ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

01
Chapitre

Introduction au génie logiciel

1I2AC1 : Génie logiciel et Conception orientée objet

Régis Clouard, ENSICAEN - GREYC

« Si les ouvriers construisaient les bâtiments
comme les développeurs écrivent leurs programmes,
le premier pic-vert venu aurait détruit toute civilisation. »
Gerald Weinberg

Plan du chapitre

1

Pourquoi un cours sur le génie logiciel ?

Formation informatique à l'ENSICAEN

- Ingénieur informatique ENSICAEN
  - Développeur / Architecte logiciel

Confusion programmeur / développeur

- Aujourd'hui tout le monde programme…

Un programmeur réalise des œuvres personnelles à usage personnel...

... plus ou mois bien réussies.

Confusion programmeur / développeur

Un développeur produit des ouvrages d'art pour un ensemble de personnes ...

..qui sont hors de portée d'un programmeur.

Confusion programme / logiciel

- Programme
  - 1 utilisateur averti et bienveillant (généralement son créateur)
  - 1 cas d'utilisation.
  - Petite taille.
  - Simple.
  - Conséquences de l'utilisation assumées par l'utilisateur.
- Logiciel
  - Tout utilisateur final.
  - Tous les cas d'utilisation prévus.
  - Grande taille.
  - Complexe.
  - Conséquences de l'utilisation portées par les développeurs.

Logiciel : utilisateur

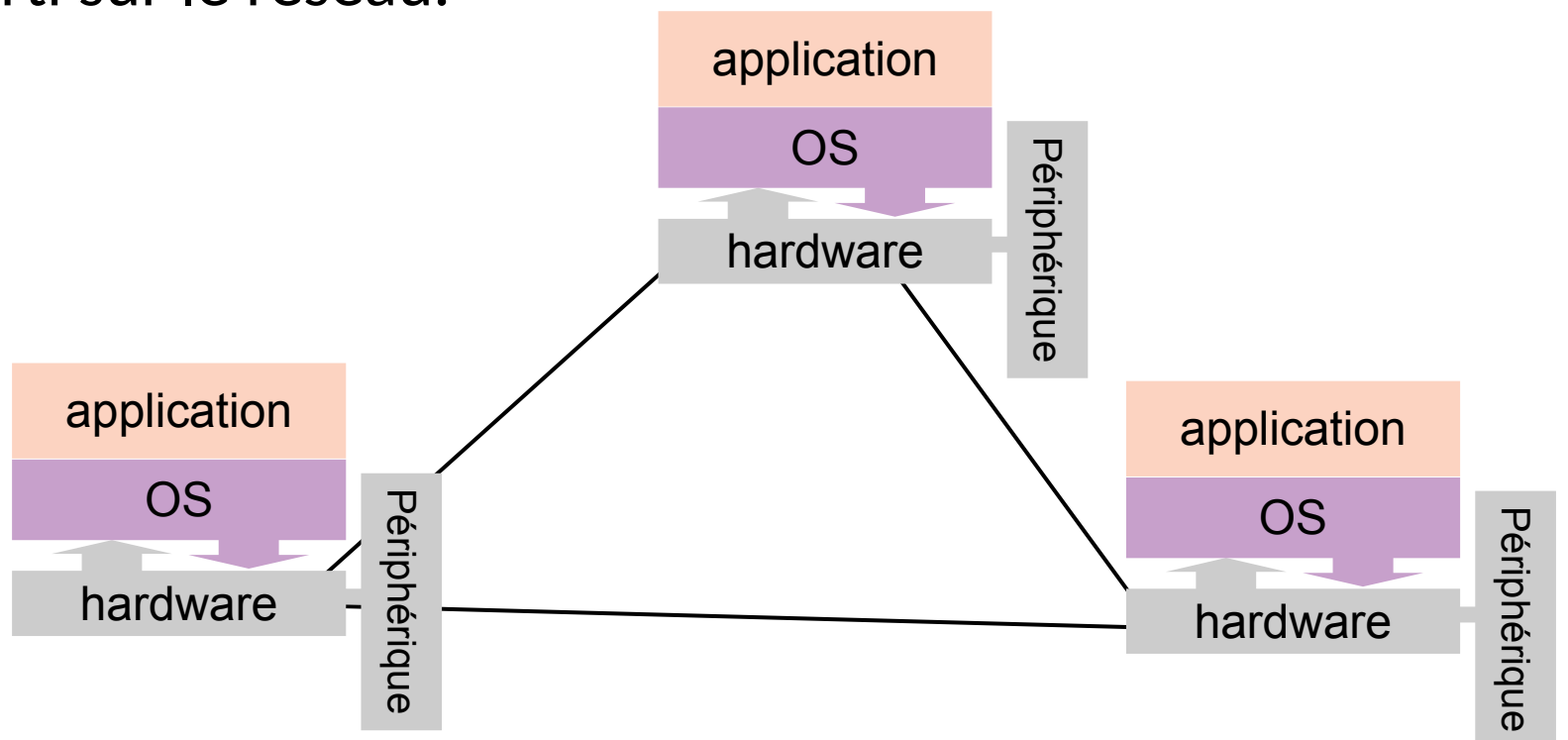
- Tout utilisateur final plus ou moins formé.
- Un logiciel doit présenter une interface ergonomique rendant son utilisation « naturelle » et «efficace».

