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

Organisation de l'enseignement

- Volume horaire
  - CM : 11 h
  - TD : 14 h
  - Pas de TP, mais
    - Lien avec le cours « *Java et Programmation objet* »
    - Katas et Coding dojos
- Plateforme pédagogique (course n°60)
  - Ressources du cours
    - Polycopié par chapitre
    - Présentation par CM
    - Katas et Coding Dojos

Organisation de l'enseignement

- Examen
  - Le cahier de TD tient lieu d'annales d'examen
  - Il portera sur tout le polycopié (*les diapositives n'en présentent qu'une partie*)
  - Document autorisé : une feuille A4 recto/verso manuscrite
- Discipline
  - Pas de vérification de présence en cours/TD
  - Respect de l'enseignant !

Question du jour

- Qu'est ce que le génie logiciel ?
  - « *Le génie logiciel est une science de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels*. » Wikipedia

Plan du chapitre

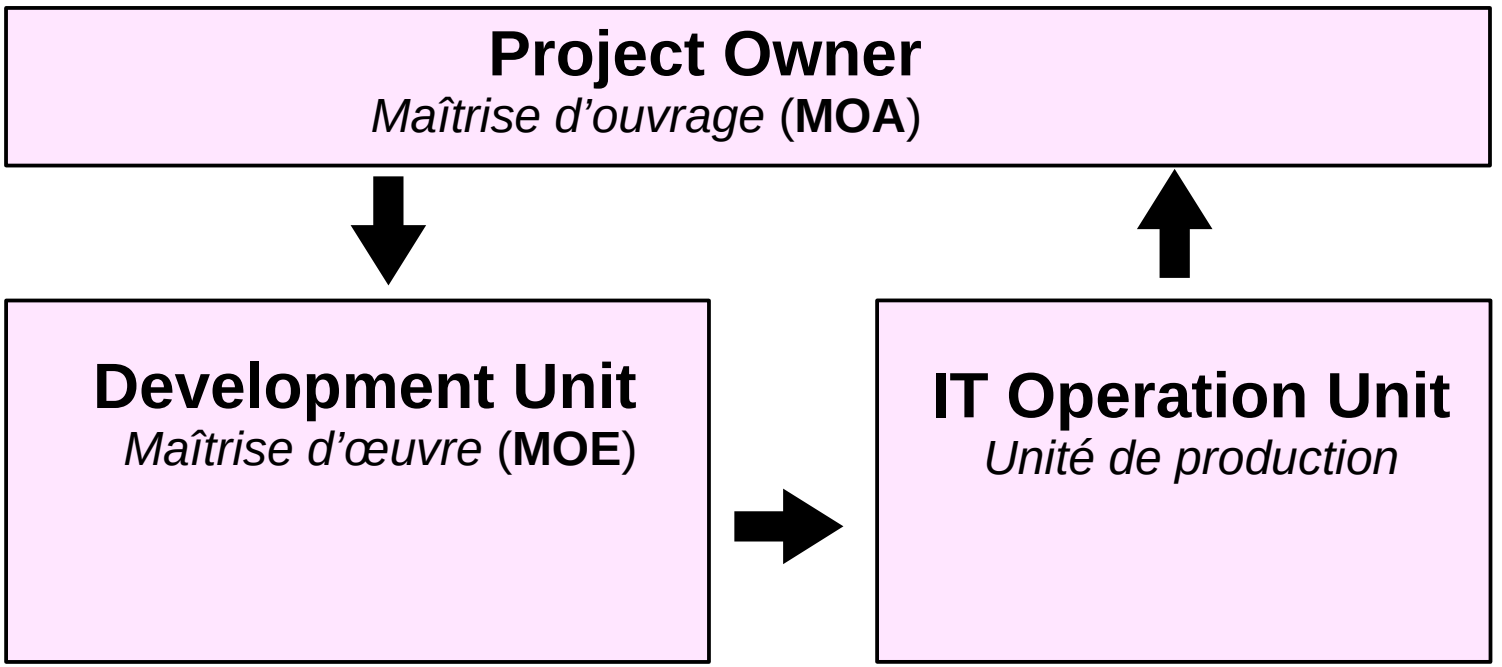
1
Pourquoi
un cours sur le
génie logiciel ?

Pourquoi un cours sur le génie logiciel ?

- La spécialité informatique de l'ENSICAEN forme des **ingénieurs développeurs logiciels**
  - *alias* **Architectes logiciels**

  - Ce sont des professionnels qui
    - conçoivent
    - déploient les systèmes logiciels
    - maintiennent
    - administrent
- Le génie logiciel est un enseignement de base pour les métiers liés au logiciel

Écosystème du développement logiciel



Pour qui ce cours ?

- Pour ceux qui choisiront de développer (**MOE, Unité de production**) :
  - Trouver sa place dans la gestion de projet informatique.
  - Se forger une culture du développement logiciel de haut niveau.
  - Prendre conscience de l'importance du code et des tests.
- Pour ceux qui choisiront de ne pas développer (**MOA**, **Conseil**, **Avant-vente**, **Ingénierie produit**) :
  - Être en mesure de formuler 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.

