Génie Logiciel

Reda Bendraou
reda.bendraou{{@}}Lip6.fr
http://pagesperso-systeme.lip6.fr/Reda.Bendraou/

Le contenu de ce support de cours a été influencé par les lectures citées à la fin de ce support.

Qu'est ce que le Génie Logiciel?

- "**Software engineering** is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software".

- " **Engineering** is the science, discipline, art and profession of acquiring and applying technical, scientific and mathematical knowledge to design and implement materials, structures, machines, devices, systems, and processes that safely realize a desired objective or inventions ".

Source: Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. p. 1-1. ISBN 0-7695-2330-7. http://www.swebok.org.

Pourquoi c'est si important?

Aujourd'hui, le logiciel contrôle le monde!

- Processus métiers (administrative etc.)
- Gouvernement
- L'industrie (Usines, chaines de fabrication)
- Transports
- Défense, finance, santé...
- Edition, médias...
- Les nouvelles technologies du web, commerce éléctronique...
- Et bien plus!!

Des enjeux aussi bien économiques que politiques

Pourquoi c'est si important?

Les conséquences en cas de problèmes peuvent être très lourdes!

- Therac 25, '85‘ 87 : 6 patients irradies, 2 morts

- Syst. Bagages Aeroport Denver, '95 : 16mois, 3.2 Mds$

- Ariane 5 vol 88/501, '96: 40s de vol, destr., 850 M$

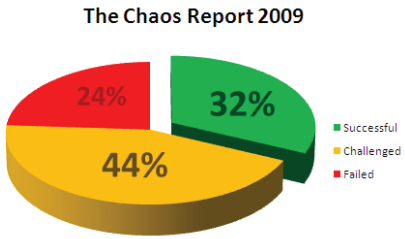
- Mars Climate Orbiter & Mars Polar Lander, '99 : destr.

Le talon d'Achilles du GL

- Le coût
- Le temps
- La qualité

Quelques chiffres!

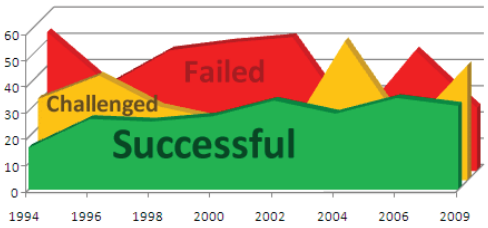
Source: The Standish Group



- **Successful** means on-time, on-budget, and with all features and functions as defined in the initial scope;
- **challenged** means late, over budget, and/or with less features and functions than defined in the initial scope;
- **failed** means cancelled prior to completion, or delivered but never used.

Quelques chiffres!

- Le taux de succès en constante hausse, mais le taux d'échec reste aléatoire



Les difficultés liées au GL

La complexité intrinsèque d'un projet

- l'ingénierie du logiciel est un métier récent en comparaison avec d'autres métiers
  - le fameux parallèle avec le bâtiment

La nature du produit informatique

- de l'information ! copiable, modifiable, malléable, bref « soft »

Les difficultés liées au GL

Les difficultés liées à la nature du logiciel

- un logiciel ne s'use pas, sa fiabilité ne dépend que de sa conception
- mais, pour rester utilisé un logiciel doit évoluer
- pas de direction clairement exprimée,
- changements fréquents,
- contradictions des besoins,...

Les difficultés liées au GL

Difficultés liées aux personnes

- ne savent pas toujours ce qu'elles veulent, ou ne savent pas bien l'exprimer
- communication difficile entre personnes de métiers différents (jargons)
- l'informaticien est souvent perçu comme introverti, peu solidaire du groupe (...ça change...)
- beaucoup d 'autodidactes qui croient savoir...

Les difficultés liées au GL

Les difficultés technologiques

- courte durée de vie du matériel,
- beaucoup de méthodes, de langages
- évolution des outils de développement,...

L'échec d'un projet informatique

Cinq raisons majeures:

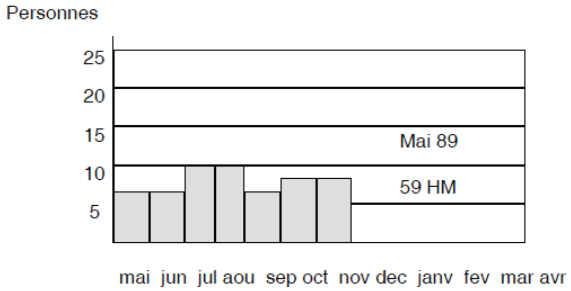
- Engagements irréalistes
- Gestion et conduite de projet inadéquates
- Manque de contrôle
  - pas de planification de la part des développeurs
  - connaissances insuffisantes en gestion de projet
- Technologies inappropriées(méthodes, outils, langages)
- Validation et vérification insuffisantes

Exemple d'un projet

- Projet de télécommunication de taille moyenne.
  - L'exemple date un peu mais malheureusement aujourd'hui encore, il est très facile de trouver des exemples similaires
- Prix forfaitaire.
- Pas de mesure de productivité, ni d'évaluation des projets antérieurs du contractant.
- Pas d'estimation de la taille du projet et de ses sous-systèmes.