Logiciel : cas d’utilisation

- Tous les cas d’utilisation doivent être prévus :
  - Cas nominaux.
  - Cas dégénérés.
  - Cas frauduleux.

Logiciel : complexité

- Portable sur différents OS et matériels.
- Interfaçant des périphériques d'entrée et de sortie.
- Réparti sur le réseau.



Logiciel : taille

- Unité de mesure de la taille d'un logiciel
  - LOC : lines of code (MLOC : 10^6 LOC → 40 romans épais).
- La taille des logiciels explose :
  - Commandes de vol A380 : 1 MLOC (contre 100 KLOC pour l'A320).
  - Jeu World of Warcraft : 5,3 MLOC (jeu de rôle en ligne massivement multijoueur).
  - OS Android : 11,8 MLOC.
  - Noyau Linux 3.1 (2011) : 17 MLOC.
  - Facebook : 62 MLOC.
  - Windows 10 : 80 MLOC (contre 40 MLOC Windows 7).
  - Google (tous les services internet) : 2 GLOC en 2015.

Conséquence : développement en équipe

- La taille des logiciels oblige à un travail en équipe.
  - Mesure en années-homme (man-year) :
    - x années-hommes = x années pour 1 homme ou x hommes en 1 année ou toute autre combinaison.
- Par exemple : le coût de développement de l'algorithme de recherche de Google est estimé à 1000 années-hommes.

Conséquence : coût de développement

- Ordre de grandeur :
  - 1 année-homme ~= 1650h.
  - 1h ~= 50€.
  - Productivité ~= 2 à 5 LOC / h.
- Donc, le code suivant :

```
static void bubbleSort( int[] array ) {
   for (int i = 0; i < array.length - 1; i++) {
      for (int j = 0; j < array.length - i - 1; j++) {
         if (array[j] > array[j + 1]) {
             swap(array, j, j + 1);
         }
      }
   }
}
```

- met 1h à traverser tout le cycle de développement et coûte 50€ !

Logiciel : responsabilisation

- Les développeurs doivent limiter voire proscrire toutes les conséquences néfastes de l'utilisation de leur logiciel.

Conséquence néfaste comique

Ordinateur de bord de la Citroën C5 d'un collègue.

« *Mise à jour du système en cours. Veuillez ne pas arrêter le moteur pendant les 25 prochaines minutes.* »

Conséquence néfaste préoccupante

Perte de la sonde Mars Climate Orbiter (1999)
lors de son entrée dans l'orbite de Mars.

C'est une erreur de traduction entre systèmes d'unités anglo-saxons et métriques pour le calcul de sa trajectoire qui est à l'origine de la défaillance.
Coût de 327 millions de $.

Conséquence néfaste tragique

Mort tragique de Lydia Cohen, 72 ans à l'hôpital de Versailles en novembre 2011.

Son allergie à un antibiotique, l'amoxicilline, était bien notée dans son dossier médical, mais le logiciel utilisé pour les prescriptions n'a pas intégré cette donnée.

Développeur / Architecte logiciel

- Un métier à haut niveau d'expertise reposant sur le génie logiciel.
- Un métier qui s'apprend.