Pourquoi un cours sur le génie logiciel ?

- Pourquoi ne pas se contenter d'un cours de programmation pour développer des logiciels ?

Question

- Soit le programme d'addition de deux nombres entiers suivant :

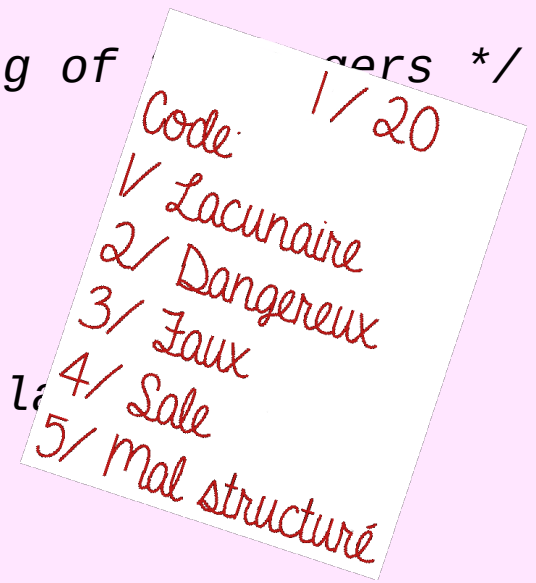
```
#include <stdio.h>
/** This program displays the result of adding of 2 integers */
int main() {
  int i1, i2; // The 2 integer operands
  scanf("%d", &i1); // Read the 2 integers
  scanf("%d", &i2);
  printf("Résultat = %d\n", i1 + i2); // Display
  return 0;
}
```

- Est-ce un bon code de programme ?

Question

- Soit le programme d'addition de deux nombres entiers :

```
#include <stdio.h>
/** This program displays the result of adding of ...gers */
int main() {
   int i1, i2; // The 2 integer operands
   scanf("%d", &i1); // Read the 2 integers
   scanf("%d", &i2);
   printf("Résultat = %d\n", i1 + i2); // Displa
   return 0;
}
```



- Est-ce un bon code de logiciel ?

Confusion programmation / développement

- Analogie avec le génie civil

Confusion programmation / développement

- Programmation (Maçonnerie) : accessible à tous

Confusion programmation / développement

- Programmation (Maçonnerie) : accessible à tous

Confusion programmation / développement

- Programmation (Maçonnerie) : accessible à tous

Confusion programmation / développement

- Développement (Architecture) : réservé aux professionnels

Confusion programmation / développement

- Développement (Architecture) : réservé aux professionnels

Confusion programme / logiciel

- Les profanes créent des **programmes**
- Les développeurs créent des **logiciels**

- Quelles sont les différences entre programme et logiciel ?

Différences programme / logiciel

1. Utilisateur
   - Programme : un utilisateur averti et bienveillant
   - Logiciel : l'utilisateur est un « client »
2. Portabilité
   - Programme : un OS
   - Logiciel : tous les OS
3. Complexité
   - Programme : résident sur 1 seul nœud
   - Logiciel : réparti sur le réseau
4. Taille des sources

Différences programme / logiciel :
La taille des sources

- Unité de mesure de la taille d'un logiciel :
  - **LOC** : lines of code
    - 1 MLOC : 10^6 LOC

Différences programme / logiciel : La taille des sources

- **Programme**
  - Quelques KLOC
- **Logiciel**
  - Commandes de vol A380 : **1 MLOC**
  - OS Android : **15 MLOC**
  - Linux kernel 5.8 (2020): **53 MLOC**
  - Facebook : **62 MLOC**
  - Windows 10: **80 MLOC**
  - Google (tous les services Internet) : **2GLOC**

Conséquence de la taille : Développement en équipe

- La taille des logiciels oblige à un travail en équipe
  - Unité de mesure :
    - **année-homme** (man-year)
    - ou mois-homme
    - ou heure-homme …
  - Par exemple :
    - Algorithme de recherche de Google est estimé à 1 000 années-hommes

Conséquence de la taille : Coût de développement

- La taille du logiciel entraîne des temps de développement et des coûts élevés
  - Combien de temps pour **développer** ce bout de code ?
  - Combien facturer le **développement** de ce bout de code ?