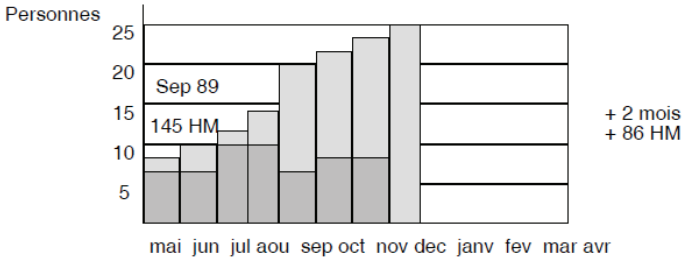
Exemple d'un projet

Charge prévisionnelle



Exemple d'un projet

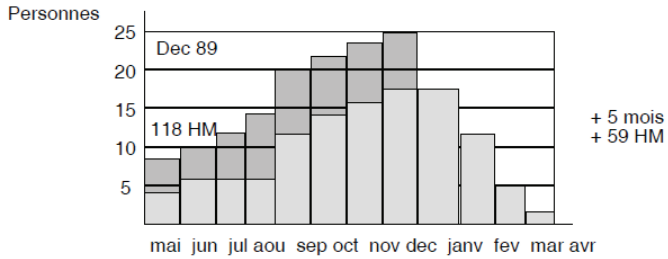
5 mois plus tard ...



Exemple d'un projet

Encore 3 mois après...

- Aucune information sur ce qui est réalisé...
- La *confiance* diminue...
- Audit...



Exemple d'un projet

Analyse de la situation

- 23 sous-systèmes avaient été isolés. Une *estimation* des tailles minimale, probable et maximale est réalisée.
- Le projet fait 67597 ± 1596 lignes.
- L'état courant après 8 mois ...
  - 2 sous-systèmes en installation et
  - 21 en conception détaillée ou codage.
- Les *productivités* souhaitées étaient :
  - en mai, 19
  - en septembre, 17
  - en décembre, 15

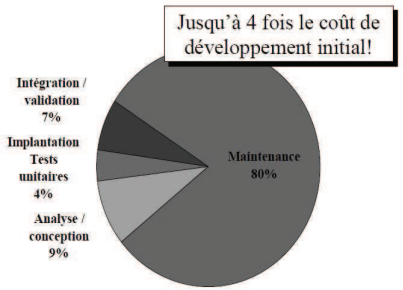
Exemple d'un projet

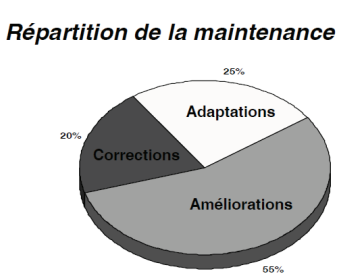
Résultat

- La productivité réelle est réévaluée à 7(-8)
- La durée du projet est réévaluée à 30 mois...(+18)
- La charge est réévaluée à 420 HM... (+300)
- Le budget a augmenté de 4 500 000 $

Ce n'est pas fini! La Maintenance

- Maintenance du projet
  - Le logiciel livré ne répond pas aux exigences du client!
  - Evolution, adaptation
- Une part très importante du coût d'un projet





Résumons!

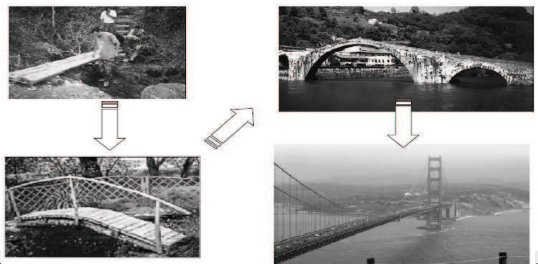
- L'Objectif du GL est de:
  - Améliorer la qualité
  - Réduire les délais
  - Optimiser les coûts
- Face à cela 3 principaux défis du logiciel d'aujourd'hui (diapos suivantes!)

