CENTRE DE
MAITRISE DES
SYSTEMES ET
DU LOGICIEL

GLG105 : Génie logiciel

Introduction

75 ans d'informatique

1801
1ère programmation binaire
(carton/trou)
Joseph-Marie Jacquard

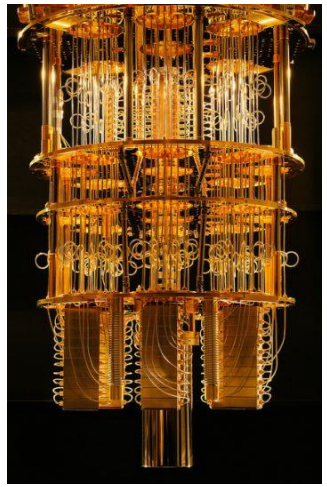
1958
Fortran
Cobol
Logiciels calculs scientifiques

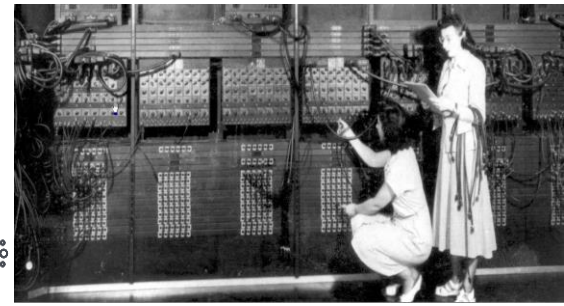

Par Trower, NASA —wikimedia.

1990
Internet &
World Wide Web

2020
5G

20XX
???


Flickr : IBM Research


US Army – Columbia University

1945
Assembleur
Logiciels calculs scientifiques
ENIAC

Crise

1970
C
SmallTalk
Logiciels calculs scientifiques
Temps réel

BDD
UNIX
Micro informatique

2000
JAVA

Clé USB

2022
IA

20XX
???

Once upon a time – Software Engineering

- Fin des années 1960, un constat est établi :
  - Délais de livraison non respectés
  - Budgets non respectés
  - Mauvaise qualité
  - Ne répond pas aux besoins des utilisateurs
  - Difficile à utiliser, maintenir, et faire évoluer
  - La maintenance n'est que corrective

- Le développement des grands projets informatiques est de moins en moins bien maîtrisé et manque de fondement rationnel

Software crisis

* The major cause of the software crisis is that the machines have become several orders of magnitude more powerful ! To put it quite bluntly : as long as there were no machines, programming was no problem at all ; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

* Edsger W. Dijkstra The Humble Programmer

Once upon a time – Software Engineering

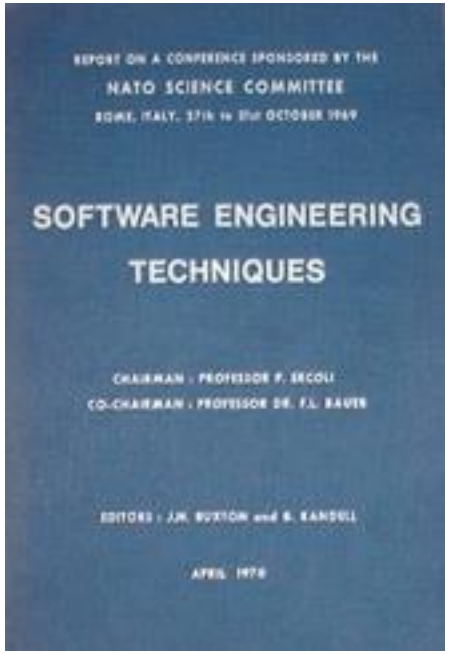
Que faire?

- **Promotion** d'un **génie logiciel** comme il existe le génie civil, électrique.
- Une conférence sur les difficultés de la production de logiciel et les moyens de les surmonter est montée sur la suggestion **de F.L. Bauer**

OTAN octobre 1968: Working conference on Software Engineering

- Actes des conférences SE'68 et SE'69

Garmisch, Allemagne,

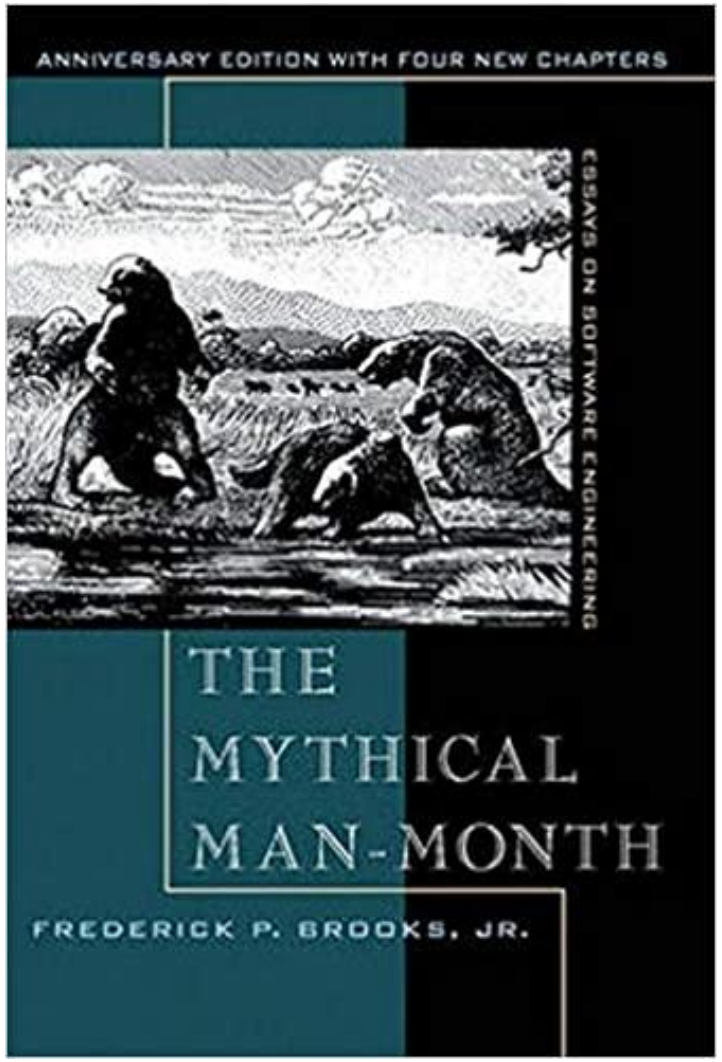


Rome, Italie

- Considéré comme le début de cette nouvelle discipline
- Conférence de 68 a mis en évidence la crise, celle de 69 les difficultés de communication au sein des acteurs de ce domaine

Le Mythe du mois-homme - The Mythical Man-Month: Essays on Software Engineering

- Parution en 1975, écrit par Fred Brooks.
- Certains points abordés dans ce livre sont toujours d'actualité de nos jours :
  - Lorsqu'un projet est en retard, ajouter de la force de travail ne fait qu'accentuer le retard.
  - Le facteur déterminant du succès ou de l'échec d'un projet est l'unité conceptuelle de son architecture.
  - Qualité de la communication. Importance de l'organisation, de la structuration des équipes, et de la répartition des rôles (RACI)
- Dans les années 1980, parution de « No Silver Bullet — Essence and Accidents of Software Engineering » qui met en doute les « technologies miracles » de son temps.