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

Conséquence de la taille : Coût de développement

- Ordre de grandeur en France :
  - 1 année-homme ≈ 1650 h
  - 1h ingénieur ≈ 50 €
  - Productivité ≈ 2 à 5 LOC/h (4 à 9 KLOC /an)
- Donc, le code coûte :
  - Temps : 1 heure-homme
  - Prix : 50 €

pour traverser tout le cycle de développement

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

Différences programme / logiciel

1. Utilisateur
2. Portabilité
3. Complexité
4. Taille des sources
5. Responsabilité des dysfonctionnements

Différences programme / logiciel : La responsabilité des dysfonctionnements

- **Programme**
  - L'utilisateur accepte les conséquences des dysfonctionnements lors de l'utilisation du programme
- **Logiciel**
  - Les développeurs sont tenus responsables par les utilisateurs des conséquences néfastes de l'utilisation du logiciel.
  - Les développeurs doivent proscrire toutes les conséquences néfastes de l'utilisation du logiciel :
    - Perte de données
    - Résultats erronés
    - Vol de données
    - Utilisation frauduleuse

Conséquence comique d'un dysfonctionnement

- Windows 95

Conséquence préoccupante d'un dysfonctionnement

- Bug sur les serveurs Microsoft Exchange en 2022
  - Les utilisateurs n'avaient plus accès à leur mail suite au changement d'année
  - Le moteur d'analyse des malwares était bugué
    - Codage des dates sur un entier signé (31 bits)
    - Or 2^31 = 2 147 483 648 ne peut pas coder les dates postérieures à 2021

Conséquence tragique d'un dysfonctionnement

- Mort tragique d'une patiente de 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

    → Ne jamais ajouter d'élément dans l'UI qui laisse à penser que la fonctionnalité associée est implémentée

Différences programme / logiciel

1. Utilisateur
2. Portabilité
3. Complexité
4. Taille des sources
5. Responsabilité des dysfonctionnements
6. Maintenance

Différences programme / logiciel : Maintenance

- Un logiciel ne s'use pas … mais il se détériore
  - Pas de logiciel sans maintenance
  - Sans maintenance un logiciel a une durée de vie estimée de 6 mois
  - La maintenance doit être préparée dès la conception
  - L'expérience montre même que la majeure partie du travail de développement commence après la livraison du logiciel au client

Conclusion en 1968

- Un programme peut se réaliser de façon empirique
- Un logiciel ne peut pas se développer sans une méthode de gestion de projet et un haut niveau d'expertise reposant sur le génie logiciel
  - Le développement est un projet étalé sur plusieurs mois
  - Le projet nécessite la collaboration de plusieurs personnes
  - La tâche est complexe et compliquée
  - La programmation ne représente qu'une très petite partie du logiciel
- Développeur logiciel :
  - Un métier à haut niveau d'expertise
  - Un métier qui s'apprend
  - L'auto-formation donne des programmeurs

Plan du chapitre

1
Pourquoi
un cours sur le
génie logiciel ?

2
Création
du génie logiciel

Génie logiciel (1968)

- Le terme anglais « *software engineering* » a été inventé par une pionnière du génie logiciel : **Margaret Hamilton**
  - Responsable de la partie logicielle embarquée du projet Apollo de la NASA
  - Le premier ingénieur en génie logiciel est une ingénieure

Margaret Hamilton

Génie logiciel (1968)

- Génie
  - Ensemble de pratiques **régulées** et basées sur des principes **scientifiques** et **économiques** (+ **RSE** et **DD**)

➡ Génie logiciel

  - Le génie logiciel postule donc que l'on peut appliquer le génie au logiciel

- Source d'inspiration : le **génie civil**
  - Le génie civil a démontré toute son efficacité depuis des millénaires

Génie civil

- La gestion de projet selon le génie civil :
  - **Découper le temps du projet en une suite de phases séquentielles**
    - *Livrable : diagramme de Gantt / Pert*
  - **Ne faire qu'une seule chose à la fois à chaque étape**
    - *Métiers spécialisés*
  - **Récolter le besoin avant de développer**
    - *Livrable : cahier des charges*
  - **Bien réfléchir avant d'agir**
    - *Livrable : documentation de conception et de réalisation*

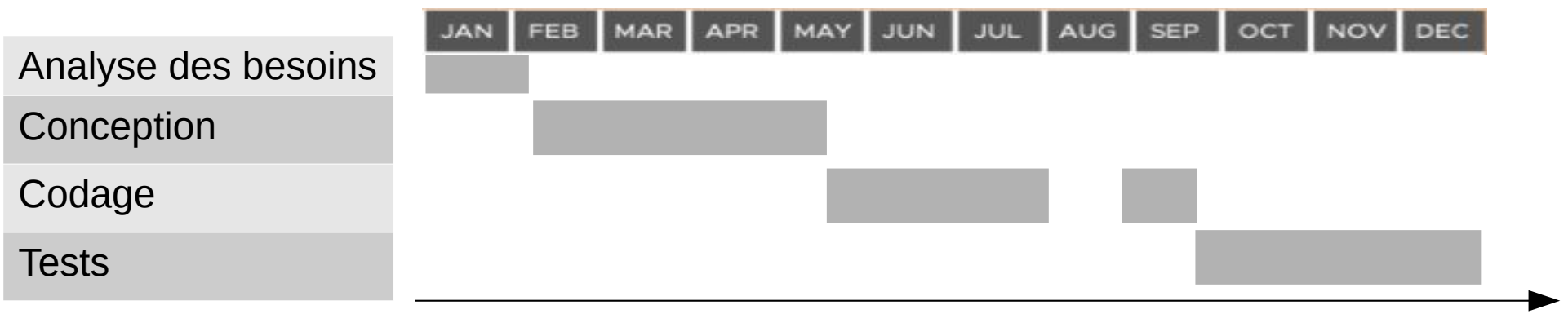
Génie logiciel

- Les éléments de base du génie civil appliqué au génie logiciel :
  - **Une méthode de gestion de projet**
    - Planifier une organisation du temps pour le travail en équipe
  - **Un paradigme de conception**
    - Fournir les briques de base de la conception et les mécanismes pour les assembler
  - **Un formalisme de modélisation**

