

Conduite de projets

- Master 1 -

Mr. Abdellah Khababa & Mr. Youssef Boulkhiout
akhababa@univ-setif.dz & youssef.boulkhiout@univ-setif.dz

Département Informatique
Faculté des Sciences
Université Ferhat Abbas - Sétif 1

01 novembre2022

Table des matières

1	Problématique de la gestion de projet	9
1.1	Objectifs	9
1.2	Introduction	9
1.3	Définition d'un projet	9
1.4	Les caractéristiques d'un projet	10
1.5	Qu'est ce qu'un projet	11
1.6	Définitions normalisées	11
1.7	Gestion d'un projet	12
1.8	L'activité de gestion d'un projet	13
1.9	Définition normalisée de la gestion de projet	13
1.10	La gestion d'un projet système d'information	14
1.11	Les acteurs du projet	15
1.12	Maitrise d'ouvrage et système d'information	15
1.13	Conclusion	16
2	Découpage d'un projet et les modèles de développement	17
2.1	Objectif	17
2.2	Introduction	17
2.3	Les principes de découpage	17
2.3.1	Le découpage temporel	18
2.3.2	Le découpage structurel	18
2.4	Le cycle de vie standard	19
2.5	Le découpage classique	19
2.6	Les modèles de développement	20
2.6.1	Le modèle du code-and-fix	20
2.6.2	Le modèle de la transformation automatique	21
2.6.3	Le modèle de la cascade	21
2.6.4	Le modèle en V	22
2.6.5	Le modèle en spirale	22
2.7	Les modèles de cycle de vie spécifiques	24
2.7.1	Le modèle RUP	24
2.8	Conclusion	24
3	Gestion de projet	25
3.1	Objectif	25
3.2	Introduction	25

3.3	Activités de gestion	25
3.4	Planification du projet	26
3.4.1	Le plan du projet	26
3.4.2	Les jalons et les livrables	26
3.5	La programmation du projet	27
3.5.1	Le graphe des antécédents	27
3.5.2	Le diagramme fléché	27
3.5.3	Méthode du chemin critique	28
3.5.4	Les types de liens	28
3.6	Gestion des risques	29
3.7	Le processus de gestion des risque	29
3.8	Conclusion	30
1	La méthode de planning PERT	31
1.1	Généralités	31
1.2	Objectifs & Domaines d'application	31
1.3	Conditions de mise en œuvre	31
1.4	Définitions & Représentations	32
1.4.1	Tâche	32
1.4.2	Tâche fictive	32
1.4.3	Étape	32
1.4.4	Réseau	32
1.5	Conventions	33
1.6	Exercices	33
1.7	Rangs ou Niveaux	34
1.8	Calcul des dates et des marges	36
1.8.1	Calcul des dates au plus tôt	36
1.8.2	Calcul des dates au plus tard	36
1.8.3	Calculs des marges	36
1.9	Exploitation du réseau	37
1.10	Exemple de calcul	37
1.11	La méthode MPM	38
2	La méthode de planning PERT Probabiliste	39
2.1	Généralités	39
2.2	Calcul des paramètres	39
2.2.1	Exemple	40
2.2.2	Calcul des risques	40
2.3	Exercice «Recette»	40
2.4	Fonction de répartition de loi normale standard	42
3	La méthode de planning GANTT	43
3.1	Généralités	43
3.1.1	Utilisation de la matrice	44
3.2	Calcul des marges	45
3.2.1	Calcul des marges totales	45
3.2.2	Calcul des marges libres	45

TABLE DES MATIÈRES	5
3.3 Utilisation du tableau	46
3.4 Optimisation des ressources	48
3.4.1 Nivellement des ressources	50
3.4.2 Lissage des ressources	51
3.4.3 Comparaison entre nivellement et lissage	52
4 Gestion de la qualité	57
4.1 Objectif	57
4.2 Introduction	57
4.3 Processus basé sur la qualité	58
4.4 Assurance qualité et standards	58
4.5 Planification de la qualité	59
4.6 Contrôle de qualité	59
4.7 Mesures du logiciel	60
4.8 Processus de mesure	61
4.9 Métriques du logiciel	61
4.10 Conclusion	61
5 Inspection du logiciel	63
5.1 Objectif	63
5.2 Introduction	63
5.3 Avantages des inspections	63
5.4 Inspections	64
5.5 Inspection de logiciel	65
5.6 Le processus d'inspection	65
5.7 Inspection des documents	66
5.8 Conclusion	66
6 Estimation du coût du logiciel	69
6.1 Objectif	69
6.2 Introduction	69
6.3 Productivité du logiciel	69
6.4 L'estimation du coût d'un logiciel	70
6.4.1 Fondé sur des lignes de code	71
6.4.2 Le modèle COCOMO	71
6.4.3 Analyse des points de fonction	73
6.5 Analyse de la valeur acquise	73
6.6 Indicateurs d'avancement	74
6.7 Conclusion	75
7 Gestion des ressources humaines	77
7.1 Objectifs	77
7.2 Introduction	77
7.3 Sélection du personnel	78
7.4 Constitution de l'équipe	78
7.4.1 Définir les rôles et responsabilités	78
7.4.2 Déterminer la composition de l'équipe	79

7.5	Gestion du groupe	80
7.6	Motivation du personnel	80
7.7	Modèle de maturité des capacités du personnel	81
7.8	Conclusion	81
8	Gestion de la configuration	83
8.1	Objectifs	83
8.2	Introduction	83
8.3	Définitions	84
8.3.1	Les avantages de la gestion de configuration	84
8.3.2	Les définition des termes et concepts	84
8.4	Les fonctions de la gestion de configuration	85
8.4.1	Les fonctions liées à la mise en place et à la gestion du référentiel	85
8.4.2	Les fonctions liées à l'activité courante des développeurs	85
8.5	La gestion de configuration et le processus de développement	86
8.6	Rôle de la gestion de configuration dans le cycle de développement	87
8.7	Les outils de gestion de version et de gestion de configuration	88
8.8	Conclusion	89
9	Métriques du logiciel	91
9.1	Objectifs	91
9.2	Introduction	91
9.3	Définitions	91
9.4	Classification des métriques	92
9.5	Les métriques du produit	92
9.5.1	Les métriques orientées taille	92
9.5.2	Métriques de Mac Cabe	92
9.5.3	Métriques de Halstead	93
9.5.4	Métriques de Henry-Kafura	94
9.6	Les métriques Objet	95
9.7	Les métriques du processus	95
9.7.1	L'approche GQM	96
9.8	Conclusion	98

Introduction

- Plébiscitée par de plus en plus **d'entreprises**, la gestion de projet s'impose dans des structures de toutes tailles comme un mode d'**organisation** particulièrement efficace.
- La gestion de projets informatiques ou **conduite de projet** est une démarche visant à structurer, assurer et optimiser le bon déroulement d'un projet. Gérer et animer un projet par le chef de projet, c'est d'abord savoir en négocier l'objectif mais aussi prévoir.
- Pour cela il faut savoir mettre en œuvre les outils de l'analyse fonctionnelle, de planification (WBS, PBS, OBS, CBS, matrice RACI, **Pert**, **Gantt**,...), gérer un budget, maîtriser les risques, motiver et animer une équipe-projet tout cela en conciliant les intérêts du **maître d'ouvrage** et parties prenantes.
- Les **livrables** doivent être clairement définis pour un résultat conforme à des normes de qualité pour le **moindre coût** dans le **meilleur délai** possible.
- La gestion de projet devient un domaine de connaissances nouveau. Elle est devenue très complexe voire exigeante, comparativement il y a plus de deux décennies.
- Dans une large mesure, la réussite des projets d'aujourd'hui nécessite non seulement un développement accéléré de techniques et outils, mais également d'habiletés de gestion et de communication, afin de maîtriser cette nouvelle discipline. D'où le défi actuel de la profession de gestion de projet.

