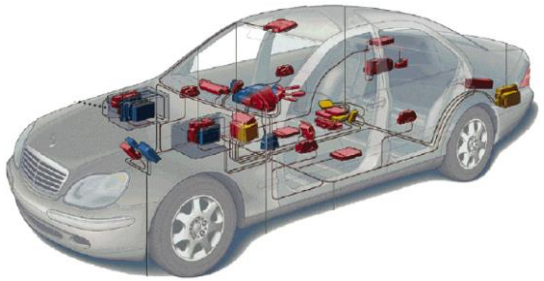
Does it have to be hard?—Essential difficulties

- **Complexité essentielle**: difficultés inhérentes difficiles à résoudre
  - Complexité, invisibilité, versatilité, conformité, discontinuité

Complexité

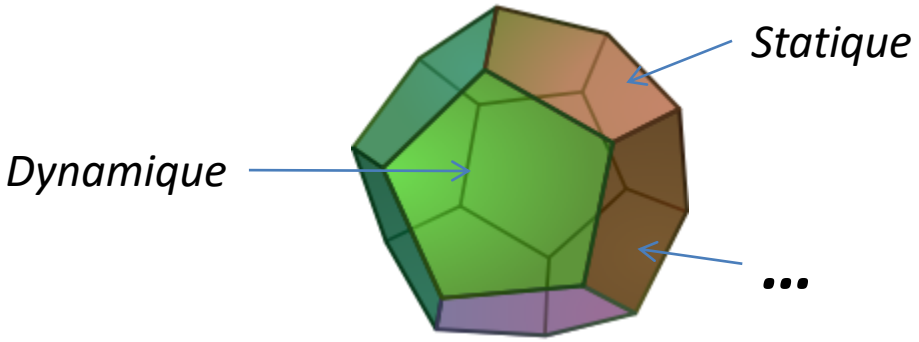
- Notre mémoire à court terme peut accommoder ±7 choses (Miller, 1956)
- Les programmes sont parmi les systèmes les plus complexes jamais créés :
  - Le nombre de composants
  - La diversité de ces composants
  - Le niveau d'interaction et de réaction
  - Le besoin d'intégration dans des systèmes réels et physiques





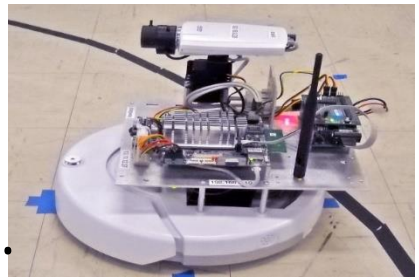
Diviser pour mieux régner

- Différentes facettes d'un système à automatiser (résultat : un joyau ou une bombe à retardement)



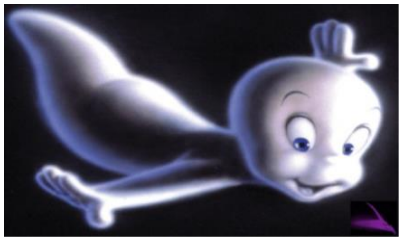
- Chaque facette représente un aspect du système
  - Dynamique
  - Statique
  - …

Conformité



- Gros logiciels composés de matériel hardware, d'usagers, d'interactions avec d'autres logiciels, etc.
  - *Cyber-Physical Systems* mélangent des composants physiques et logiciel, ex: robots
- Logiciel intègre toutes ces parties dans un seul système
  - Préserve l'**unité du domaine**
  - Communique avec les **composants hétérogènes** du domaine
- Logiciel incarne le domaine
  - Le logiciel doit « **représenter** » le domaine
    - Propriétés du domaine s'infiltrent dans le logiciel
  - Besoin de **connaissance/compréhension du domaine**
    - Logiciel doit se conformer au domaine