    Fournir un langage formel pour parler du code de manière abstraite et non ambiguë

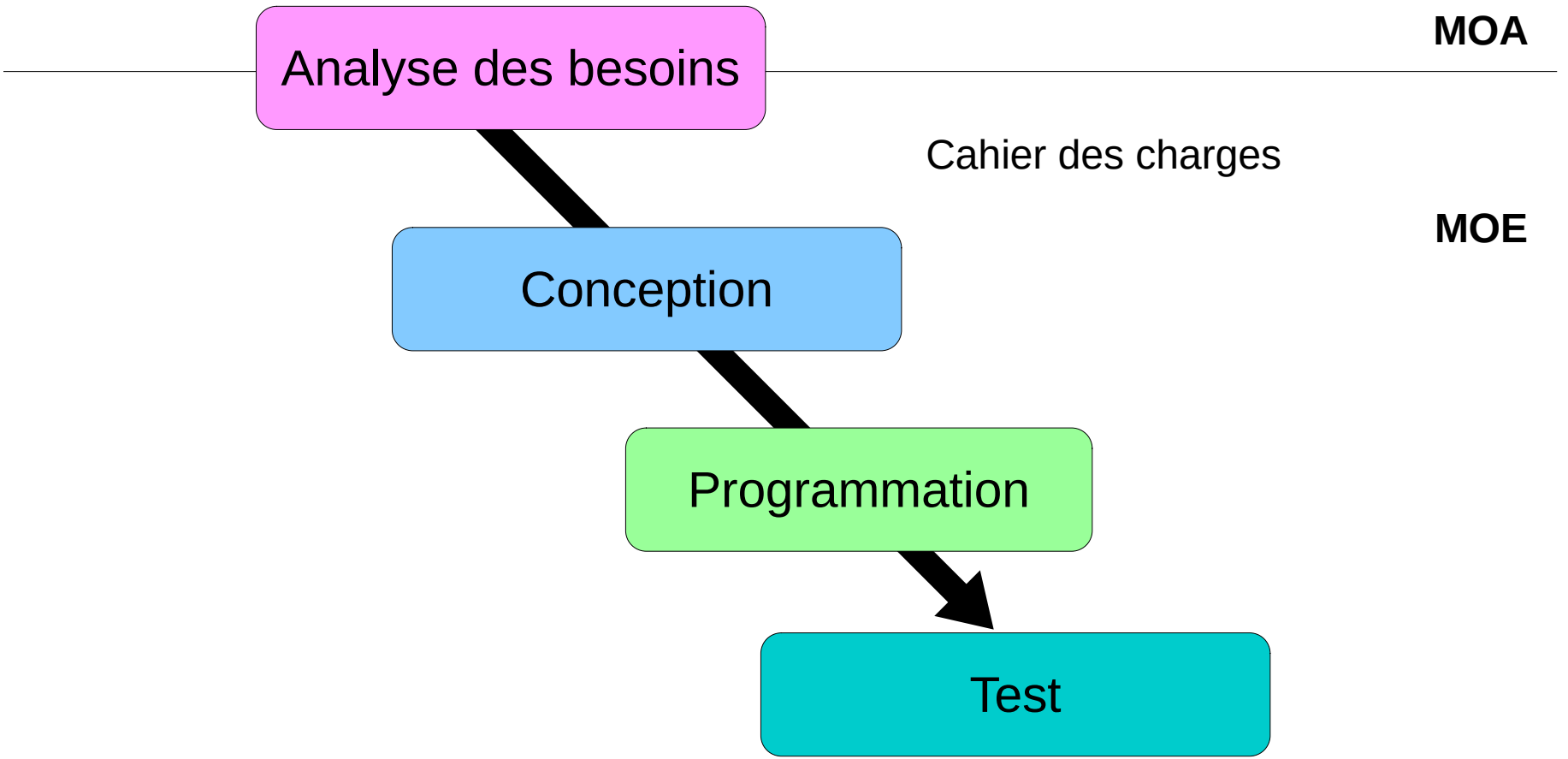
1/ Une méthode :
Le cycle en cascade

- **Découper le temps du projet en étapes séquentielles**



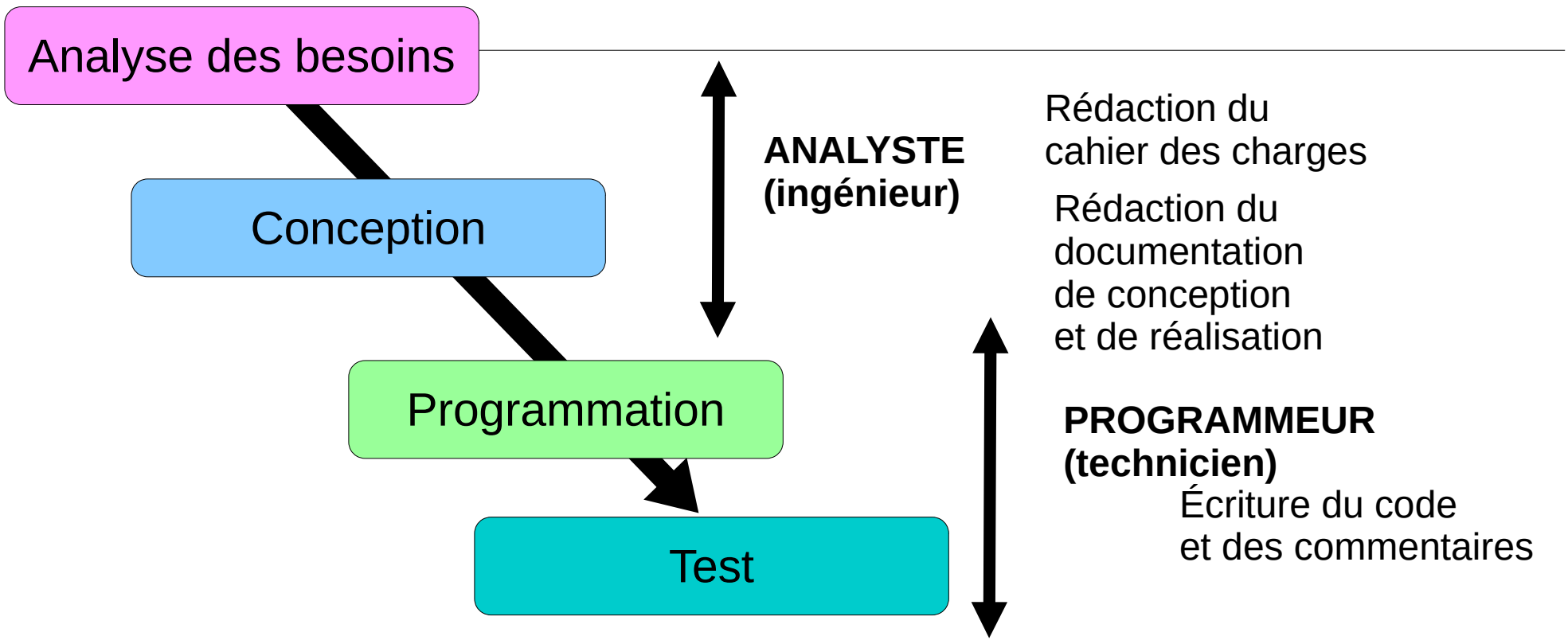
1/ Une méthode : Le cycle en cascade

- **Récolter le besoin avant de développer**



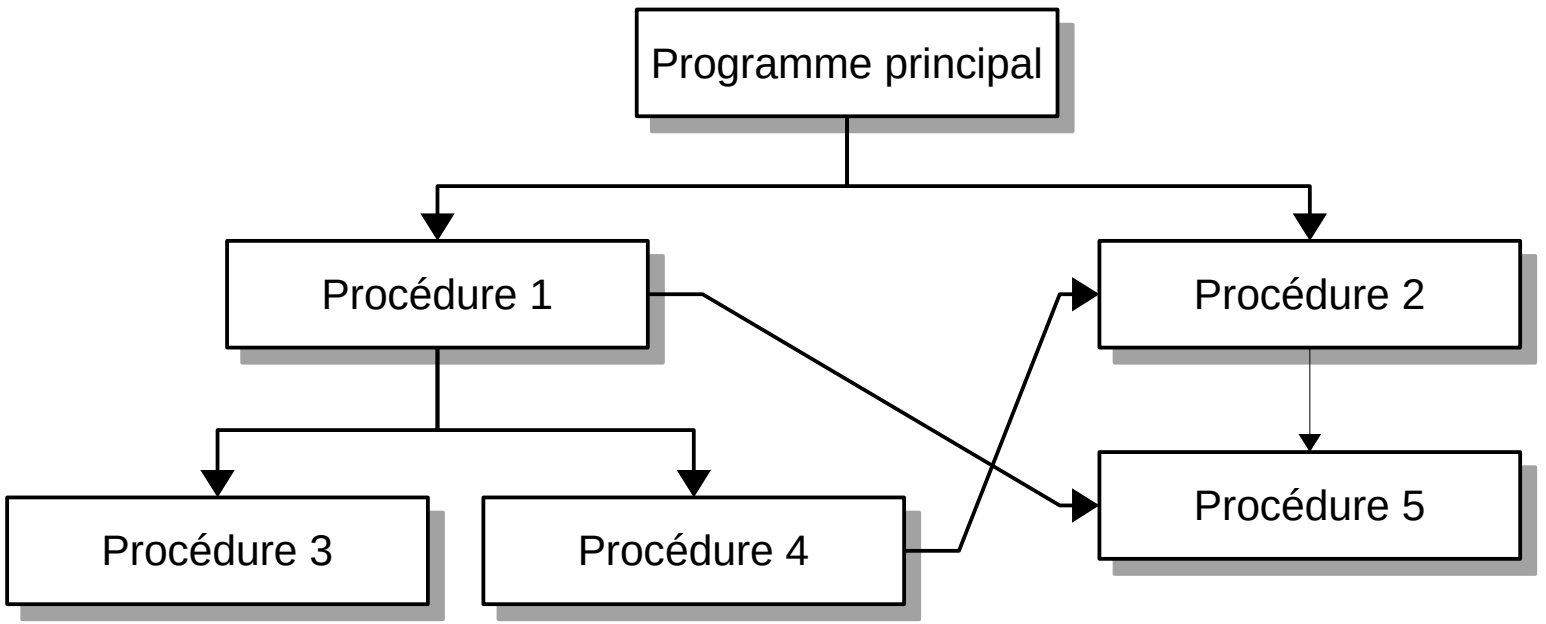
1/ Une méthode : Le cycle en cascade

- *Ne faire qu'une seule chose à chaque étape*
  - Il en résulte deux métiers dans la MOE



2/ Un paradigme : La conception procédurale

- Graphe d'appel des procédures



3/ Un formalisme : L'algorithmique

- *Bien réfléchir avant d'agir*

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	$n-1$

Ada Lovelace
vers 1843

- Le premier programmeur est une programmeuse : Ada Lovelace
  - Algorithme de calcul des nombres de la suite de Bernoulli $\sum_{k=0}^{n-1} k^m$

