Génie logiciel

Concepts fondamentaux

Bruno MERMET, Université du Havre

Nécessité du Génie Logiciel

Développement d'un logiciel

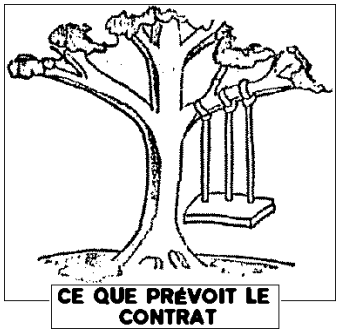
- Caractéristiques souhaitées :
  - Adéquation avec les besoins
  - Maintenance aisée
  - Bon marché
  - Rapidement développé
- Comment ?

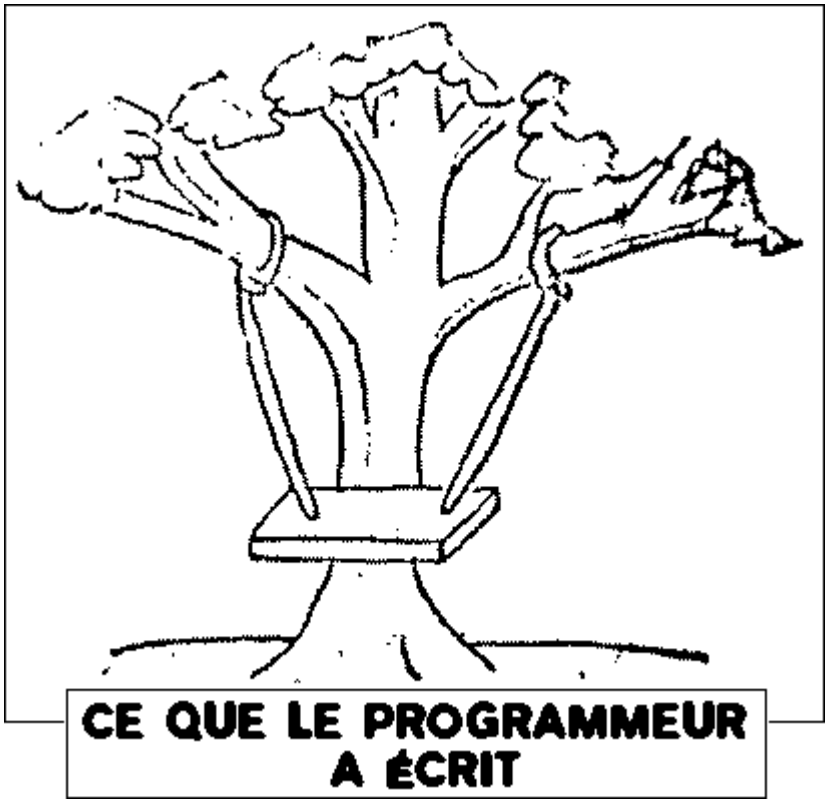
Le génie logiciel = outils + méthodes

Sans Génie Logiciel (1)

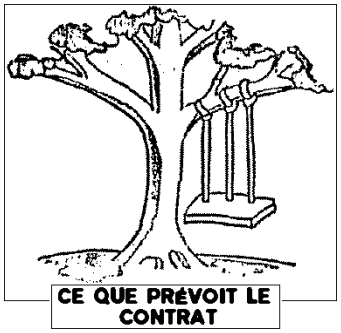
Sans Génie Logiciel (2)

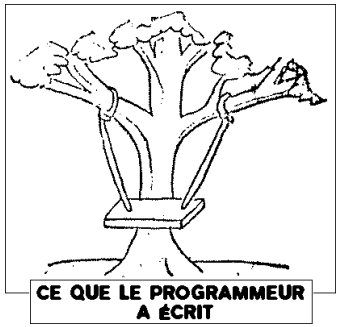
Sans Génie Logiciel (3)

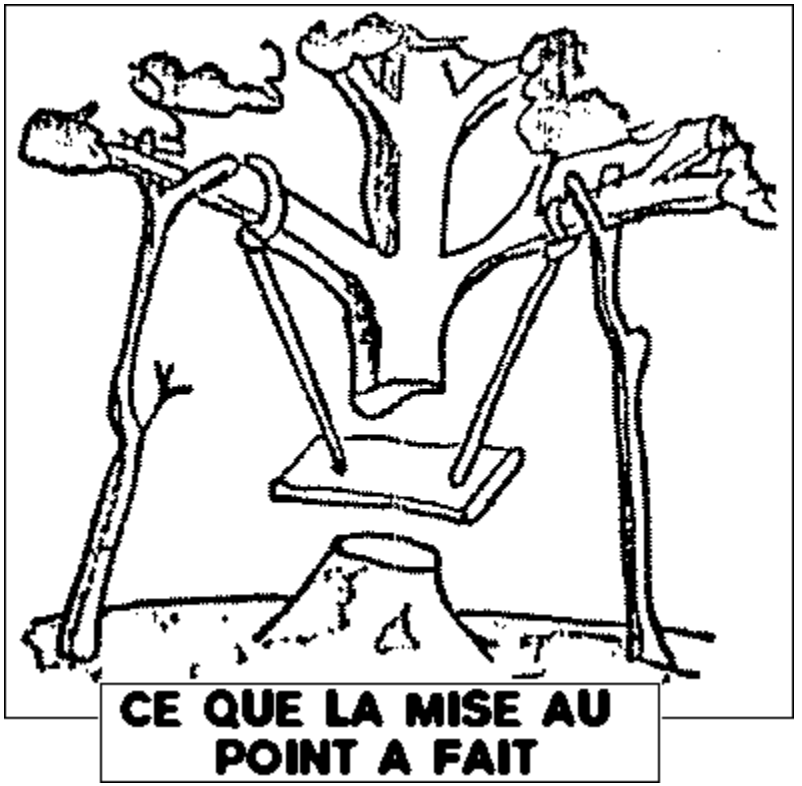




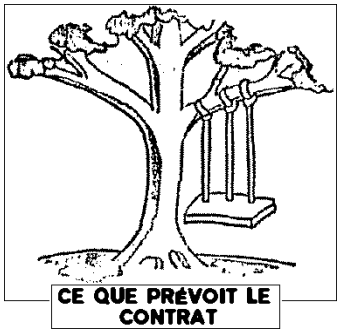
Sans Génie Logiciel (4)

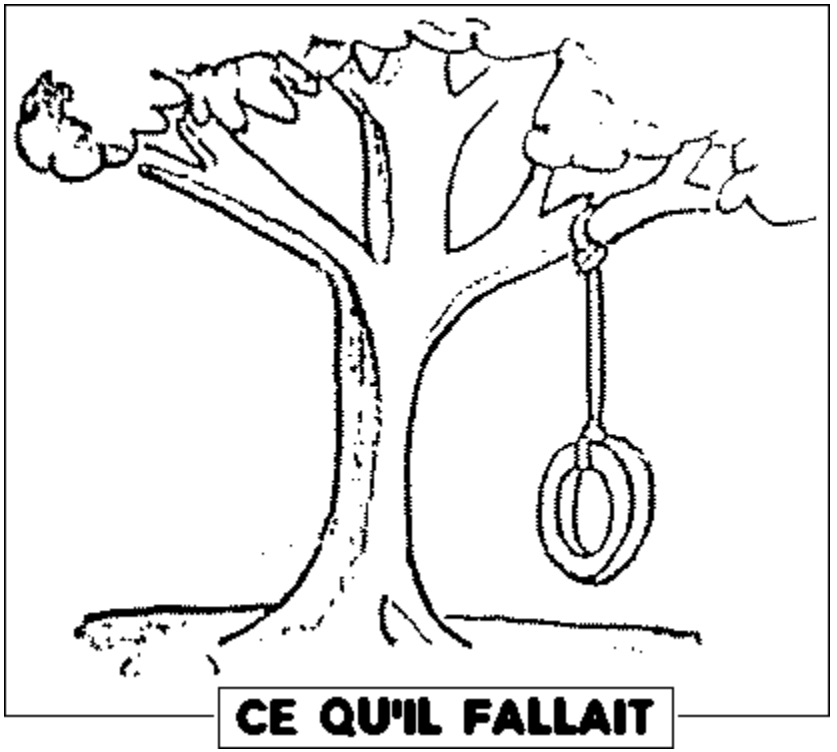


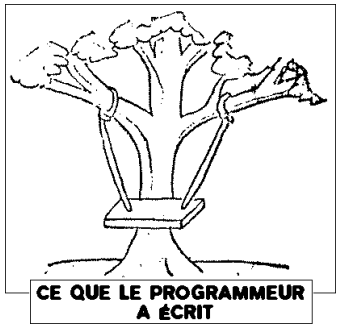


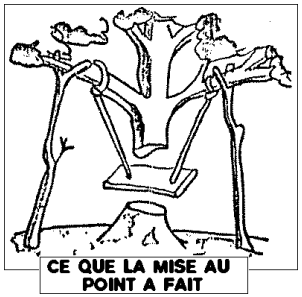


Sans Génie Logiciel (5)