Invisibilité



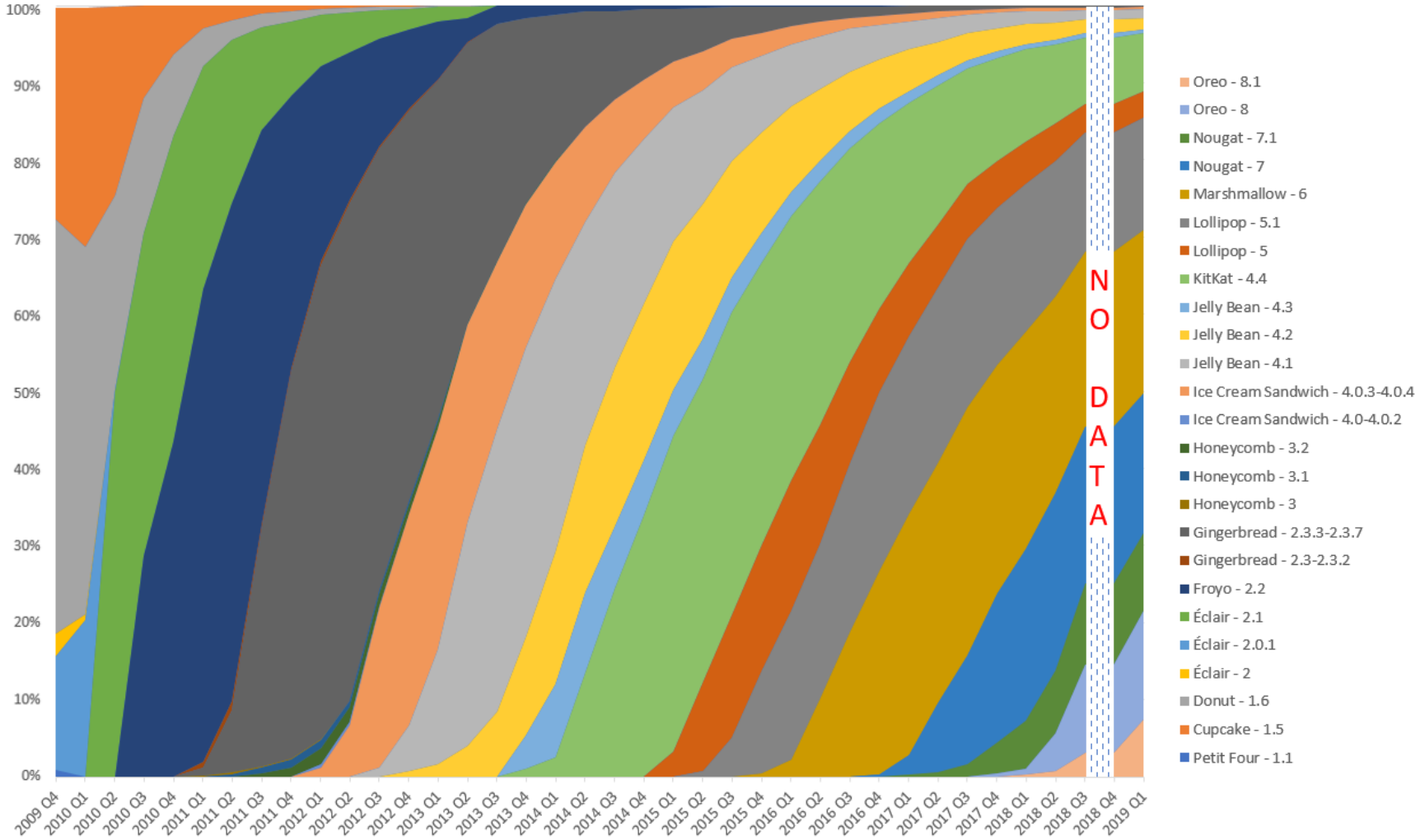
- Pouvez-vous toucher une classe?
- Logiciel est intangible mais il existe des techniques de visualisation.
- Diffère des lois de la physique et des mathématiques (continues)
- Pas moyen de représenter un produit au complet
  - Vues complètes (ex: code) sont incompréhensibles
  - Vues partielles/incomplètes (ex: modèles) et peut-être illusoires
- Nos sens ne peuvent pas être facilement utilisés pour comprendre
  - Visualisation et sonification du logiciel très laborieux

Versatilité

- Logiciels évoluent et changent constamment
  - Changement dans les besoins
- Ce qu'on savait hier peut être obsolète aujourd'hui
  - Nouvelle/meilleure techno émerge continuellement (loi de Moore, 1965 Intel) : nombre de composants par cm^2 double chaque année.
- Le logiciel est censé être facilement modifiable (plus facilement qu'un système physique).
- Mais le défi est de le modifier **correctement**