Plan du chapitre

1

Pourquoi un cours sur le génie logiciel ?

2

Première génération du génie logiciel

La crise du logiciel

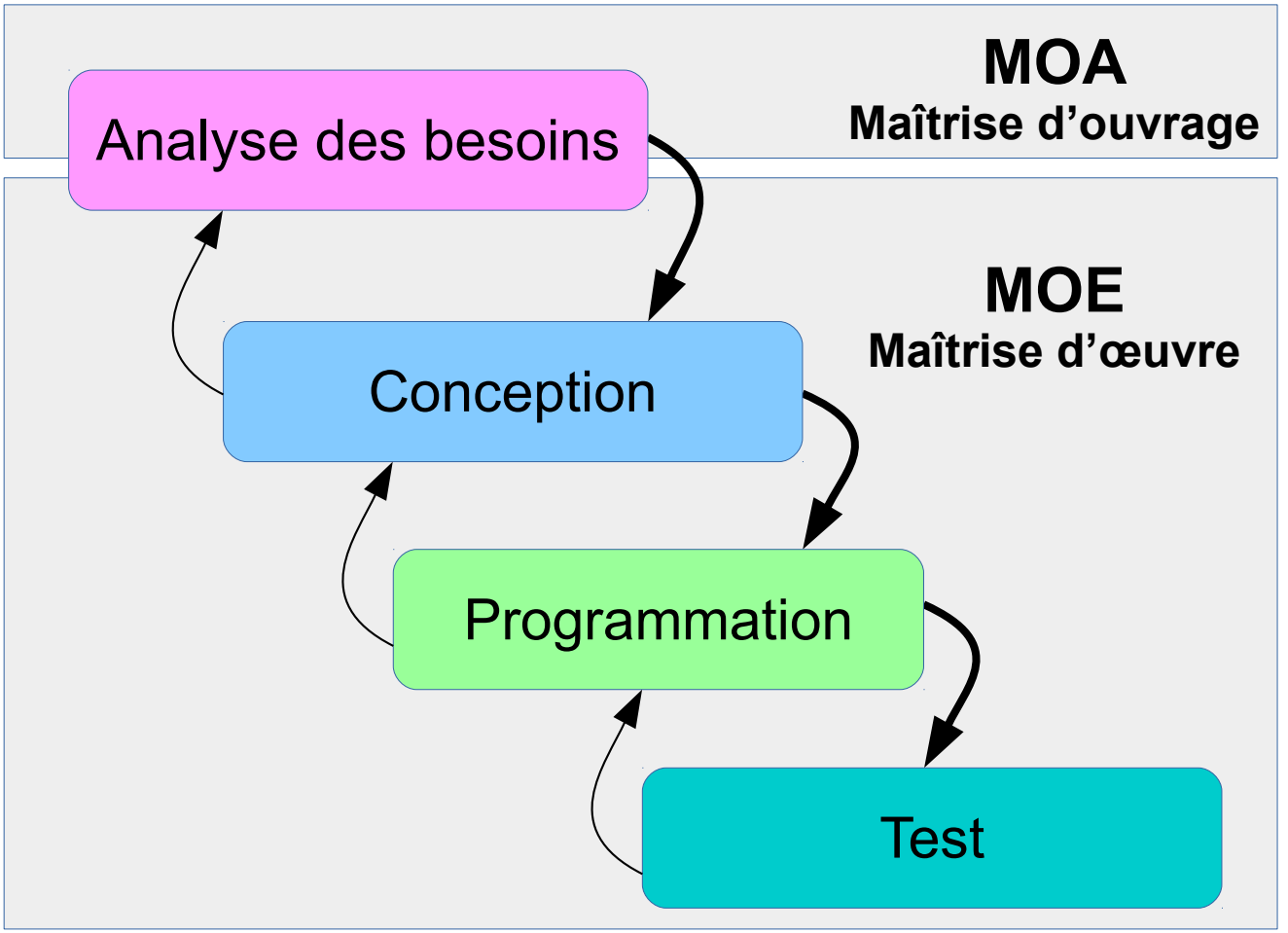
- Expression née d'un constat en 1968 :
  - Il est incroyablement difficile de réaliser dans les délais prévus des logiciels satisfaisant la qualité attendue.
  - La taille et la complexité conduisent à une incapacité à maîtriser le logiciel.
- Raison majeure
  - Il n'existe pas de théorie générale de la construction de logiciel.
    - « *Nous sommes toujours à la recherche d'une théorie générale de la construction de logiciels, à l'image des équations de la mécanique classique qui permettent de concevoir un pont robuste.* » Joseph Sifakis (prix Turing 2007).