Chapitre 1

Problématique de la gestion de projet

1.1 Objectifs

L'objectif de ce chapitre est de fournir une introduction sur la gestion de projets informatiques à savoir :

- Définition du terme Projet.
- Définition de ce qu'est la gestion de projet.
- Gestion de projet système d'information.
- Rôles et responsabilités des principaux acteurs impliqués dans le développement d'un projet.

1.2 Introduction

- Les premières réflexions sur **la conduite de projets** (grands projets engagés dans les différents domaines industriels : aéronautiques, travaux publics, armement) datent des années 50.
- La conduite d'un projet correspond à une mise en œuvre d'une organisation méthodologique pour faire en sorte que **l'ouvrage** réalisé par **le maître d'œuvre** réponde aux attentes du **maître d'ouvrage** dans les contraintes de **délai, coût et qualité**.
- L'objectif est de développer des techniques et des méthodes pour augmenter la maîtrise des travaux et la coordination des différents corps métier. La formalisation mathématique des problèmes de gestion pour prendre des décisions optimales.
- Certaines offres variées de prestation comprenant la planification et la surveillance des délais et des coûts, la gestion de la qualité du projet, etc.... sont apparues.
- **Conclusion : Globalement, il y a absence de maîtrise de projets..**

1.3 Définition d'un projet

- Un projet est l'image plus ou moins précise d'un futur que l'on pense atteindre.
- Un projet est défini comme **un ensemble d'activités à effectuer pour atteindre un but défini de façon spécifique.**
- On le représente sous la forme d'un triangle :
- Donc, un projet consiste à vouloir réaliser une idée **ayant un caractère nouveau.**

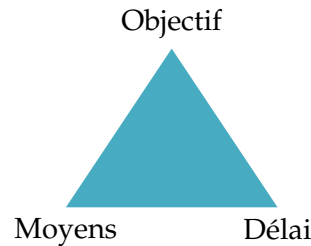


FIGURE 1.1 – Triangle de projet.

- Cette réalisation est **unique**, éphémère et il faut un certain temps pour la réaliser.
- Comment réaliser l'idée en projet ?
 1. Transformer l'idée en objectifs.
 - **De techniques** : ce qu'on veut faire.
 - **De délai** : en combien de temps.
 - **De coût** : avec quel budget.
 2. Définir les moyens nécessaires.
 3. Prévoir l'organisation et la gestion du projet.

1.4 Les caractéristiques d'un projet

- Le projet a des **enjeux importants** : c'est le caractère stratégique d'un projet
- Le **caractère novateur** : la démarche projet repose sur la créativité
- Un cycle de **vie borné** : un début et une fin s'imposent
- La **multiplicité des intervenants** : coopération, coordination et qualités relationnelles sont indispensables
- **Caractère aléatoire** : existence d'éléments non maîtrisables, liés aux facteurs humains et à la créativité du produit.
- **Plusieurs disciplines en cause et intérêts divergents (transversalité)** : le projet est développé par plusieurs entreprises et/ou par plusieurs services d'une même entreprise.

Les caractéristiques spécifiques aux projets informatiques

- L'objet informatique produit est immatériel.
- Finalité nouvelle est unique : mélange de «déjà fait» et «jamais fait».
- Incertitude dans la réussite et dans les choix techniques, **remise en cause des techniques, des délais et des coûts envisagés** :
 - Modifications, causes de nombreux feedback.
 - Pour chaque partie d'un projet, les feedback entre études et réalisations sont inévitables.
 - Faisabilité, définition, conception, réalisation des différentes parties d'un projet s'imbriquent et se conditionnent du début à la fin.

1.5 Qu'est ce qu'un projet

- Le terme «Projet» fait référence à une activité unique et contrainte, réalisée pour atteindre un objectif.
- Un objectif est toujours SMART

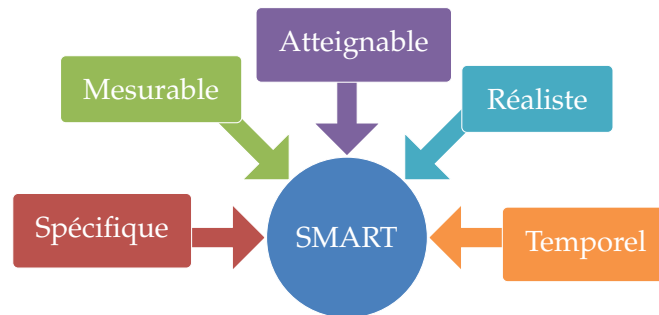


FIGURE 1.2 – SMART.

1. **Spécifique (specific)** : l'objectif doit être en lien direct avec le travail de la personne chargée de réaliser l'objectif, il doit être personnalisé. Par ailleurs, un objectif doit être simple à comprendre, clair, précis et compréhensible par la personne pour que soit efficace car la complexité ralentit l'action. De plus, il doit être compréhensible également par tous pour que l'objectif ait une légitimité aux yeux de tous.
2. **Mesurable (Measurable)** : un objectif mesurable doit être quantifié et qualifié. Pour réaliser un objectif, la définition d'un seuil est nécessaire afin de savoir quel le niveau à atteindre, la valeur de la mesure à atteindre ou à rencontrer. Il n'est pas possible de choisir un objectif que l'on ne peut quantifier ou qualifier par souci d'évaluation des moyens nécessaires pour l'atteindre.
3. **Acceptable ou atteignable (Achievable)** : il doit être partagé par les participants et orienté action, on dit parfois également Acceptable, atteignable et ambitieux, un objectif atteignable est un objectif suffisamment grand, ambitieux pour qu'il représente un défi et qu'il soit motivant. Par ailleurs cet objectif doit être atteignable et donc raisonnable favorisant ainsi l'adhésion des participants à ce dernier. Ainsi, l'objectif sera plus facilement accepté par chacun des acteurs.
4. **Réaliste (Realistic)** : il est réaliste pour lequel le seuil de réalisme est défini. C-à-d. un niveau pour lequel le défi motivera un grand nombre de participants et évitera au mieux l'abandon de certains participants au fur et à mesure de la progression de l'objectif.
5. **Temporel (Time-bound)** : un objectif temporellement défini est délimité dans le temps. L'objectif doit être clairement défini.

1.6 Définitions normalisées

- **Définition** : Un projet est un ensemble d'activités organisées en phases ou étapes et formant l'unité de gestion permettant la réalisation d'un objectif défini et précis.
- Un projet est unique, il a une durée limitée et démarche (un début et une fin), comporte plusieurs activités distinctes et se termine par la mise en place d'un procédé (un système dans le domaine informatique) voire une annulation.