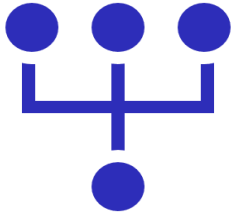


Les réponses à la crise

Maîtrise du processus de développement

Application d'un modèle

- Analyse des besoins
- Spécification
- Conception
- Programmation
- Validation et vérification
- Livraison
- Maintenance (MCO)
- Cadre pour la gestion de projet



Mise en place de méthodes structurées et d'outils CASE

SART : Structured Analysis real-time

SADT : Structured Analysis and Design Technique

AGL ou outils CASE (Computer Aided Software Engineering)

Les réponses à la crise

- Dans les années 90, les méthodes structurées commencent à « s'essouffler» car les développements sont de plus en plus complexes, de plus en plus longs et l'obsolescence est de plus en plus rapide

- Nécessité à réutiliser les connaissances et à les partager
  - Apparition du paradigme objet
  - Modernisation des AGL

- De nos jours, l'Agilité, devOps, micro-services, Cloud Computing …


Crisis

Etude du « Department of Defense USA »

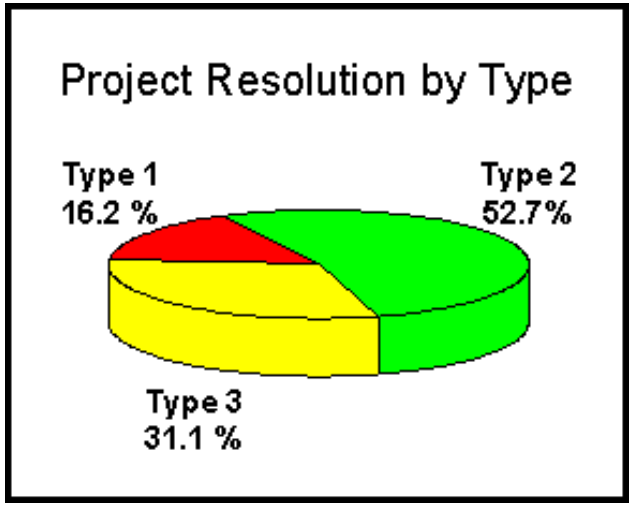
- Selon une étude du DoD aux US en 1988, 70 % des projets initialisés sont abandonnés avant leur achèvement pour :
  - non-conformité (non-respect du cahier des charges),
  - inadéquation de la réalisation par rapport au besoin,
  - retard trop important,
  - dépassement de budget,

- Et les 30 % restants connaissent un retard moyen de plusieurs années; Il s'agit bien sûr de gros projets, militaires ou technologiquement sensibles, dont la durée de développement est de plusieurs années

Crise du logiciel: Rapport CHAOS

- En 1995, le Standish group a estimé que les compagnies et gouvernements américains ont dépensé :
  - 81 milliards $ pour des projets informatiques qui n'ont jamais abouti
  - 59 milliards $ sur des projets qui auront abouti mais en dépassant les plannings estimés

- 8380 applications recensées pour l'étude ont été réparties suivant 3 catégories :
  - Type 1 : Le projet est arrivé à terme dans les temps et dans le budget tel que défini au départ
  - Type 2 : Le projet est terminé mais avec un dépassement des délais et du budget avec des modifications par rapports à la définition du départ
  - Type 3 : Le projet échoue durant sa réalisation



Le rapport CHAOS est publié annuellement depuis 1994 et présente un instantané de l'état de l'art du développement logiciel

Crise du logiciel : Rapport CHAOS

Suivant l'étude un projet peut, ou non, réussir :

- Type 1 : Succès du projet
  - Implication des utilisateurs
  - Executive Management Support
  - Bonne définition des besoins
- Type 2 : Challenge du projet
  - Manque de données concernant les utilisateurs
  - Définition des besoins incomplète
  - Modification de la définition des besoins
- Type 3 : Echec du projet
  - Demande incomplète
  - Manque d'implication des utilisateurs
  - Manque de ressources

Crise du logiciel : De nos jours

- Etude du Standish group

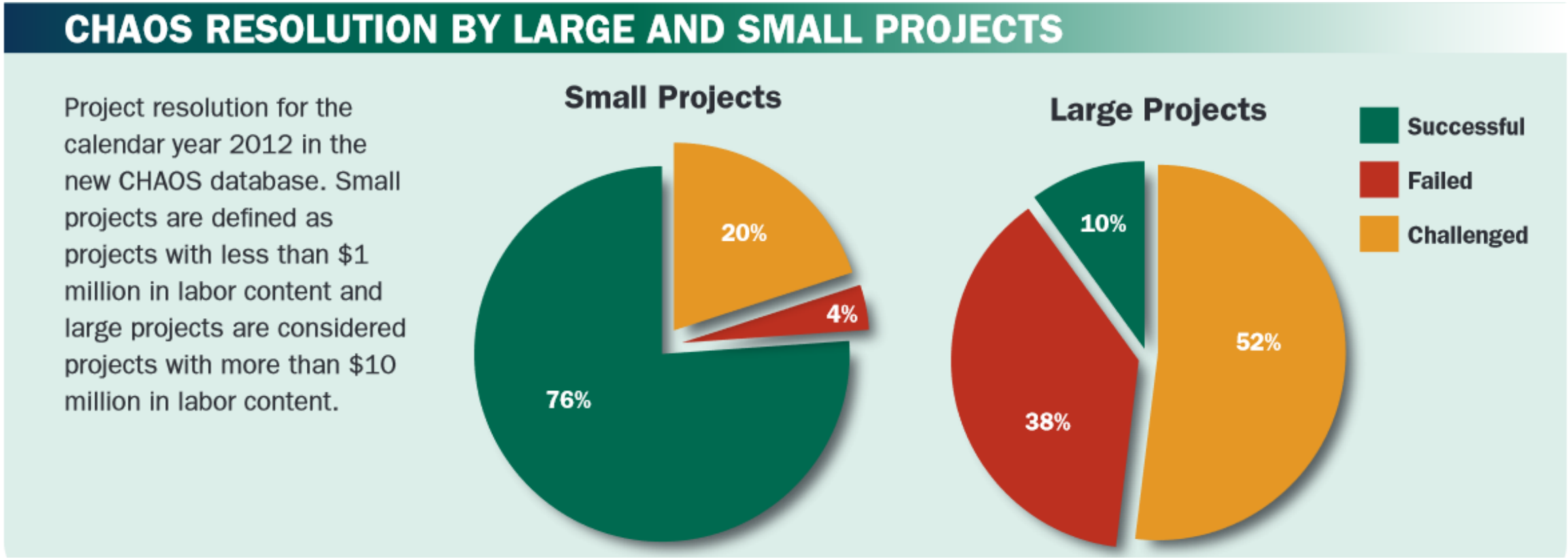
MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

- Projets réussis : achevés dans les délais et pour le budget impartis, avec toutes les fonctionnalités demandées
- Projets mitigés : achevés et opérationnels, mais livrés hors délais, hors budget ou sans toutes les fonctionnalités demandées
- Projets ratés : abandonnés avant la fin ou livrés mais jamais utilisés