Une réponse : le génie logiciel (1968)

- Le génie logiciel* est l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.
- Concrêtement, le génie logiciel est défini par :
  - **Une méthode**
    - Découper le développement en étapes et introduire du contrôle entre les étapes.
    - Organiser le travail en équipe.
  - **Un paradigme**
    - Fournir les briques de base de la conception et les mécanismes pour les assembler.
  - **Un formalisme**
    - Fournir un langage pour parler du code des logiciels de manière abstraite et non-ambiguë.

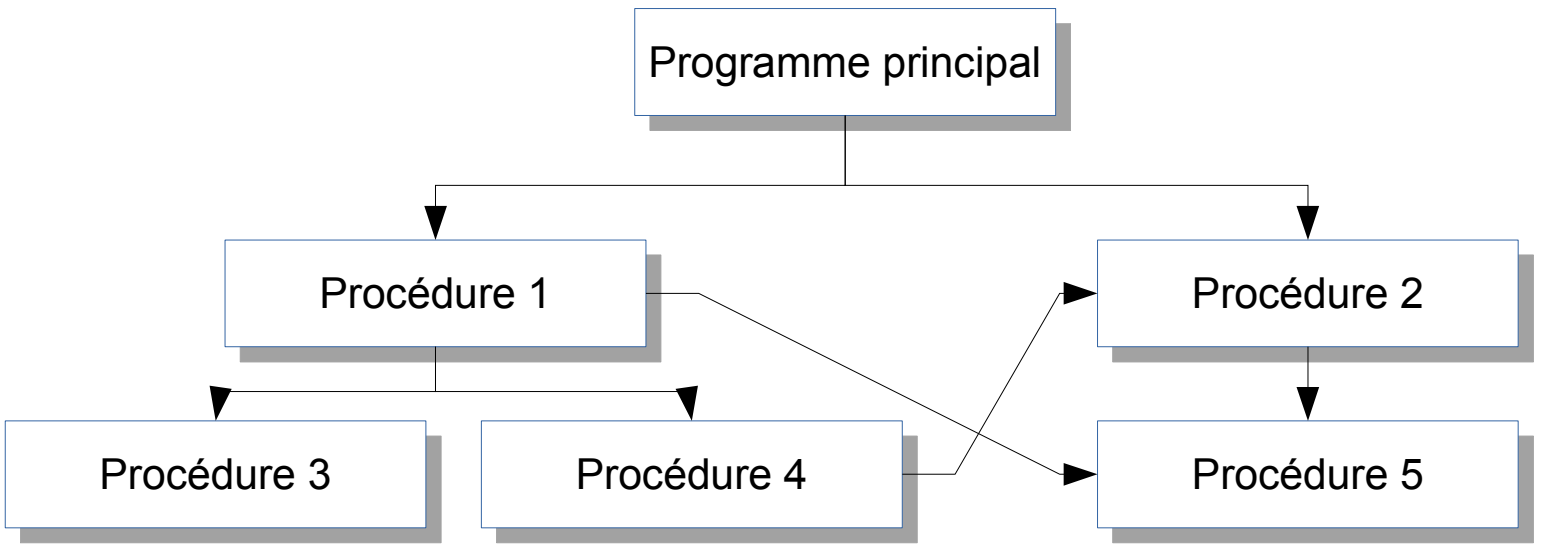
* le terme anglais *software engineering* est du à **Margaret H. Hamilton**, créatrice du système embarqué du programme spatial Apollo et l'une des pionnières du génie logiciel.

Une méthode : le cycle en cascade



Un paradigme : conception procédurale

- Un programme est conçu comme une suite finie de procédures plus ou moins élémentaires qui s'enchaînent.
  - brique : procédure paramétrée.
  - assemblage : appel de procédures avec la valeur des paramètres.