Etapes de développement

- Analyse (du problème)
  comprendre et recenser les besoins
  ¶ spécification (par exemple cahier des charges)
- Conception (du logiciel)
  préliminaire
  - éclater le logiciel en sous-parties
  - définir les interfaces entre ces sous-parties

  ¶ architecture du logiciel
  détaillée
  - préciser l'architecture des sous-parties
- Implantation
  codage

Validation du logiciel

- Définition
  - assurer la cohérence entre les besoins et le logiciel obtenu
  - garantir au mieux l'absence d'erreur
- Moyens
  - prototyper

    développer et « essayer » une partie du logiciel à concevoir
  - tester

    effectuer des essais de fonctionnement et vérifier le résultat obtenu par rapport au résultat attendu
  - prouver

    vérifier mathématiquement la cohérence de la conception/du code par rapport à la spécification (qui doit être formelle)

Comparaison des moyens de validation

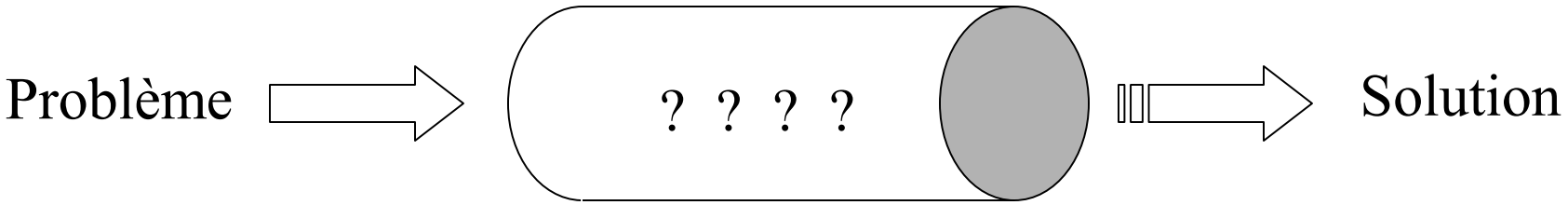
	Ce qui est vérifié	Cible	Avantage	Inconvénient
Prototyper	La bonne compréhension du problème	Développeurs	Intervient très tôt	Aucune garantie sur résultat
Tester	Un comportement correct dans des cas bien précis	Programme exécuté	Facilité de mise en œuvre Nombreux outils	Intervient à la fin pas exhaustif
Prouver	La correction par rapport aux propriétés spécifiées formellement	Code et/ou conception	Garantie obtenue	Lourd à mettre en place

Conséquence :

- Preuve réservée aux « systèmes critiques »
- Test toujours utilisé

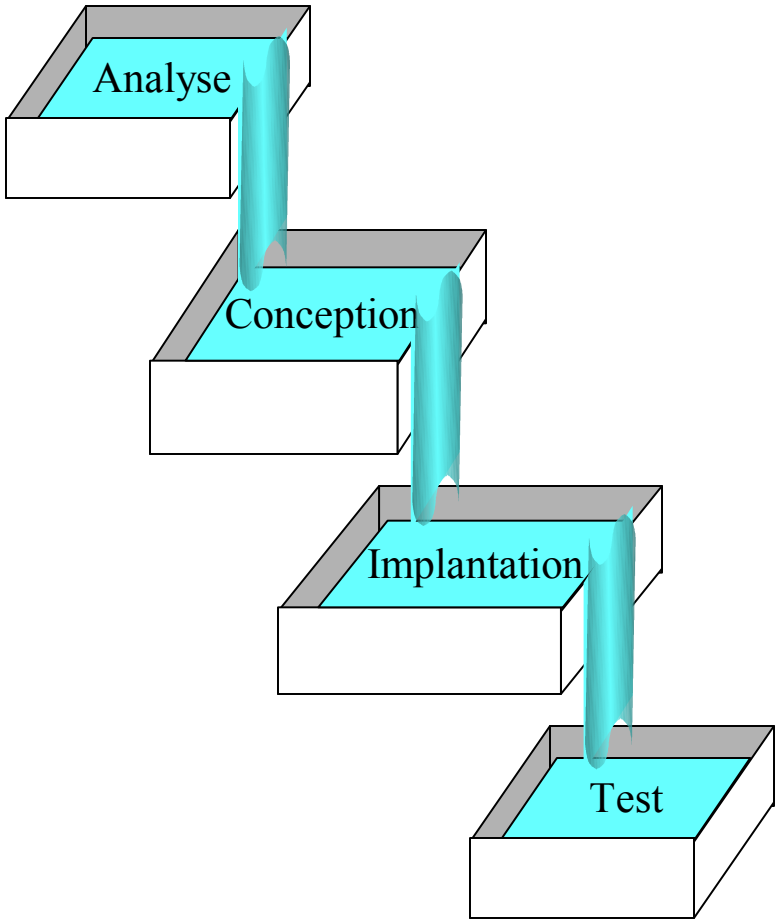
Le « cycle de vie » du logiciel

1. Le modèle en tunnel



- Caractéristique
  - absence de modèle !
- Utilisation
  - A éviter !
  - réservée aux petits projets