Crise du logiciel : De nos jours



25 % des projets en France ne respectent pas leurs délais (mars 2011)

Projets 2010	Durée réelle constatée			
Durée prévue	<3 mois	3 à 6 mois	6 à 12 mois	> 12 mois
< 3 mois	66,67%	26,67%	0,00%	6,67%
3 à 6 mois	0,00%	71,79%	25,64%	2,56%
6 à 12 mois	0,00%	1,59%	77,78%	20,63%
> 12 mois	0,00%	0,00%	5,26%	94,74%
Total	5,75%	18,97%	35,63%	39,66%

Source : société Sciforma (réalisé auprès de 4 000 responsables d'entreprises de toutes tailles et de tous secteurs),

Le génie logiciel

Software Engineering

Quelques définitions (1/2)

Programme informatique

- Ensemble des instructions écrites dans un langage de programmation permettant à un système informatique d'exécuter une tâche ou un ensemble de tâches donnés
- C'est un assemblage :
  - D'algorithmes (fonction à calculer)
  - De représentations (codage, structures de données)
  - De gestion de ressources (mémoire, processus, etc.)
  - D'organisation et de paramétrage (évolution, interfaces, généricité)
  - De directives de packaging (topologie statique et dynamique, granularité, modularité de la programmation objet)
  - D'observateurs abstraits (exceptions pour la fiabilité)

Logiciel

- Ensemble d'artifacts coopérants via des interfaces et de procédures (scripts) dédiés à la résolution d'un ensemble de besoins de ses utilisateurs
- Pas uniquement du code informatique…
  - Documentation (manuels utilisateurs, de référence…)
  - Jeux de tests
  - Informations de configuration

SUFFISAMMENT COMPLEXE POUR FAIRE TRAVAILLER PLUSIEURS PERSONNES EN MEME TEMPS

Quelques définitions (2/2)

Ingénierie du logiciel
(source Adeli)

- Domaine de l'informatique qui concerne la **RÉALISATION** de systèmes logiciels qui sont suffisamment **GRANDS** et **COMPLEXES** pour être réalisés par des **ÉQUIPES** d'ingénieurs (École Polytechnique Montréal).

- L'art et la science de **concevoir** et de **construire**, avec **économie** et **élégance**, des applications, des objets, des frameworks et d'autres systèmes informatiques, qui soient corrects, robustes, extensibles, réutilisables, sûrs, efficaces, faciles à maintenir et à utiliser.

- Software Engineering is the **technological** and **managerial** discipline concerned with systematic production of software products that are developed and modified on **time** and within **cost estimates** [Fayrley 85].

- Application systématique des connaissances, des méthodes et des acquis scientifiques et techniques pour la conception, le développement, le test et la documentation de logiciels, afin d'en rationaliser la production, le suivi et la qualité [arrêté du 30 décembre 1983 (J.O. du 19 février 1984)].

Le génie logiciel - Software Engineering

Le GENIE LOGICIEL vise à rationaliser le processus de développement du logiciel à l'aide de méthodes, procédures

Doit garantir que :

√ Les spécifications correspondent aux besoins réels du client

√ Le logiciel respecte ses spécifications

√ Les coûts affectés à la réalisation soient tenus

√ Les délais de réalisation soient honorés

Les principes et les méthodes ne sont pas fautifs, le PROBLEME est plutôt qu'ils ne sont PAS APPLIQUES

Le génie logiciel - Software Engineering

Le GENIE LOGICIEL

INCLUT DES ASPECTS TECHNIQUES ET NON TECHNIQUES

BASÉ SUR DES MÉTHODOLOGIES ET DES OUTILS

DEMANDE DES CAPACITÉS DE COMMUNICATION ET D'ANTICIPATION

Implique

√ Du savoir

Et

√ Du savoir-faire

Les grands principes du génie logiciel

Les principes et méthodes

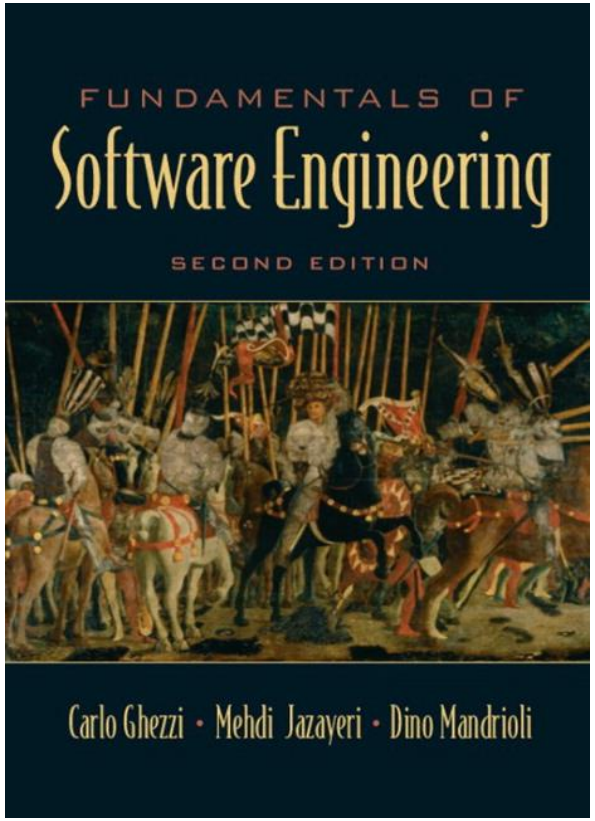
Le bon sens

Des principes

Des méthodes

Les grands principes du génie logiciel

- La rigueur
- La décomposition des problèmes en sous-problèmes
- La modularité
- L'abstraction
- L'anticipation des évolutions
- La généricité
- La construction incrémentale



C. Ghezzi, M. Jazayeri, D. Mandrioli.Fundamentals of Software Engineering. Prentice-Hall, 1991.

Le génie logiciel - Software Engineering

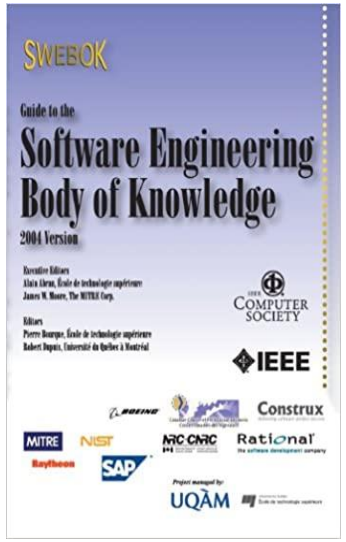
- Le guide SWEBOK du IEEE définit les champs de connaissance du génie logiciel

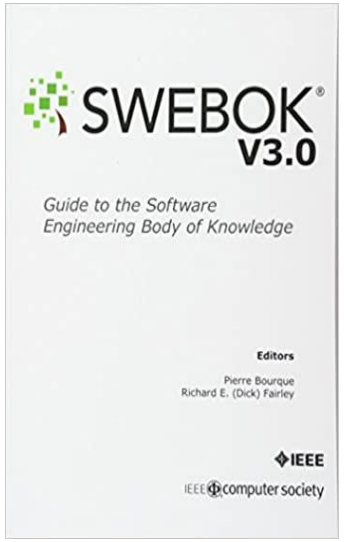
- Le SWEBOK « Guide to the Software Engineering Body of Knowledge », est un référentiel pour structurer les connaissances relatives au génie logiciel.