Un formalisme : algorithmique

- Langage semi-formel incluant :
  - Structures de données.
  - Séquence d'instructions organisée par des structures de contrôle.
  - Analyse de complexité.

INSERTION-SORT(A)		*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4	$i = j-1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$

Ingénierie logicielle

- Il en résulte deux métiers :
  1. **Analyste**
     - Analyse des besoins. } Rédaction du cahier des charges.
     - Spécification fonctionnelle. } Rédaction de la documentation technique.
     - Conception générale.
     - Conception détaillée.
  2. **Programmeur**
     - Écriture du code.
     - Écriture des tests.
     - Déploiement.

Plan du chapitre

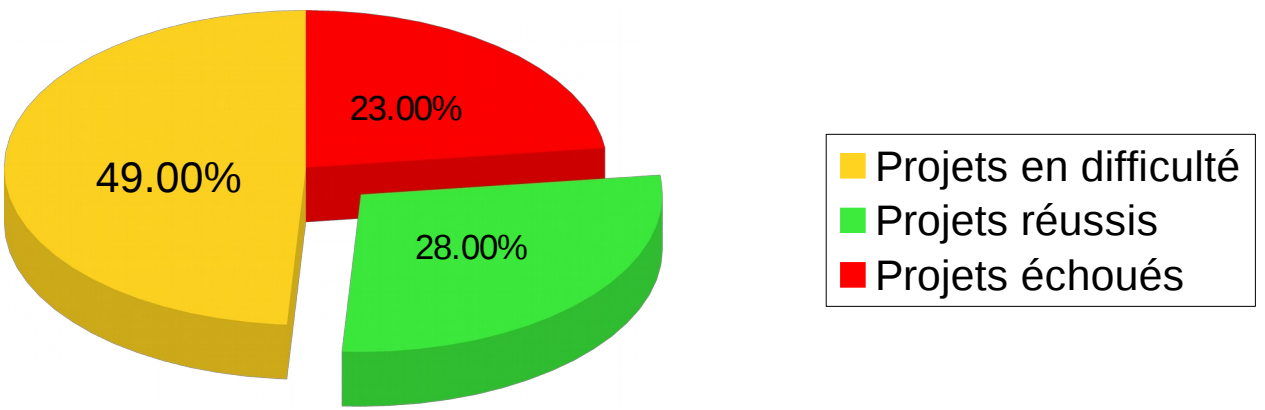
1
Pourquoi un cours sur le génie logiciel ?

2
Première génération du génie logiciel

3
Constat d'échec

Constat d'échec (2001)

- Malgré l'application du génie logiciel, plus des deux tiers des projets non satisfaisants.



Source : The Standish group (2000)

35.000 projets étudiés

Exemples de projets en échec

- Le projet TAURUS
  - La Bourse de Londres a renoncé en mars 1993, après quatre ans de développement, au projet informatique Taurus qui devait assurer le suivi complet de l'exécution des transactions. Ce système a coûté directement 60M£ et les opérateurs sur le marché ont dépensé 400M£ pour y adapter leurs propres logiciels.
- Système de paiement d'Atos
  - La saturation du système d'autorisation de paiement dépassant 100€ a provoqué en pleine en pleine période d'achats de Noël 2001 de longues files d'attente de clients excédés dont beaucoup finiront par abandonner leurs chariots. Les autorisations de débit qui prenaient habituellement quelques dizaines de secondes, nécessitèrent ce jour-là quasiment une demi-heure. Le coût du préjudice pour le seul groupe Leclerc est de 2M€.

Cause n°1 : suivre un plan coûte que coûte

- Le cycle en cascade définit des étapes séquentielles précises.
  - Engagement sur plusieurs mois.
    - cf. diagramme prévisionnel de Gantt.
  - Il n'y a pas (ou peu) de remise en cause possible.
- Conséquence
  - Si l'une des étapes en amont est imparfaite, les étapes en aval seront imparfaites et conduiront à l'échec du projet.