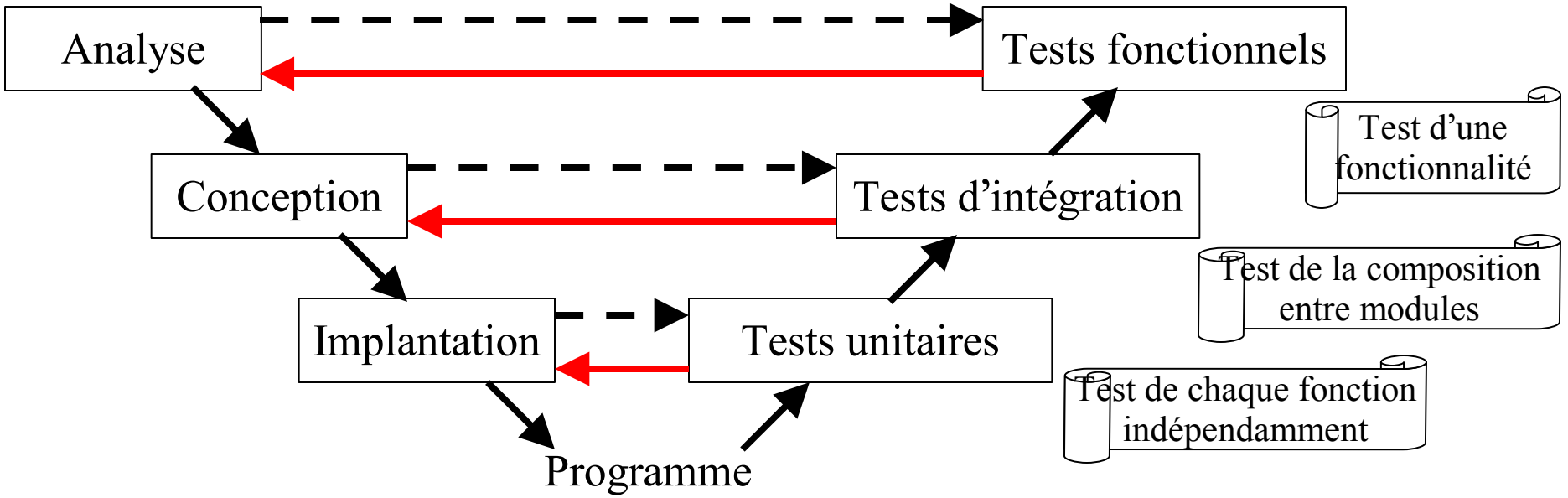
2. Le modèle en cascade

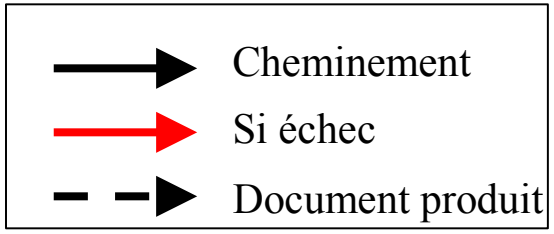


Caractéristiques

- phases bien identifiées
- test :
  - intervient tard
  - est mal cadré

3. Le modèle en V



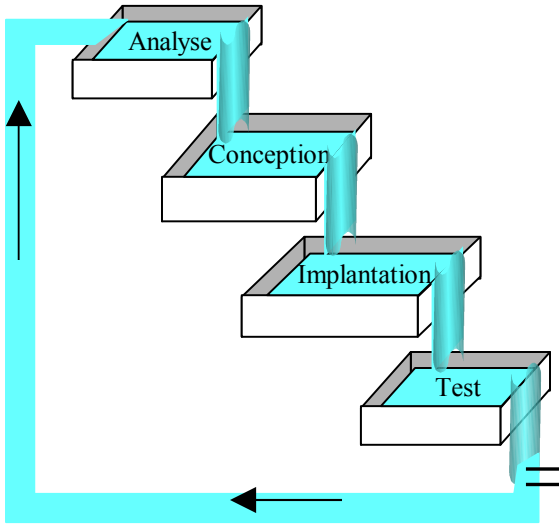


Caractéristiques

- Modèle en cascade amélioré
- Le plus utilisé

4. Le modèle itératif

Etape n :



⇒ Prototype n

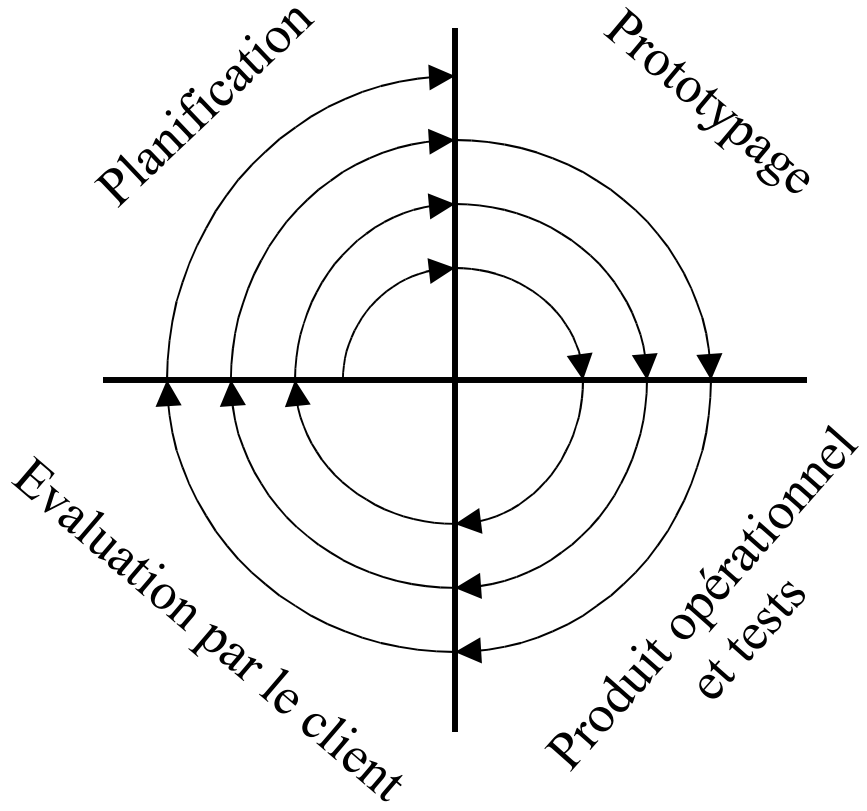
- Principe

  A chaque étape, on rajoute de nouvelles fonctionnalités

- Caractéristiques
  - Chaque étape est relativement simple
  - On peut tester et essayer au fur et à mesure le logiciel que l'on développe

4 bis. Le modèle en spirale

Autre vision du modèle itératif



Critères de qualité d'un logiciel

Qualité et Assurance Qualité

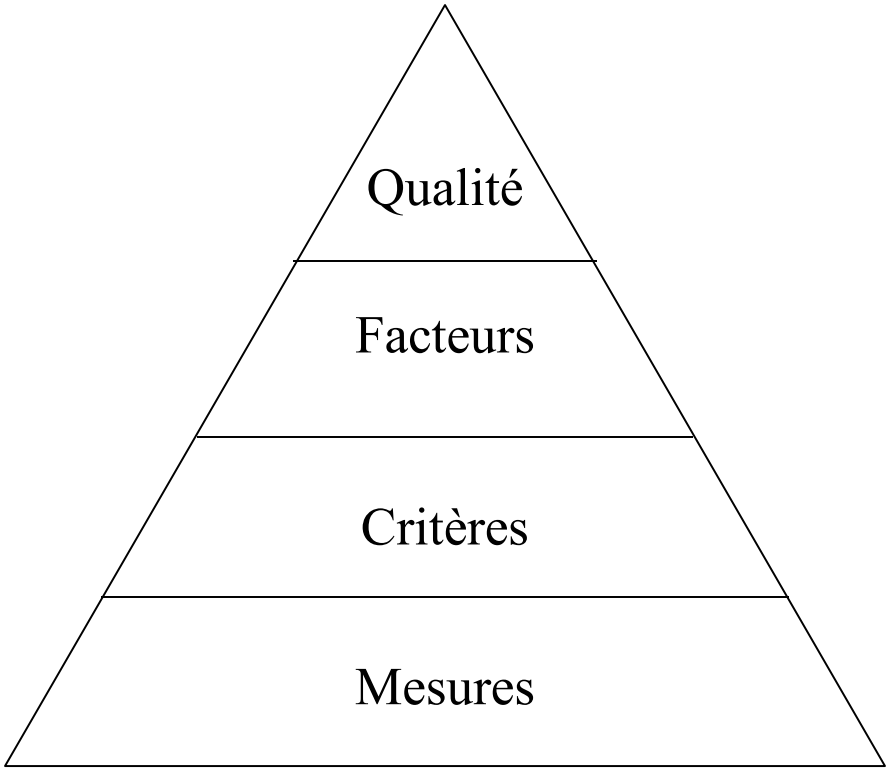
- Qualité

  ensemble de caractéristiques que doit satisfaire un produit pour répondre aux besoins
- Corollaire

  un produit est « de qualité » dès qu'il répond aux critères de qualité qu'on lui a attribués. Donc un produit bas de gamme peut être considéré « de qualité ».
- Assurance qualité

  processus permettant normalement d'assurer la qualité du produit

Evaluer la qualité



Les facteurs de la qualité

- Point de vue utilisateur
  - Fiabilité : pas de « plantage »
  - Sécurité : pas de mise en danger de vies humaines/de machines
  - Intégrité : protection des données contre les intrusions
  - Ergonomie : utilisation aisée du logiciel
  - Efficacité : minimisation des ressources (temps, mémoire, etc.)
- Point de vue développeur
  - Testabilité : facilité de vérification du code
  - Maintenabilité : détection et correction aisée des erreurs
  - Flexibilité : évolution facile
- Point de vue communication
  - Réutilisabilité : utiliser les modules développés dans de futurs projets
  - Portabilité : possibilité de faire tourner le logiciel sur d'autres architectures
  - Compatibilité : échange de données avec d'autres logiciels