Évolution en continu de logiciels

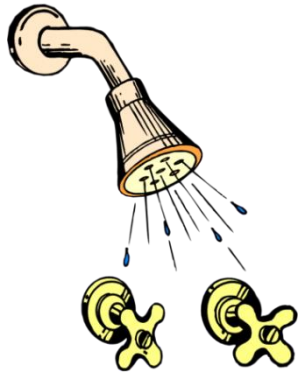
Distribution des versions Android sur les appareils



2018: jusqu'à 11 versions concurrentes

Discontinuité

- L'humain comprend facilement les systèmes linéaires ou semi-linéaires :
  - Tourne un peu le robinet d'eau chaude/d'eau froide.
  - Petites variations de température de l'eau.



- Les logiciels sont discontinus : petits changements en entrée résultent en un énorme changement en sortie :
  - Entre le mot de passe correctement et toutes les fonctionnalités sont disponibles.
  - Fais une petite erreur et l'appli reste fermée

Origines du génie logiciel

- Jusqu'au milieu des années 1960, logiciels construits à l'improviste
- Programmes codés par des experts de divers métiers
  - Ingénieurs matériels, mathématiciens
  - Le temps d'utilisation d'un ordinateur coûtait plus cher (600$/h) que le salaire de ses opérateurs (2$/h)
- Programmes devenaient de plus en plus complexes avec de nouvelles techno, des besoins qui évoluent et la diversité des programmeurs
- Il n'y avait pas de méthodologie pour construire un logiciel et le **changer**
  - Coder puis corriger *(code-and-fix programming)*

Origines du génie logiciel

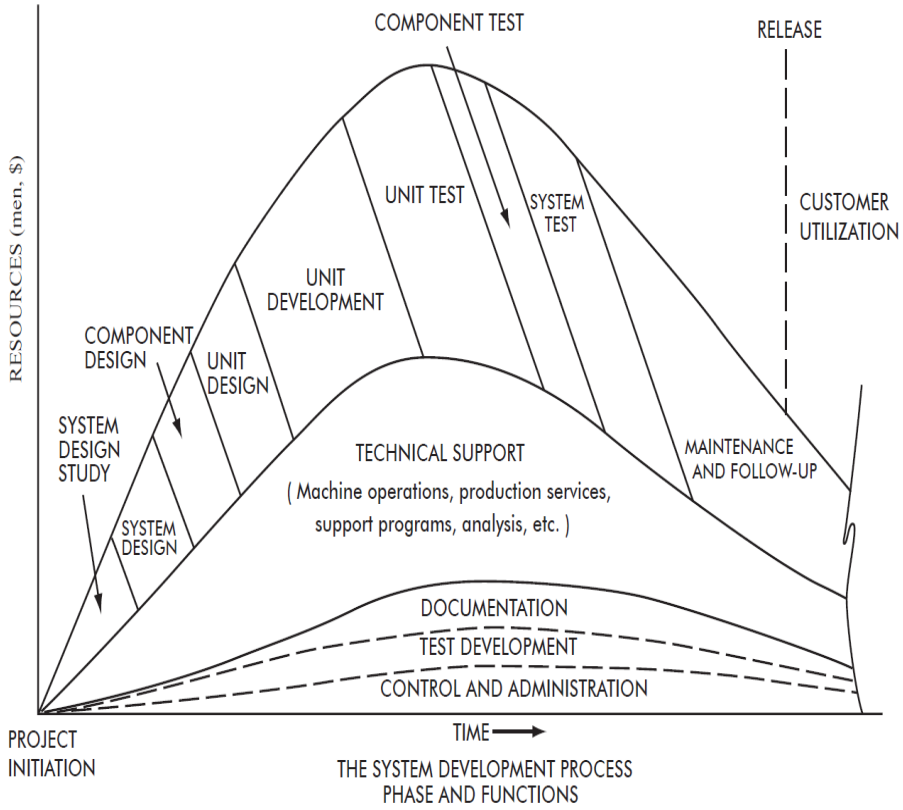
- À la fin des années 1960 éclate la « crise du logiciel ».
- Prise de conscience des difficultés que rencontre le développement des grands projets informatiques :
  - le développement est de moins en moins bien maîtrisé
  - Les équipes de programmeurs sont confrontées à des problèmes de communication que l'encadrement peine à résoudre.
  - le logiciel produit ne répond pas toujours aux attentes car les spécifications ne capturent pas toujours correctement les besoins.
- Sur la suggestion de F. L. Bauer, professeur à l'université technique de Munich, une conférence de travail sur les difficultés de la production de logiciel et les moyens de les surmonter se tient.