SWEBOK – Domaines de connaissances

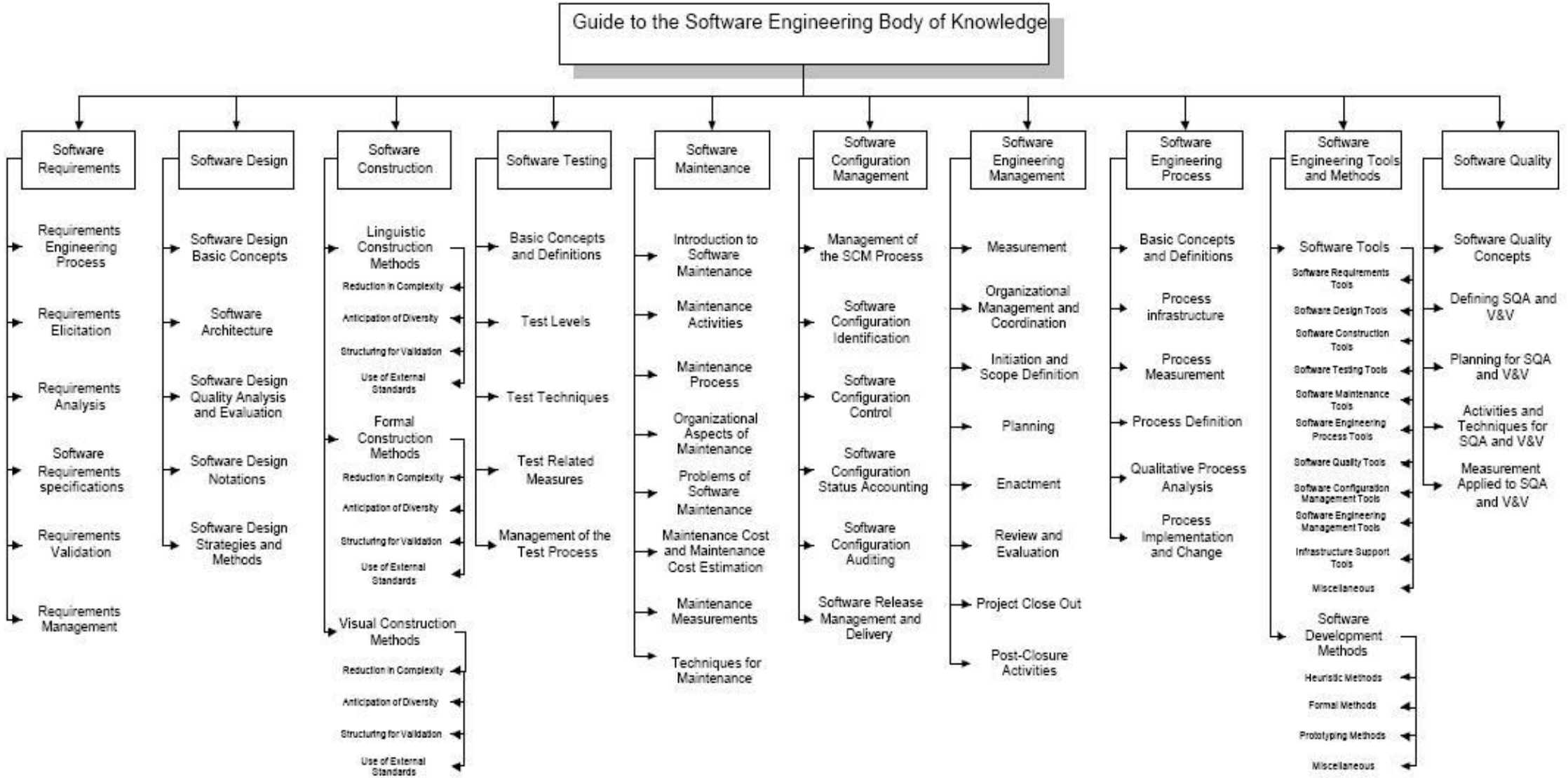
SWEBOK Knowledge Areas	Additional linked disciplines
• Software requirements	• Computer engineering
• Software design	• Systems engineering
• Software construction	• Project management
• Software testing	• Quality management
• Software maintenance	• General management
• Software configuration management	• Computer science
• Software engineering management	• Mathematics
• Software engineering process	
• Software engineering models and methods	
• Software quality	
• Software engineering professional practice	
• Software engineering economics	
• Computing foundations	
• Mathematical foundations	
• Engineering foundations	

Foundations Knowledge Areas





SWEBOK – Domaines de connaissances



INCOSE
International Council on Systems Engineering

AFIS Association Française d'Ingénierie Système

- Association à but non lucratif (1990)
  - consistant à relancer des travaux d'intérêt général sur les notions de systèmes et d'ingénierie des systèmes
  - suite aux nombreux constats
    - de défaillances,
    - d'accidents,
    - de pertes financières,
    - d'inefficacités
    - constatés dans les systèmes des années 80 et 90.

Comment

- Spécifier un logiciel
- Créer un logiciel de qualité
- Gérer un projet logiciel

Activités principales

- L'analyse des besoins
- La spécification
- La conception (40% de l'effort)
- La programmation (de 15 à 20% de l'effort)
- La validation et la vérification
- Le suivi et la maintenance