Des critères de qualité

Traçabilité
Complétude
Précision
Cohérence
Robustesse
Simplicité
Modularité
Généralité
Extensibilité
Efficacité de stockage
Efficacité d'exécution

Traçage des accès
Contrôle des accès
Souplesse des interfaces
Facilité d'exploitation
Facilité d'apprentissage
Indépendance machine
Indépendance système
Normalisation des communications
Standardisation des structures de données
Concision
Conformité

Relations entre facteurs et critères

- *Fiabilité* : cohérence, robustesse, conformité, précision
- *Sécurité* : complétude, précision, cohérence, robustesse
- *Intégrité* : traçage et contrôle des accès
- *Ergonomie* : souplesse des interfaces, facilité d'apprentissage et d'exploitation
- *Efficacité* : efficacité de stockage et d'exécution, (concision)
- *Testabilité* : traçabilité, simplicité, modularité
- *Maintenabilité* : traçabilité, modularité, traçage des accès, simplicité
- *Flexibilité* : généralité, extensibilité, modularité
- *Réutilisabilité* : généralité, normalisation des com. et structures de données
- *Portabilité* : indépendance machine et système
- *Compatibilité* : normalisation des communications et structures, de données

Mesure

- Définition
  - un nom
  - une méthode de calcul
  - une valeur optimale
  - un intervalle dont il ne faut pas sortir
- Exemples
  - nombre d'imbrications maximal, (souvent, $nim \leq 5$)
  - taux de couverture des tests
  - proportion de commentaires par rapport aux lignes de code
- Attention !
  - Une mesure :
    - indicateur quantitatif
    - pas d'aspect qualitatif

Mise en place de la qualité

- Plan Qualité Logiciel (PQL)

  document précisant pour un logiciel donné les phases de développement et les facteurs et critères de qualités, ainsi que les niveaux requis pour ces derniers.

- Système Qualité

  dispositif mis en place par une entreprise pour vérifier le respect de la procédure d'Assurance Qualité

Le Suivi de la qualité : la revue

- Définition
  - réunion de présentation/prise de décision
  - menée par le chef de projet ou le chef d'une phase
- Types de revues :
  - Revue de début de projet
    - présentation du projet, du planning, de la démarche
  - Revue de fin de phase
    - existence et qualité de la documentation
    - adéquation du point atteint avec les objets fixés au départ
  - Revue de fin de projet
    - analyse des problèmes rencontrés, conséquences méthodologiques
    - préparation de la maintenance

Le suivi de la qualité : l'inspection

- Définition
  - contrôle approfondi d'un point particulier
  - menée par un spécialiste du domaine inspecté
- Documents inspectés
  - la documentation
  - les sources
  - les dossiers de test
- Principe
  - examen systématique de certains points
  - recherche de potentiels défauts
  - vérification de l'application de certaines règles

Le suivi de la qualité : l'audit

- Définition
  - examen méthodique d'un aspect donné (produit, processus)
  - mené par quelqu'un d'extérieur à l'équipe de développement :
    - client, organisme de contrôle ⇒ audit externe
    - responsable Assurance Qualité ⇒ audit interne
- But
  - existence et conformité du PQL avec les exigences requises
  - application correcte du PQL
  - état d'avancement du projet
  - vérification du processus de gestion de configuration

Méthodes d'analyse et de conception

Analyse fonctionnelle

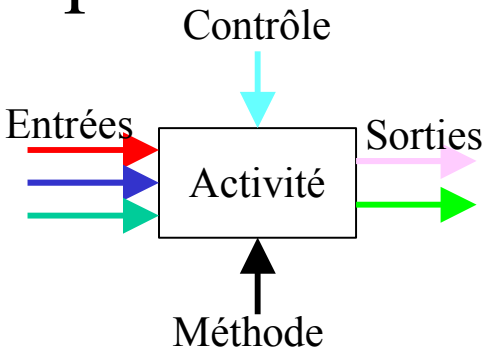
- Démarche

  Recenser les fonctionnalités à implanter
- Résultat

  cahier des charges fonctionnel
- On distingue
  - fonctions de service : besoins des utilisateurs
  - fonctions techniques : requises pour implanter les fonctions de service
- Pour chaque fonction, préciser
  - son importance
  - des critères de qualité
- Application : SADT

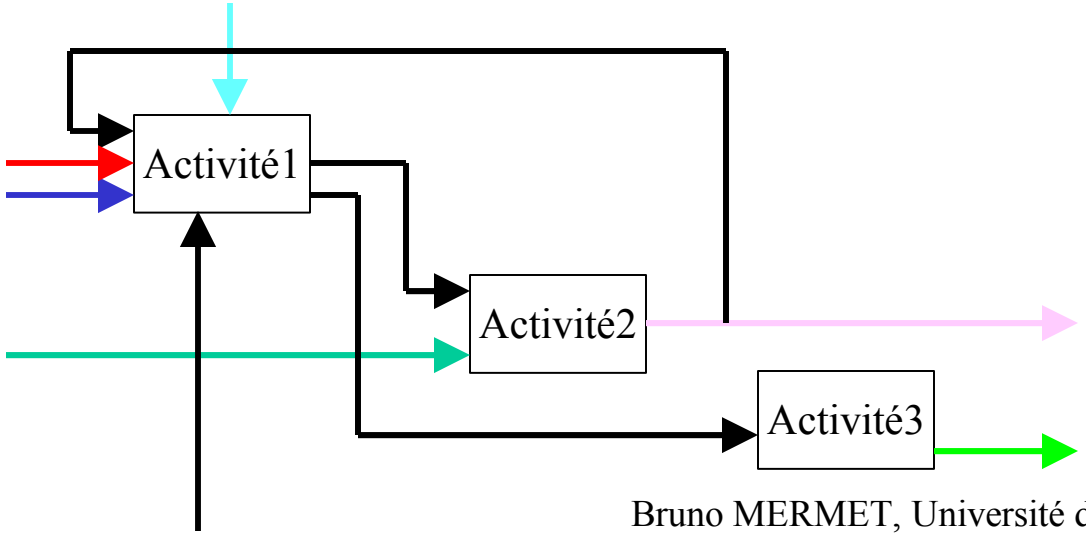
Présentation de S.A.D.T.
(Structured Analysis and Design Technique)

- Représentation sous la forme d'actigrammes :



Les entrées et sorties sont des données

- Décomposition de la « boîte » *Activité* :



N.B. : il existe aussi la vue duale, les datagrammes, dans lesquels les boîtes représentent des données et les entrées/sorties des activités

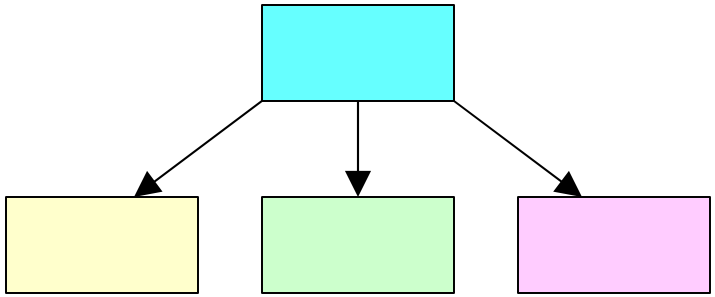
Approche objet

- Démarche

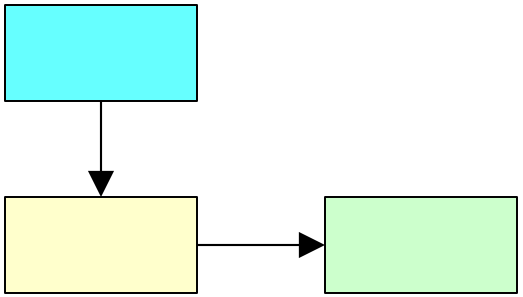
  analyser les objets intervenant dans le système
- Résultat
- diagramme de classes
  - ensemble des classes nécessaires
  - liens entre les classes
  - méthodes et éventuellement attributs des classes
- Applications
  - OMT, BOOCH, OOSE, UML