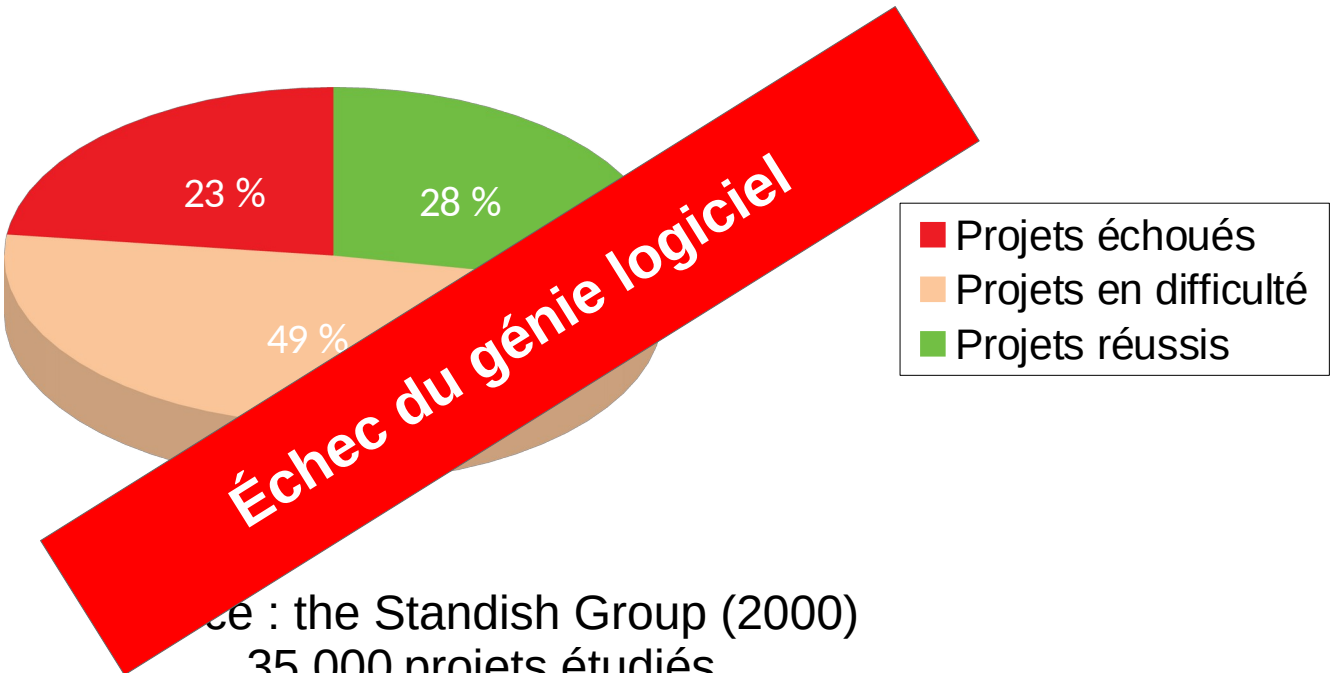
Plan du chapitre

1
Pourquoi
un cours sur le
génie logiciel ?

2
Création
du génie logiciel

3
Bilan

Bilan en 2001

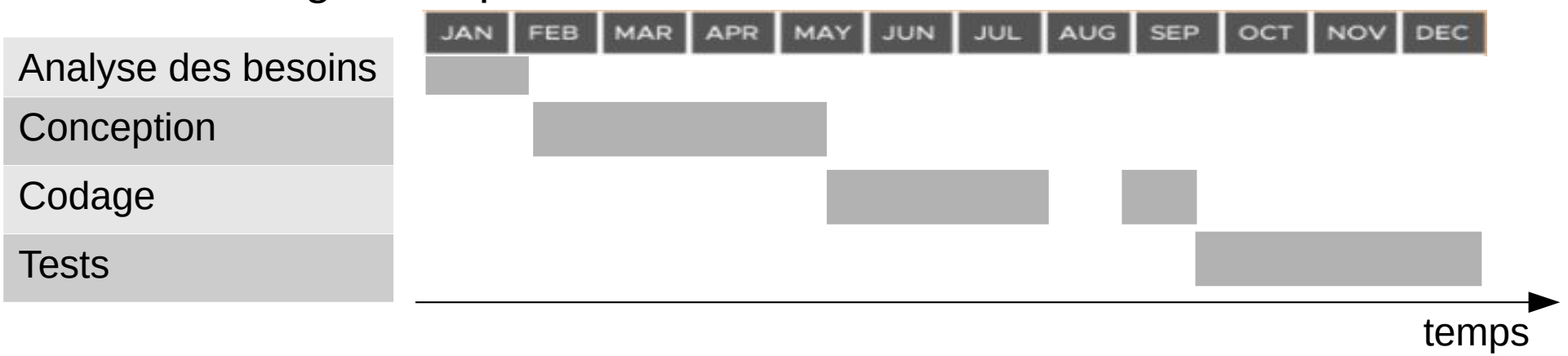


ce : the Standish Group (2000)
35.000 projets étudiés

Causes de l’échec

Cause d'échec 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



  - Conséquences
    - La moindre modification du planning met le projet en danger