- **Selon ISO 10006-2003** : Un projet est un processus unique, qui consiste en un ensemble d'activités coordonnées et maîtrisées comportant des dates de début et de fin, entrepris dans le but d'atteindre un objectif conforme à des exigences spécifiques telles que les contraintes de délais, de coûts et de ressources.
- **Le référentiel du PMI** (Project Management Institute), appelé Guide du PMBOK (Project Management Body of Knowledge) donne la définition suivante : Un projet est une entreprise temporaire décidé pour obtenir un produit ou un service unique.
- **AFITEP et AFNOR** : définissent un projet comme un ensemble d'actions à réaliser pour satisfaire un objectif défini, dans le cadre d'une mission précise et pour la réalisation desquelles on a identifié non seulement un début, mais aussi une fin.

Exemples :

1. Introduction d'un logiciel ERP (entreprise ressources planning) ou PGI (progiciel de gestion intégré)
2. Acquisition et mise en place d'un LAN dans une entreprise
3. Développement d'un logiciel spécifique,
4. Upgrade d'un parc de PCs à une nouvelle version de système d'exploitation (améliorer ou augmenter en puissance)

1.7 Gestion d'un projet

- La gestion d'un projet a pour but de mener un projet à son terme en organisant et en surveillant son déroulement.
- Les trois aspects représentés par le triangle Projet doivent être mis sous contrôle. Chacun fait l'objet d'une gestion spécifique et qui prend en compte l'existence des deux autres.
- Enjeux : **Améliorer la qualité, diminuer les coûts et maîtriser les délais**. D'où le triangle gestion de projet suivant :

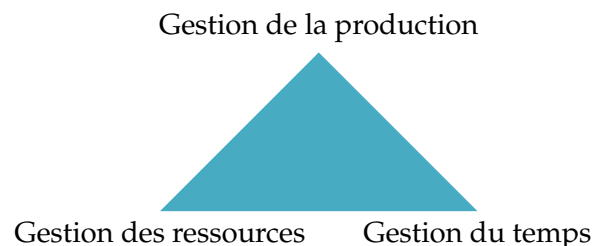


FIGURE 1.3 – Le triangle de gestion de projet.

- **Le délai donne lieu à la gestion du temps dont le rôle est définir le parcours et de le jalonner, d'établir des calendriers et de maîtriser la consommation de l'enveloppe temps.**
- Les moyens affectés constituent le budget du projet qui est transformé en travail, locaux, matériel, temps machine, déplacement, etc.
- Cette transformation nécessite une gestion des ressources portant sur les ressources humaines et les moyens matériels.

- L'objectif du projet doit à son terme être concrétisé par une ou plusieurs fournitures. Ce sommet donne naissance à la gestion de la production, qui a pour but de suivre et diriger l'avancement vers l'objectif tout au long du projet.

1.8 L'activité de gestion d'un projet

L'activité de gestion de projet peut être décomposée en trois activités principales autour de la production proprement dite :

1. Analyser :

- Déterminer le chemin que l'on va emprunter pour avancer vers l'objectif.
- Étudier les caractéristiques du projet, son contexte, les risques qui les menacent et l'état de son avancement. D'où un découpage du projet en activités à entreprendre et à une estimation de l'effort est nécessaire.

2. Organiser :

- Repérer les contraintes d'enchaînement entre les tâches afin de les ordonnancer (établir un calendrier).
- L'organisation recouvre la constitution d'une équipe et la prise en compte des relations avec tous les partenaires.

3. Piloter :

- Suivre l'avancement du projet, en quantité et en qualité ainsi que l'analyse et le traitement de l'écart avec ce qui est prévu, les orientations et les décisions à prendre ou à faire prendre.
- Inclure la gestion de l'équipe et la gestion des conflits.

Un projet est managé du début jusqu'à la fin ce qui explique la boucle de rétroaction (Voir la figure ci-dessous)

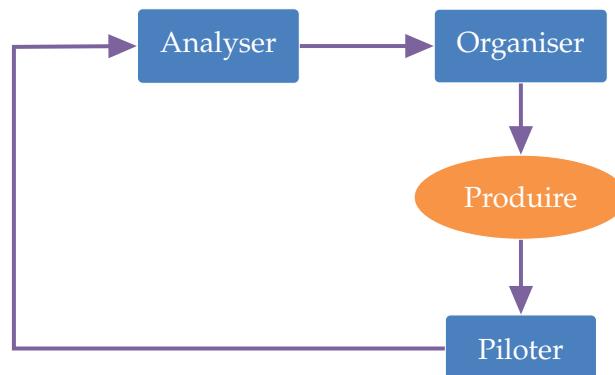


FIGURE 1.4 – La boucle de rétroaction.

1.9 Définition normalisée de la gestion de projet

- Le référentiel de l'IPMA (International Project Management Association, basée en Suisse, est une fédération d'associations locales, dont par exemple l'AFITEP est le représentant pour la France, la GPM pour l'Allemagne, ...) donne la définition de l'ensemble des activités de gestion de projet.

- La gestion de projet consiste à planifier, organiser, suivre et maîtriser tous les aspects du projet ainsi que la motivation de tous ceux qui sont impliqués dans le projet, de façon à atteindre les objectifs de façon sûre et dans les critères définis de coûts, délais et performance.
- La gestion est une responsabilité du chef de projet et se compose de quatre activités, pouvant correspondre à une fonction :
 1. **La direction de projet** : est exercée par le chef de projet et sert à :
 - Fixer les objectifs, la stratégie et les moyens
 - Coordonner les actions successives.
 - Maîtriser, modifier l'itinéraire et l'horaire si un des objectifs évolue.
 - Optimiser la répartition des ressources en vue d'arriver à une solution optimale.
 2. **La gestion de projet** : est effectuée afin d'accomplir la direction du projet (analyser les risques, estimer la charge, organiser le travail, le planifier et le suivre).
 3. **L'activité de maîtrise** : correspond à des tâches qui ont été extraites de la gestion de projet et qui sont centrées sur une préoccupation (la maîtrise de la qualité, par exemple).
 4. **L'activité de pilotage** : est une activité périodique d'orientation du projet faite par le comité de pilotage (si elle est isolée de la fonction de direction de projet).
- Enfin, la gestion d'un projet est un processus difficile à maîtriser à cause des facteurs (de risque) suivants :
 - coûts et délais à respecter .
 - technologies à maîtriser.
 - ressources humaines à gérer.

1.10 La gestion d'un projet système d'information

- **Définition** : un système d'information est un ensemble organisé de ressources : matériel, logiciel ; personnel, données, procédures... permettant d'acquérir, de traiter, stocker, communiquer des informations (sous formes données, textes, images, sons..) dans des organisations.
- **Caractéristiques** : Le triplet (objectif, délai, moyens) dans le domaine des systèmes d'information présente trois caractéristiques :
 - Il y a interaction entre l'objectif d'une part et les moyens/délais d'autre part.
 - L'objectif d'un projet n'est parfaitement défini qu'à l'achèvement du projet.
 - Le développement d'un système d'information se déroule dans une organisation.
- Objectifs des projets systèmes d'information :
 1. **Productivité administrative** : possible grâce à l'automatisation d'une partie des tâches qui contribue à la diminution de la main d'œuvre.
 2. **Aide au management** : en bâtissant une mémoire de l'organisation et de ce qui l'entoure à partir de laquelle on pourra construire des tableaux de bord, faire des analyses, etc.
 3. **Efficacité opérationnelle** : le meilleur fonctionnement est obtenu par l'usage créatif des technologies de l'information et de la communication.
 4. **Evolutivité** : un système flexible peut être modifié en cas d'évolution des contraintes et/ou de stratégies.
 5. **Utilisation d'une nouvelle technologie** : pour obtenir un «effet vitrine» vis-à-vis de l'extérieur. Un délai court est un élément essentiel de la réussite du projet.