Défis et obstacles du logiciel d'aujourd'hui

- **Composants de plus en plus complexes**
  - Problème de la validation intra-composants
  - Le coût très élevé des programmes prouvés
    - Preuve au moins aussi complexe que le code
    - Autant de chances de se tromper dans la preuve...
    - En pratique :
      - Réservé à des (petits) sous-systèmes (très) critiques
      - Recours aux tests pour le reste

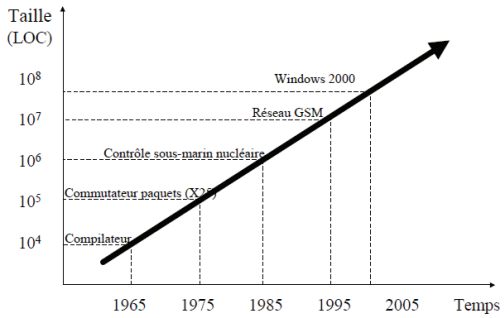
Défis et obstacles du logiciel d'aujourd'hui

- **Applications de plus en plus larges/distribuées**
  - Problème de la validation inter-composants
  - Problème de gestion des ressources (humaines et matérielles)
    - Avoir une ressource qualifiée, utiliser la bonne techno
    - Conduite et suivi du projet
      - Utiliser le bon processus, planification et gestion des délais, offshore...



Défis et obstacles du logiciel d'aujourd'hui

- **Applications de plus en plus larges/distribuées**
  - Le nombre de lignes de code ne cesse d'augmenter



Défis et obstacles du logiciel d'aujourd'hui

- **Des besoins qui évoluent en cours de route**
  - Problème de maintenance **évolutive** et **corrective**
  - Nokia rapporte à propos de son infrastructure GSM
    - 50% des exigences (besoins numérotés) ont changé *après* le gel du cahier des charges
    - 60% de ceux-ci ont changé au moins 2 fois!
    - C'est le cas général plutôt que l'exception
    - Cahier des charges figé = rêve des années 70

Des solutions?

Quelques solutions que nous allons aborder dans ce cours

- **Comment gérer la complexité: (3 séances)**
  - La modélisation
    - Modèle Vs. Code
    - UML: The Unified Modeling Language
    - La documentation
- **Comment augmenter la productivité: (1 séance)**
  - La génération de code & Le reverse engineering
    - Java comme langage de référence
  - La vision MDE (Model-Driven Engineering)
- **Comment améliorer la qualité et la fiabilité: (2 séances)**
  - Designs patterns & Refactoring (qualité du design)
  - Les tests (loin d'être exhaustifs!) (fiabilité de l'application)
- **Comment maîtriser les délais et réduire les risques: (1 séance)**
  - Les processus, méthodes et cycles de développement
    - Conduite de projets, Process Agiles Vs. Process lourds
    - Outils de documentation de procédés et de suivi de projets (Eclipse Process Framework)

Modalités du cours

- **8 séances** de **cours (2h30)**, présence obligatoire et notée!!
- **8 séances** de **Tds (2h)**, présence obligatoire!
  - Exercices au tableau
- **8 séances** de **Tps (3h)**, présence obligatoire!
  - Exercices sur machine
- **1 Projet** comme fil conducteur, à faire pendant le tp (en plus des exos) et à rendre (code + modélisation UML + documentation)
- **1 Partiel**
- **1 Examen**
- Un TP solitaire (en option)
- Note CC=Partiel*60%+Projet *30%+[TP Solitaire]*10%
- **Note Finale= Note Examen *60% + Note CC*40%**

Lectures

- Software Engineering,
  - Ian Sommerville, Addison Wesley; 8 edition (15 Jun 2006), ISBN-10: 0321313798
- The Mythical Man-Month
  - Frederick P. Brooks JR., Addison-Wesley, 1995
- Cours de Software Engineering du Prof. Bertrand Meyer à cette @:
  - http://se.ethz.ch/teaching/ss2007/252-0204-00/lecture.html
- Cours d'Antoine Beugnard à cette @:
  - http://public.enst-bretagne.fr/~beugnard/

- UML Distilled 3rd édition, a brief guide to the standard object modeling language
  - Martin Fowler, Addison-Wesley Object Technology Series, 2003, ISBN-10: 0321193687
- UML2 pour les développeurs, cours avec exercices et corrigés
  - Xavier Blanc, Isabelle Mounier et Cédric Besse, Edition Eyrolles, 2006, ISBN-2-212-12029-X
- UML 2 par la pratique, études de cas et exercices corrigés,
  - Pascal Roques, 6ème édition, Edition Eyrolles, 2008
- Cours très intéressant du Prof. Jean-Marc Jézéquel à cette @:
  - http://www.irisa.fr/prive/jezequel/enseignement/PolyUML/poly.pdf
- La page de l'OMG dédiée à UML: http://www.uml.org/

- Design patterns. Catalogue des modèles de conception réutilisables
  - Richard Helm (Auteur), Ralph Johnson (Auteur), John Vlissides (Auteur), Eric Gamma (Auteur), Vuibert informatique (5 juillet 1999), ISBN-10: 2711786447