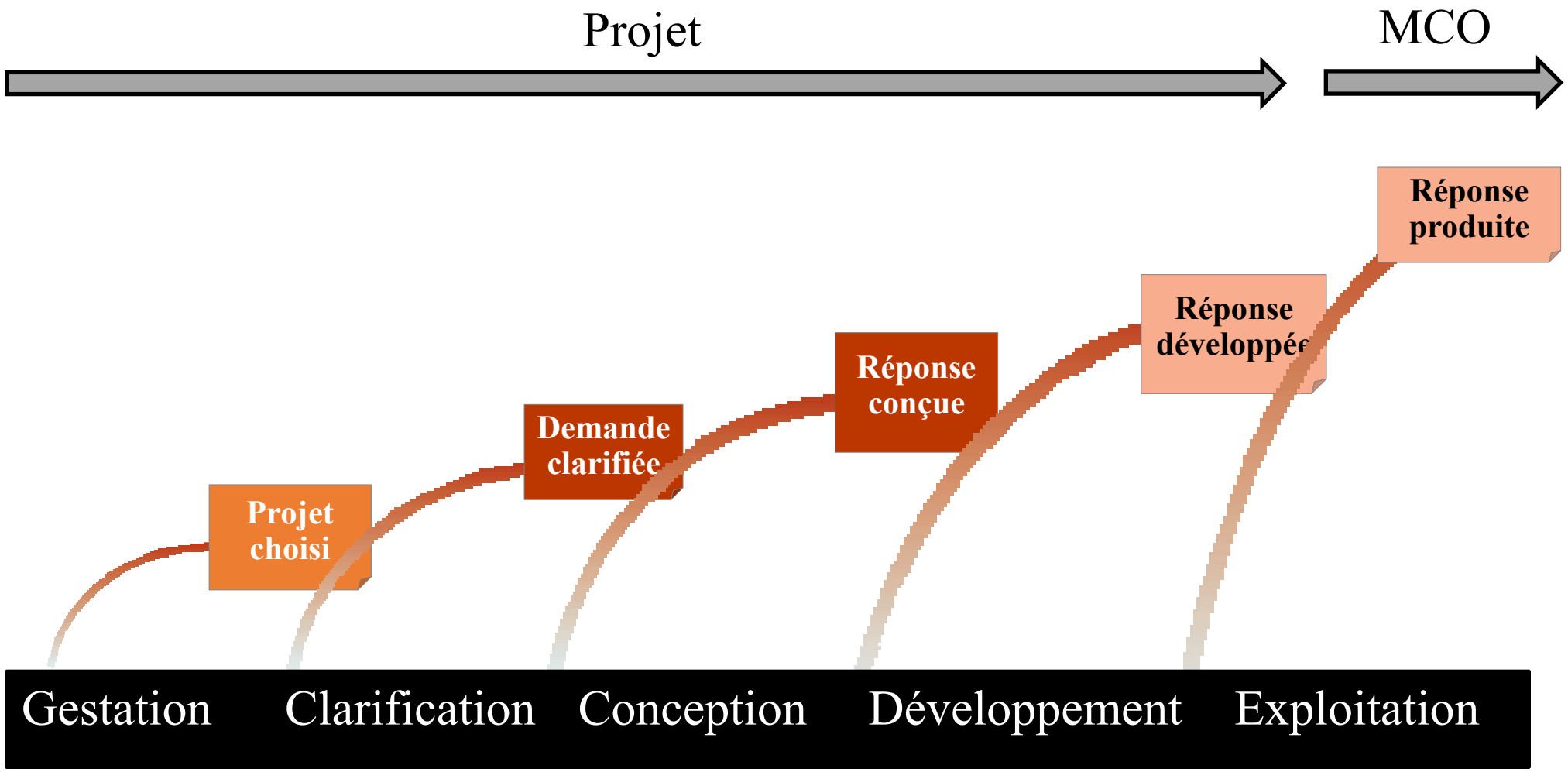
Que doit réaliser le logiciel?

- Exercice complexe
- Dépendant du contexte d'utilisation du logiciel
- Et de ses objectifs

Comment?

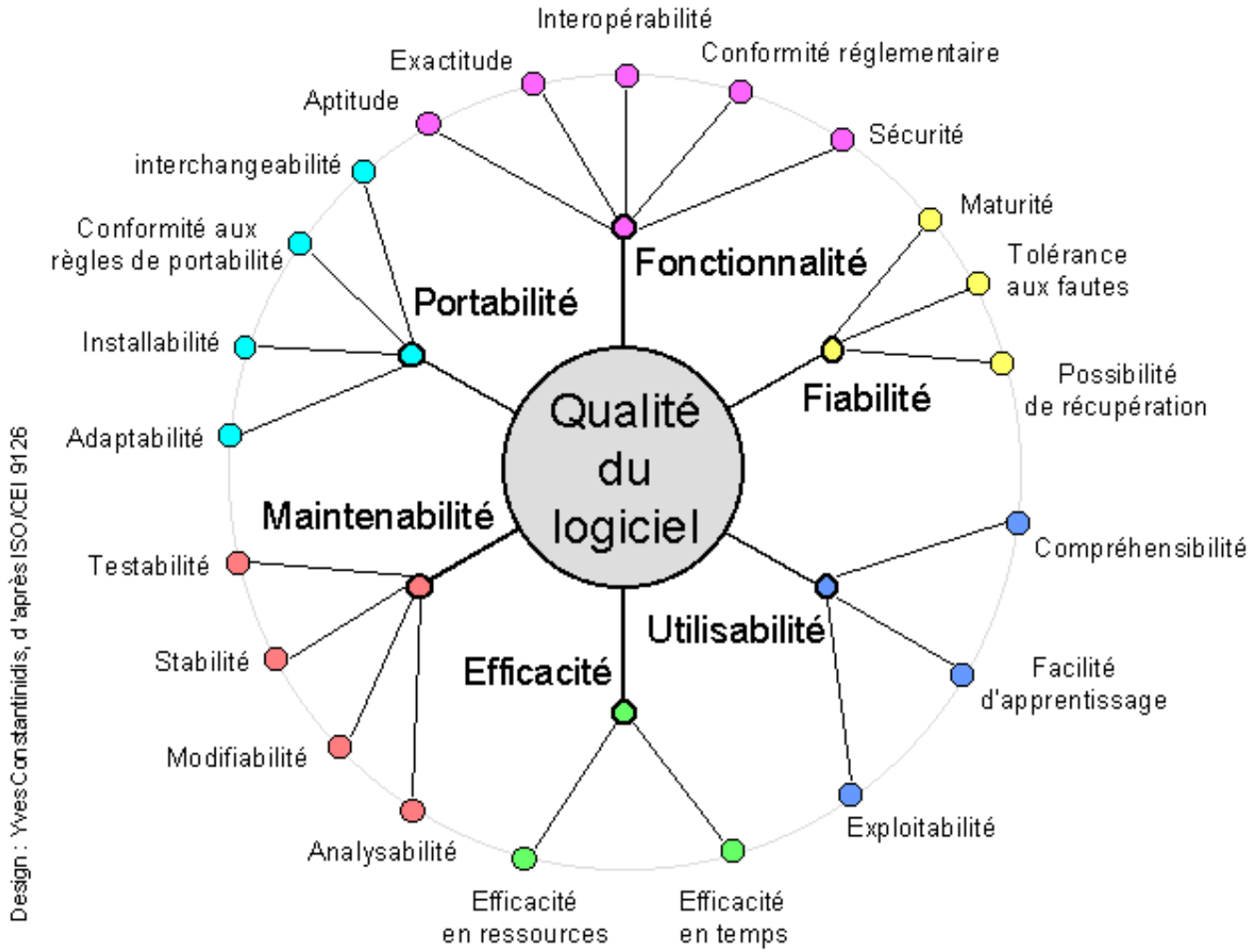
- Généralement en langage naturel (français)
- Mais aussi dans des langages :
  - Informels
  - Semi-formels
  - Formels

Processus de conception d'un projet



Respect des spécifications

Respect des critères de qualité norme ISO/CEI 9126



Qualité

Définition d'objectifs

Contrôlée à travers ces objectifs

AGL / Outils CASE

- AGL : Atelier de génie logiciel / Outils CASE (Computer Aided Software Engineering) : outils d'aide au dévelopement
- L'AGL est adapté aux différentes phases d'un projet et est basé sur les méthodologies
- 2 catégories d'AGL
  - Horizontaux : Editeur de texte, gestion de projet, gestion de configuration …
  - Verticaux : Spécifique aux phases d'un projet
    - Spécification, conception, génération de code ….
- 2 types d'AGL
  - Upper-case : Environnement de conception
  - Lower-case : Environnement de développement
- IDE : Integrated Development Environment est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs

Le projet


Morris Charts
Line Chart
Area Chart
Bar Chart
Donut Chart
25%
Donut Chart
Sparkline Charts
Line Chart
Bar Chart
Pie Chart
Easy Pie Charts
25%
50%
75%

Naissance des projets

De sources externes

- Besoins, désirs, attentes ou suggestions des clients
- Offre existantes (concurrents ou produits/services substituts)
- Nouvelles technologiques
- Nouvelles dispositions légales
- Expositions, foires, kiosques,
- Etc...