1.11 Les acteurs du projet

1. **La maîtrise d'ouvrage ou le maître d'ouvrage (MOA) :** est le donneur d'ordre au profit duquel l'application est conçue. Ses rôles sont :
 - Décrire les besoins et définir le cahier des charges.
 - Établir le financement et le planning général des projets.
 - Fournir les spécifications fonctionnelles générales et valider la recette.
 - Assurer la responsabilité de pilotage du projet dans ses grandes lignes.
 - Adapter le périmètre fonctionnel en cas de retard afin de respecter la date de la livraison.
2. **La maîtrise d'œuvre ou le maître d'œuvre (MOE) :** répond au programme fonctionnel déterminé par la maîtrise d'ouvrage en proposant une solution qui permette la réalisation de ce programme tout en respectant les contraintes préétablies (moyens, budget, planning, ...). Ses rôles sont :
 - Conseiller la maîtrise d'ouvrage.
 - Participer à la conception de l'application.
 - Garantir la bonne réalisation technique de la solution proposée.
 - Vérifier la qualité de la réalisation (recette).

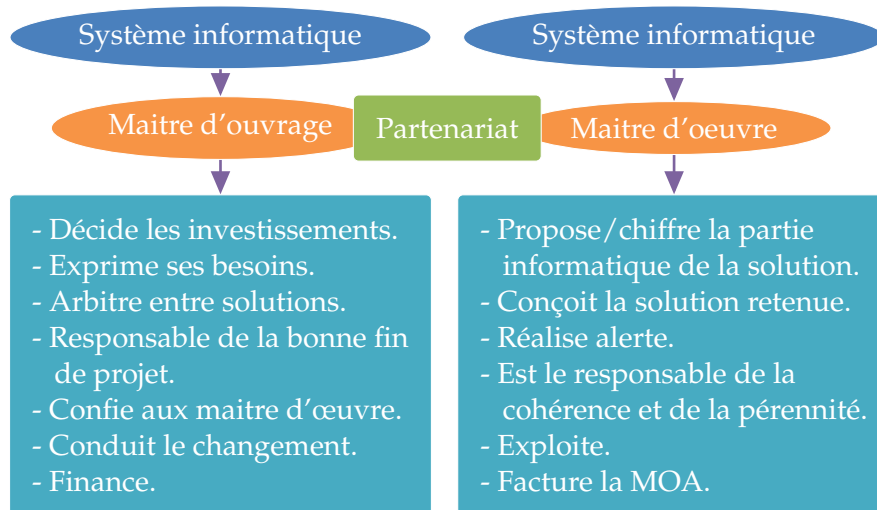


FIGURE 1.5 – Les acteurs du projet.

1.12 Maitrise d'ouvrage et système d'information

Maitrise d'ouvrage

- Propriétaire du système construit ou corrigé par le projet et est responsable de son financement.
- Assume l'entière responsabilité du fonctionnement futur de ce système.
- Délimite le champ du projet, et spécifie les résultats à produire.
- Définit les objectifs assignés au MOE.
- Précise éventuellement les contraintes réglementaires calendaires et budgétaires.
- Valide les propositions faites par la MOE.
- Préside le comité de pilotage s'il existe.

- Fait appel à une ou plusieurs cellules d'assistance :
 - Dans ses relations avec la MOE aux plans de l'expertise technique et l'expertise fonctionnelle.
 - Dans ses relations avec les directions de l'entreprise concernées par ce projet.
 - Dans ses relations avec les partenaires extérieurs (fournisseurs, sous-traitants, organisme de financement. .).

Maitre d'œuvre

- Elle a la responsabilité de construire un système répondant aux besoins des utilisateurs avec le souci :
 - De respecter les contraintes fixées par le MOA (délais, budget, qualité. .)
 - D'assurer la cohérence de ce système avec l'ensemble du SI de l'entreprise.
 - De respecter les règles d'ingénierie de l'entreprise.
- Elle définit et choisit les méthodes et moyens nécessaires.

1.13 Conclusion

Un projet regroupe de **nombreuses activités**. Il doit concilier :

- **Les objectifs fonctionnels**.
- Les **spécifications** (Aspects techniques).
- Les **contraintes temporelles**.
- Les **contraintes budgétaires**.
- Les **contraintes matérielles** (Ressources allouées).

Les principaux acteurs impliqués dans le développement d'un projet sont **le maitre d'œuvre** et le **maitre d'ouvrage** dont chacun un rôle spécifique.

Chapitre 2

Découpage d'un projet et les modèles de développement

2.1 Objectif

L'objectif de ce chapitre est d'introduire la notion de découpage d'un projet logiciel et par conséquent de processus logiciel.

Ce chapitre permet de :

- Comprendre le principe de découpage d'un projet logiciel.
- Expliciter les différents modèles de processus.
- Expliciter le processus RUP, un processus moderne et générique.

2.2 Introduction

- Un processus logiciel est un ensemble d'activités qui conduit à la réalisation du produit logiciel. Les processus logiciels sont complexes et nécessitent certaines décisions prises par les différents intervenants.
- Bien qu'il existe plusieurs modèles de processus logiciel, certaines activités fondamentales sont communes à tous les processus logiciels telles que :
 1. la spécification du logiciel,
 2. la conception et l'implémentation du logiciel,
 3. la validation du logiciel,
 4. l'évolution du logiciel.

2.3 Les principes de découpage

- La tâche du chef de projet est de découper le projet afin de pouvoir répartir dans le temps la production et les ressources nécessaires.
- Découper un projet consiste à identifier des sous-ensembles quasi autonomes, ayant les caractéristiques suivantes :
 - Chaque sous ensemble donne lieu à un résultat bien identifié
 - La charge propre à chacun peut être évaluée.

- Les contraintes d'enchaînement entre les sous ensembles sont repérables : certains sous ensembles peuvent être réalisés parallèlement, d'autres sont liés par des contraintes d'antériorité.
- Le découpage est fait à des mailles différentes, un sous ensemble étant souvent à son tour décomposé.

On utilise deux grands critères pour découper un projet :

2.3.1 Le découpage temporel

Le critère temporel est utilisé dans la plupart des projets répartir le travail dans le temps. Il se base sur les modèles de développement ou modèle de cycle de vie.

- Un projet se compose de phases.
- Chaque phase comprend un certain nombre d'activités (sous phase).
- Une activité est définie par une ou plusieurs tâches à effectuer. A chaque élément de décomposition, on attache un résultat à atteindre appelé livrable.
- Les principaux livrables à fournir sont : le cahier de charges, le plan de projet, les spécifications détaillées, l'architecture fonctionnelle et technique, planification, le plan assurance qualité, la gestion des risques, le plan de test, le logiciel, manuel d'utilisateur, etc.
- L'ensemble ordonnancé des phases d'un projet s'appelle le cycle de vie du projet.