Cause n°2 : estimer la durée d'un projet

- Il est extrêmement difficile (voire impossible) d'estimer la durée/coût d'un projet informatique.
  - Il y a trop d'aléas.
- *Remarque : si vous êtes dans l'obligation de donner une estimation de la durée d'un projet ou d'une tâche, voici une règle empirique :*
  - *Estimer honnêtement le temps de développement.*
  - *Multiplier par 3.*
  - *Passer à l'unité supérieure.*

Cause n°2 : estimer la durée d'un projet

- Mythe du année-homme (cf. «The Mythical Man-Month », Fred Brooks, 1995)
  - Non-linéarité de la charge de travail (source *Borland Software Corporation*) :

Taille équipe	Productivité par personne (KLOC/année)	Productivité de l'équipe (KLOC/année)	Gain
1	15.0	15.0	
2	11.9	23.8	1.6
10	7.0	69.6	4.6
25	5.1	128.2	8.5

- Contrairement à l'intuition, ajouter des personnes à une équipe d'ingénieurs ne permet pas de rattraper le retard, au contraire.
  - Les nouvelles personnes doivent être formées et informées sur le logiciel en cours de construction par les autres ingénieurs, ce qui entraîne des retards supplémentaires (loi de Brooks).

Cause n°3 : définir les besoins au début

- Les clients ne savent pas identifier exactement ce qu'ils veulent.
- Les clients ne savent pas exprimer clairement les besoins identifiés.
- Les besoins des clients changent au cours même du projet.

Cause n°4 : réaliser les tests à la fin

- Les tests sont généralement la variable d'ajustement du temps.
  - Étant réalisés à la fin, les tests sont faits selon le temps restant ou pas.
- Pourtant, il est impossible de garantir des logiciels sans défaut.
  - Estimation : 1 à 10 bugs / KLOC
    - Windows 10 (80 MLOC) → 80,000 bugs !
  - En 2006, 500 bugs déclarés dans la station spatiale internationale.

Cause n°5 : raisonner au niveau procédure

- La conception procédurale est inadaptée à la conception de programmes de très grande taille.
  - Le niveau procédure est trop bas pour appréhender la complexité des logiciels d'aujourd'hui.
    - Programmer au niveau procédural est une activité complexe qui revient à construire une voiture au niveau moléculaire.

Plan du chapitre

1
Pourquoi un cours sur le génie logiciel ?

2
Première génération du génie logiciel

3
Constat d'échec

4
Le génie logiciel aujourd'hui

Constat

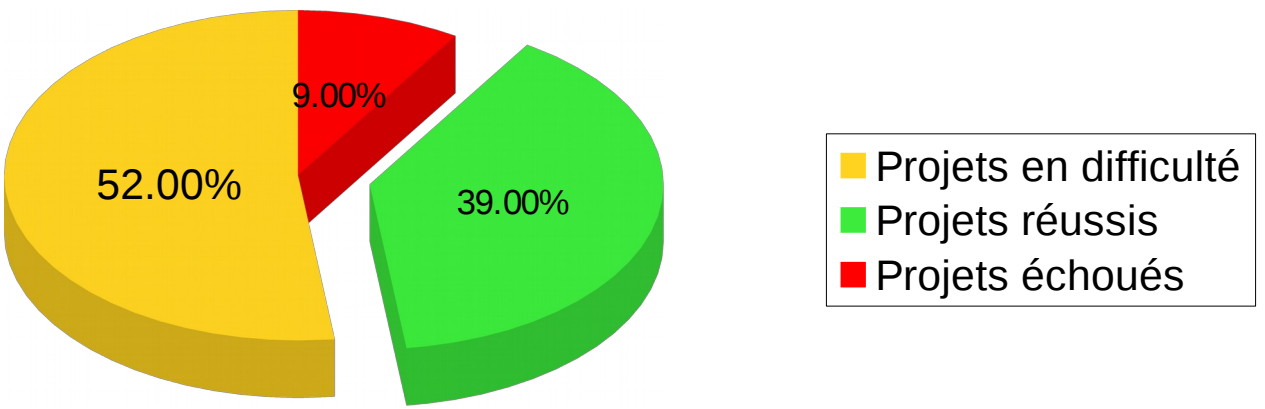
- Le logiciel n'est pas le matériel.
  - On peut construire une tour de 52 étages en commençant par la salle de bain du 12^{e} étage.
  - On peut installer le chauffage par le sol après avoir mis le carrelage.
- Il doit donc être possible de repenser le génie logiciel en s'émancipant du génie civil.