De sources internes

- Direction
- Unités organisationnelles
- Démarches individuelles

CNAM/ EPN05 G. Rocheteau - Génie logiciel
Introduction – page 35

La gestion de projet - Les problèmes du chef de projet (1/2)

- ✓ Comment **inventorier** tout ce qui et rien que ce qui est constitutif du projet
  - Le périmètre du projet –Les nomenclatures produits et/ou services
- ✓ Comment **coordonner** et **planifier** les actions
  - «The plan is nothing. Planning is everything», D.Eisenhower
- ✓ Comment **allouer** au mieux les ressources disponibles et **contrôler** les coûts
  - Ordonnancement, méthode PERT, etc.
    - cf. la recherche opérationnelle à la RAND Corp. dans les années 40s-50s
- ✓ Comment **synchroniser** au mieux différentes tâches interdépendantes mais qui peuvent être menées en parallèle
  - Gestion optimale du temps (le temps est une ressource rare)

La gestion de projet - Les problèmes du chef de projet (2/2)

- ✓ Comment **arbitrer** les conflits inhérents aux activités humaines et les ressources humaines réelles
- ✓ Comment **anticiper** un événement mineur qui risque d'engendrer le chaos
  - Analyse de risque, inventaire des ressources critiques, etc.
- ✓ Comment **acquérir** les ressources de toute nature nécessaires au projet en évitant le stockage inutile ou la rupture de stock, i.e. la logistique
- ✓ Comment **garantir** la qualité, i.e. les ressources affectées à la qualité en évitant la sous-qualité ou la sur-qualité
- ✓ Comment **communiquer** efficacement, i.e. sans perturber les équipes, avec l'environnement et les parties prenantes

Profil et rôle du chef de projet

NIVEAU D'EXPÉRIENCE ET D'EXPERTISE DU CHEF DE PROJET

CAPACITÉ À JOUER LE RÔLE DE POINT D'ANCRAGE OU DE PONT ENTRE LES DIRECTIONS CONCERNÉES ET L'ÉQUIPE DE PROJET

SUFFISANCE DU POUVOIR DÉTENU PAR LE CHEF DE PROJET, AUX PLANS DÉCISIONNEL, ORGANISATIONNEL ET HIÉRARCHIQUE

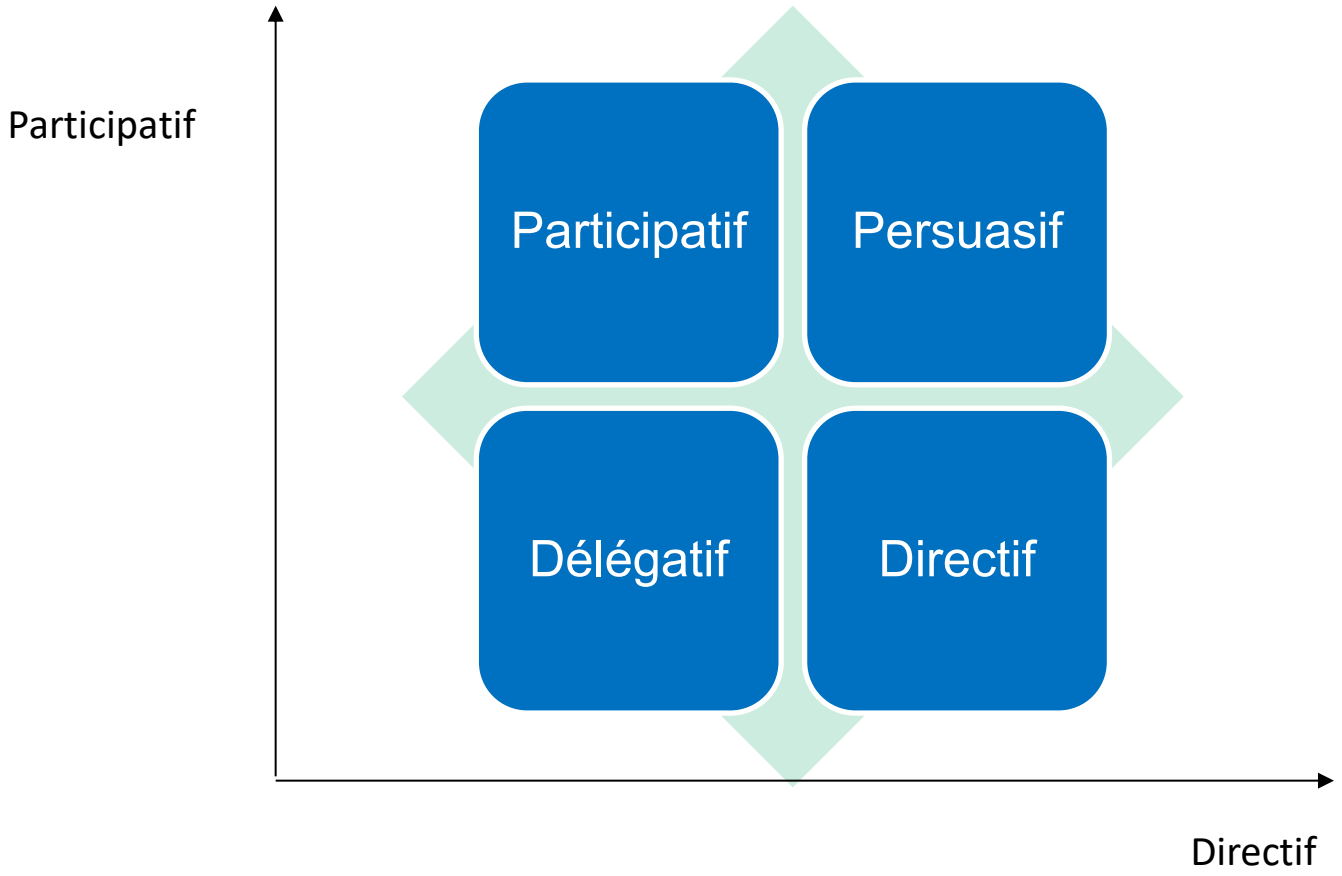
NIVEAU DE RESPECT SUSCITÉ

APTITUDE À INTÉGRER ET À COMMUNIQUER DIFFÉRENTES PERSPECTIVES OU ASPECTS EN UN TOUT COHÉRENT ET COMPRÉHENSIBLE*

Capacité d'anticipation

* marketing, design, ingénierie, achats, fournisseurs, production et qualité

Styles de leaderships



Selon Hersey et Blanchard – créateurs du leadership situationnel – il n'existe pas de « bon » style de leadership : un leader doit adopter le style le plus adapté à la situation