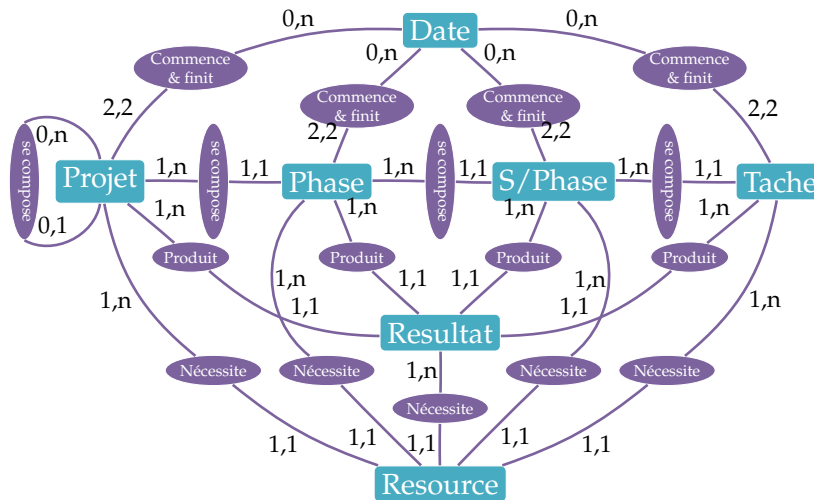


FIGURE 2.1 – La modélisation de découpage.

2.3.2 Le découpage structurel

Le critère structurel permet d'organiser le travail en se basant sur la structure du produit final : la décomposition fait apparaître les différents modules qu'il faut obtenir. L'utilisation de ce critère **nécessite une visibilité suffisante** sur le résultat à produire. En plus du découpage temporel et surtout si le projet est de taille importante, on recourt au découpage structurel pour les raisons suivantes :

- **Maîtriser le projet** : les sous ensembles sont de taille plus réduite et plus facile à maîtriser.
- **Répartir les responsabilités** : l'autonomie des modules permet de confier différents responsables pour différents modules.

- **Réduire les délais planifiés** : certains modules indépendants sont développés en parallèle ce qui permet d'avancer la date théorique d'achèvement du projet.
- **Avoir un développement incrémental** : pour différentes raisons, on choisit de développer un système d'information en versions successives dont chaque version comporte un nombre croissant de modules par rapport à la précédente.

2.4 Le cycle de vie standard

Le cycle standard se compose des phases suivantes :

1. **Etude de faisabilité** : comprend les travaux d'analyse, des travaux de recherche, des études sur terrain où il s'agit de vérifier si le projet est techniquement réalisable.
2. **Définition des solutions** : donne une représentation précise de l'objectif à atteindre. Les solutions possibles sont étudiées de façon détaillée. A terme de cette phase, une solution est choisie et l'on dispose de spécifications exactes.
3. **Conception détaillée** : sert à préparer les contrats de réalisation. Ces contrats contiennent le cahier des charges pour les sous-traitants.
4. **Réalisation** : est l'exécution des contrats par les sous-traitants, conformément aux cahiers des charges. Cette phase se termine par une procédure d'acceptation officielle.

Problèmes posés par le découpage standard :

- La **notion de cahier des charges est déclinée à plusieurs moments**. La plupart des phases du cycle de développement peuvent conduire à un cahier des charges qui oriente le travail de l'étape ultérieure.
- **Le découpage temporel standard suppose que le client possède une description complète de ce qu'il attend**. Or la détermination des besoins et des solutions adéquates est un problème majeur.
- L'élaboration d'un cahier des charges est un travail **coûteux**.
- L'écriture des spécifications souffre de l'absence de composants réutilisables.

2.5 Le découpage classique

Les méthodes de développement des systèmes d'information ont proposé un découpage temporel de référence (**cycle de vie classique**) qui correspond au vocabulaire de la méthode Merise :



FIGURE 2.2 – Le découpage classique.

1. **Le schéma directeur (SD)** : sert à définir les scénarios d'évolution du patrimoine informatique sous l'un ou l'autre de ces trois **angles** :
 - Evolution de l'**architecture technique** (matériels, réseaux)
 - Evolution de l'**architecture applicative** (données commune, identification des domaines, etc.)
 - Evolution **des fonctions informatiques** (méthodes, normes, outils)
2. **L'étude préalable (EP)** : sert à repenser une application vieillissante sur un domaine bien identifié ou pour répondre à un nouveau besoin. L'objectif est double :

- Les choix structuraux pour la future application à savoir évaluer l'adéquation de la solution aux objectifs, évaluer l'investissement (budget, temps), ajuster la solution à l'enveloppe si cela est nécessaire.
 - Fournir une base de référence pour la suite du projet. Le rapport de l'étude préalable est considéré comme un cahier des charges pour l'étude détaillée.
3. **L'étude détaillée (ED)** : sert à concevoir et à décrire de façon exhaustive la solution qui sera ensuite complétée par l'étude technique. Le résultat comprend toute la vision externe du système.
 4. **L'étude technique (ET)** : concerne uniquement les informaticiens et consiste à optimiser les structures physiques de données et de construire les traitements en essayant de préparer la réutilisation du code.
 5. **La réalisation (REAL)** : l'objectif est de produire un logiciel testé. Elle se termine par une procédure d'acceptation officielle appelée recette.
 6. **La mise en œuvre (MEO)** : consiste à préparer le démarrage effectif de la nouvelle application.
 7. **La qualification (QUALIF)** : l'objectif est de réaliser des tests dans l'environnement opérationnel et de tirer un bilan du système d'information installé.

2.6 Les modèles de développement

- Il n'existe pas une démarche unique mais on peut construire le découpage temporel en fonction des caractéristiques de l'entreprise et du projet.
- Les découpages temporels génériques, appelés les modèles de développement (process models) ou modèles de cycle de vie sont les suivants :

2.6.1 Le modèle du code-and-fix

Après une étape brève de compréhension de l'objectif, l'application est développée. Plusieurs cycles de mise au point, permettent d'atteindre le résultat visé, une collaboration avec l'utilisateur du futur système.

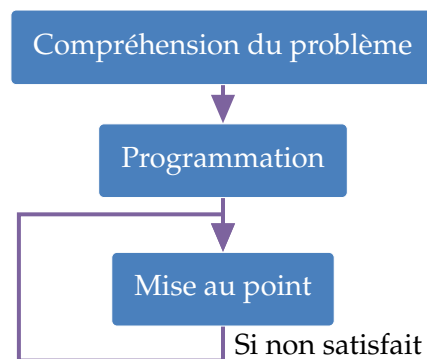


FIGURE 2.3 – Le modèle du code-and-fix.

2.6.2 Le modèle de la transformation automatique

Il est basé sur la possibilité de transformer automatiquement des spécifications en programmes. Cela suppose que les spécifications sont complètement validées. Une succession de cycles de spécification/validation s'achève par la génération de code.

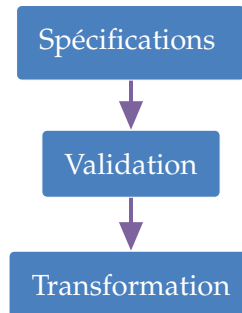


FIGURE 2.4 – Le modèle de la transformation automatique.

2.6.3 Le modèle de la cascade

- L'objectif de ce modèle est de jalonner rigoureusement le développement et de définir de manière précise les rôles respectifs du fournisseur (qui produit le livrable) et du client (qui accepte ou refuse le résultat).
- C'est une succession de phases qui correspond à une approche descendante.
- Chaque phase donne lieu à une validation officielle : on ne passe à la suivante que si le résultat du contrôle est satisfaisant. Sinon, on modifie le livrable pour qu'il devienne acceptable.