« Bon, les gars, il y a eu un accident au quai.
Une petite auto blanche est tombée à l'eau.
Qu'est ce qu'on fait ? »

« Pas de problème, on va appeler une grue
pour remonter l'automobile. »

« Ça va très bien, la voiture est toute petite,
dans une demi-heure tout est terminé."

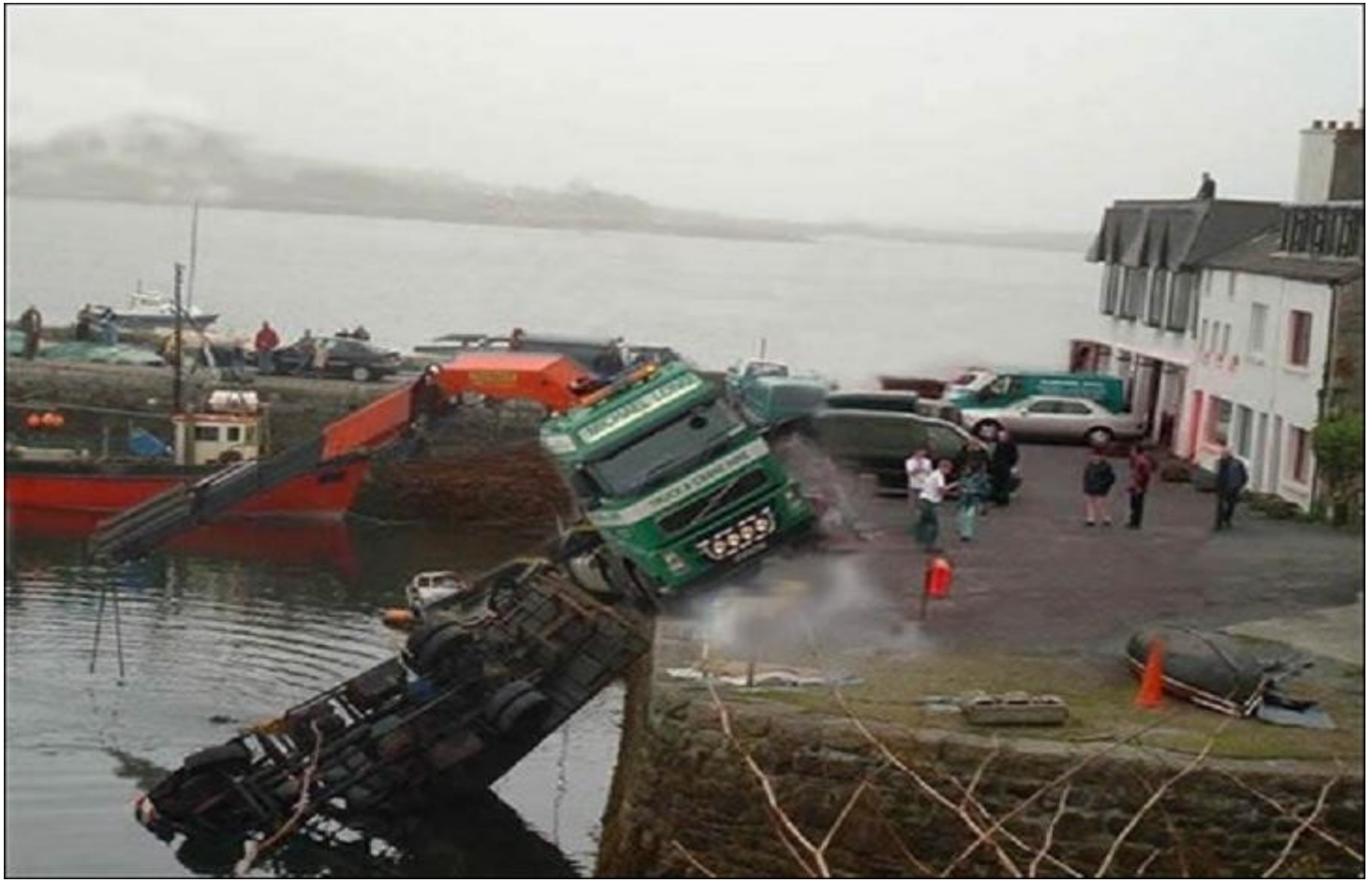
« ARGH!!!! La grue est tombée aussi à l'eau avec la petite voiture ! »