Approches structurées

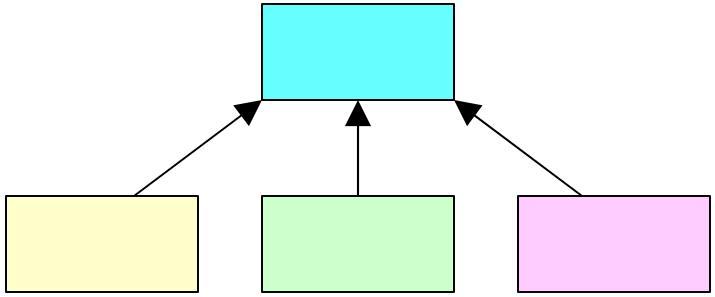
- Descendante



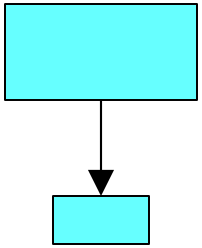
- Séquentielle



- Ascendante



- Récursive



Les différents types de langages de programmation

Taxonomie de base

- Langages impératifs
  - programme = suite d'instructions (éventuellement parallèles)
  - exemples : C, C++, ADA, PASCAL, BASIC, etc.
- Langages fonctionnels
  - programme = composition de fonctions
  - exemples : LISP, CAML, SML
- Langages logiques
  - programme = ensemble de faits et de règles d'inférence
  - exemples : PROLOG, Mercury

Niveaux de langages

- Assembleur/langage machine
- Langage « intermédiaires »
  - exemple : C
  - structure de l'implantation doit encore être connue
- Langage « évolué »
  - exemple : ADA, Eiffel, Perl
  - structure interne complètement masquée
- Langage de quatrième génération (L4G)

Classification selon le code exécuté

- Langage interprété (ex. Perl)
  - code exécuté = le source du programme
  - chaque instruction est d'abord traduite puis exécutée
  - exécution lente, mais mise au point aisée
- Langage compilé (ex. C)
  - code exécuté = une version en langage machine du programme
  - le source est traduit une fois pour toute
  - exécution rapide, mais recompilation nécessaire à chaque modification
- Langage semi-interprété (ex. Java)
  - code exécuté = traduction en un pseudo-assembleur du source

Les langages orientés objet

- Idée de base
  - Toute donnée est une structure à laquelle sont rattachées des fonctions appelées *méthodes*
  - Programme = ensemble de données reliées entre elles
- Langages concernés : tous
  - Perl (impératif interprété)
  - Objetive Caml (fonctionnel interprété)
  - Eiffel (impératif compilé)
  - Java (impératif semi-interprété)

  ...

Tests et mise au point

La notion de test

- Principe du test
  - des données en entrée
  - un résultat attendu

  } jeu de test

  - une exécution du logiciel obtenu sur les données
  - comparaison du résultat obtenu avec le résultat attendu
- Interprétation du test
  - si échec : il y a un problème
  - si réussite : ??

Garanties apportées par le test

- Si ensemble exhaustif de tests :

  Réussite de tous les tests = preuve (*model checking*)

- Sinon (cas le plus courant) :

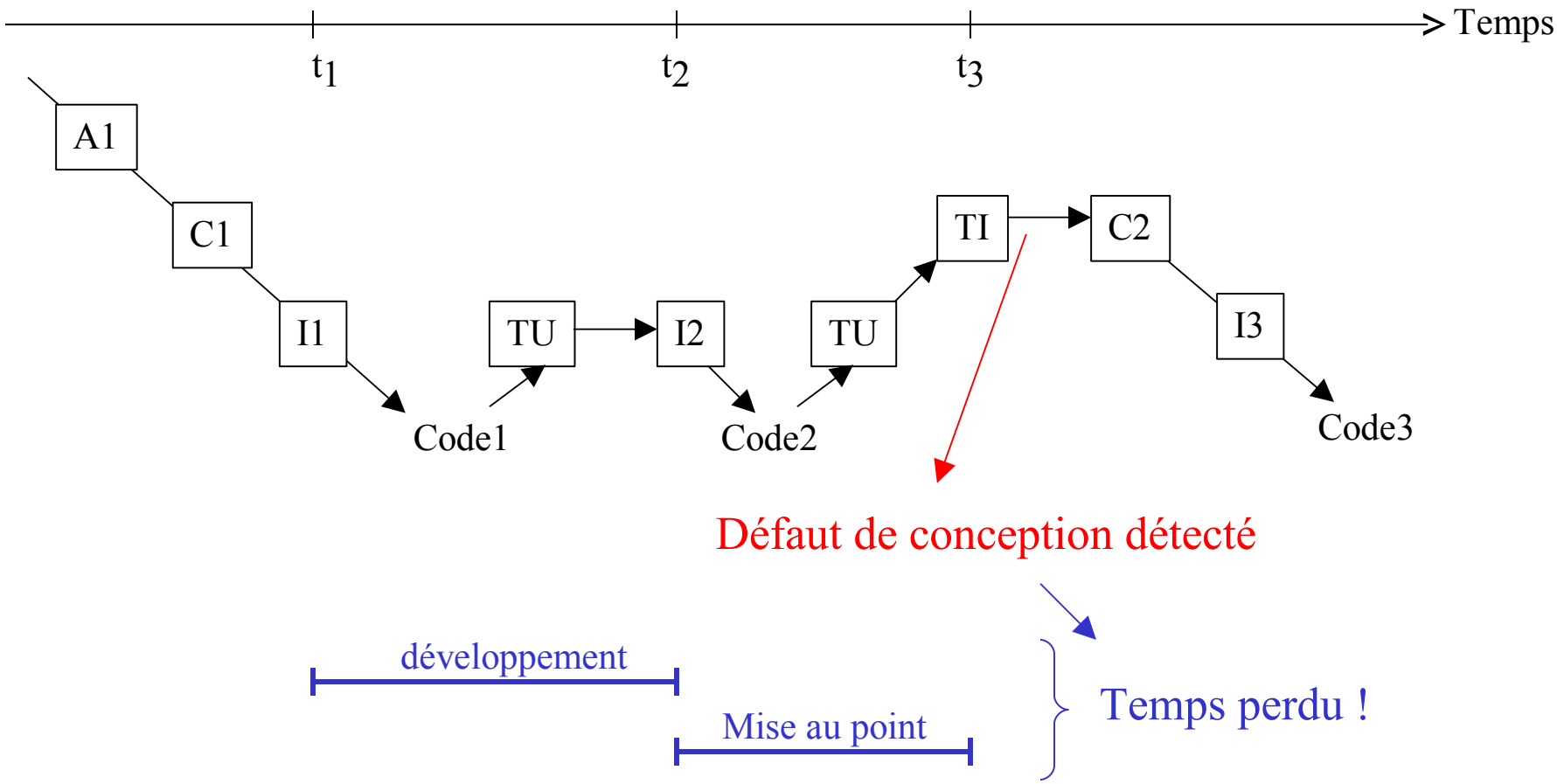
  Réussite de tous les tests ne garantit rien !