Développement de l'expertise des chefs de projets

Expertise

Stade N°1: chef de projet novice

Stade N°2: chef de projet débutant/confirmé

Stade N°3: le chef de projet compétent

Stade N°4: la maîtrise

Stade N°5: l'expertise

Les facteurs humains

- ✓ C'est le facteur essentiel de la gestion de projet
  - Quelques causes suffisantes d'échec
    - Les données collectées sont utilisées pour juger les personnes
    - Les tâches / responsabilités ne sont pas clairement attribuées
    - Les métriques adoptées sont bureaucratiques et ne mesurent pas les aspects essentiels de l'activité
    - Être négatif et culpabiliser sans cesse les collaborateurs
    - Ne pas tenir compte des informations remontées par l'équipe
    - Ne pas tenir l'équipe informé de l'avancement, des risques
- ✓ Le chef de projet doit être attentif et vigilant sur le «moral» de l'équipe
  - Facteur fondamental de productivité

Les dix facteurs de succès des projets (Pinto & Sleven)

1. Clarification initiale de la mission du projet (finalité, but et orientations ou instructions générales)
2. Appui manifesté par la haute direction envers le projet
3. Planification et échéancier de projet complets et détaillés
4. Implication initiale et continue du client
5. Recrutement, sélection et entraînement de l'équipe
6. Capacité technique de production (technologie et expertise technologique)
7. Acceptation du projet finalisé par le client ou l'utilisateur
8. Suivi et contrôle constamment alimentés en informations
9. Communication et circulation des informations
10. Résolution des imprévus, déviations, conflits ou crises

Facteurs de succès des projets de développement de produit (Brown & Eisenhardt)

- Équipe de projet : constitution et composition
- Communication de l'équipe : mode et processus utilisés
- Organisation du travail : niveau de rationalité préconisé
- Chef de projet : pouvoir, vision, habiletés de gestion
- Direction : support et type de contrôle exercé
- Fournisseurs : degré et début de l'implication
- Clients : degré et début de l'implication
- Performance du processus : durée et productivité
- Concept de produit : pertinence du concept développé
- Marché : grand, en croissance, à faible compétition

L’équipe de projet constituée

- Champs d'expertise composant l'équipe de projet
- Pertinence des profils des personnes choisies
- Capacité de l'équipe de projet à installer une dynamique « projet » entre les acteurs internes et externes impliqués
- Niveau de familiarité et de confiance entre les membres de l'équipe (mutualisation des informations)

Communication de l’équipe

- Obtention, circulation et mise en commun des informations pertinentes à l'accomplissement du projet
  - Richesse et variété des informations
  - Fréquence du recueil et de la mise à jour des données
  - Volume suffisant des données obtenues et mises en commun
- Capacité de trouver les informations pertinentes à la bonne marche du projet
- Pertinence des mécanismes de circulation et de mise en commun des informations utiles entre les différents acteurs internes et externes

Degré de rationalité

\+

Diagnostics complets, structurés et fiables
Prévisions possibles, précises et robustes
Planification plutôt détaillée, rigoureuse et difficile à changer
Contrôle plutôt pointu, fréquent et extensif

\-

Diagnostics incomplets, peu structurés, peu fiables
Prévisions difficiles, moins précises et moins robustes
Planification peu détaillée, peu rigoureuse, peu souple
Contrôle plutôt général, assez souple, peu fréquent et peu extensif

Informatisation et circulation de l’information

- Présence et utilisation d'un langage commun
- Développement d‘outils afin d’arbitrer les contraintes des différents acteurs du projet
- Mise en réseau informatique des différents acteurs du projet
- Possibilité d’interfaçage ou compatibilité entre les ordinateurs et les logiciels utilisés par les différents acteurs du projet
- Normes de conception
- Rapidité d'accès aux données requises

Implication des clients / fournisseurs

- Moment où les clients / fournisseurs sont impliqués dans le projet (en général, le plus tôt possible)
- Ampleur ou intensité de l'implication des fournisseurs dans le projet
- Niveau d'implication des clients / fournisseurs permettant l'identification la plus tôt possible des problèmes susceptibles de se produire plus loin dans le déroulement du projet
- Niveau d'implication des clients permettant d'améliorer la pertinence du service développé, offrant ainsi une meilleure satisfaction des clients

Evaluation des projets

- Grâce à une démarche formalisée (processus de planification stratégique de l'entreprise)
- Grâce à une démarche informelle (réaction à un événement imprévu)
- Evaluation succincte ou rapide
- Evaluation longue

La relation économique Coût Qualité Fonctionnalité Délai

CQFD
stabilisation

Coût

- Fixé par le client dès le début

Qualité

- Dépend des actions du chef de projet, et en particulier de l'effort de tests; en théorie, elle est fixée dès que le plan qualité est approuvé, généralement en début de projet. Il est particulièrement malvenu et maladroit de réviser le plan qualité à la baisse.

Délai

- Fixé par le client qui en général synchronise le travail avec d'autres projets ; peut varier en cours de projet.

Fonctionnalités

- Service rendu (i.e. fonctions offertes) proposé par le maître d' oeuvre à son client ; les fonctionnalités peuvent souvent être négociées en contre partie du coût et du délai.

Maturité: CMMI - Capability Maturity Model Integration

- N5 Optimiser : anticipation des changements, prévention des défauts, adaptation dynamique, recherche constante de l'optimum CQFD durable.
- N4 Piloter : toute action est mesurée et régulée de façon quantitative en fonction de la finalité du projet,
- N3 Définir : approche systémique et constructive, fondée sur les processus, de ce qu'il faut faire pour satisfaire le client final,
- N2 Reproduire (discipliné) : on s'efforce de reproduire ce que font les experts par adoption de bonnes pratiques fondées sur la confiance que l'on a dans l'expert,
- N1 Initial : c'est le « laissez faire »,