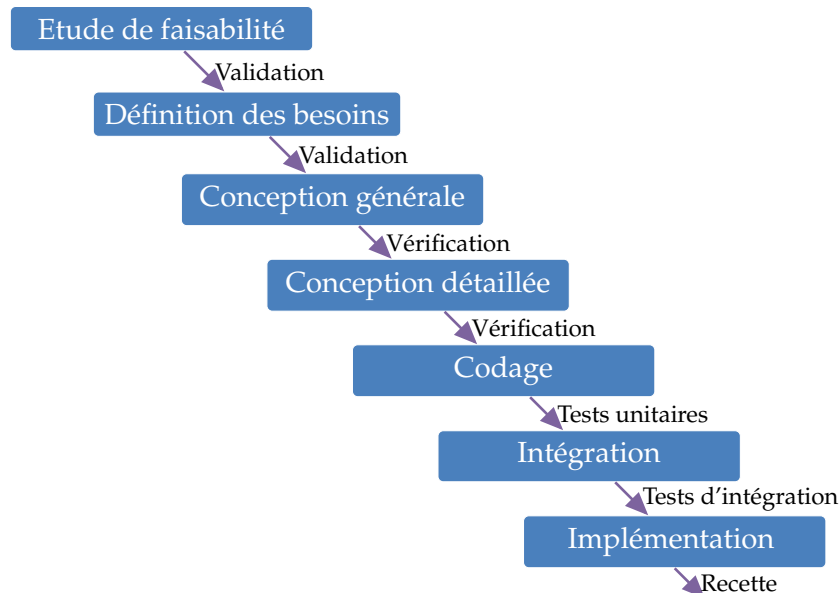


FIGURE 2.5 – La modélisation de découpage.

2.6.4 Le modèle en V

Le modèle en V est une amélioration du modèle de la cascade. Chaque phase de la première branche du V va être explicitée par des critères d'appréciation et d'acceptation du système aux étapes correspondantes de la deuxième branche du V.

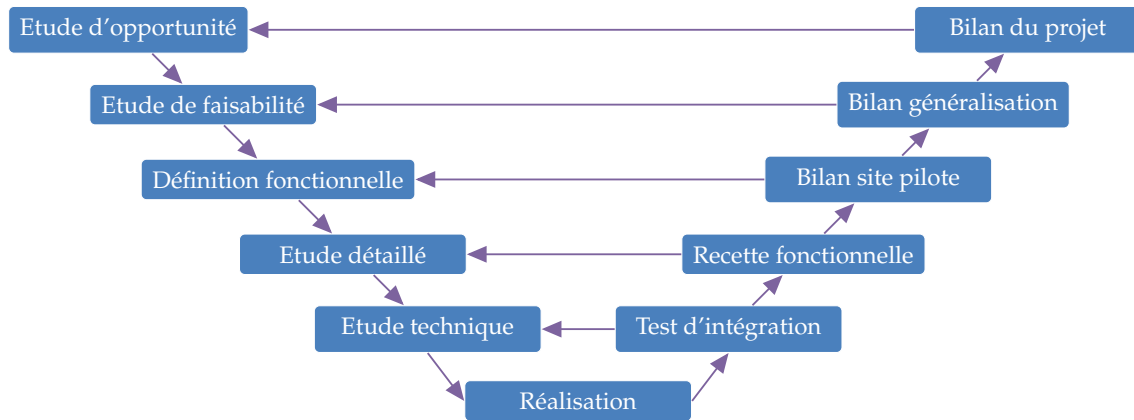


FIGURE 2.6 – Le modèle en V.

2.6.5 Le modèle en spirale

Le modèle en spirale est proposé par B.Boehm en 1988. C'est un modèle qui repose sur le fait qu'il y a une relation contractuelle entre le fournisseur et le client. Plusieurs cycles sont effectués. Chaque cycle donne lieu à une contractualisation s'appuyant sur les besoins exprimés lors du cycle précédent. Donc il sert à :

- Détermination des objectifs du cycle, des alternatives pour les atteindre et des contraintes ; à partir des résultats des cycles précédents , ou de l'analyse préliminaire des besoins ;
- Analyse des risques, évaluation des alternatives à partir de maquettage et/ou prototypage ;
- Développement et vérification de la solution retenue, un modèle classique (cascade ou en V) peut être utilisé ici ;
- Revue des résultats et vérification du cycle suivant.

Chaque cycle de la spirale est comporte les étapes suivantes :

- Analyse du risque.
- Développement d'un prototype.
- Simulation et essais du prototype.
- Détermination des besoins, à partir des résultats des essais.
- Validation des besoins par un comité de pilotage.
- Planification du cycle suivant.
- Le dernier cycle comprend :
 - en phase 2 développement de la version finale
 - en phase 3 tests et installation
 - et s'arrête là.

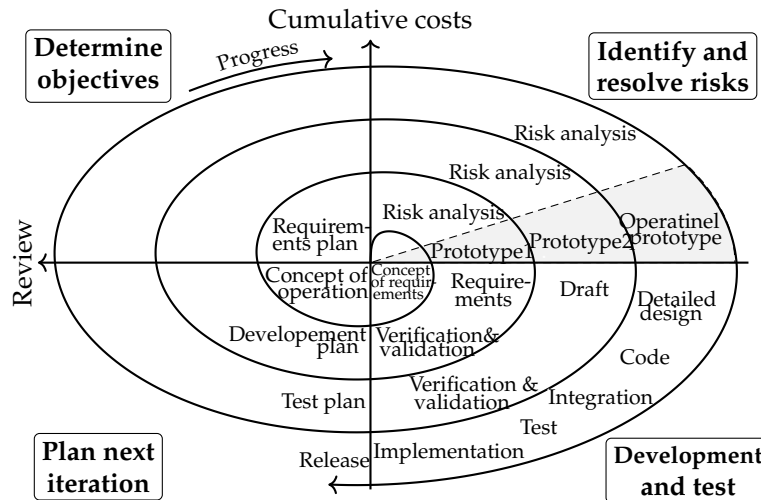


FIGURE 2.7 – Le modèle en spirale.

Analyse des risques

Le modèle en spirale tient compte de la possibilité de réévaluer les risques en cours de développement, la démarche est la suivante :

- Identifier les risques, leur affecter une priorité.
- Développer une série de prototypes pour identifier les risques.
- Utiliser un modèle en V/cascade pour implémenter chaque cycle.
- Si un cycle concernant un risque a été achevé avec succès, évaluer le résultat du cycle et planifier le cycle suivant. Si un risque n'a pu être résolu, terminer le projet immédiatement.