  ¶ Choisir judicieusement l'ensemble des jeux de tests

Les niveaux de test

- Tests unitaires :
  - vérification du bon comportement d'une fonction, d'un module, par rapport à sa spécification
- Tests d'intégration :
  - vérification du bon fonctionnement de la collaboration entre modules
- Tests fonctionnels :
  - vérification globale d'une fonctionnalité du système décrite lors de l'expression des besoins

Tests et modèle en V

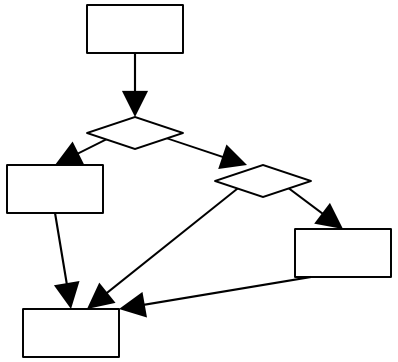


Typologie des tests

- Tests « boîte noire »
  - écrits indépendamment du code
  - vérifient le codes sur différents types d'entrées possibles
- Tests « boîte blanche »
  - écrits pour avoir un *taux de couverture* (Tc) maximum

$$Tc = \frac{Nb\_branches\_parcourues}{Nb\_branches\_total}$$

Exemple à 3 branches :



Mise au point

Si un test échoue

- créer un *rapport d'anomalie*
- selon le niveau du test, revoir la phase d'analyse, de conception ou de codage
- générer les nouvelles versions de documents

Composants auto-testables

- Définition

  composant incluant :

  - des fonctions de tests unitaires
  - une fonction de test du composant

- Intérêt

  – réutilisabilité du composant étendue à son test

Documentation

Les types de documents

- Documents de pilotage de projet

  précisent le déroulement du projet (phases, durées, intervenants)
- Documents d'étude et de développement

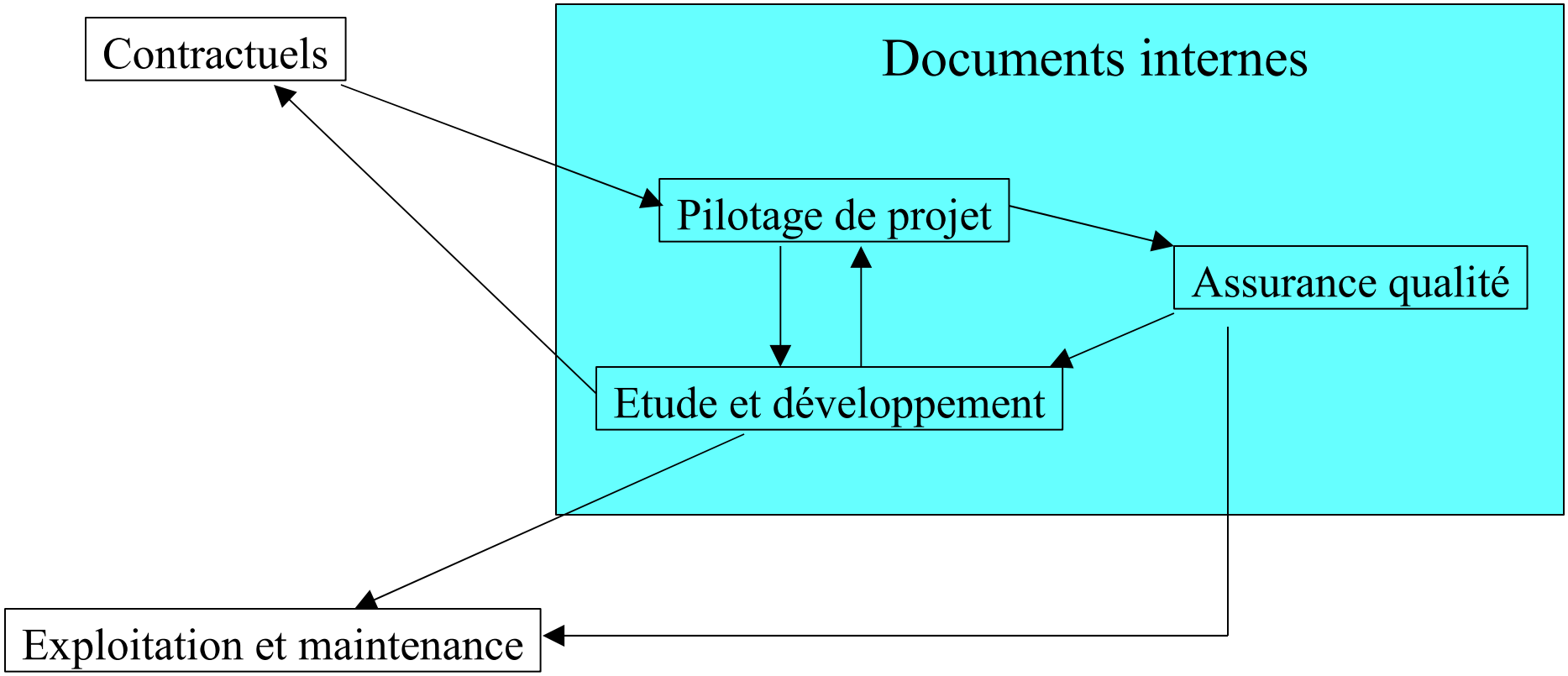
  expliquent les problèmes, les solutions choisies, les choix effectués
- Documents d'assurance qualité

  caractérisent les facteurs, critères, mesures de qualité
- Documents d'exploitation et de maintenance

  détaillent l'installation du logiciel, l'origine des problèmes
- Documents contractuels

  servent à établir les points d'accord entre client et fournisseur

Influences entre documents



Gestion de la documentation

- Utilisation de documents standardisés
- Donner à tout document :
  - un type
  - un nom
  - une date de dernière modification
  - un numéro de version
- Sur chaque document, préciser de plus :
  - son lieu de rangement
  - les participants à sa rédaction

Quelques exemples standards de documents

- Plan Qualité Logiciel
  mise en œuvre de la qualité
- Dossier de traçabilité
  lien entre les différents documents, le code
- Dictionnaire
  lien entre termes et concepts (1 terme = 1 concept)
- Dossier de test
  que tester, comment
- N.B. : le code produit est un document !

Maintenance

Pour tout problème :

- noter la date de mise en évidence
- le caractériser par un ou plusieurs tests
- préciser les conséquences
- expliquer les causes une fois celles-ci localisées et justifier
- décrire et mettre en œuvre la solution adoptée
- modifier le code et sa documentation
- préciser le nouveau numéro de version et la date de correction

Gestion de projet

Les différentes tâches

- Phases préliminaires
  - Estimation de charge
  - Ecriture du PQL
  - Constitution de l'équipe
- En cours de développement
  - Mise en place des différentes phases (de la spécification aux tests)
  - Gestion des anomalies
  - Contrôle de la qualité
  - Suivi de projet

Types de projet

D'après Boehm, Trois types de projet :

- Organiques (environnement stable, pas de contraintes temps réel)

  exemples : compilateur, calcul scientifique

- Médians (environnement instable, contraintes temps réel)

  exemples : automates programmables, systèmes de régulations

- Imbriqués (environnement très instable, performances en temps et précision difficiles à atteindre)

Estimations

- Nombre de lignes à produire
  - découper le projet en modules
  - pour chaque module, faire la moyenne de plusieurs estimations
  - faire la somme des moyennes obtenues pour chaque module

  ¶ Taille : nombre approximatif de milliers de lignes à produire
- Charge (en Homme.Mois)

  $\text{Charge} = f \times C \times \text{Taille}^P$ avec :

  (charge hors spécification)

  f : facteur de charge, vaut 1 en général

Projet	C	P
Organique	2,4	1,05
Médian	3,0	1,12
Imbriqué	3,6	1,20

Quelques facteurs de charge

- Fiabilité requise
  si risque de pertes de vies humaines, pertes financières important
  f = 1,29
- Complexité du logiciel
  si problèmes mathématiques complexes, communications
  f = 1,45
- Temps d'exécution
  f = 1,22
- Contrainte mémoire
  f = 1,19
- Equipe peu qualifiée
  f = 2,06

Autres évaluations

- Délai (en mois)

  Délai = C × Charge^{P} avec :

Projet	C	P
Organique	2,5	0,38
Médian	2,5	0,35
Imbriqué	2,5	0,32

- Productivité
  - Prod = Taille/Charge
- Effectif moyen
  - Em = Charge/Délai

Exemples d'évaluation

Développer un projet standard de 50 000 lignes :