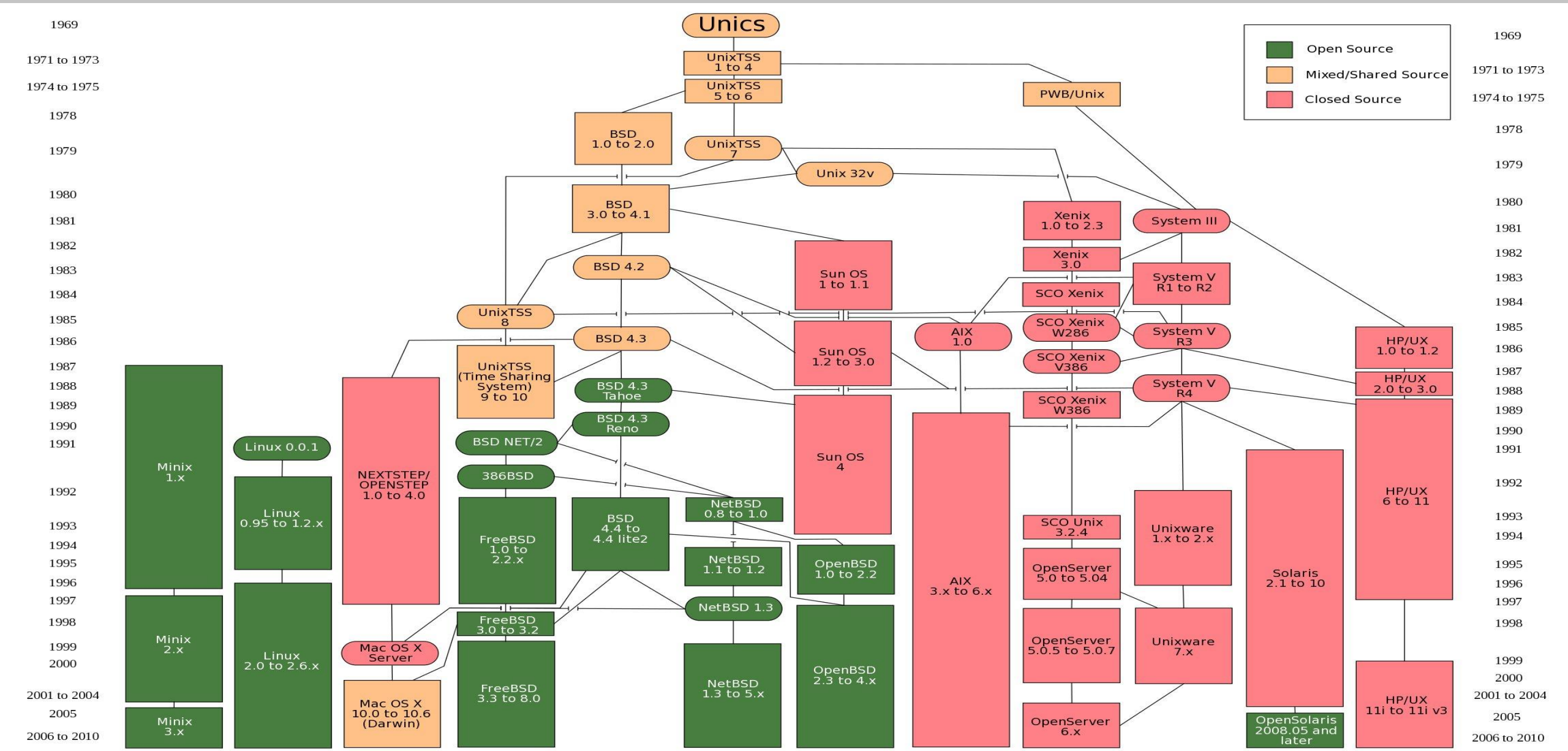
CMMI créé par le département de la défense US (DoD)

Lois « informatiques »

- **Loi de Murphy (LEM)**
  - « S'il y a plus d'une façon de faire quelque chose, et que l'une d'elles conduit à un désastre, alors quelqu'un le fera de cette façon »
- **Loi de Parkinson**
  - Elle affirme que « le travail s'étale de façon à occuper le temps disponible pour son achèvement ».
- **Loi de Moore**
  - Loi selon laquelle la complexité des circuits intégrés, le nombre de transistors et la puissance des processeurs doublaient tous les deux ans.
- **Loi de Pareto**
  - Loi des 80/20; environ 80 % des effets sont le produit de 20 % des causes
- **Loi de Say**
  - « l'offre crée sa propre demande »

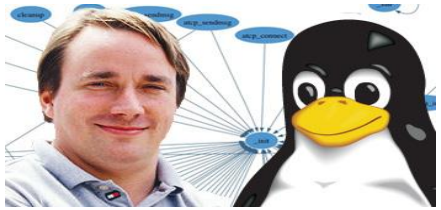
THE METHOD WON'T SAVE YOU
(but it can help !)

UNIX


1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2010
Open Source
Mixed/Shared Source
Closed Source
Unics
UnixTSS 1 to 4
UnixTSS 5 to 6
PWB/Unix
BSD 1.0 to 2.0
UnixTSS 7
Unix 32v
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
UnixTSS 8
SCO Xenix W286
AIX 1.0
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
HP/UX 1.0 to 1.2
SCO Xenix V386
UnixTSS (Time Sharing System) 9 to 10
HP/UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
SCO Xenix W386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
Sun OS 4
BSD NET/2
NEXTSTEP/ OPENSTEP 1.0 to 4.0
386BSD
HP/UX 6 to 11
NetBSD 0.8 to 1.0
SCO Unix 3.2.4
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
Unixware 1.x to 2.x
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
AIX 3.x to 6.x
OpenServer 5.0 to 5.04
Solaris 2.1 to 10
Minix 2.x
NetBSD 1.3
Linux 2.0 to 2.6.x
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
Unixware 7.x
Mac OS X Server
FreeBSD 3.3 to 8.0
NetBSD 1.3 to 5.x
OpenBSD 2.3 to 4.x
HP/UX 11i to 11i v3
Minix 3.x
Mac OS X 10.0 to 10.6 (Darwin)
OpenServer 6.x
OpenSolaris 2008.05 and later

SLOC

Année	Nbre LC (en million)	Linux	Windows
1991	0.01	0.11	
1992	2.5		3.1
1992	4.5		NT 3.1
1993	7.5		NT 3.5
1994	0.176	1.0.0	
1996	11		NT 4.0
2000	29		2000
2001	40		XP
2003	5.93	2.6.0	
2007	50		VISTA
2009	40		7
2013	17	3.10	
2017	25	4.12	
Dernière version	LINUX : 6.5.3 (13 septembre 2023 WINDOWS 11, version 22H2		



Modèle d’estimation

- La règle 40 - 20 - 40 stipule qu'en moyenne :

40% de l'effort est requis pour spécifier et concevoir,

20% de l'effort va à la programmation,

40% de l'effort va aux tests de vérification et de validation.

- \+ *Se focaliser uniquement sur la programmation, c'est se préparer à une catastrophe qui ne manquera certainement pas d'arriver.*
- Il est normal de consacrer 15 à 20% des ressources du projet à des tâches dites transversales (gestion de l'équipe, assurance qualité, formation, documentation, configuration matériel et logiciel, etc.).

Le marché de l'informatique et du digital

- **Ingénieur cloud.** En 2024, les salaires devraient grimper à des niveaux compris entre 55.000 et 70.000 euros après deux à cinq ans d'expérience, et atteindre 85.000 euros à partir de 10 ans d'expérience. Les hausses de salaire oscilleront ainsi entre 7 et 13% par rapport à l'année 2023.
- **Ingénieur DevOps.** Ce métier consiste à introduire des processus, des outils et des méthodes pour équilibrer les besoins tout au long du cycle de développement d'un logiciel. L'an prochain, les salaires atteindront de 55.000 à 70.000 euros après deux à cinq ans d'expérience, soit une hausse d'environ 10 à 15% par rapport à 2023.
- **Pentester.** Il s'agit d'un professionnel de la cybersécurité dont les missions visent à limiter et anticiper au maximum les failles et intrusions dans un système informatique. En début de carrière, un pentester pourra toucher entre 40.000 et 50.000 euros (+13% par rapport à 2023). Les profils ayant entre deux et cinq d'expérience pourront eux prétendre à un salaire compris entre 45.000 et 60.000 euros (+9 à 13%).

Source: Capital