« Ça va vraiment mal.
Et là, qu'est ce qu'on fait ? »

« Pas de problème, on a appelé une autre grue, une GROSSE celle-là. »

« Bon, la petite automobile est déjà sortie,
on aurait du appeler cette grue la première fois... »

« Il ne reste qu'à sortir la petite grue maintenant, dans une demi-heure tout devrait être fini. »

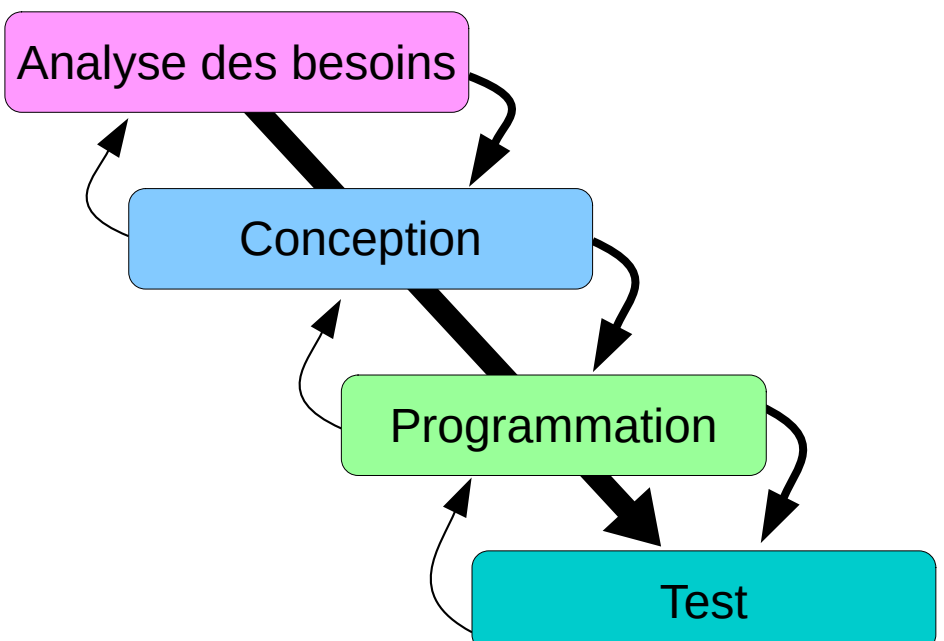


« ARGHHHHHH !!! ENCORE !!!!!!!!! »

« Pas de problème, on a appelé une autre grue, plus grosse encore.. »

Un plan est rarement respecté

- Une amélioration
  - Ajouter des allers-retours entre les phases
  - Question : comment anticiper la durée des allers-retours dans la planification ?



Cause d'échec n°2 :
Estimer la durée d'un projet

- Déterminer la durée d'un projet en années-hommes est extrêmement difficile
  - Trop d'aléas et d'incertitudes
  - Pas de règles et pas de mesures systématiques

Cause d'échec n°2 : Estimer la durée d'un projet

- Mythe de l'**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**

  - *Note : 15 KLOC/année → 9 LOC/h*
  - Certaines tâches ne sont pas parallélisables :
    - « Neuf femmes ne font pas un enfant en un mois »

Cause d’échec n°2 :
Estimer la durée d’un projet

- Rattraper le retard
  - Contrairement à l’intuition, ajouter des personnes à une équipe d'ingénieurs ne permet pas de rattraper le retard, au contraire :
    - Les nouveaux ingénieurs doivent être formés par le anciens qui ne sont donc plus à leur tâche et le retard s’accroît encore