Charge = $2,4 * 501^{,05}$ = 146 HM*

Délai = $2,5 * 1460^{.38}$ = 17 mois

Productivité = 50 000/146 = 342 lignes/HM

= 17 lignes/homme/jour

Effectif moyen =146 / 17 = 8,5 Hommes

*1 HM = 1 Homme.Mois = 152 heures = 19 jours

Répartitions par phase

Type projet	Phase	Charge	Délai
Organique	Spécification	6%	12%
	Conception préliminaire	16%	19%
	Conception détaillée	25%	55%
	Codage et Tests unitaires	40%	
	Intégration	19%	26%
Médian	Spécification	7%	20%
	Conception préliminaire	17%	26%
	Conception détaillée	25%	48%
	Codage et Tests unitaires	35%	
	Intégration	23%	26%
Imbriqué	Spécification	8%	32%
	Conception préliminaire	18%	34%
	Conception détaillée	26%	
	Codage et Tests unitaires	28%	40%
	Intégration	28%	26%

Exemple (suite)

- Spécification
  - charge : 0.06 * 146 = 8,76
  - délai : 0,12 * 17 = 2,04

  } 8,76/2,04 = 4,29 hommes
- Conception préliminaire
  - charge : 0,16 * 146 = 23,36
  - délai : 0,19 * 17 = 3,23

  } 23,36/3,23 = 7,23 hommes
- Réalisation
  - charge : 0,65 * 146 = 94,9
  - délai : 0,55 * 17 = 9,35

  } 94,9/9,35 = 10,14 hommes
- Intégration
  - charge : 0,19 * 146 = 27,74
  - délai : 0,26 * 17 = 4,42

  } 27,74/4,42 = 6,27 hommes
- Au total
  - charge = 154,76 HM; délai = 19,04 mois

Contraintes en personnel

- Hypothèses :
  - charge C, délai D avec un effectif E
  - contrainte : effectif limité à $E_0 < E$

- Résultat :
  - $D' = D*(E/E_0)^{0,76}$ $(D' > D)$
  - $C' = C*(D/D')^{0,32}$ $(C' < C)$

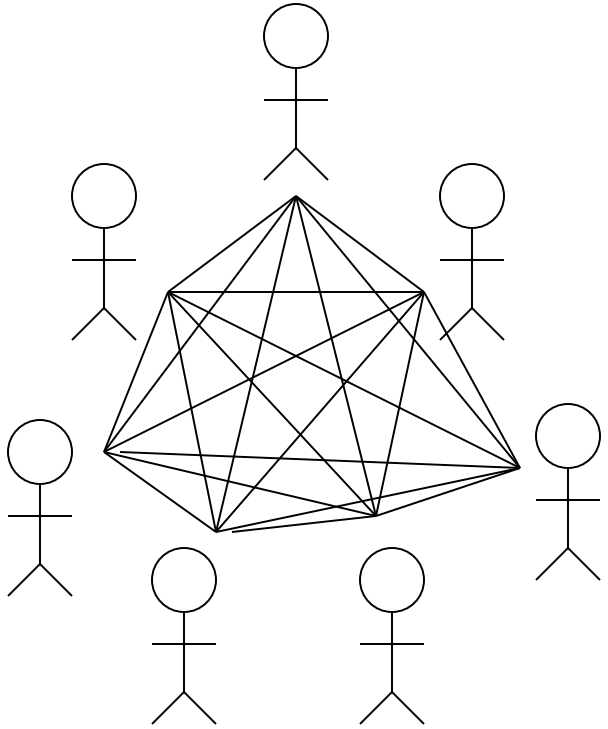
Application à l'exemple

- On suppose 7 personnes disponibles maximum
- Application
  - Spécification : rien de changé
  - Conception préliminaire :
    - $D' = 3,23 * (7,23/7)^{0,76} = 3,31$
    - $C' = 23,36 * (3,23/3,31)^{0,32} = 22,8$
  - Réalisation :
    - $D' = 9,35 * (10,14/7)^{0,76} = 12,39$
    - $C' = 94,9 * (9,35/12,39)^{0,32} = 86,72$
  - Intégration : rien de changé
- Au total

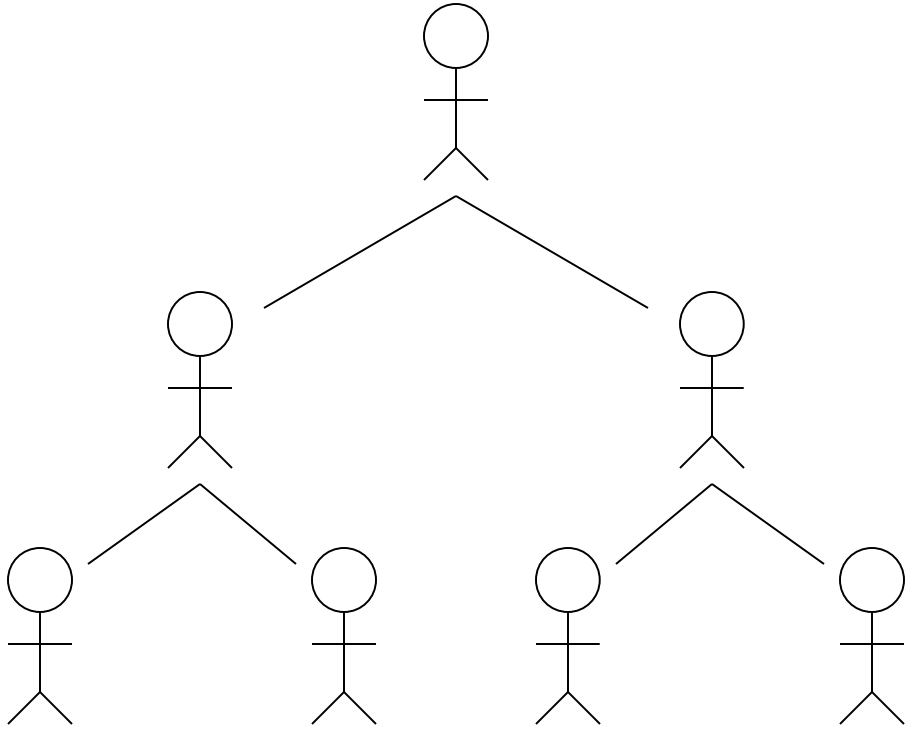
  charge = 146,02 HM ; délai = 22,16 mois

Structuration de l'équipe

Pourquoi structurer ?