Naissance du génie logiciel

- Working Conference on Software Engineering est considérée comme l'événement fondateur qui popularise l'expression software engineering (génie logiciel).
- En 1968, la conférence de l'OTAN se réunit pour discuter d'un nouveau domaine: le génie logiciel (GL)
  - Rendre GL une discipline à part entière
  - Suivant les mêmes méthodologies que les disciplines de génie traditionnelles
- Dijkstra, Naur, Bauer, Perlis, Gries, McIlroy, Randell
- Ils proposent des recommandations sur comment développer du logiciel

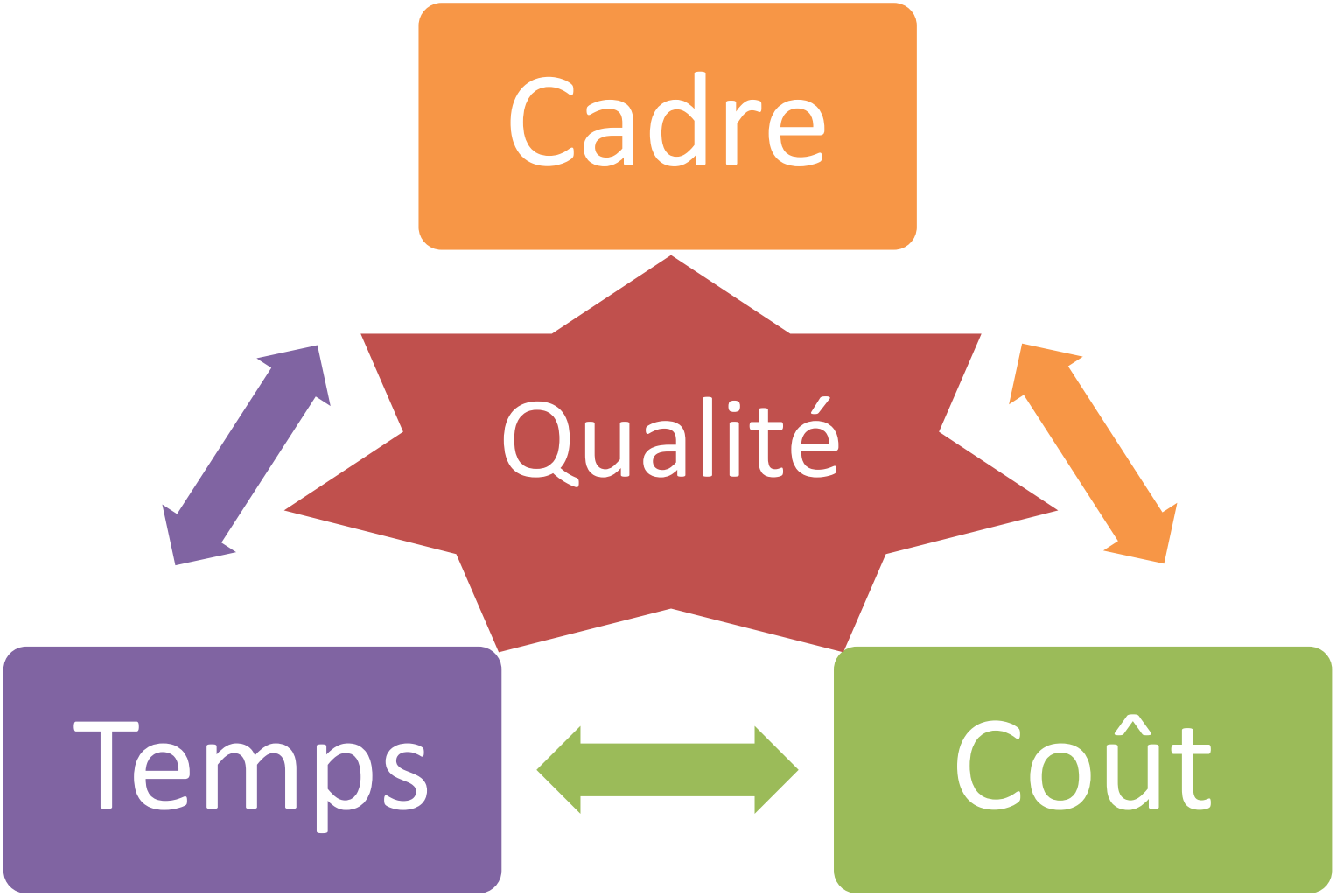
Problème des projets logiciels (Nash)