La mise en œuvre demande des compétences managériales et devrait être limitée aux projets innovants à cause de l'importance que ce modèle accorde à l'analyse des risques. Citons, par exemple :

1. **Risques humains** :
 - Défaillance du personnel ; surestimation des compétences ;
 - Travailleur solitaire, héroïsme, manque de motivation.
2. **Risques technologiques** :
 - Produit miracle, "plaqué or" ;
 - Changement de technologie en cours de route ;
 - Problèmes de performance .
 - Exigences démesurées par rapport à la technologie.
 - Incompréhension des fondements de la technologie.
3. **Risques processus** :
 - Pas de gestion de projet
 - Calendrier et budget irréalistes ;
 - Calendrier abandonné sous la pression des clients ;
 - Composants externes manquants ;
 - Tâches externes défaillantes ;
 - Insuffisance de données.
 - Validité des besoins ;
 - Développement de fonctions/interfaces utilisateurs inappropriées ;

2.7 Les modèles de cycle de vie spécifiques

2.7.1 Le modèle RUP

Le modèle RUP (Rational Unified Process) est représentatif d'une approche combinant plusieurs modèles. Sa structure fait l'objet d'un assez large accord, notamment parmi les praticiens. Il peut être lu de la façon suivante :

- Le cycle est constitué de quatre phases principales : étude préalable (opportunité), conception de la solution détaillée (élaboration), développement de la solution (construction) et mise en œuvre (transition).
- Il existe six types de tâches qui se retrouvent à des degrés différents dans chacune des phases. Par exemple, l'étude des besoins peut apparaître jusqu'à la fin du projet mais la plus grande partie est effectuée dans les deux premières phases.
- Certaines phases peuvent être menées de façon cyclique. L'élaboration se fait en deux cycles, conduisant à la production des spécifications externes (vision utilisateur) et des spécifications techniques (vision développeur).
- La construction s'effectue en spirale.

Les deux visions rassemblées (vision utilisateur et vision développeur) - Le Modèle Itératif :

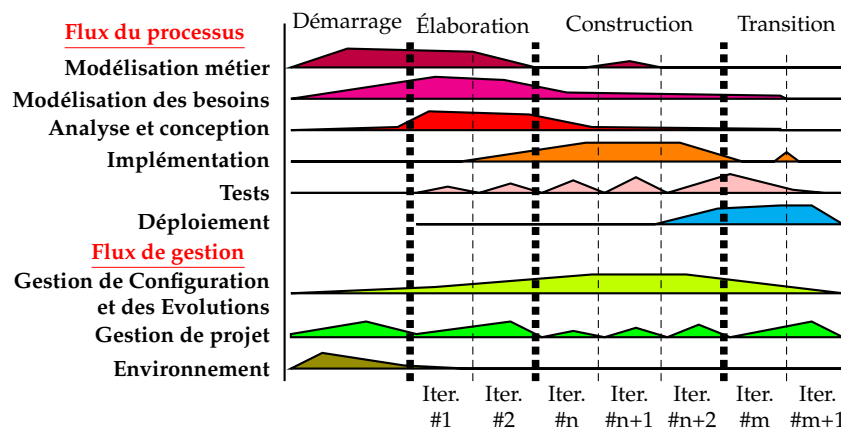


FIGURE 2.8 – Le modèle RUP.

2.8 Conclusion

- Les modèles de processus logiciel sont des représentations abstraites des processus logiciel. Ils visent à ordonner et à structurer le développement du logiciel. Tous les processus logiciel incluent certaines activités communes. Les modèles de processus itératifs présentent le processus logiciel comme étant un cycle d'activités.
- Le choix d'un modèle de développement s'appuie sur l'analyse des caractéristiques et surtout sur l'analyse des risques.
- Il existe aussi d'autres modèles tels que **Extreme Programming**, développement orienté service, etc. qui ne sont pas traités dans ce chapitre

Chapitre 3

Gestion de projet

3.1 Objectif

L'objectif de ce chapitre est de fournir une vue générale sur la gestion de projet. Cela se traduit par :

- Déterminer les principales tâches du chef de projet
- Comprendre le besoin d'une planification pour tout projet logiciel
- Définir les deux représentations graphiques qui correspondent à la programmation d'un projet
- Introduire la notion de gestion des risques et certains types de risques qui peuvent affecter des projets logiciels.

3.2 Introduction

- La gestion d'un projet logiciel est une partie essentielle du génie logiciel. Une mauvaise gestion conduit à un échec de réalisation du logiciel : non satisfaction des besoins des usagers, dépassement des coûts et des délais prévus.
- Le chef du projet est responsable de la planification et de la programmation du développement du projet.

3.3 Activités de gestion

Les activités de gestion suivies par la plupart des chefs de projets sont :

1. **Écriture de la proposition du projet** : la proposition décrit les objectifs du projet et comment y parvenir (les grandes lignes de sa réalisation). Une proposition doit aussi contenir une évaluation des risques et des coûts. Elle peut servir aussi d'argumentaire pour justifier la mise en route du projet. De plus, elle nécessite de l'expérience et de la compréhension du domaine d'activité.
2. **Planification et programmation du projet** : consiste à identifier les activités, les jalons et les livrables produits par le projet. Il s'agit d'un travail d'ordonnancement qui nécessite des connaissances très précises du domaine, des équipes de développement, etc.
3. **Évaluation des coûts du projet** : consiste à estimer les ressources nécessaires pour accomplir le plan du projet.
4. **Contrôle et révision du projet** : c'est une activité continue qui consiste à observer la progression du projet et à le comparer avec le progrès planifié. Cela peut avoir lieu à l'aide d'un contrôle informelle (discussion entre équipe) ou durant les revues formelles qui sont effectuées.

5. Sélection et évaluation du personnel : consiste à choisir un personnel compétent

6. Écriture du rapport et des présentations : consiste à établir un rapport sur le projet pour le client et pour l'organisation contractuelle. C'est un document concis, cohérent qui est présenté lors des contrôles de progression.

Par conséquent, le chef de projet doit pouvoir communiquer une vue synthétique du projet à différents publics (autres chefs de projet, clients, responsables, etc.)

3.4 Planification du projet

- Une gestion effective d'un projet dépend de la planification du progrès du projet.
- Le plan initialement établi doit être le meilleur possible et il doit évoluer au fur et à mesure que le projet avance.
- Au début, le plan envisagé doit établir les estimations concernant certaines contraintes affectant le projet (date de livraison, équipe disponible, budget).
- De plus, certains autres paramètres sont aussi à prendre en considération (la taille, la structure et la répartition des fonctions) pour enfin définir les **jalons** et les **livrables**.

3.4.1 Le plan du projet

Dans la majorité des cas, le plan comprend les sections suivantes :

- **Introduction** : description des objectifs du projet et des contraintes (délai, budget)
- **Organisation du projet** : description de l'organisation de l'équipe, le personnel impliqué et le rôle de chacun joué dans l'équipe.
- **Analyse des risques** : description des risques possibles du projet, leur probabilité d'apparition ainsi que les stratégies proposées pour les réduire.
- **Besoins des ressources en matériel et logiciel** : description du matériel et du logiciel de support pour le développement du projet.
- **Découpage du projet** : découpage du projet en activités en identifiant les jalons et les livrables.
- **Programmation du projet** : description de la dépendance entre les activités, le temps estimé pour déterminer chaque jalon et l'affectation du personnel dans chaque activité.
- **Mécanisme de contrôle et les rapports** : définition des rapports de gestion établis, quand ils sont établis, et le mécanisme de contrôle utilisé.