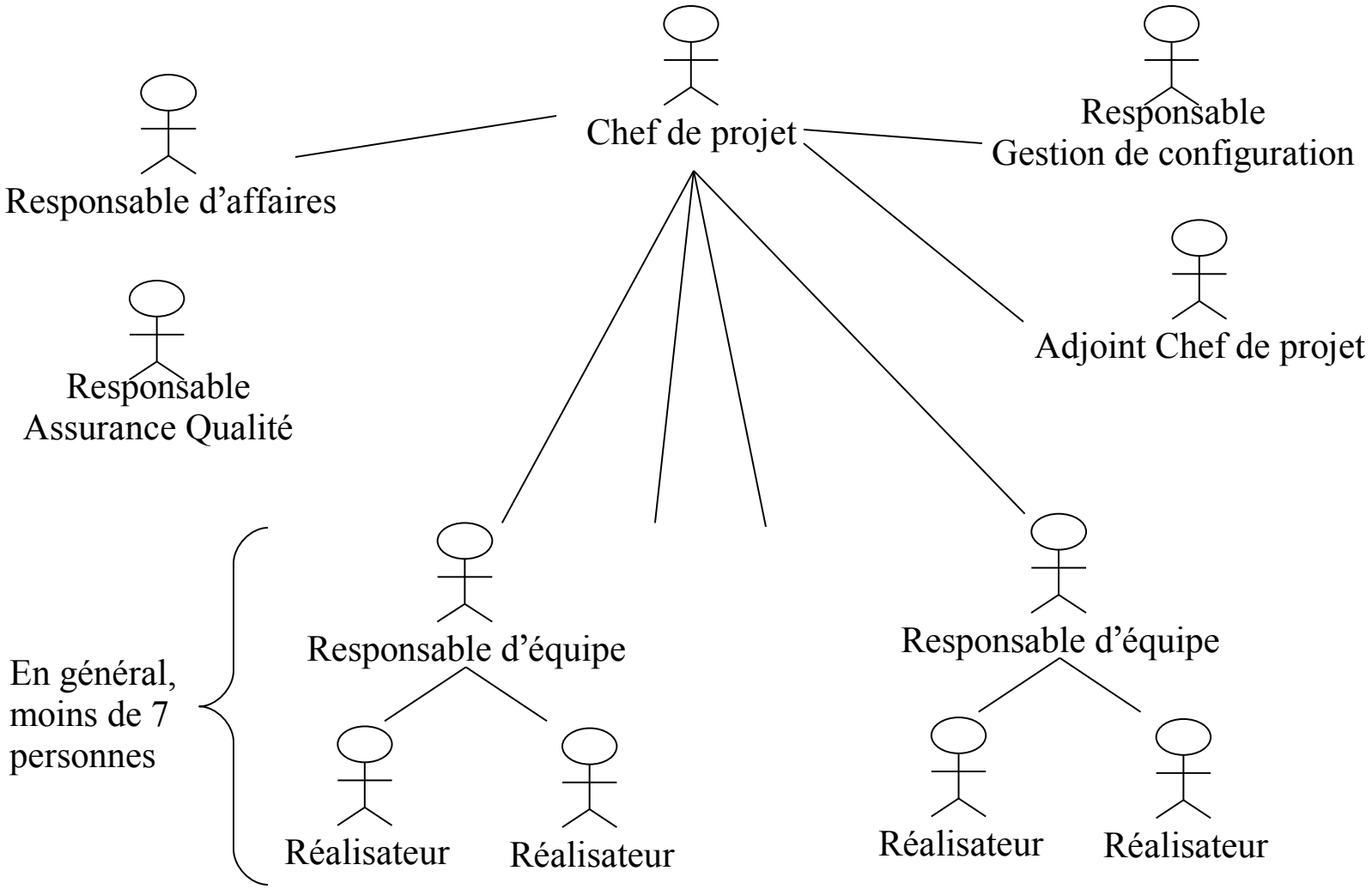


n(n-1)/2 interactions
21 interactions pour 7 personnes



$b.\log_b n$ interactions si b branches
6 interactions pour 7 personnes

Structure traditionnelle d'une équipe



Rôles des intervenants (1)

- Chef de projet :
  - mise en œuvre de la qualité
  - assurer le suivi du projet
  - coordonner les équipes
  - diriger les différentes phases (sauf conception détaillée et tests unitaires)

- Chef d'équipe :
  - diriger la conception détaillée et les tests unitaires le concernant
  - assurer l'interface entre son équipe et le chef de projet

Rôles des intervenants (2)

- Responsable Assurance Qualité
  - Mettre au point le Plan Qualité Logiciel
  - Définir les mesures nécessaires et leur interprétation
  - Mener le suivi de la qualité
  - Indépendant du Chef de Projet
- Responsable Gestion de Configuration
  - Gérer les différentes versions, les noms, les anomalies
- Responsable d'Affaires
  - Chiffrer les évolutions envisagées
  - Négocier avec le client

Répartition des activités

Pourcentage pris par les différentes activités pour chaque phase

	Phase	Gestion projet	Gestion conf.	Assurance Qualité	Dvlpt
Organique	Spécification	13,5%	1%	2%	84,5%
	Conception préliminaire	11%	0,8%	1,7%	86,5%
	Réalisation	6,5%	2,2%	4,3%	87%
	Intégration	7,5%	2,7%	5,3%	84,5%

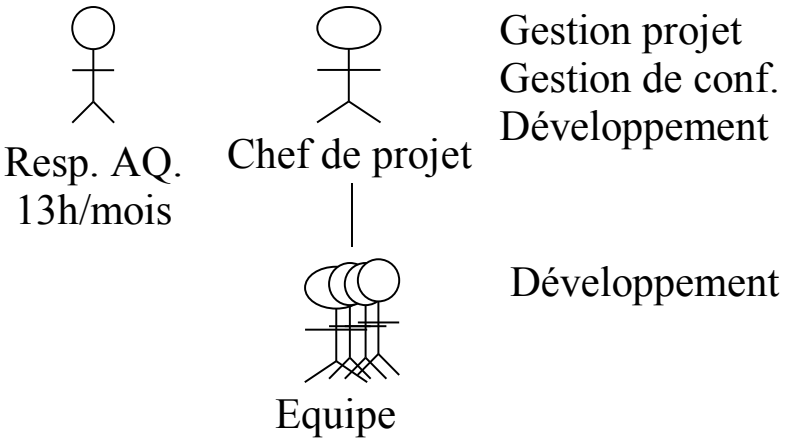
Répartition des activités sur l'exemple

- Rappel des effectifs pour chaque phase :
  - spécification : 4,29
  - Conception préliminaire : 7
  - Réalisation : 7
  - Intégration : 6,27
- Répartition :

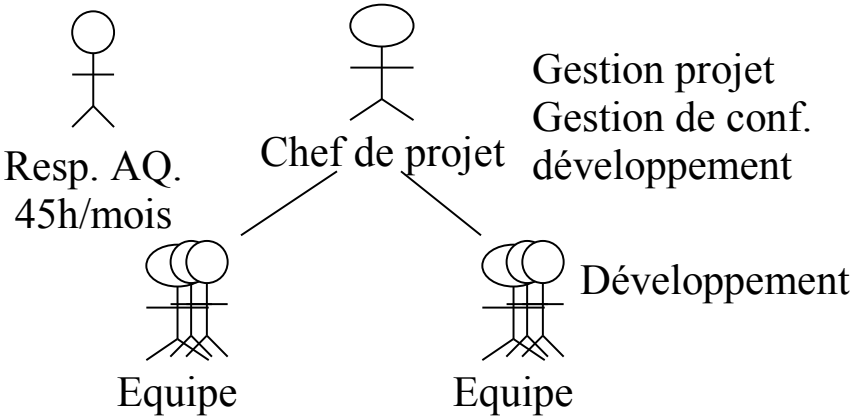
	Phase	Gestion projet	Gestion conf.	Assurance Qualité	Dvlpt
Organique	Spécification	0,57	0,04	0,09	3,62
	Conception préliminaire	0,77	0,06	0,12	6,05
	Réalisation	0,45	0,15	0,30	6,09
	Intégration	0,47	0,17	0,33	5,3

Constitution de l'équipe sur l'exemple

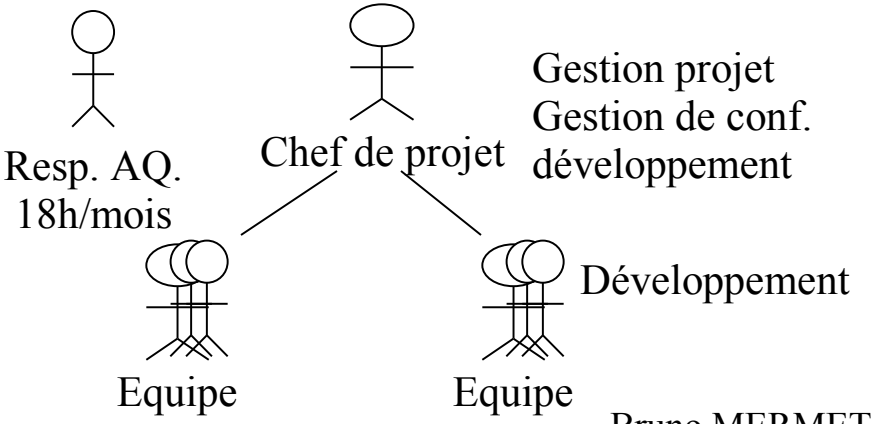
- Spécification



- Réalisation



- Conception préliminaire



- Intégration