Repenser le génie logiciel

- Nouvelle vision du génie logiciel : Agilité
  - Nouvelle définition : Le génie logiciel est l'**art** et l'ensemble des moyens techniques, industriels et humains qu'il faut réunir pour construire, distribuer et maintenir des logiciels.
  - Ajoute une dimension artisanal au développement (*software craftsmanship)*.
- Depuis quelques années, le génie logiciel a beaucoup changé y compris au niveau de la programmation.
  - Nouveau paradigme de conception logicielle.
    - **Conception orientée objet**.
  - Nouveau formalisme de modélisation.
    - **UML**
  - Nouvelle méthode de gestion de projet.
    - **Méthode itérative et incrémentale**

Effet sur la réussite des projets

- Amélioration (lente mais réelle) … le temps que l'agilité mature dans les équipes.
  - Rappel en 2000 : 29 % en échec / 28 % réussis / 49 % en difficulté.



Source : The Standish group (2015)
10.000 projets étudiés

Remarques

- Le paradigme procédural et l'algorithmique sont encore des principes de base du développement.
  - Mais, ils sont localisés dans les aspects plus spécialisés.
- Il n'y a plus de séparation entre les métiers analyste et programmeur.
  - Un seul métier : ingénieur développeur.
    - Les ingénieurs qui composent une équipe travaillent sur tout le cycle de vie d'une application depuis l'analyse des besoins, la conception, la programmation et les tests.
    - Aujourd'hui, l'ingénieur développeur est aussi en charge de la production voire du déploiement du logiciel chez le client.
    - On parle de **DevOps** pour décrire toute la chaîne complète.

Plan du chapitre

1
Pourquoi un cours sur le génie logiciel ?

2
Première génération du génie logiciel

3
Constat d'échec

4
Le génie logiciel aujourd'hui

5
Cours de génie logiciel

Plan du cours

- Ce cours est accès sur la conception et la programmation :
  - Paradigme objet.
  - Langage UML avec ses principaux diagrammes.
  - Processus de développement itératif et incrémental.
  - Qualité du code.
  - Tests logiciels.
- Les méthodes de développement seront vues en 2^{e} année.

Pourquoi ce cours ?

- À l'issue du cours, vous serez en mesure de :
  - Concevoir un logiciel de taille respectable en équipe.
  - Comprendre une conception modélisée.
  - Évaluer la qualité d'une conception.
  - Mettre en application les principes agiles du développement.
- Important
  - « Les livres d'art ne permettent pas de vous transformer en artistes, pas plus que les livres sur le génie logiciel ne vous transforment en développeur. Ils ne peuvent que vous apporter les outils, les techniques et les processus de réflexion employés par d'autres développeurs qui sont sources d'inspiration. » Robert Martin.
  - Il existe aujourd'hui 50 ans d'expérience en génie logiciel dont il est nécessaire de s'inspirer. Le développement logiciel est un domaine qui souffre de beaucoup d'incompétence avec beaucoup d'autodidactes qui croient savoir.

Pour qui ce cours ?

- Pour ceux qui choisiront de développer (MOE)
  - Trouver sa place dans la gestion de projet informatique.
  - Forger une culture du développement logiciel de haut niveau.
  - Prendre conscience de l'importance du code et des tests.
- Pour ceux qui choisissent de ne pas développer (MOA, Conseil)
  - Être en mesure d'exprimer des besoins et de suivre un développement de logiciel.
  - Comprendre comment sont construits les ouvrages à spécifier et apprécier leurs contraintes.
  - Gagner en crédibilité face aux personnes de la MOE.

Que retenir de ce chapitre ?

- Le développement de logiciels est plus que la programmation d'applications.
- Le développement de logiciels de grande taille ne peut se faire sans maîtrise du génie logiciel.
- Le génie logiciel reste de l'artisanat qui repose sur une forte culture théorique et pratique de 50 ans.
- Le génie logiciel a beaucoup évolué depuis quelques années. Il est fortement influencé par l'agilité.
- Aujourd'hui le génie logiciel définit :
  - Un paradigme de conception basé sur la notion d'objet.
  - Un formalisme de modélisation basé sur le langage UML.
  - Une méthode pour structurer le processus de développement et organiser le travail en équipe basé sur l'agilité.