- Extrait du rapport de la conférence originale de 68 page 20.
- Introduit la terminologie.
- Le plus coûteux durant le développement est les test.
- Maintenance plus long que développement (regarder l'intégrale, pas la hauteur).
- Ce processus n'est pas très différent aujourd'hui !



NATO SOFTWARE ENGINEERING CONFERENCE 1968

Les forces d'impact du logiciel

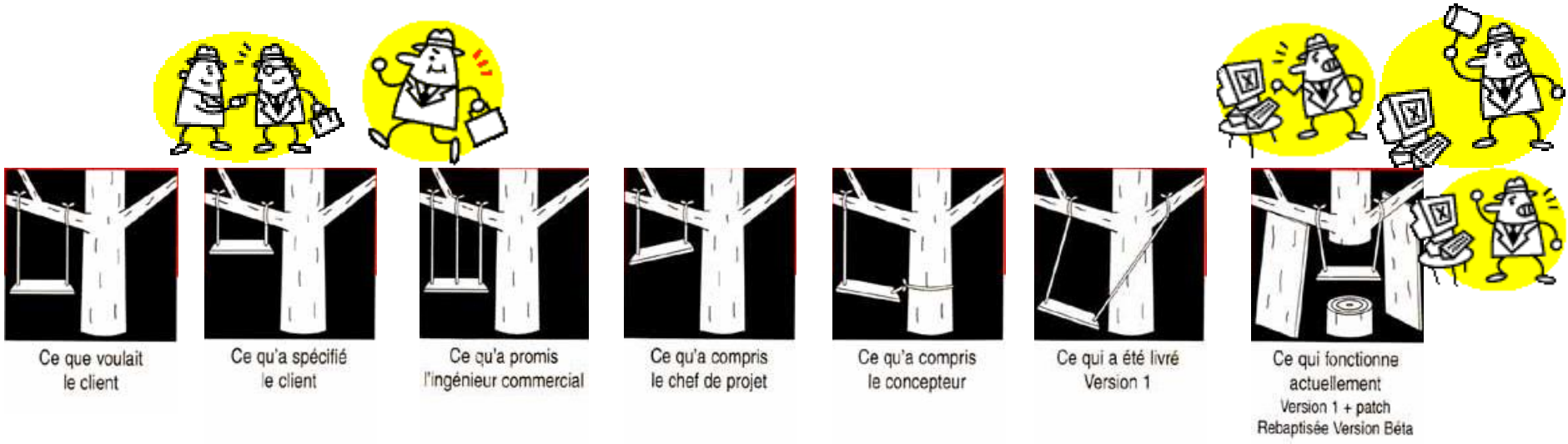


Défis de l'ingénieur logiciel

- Logiciel doit correspondre aux **besoins** du client
  - Tout en fournissant une solution **efficace** au problème
- Logiciel doit être **rentable**
  - Coûts et temps de développement
  - Exigences croissantes
- La solution doit être de **haute qualité**
  - Réduire efforts de maintenance

Utilisateurs & développeurs

Y a-t-il vraiment un dialogue ??



Inspiré de *http://www.bric-a-brac.org/humour/images/informatique*

Les grandes activités d'un ingénieur logiciel

DÉCRIRE: BESOINS, SPÉCIFICATION DE CONCEPTION, DOCUMENTATION

IMPLÉMENTER: CONCEPTION, PROGRAMMATION

ÉVALUER: TEST, VÉRIFICATION, VALIDATION, RÉVISION

GÉRER: PLANIFICATION, ÉCHELONNAGE, COMMUNICATION

FAIRE FONCTIONNER: DÉPLOIEMENT, INSTALLATION, MAINTENANCE