3.4.2 Les jalons et les livrables

Dans la majorité des cas, le plan comprend les sections suivantes :

- **Le logiciel** : est un produit intangible ce qui rend impossible d'estimer l'avancée du projet, le coût du projet ainsi que la mise à jour de la programmation du projet d'où la nécessité d'établir une série de jalons.
- **Un jalon (milestones)** : est une marque de fin reconnue d'une activité du processus logiciel. Pour chaque jalon, il faut établir un sortie formelle (rapport) qui n'est pas nécessairement un document important. Il peut correspondre à un court rapport synthétisant ce qui a été fait.
- **Un livrable** : est un résultat du projet qui est délivré au client. Il est livré à la fin d'une phase donnée du projet telle que la phase de spécification ou de conception. Il est mesurable, tangible et/ou vérifiable.

Par conséquent, les livrables sont des jalons mais les jalons ne sont pas nécessairement des livrables.

3.5 La programmation du projet

- Elle représente une tâche difficile du chef de projet où le délai et les ressources nécessaires sont estimés afin de compléter les activités, prévues et de les organiser en une séquence cohérente.
- Le programme sera défini et représenté sous la forme d'un diagramme.
- Deux techniques complémentaires sont utilisées :
 1. La technique des graphes : deux méthodes existent :
 - (a) la méthode des antécédents.
 - (b) la méthode du diagramme fléché.
 2. La technique de Gantt.

3.5.1 Le graphe des antécédents

- L'établissement de ce graphe correspond à la méthode du Chemin critique ou encore CPM (Critical Path Method). Ce graphe ou réseau est aussi appelé PDM (Precedence Diagramming Method).
- Chaque activité y est représentée par une boîte. Les activités sont liées entre elles par des liaisons de dépendances représentées par des flèches. C'est une représentation synthétique des relations logiques entre activités, construit de gauche à droite pour représenter la chronologie d'un projet.

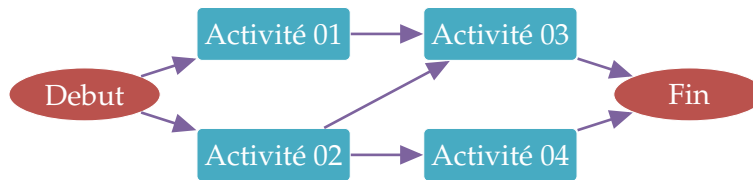


FIGURE 3.1 – Le graphe des antécédents.

3.5.2 Le diagramme fléché

- Les activités figurent sur les flèches et les ronds représentent les jalons.

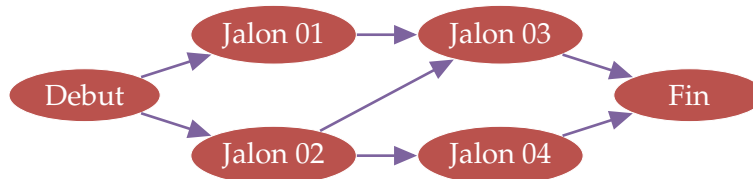


FIGURE 3.2 – Le diagramme fléché.

- **Remarque :** Dans la majorité des progiciels de planification, le formalisme du graphe des antécédents est retenu parmi lesquels on cite PMW (Project Management Workbench) et Microsoft Project.

- L'avantage du graphe des antécédents est qu'il permet une visualisation claire de la logique des dépendances. Il donne aussi la possibilité de relations avec des délais (écart/recouvrement).

3.5.3 Méthode du chemin critique

- L'analyse du graphe permet de mettre en évidence des chemins qui comportent des tâches critiques dits chemins critiques.
- Le chemin critique correspond à la séquence de tâches qui détermine la durée totale du projet. Ce chemin est continu depuis le début jusqu'à la fin du projet.
- Tout retard affectant une tâche du chemin critique est intégralement répercuté sur la durée du projet et donc sa date de fin. La tâche critique est une tâche du chemin critique.
- Pour cela, on calcule les paramètres clés attachés à chaque graphe :
 - **Dates au plus tôt** : Début au plus tôt et fin au plus tôt.
 - **Dates au plus tard** : Début au plus tard et fin au plus tard.
 - **Marges** : Marge totale et marge libre.
- **Exemple** :

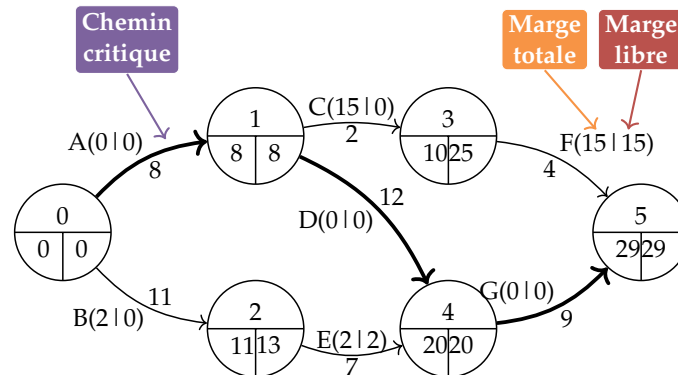


FIGURE 3.3 – Le réseau PERT.

3.5.4 Les types de liens

- Les liens entre les tâches représentent des contraintes provenant de la nature des tâches elles-mêmes. Les types de liens qui existent sont les suivants :

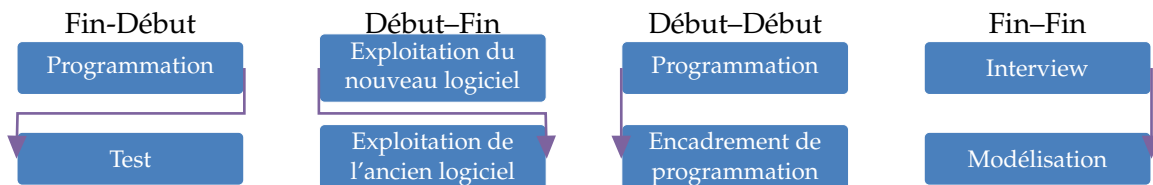


FIGURE 3.4 – Les types de liens.

3.6 Gestion des risques

- Un risque correspond à **la probabilité qu'un événement indésirable ait lieu**. Une des principales tâches du chef de projet : anticiper les risques qui peuvent affectés la programmation du projet ou la qualité du logiciel et par conséquent déterminer les actions à entreprendre a fin de les éviter.
- Différents types de risque peuvent être identifiés , parmi lesquels on cite les catégories des risques Suivants :
 - **Risques liés au projet** : risques qui affectent la programmation du projet ou les ressources qui lui sont alloués.
 - **Risques du produit** : risques qui affectent la qualité ou la performance du logiciel.
 - **Risques métiers ou commerciaux** : risques qui affectent l'organisation développant ou procurant le logiciel.

TABLE 3.1 – Type des risques.

Risque	Type	Description
Rotation du personnel	Projet	Personnel expérimenté abandonne le projet avant sa fin.
Changement du besoin	Produit	Grand nombre de changement des besoins est effectué.
Compétition du produit	Métier	Produit compétitif est disponible sur le marché avant que le système soit complet.

3.7 Le processus de gestion des risque

Le processus de gestion des risques comprend les étapes suivantes :

1. **Identification des risques** : L'identification du risque consiste à découvrir les risques possibles liés au projet. Elle se base sur une approche intuitive ou simplement sur les expériences.
2. **Appréciation des risques** : Elle comporte deux tâches distinctes : la première consiste à estimer la probabilité que l'événement se produise et la seconde consiste à évaluer l'impact, le coût des conséquences d'un tel événement.
3. **Calculs des risques** : L'attribution d'une valeur chiffrée à un risque consiste à multiplier sa probabilité qu'il se produise par le coût estimé de ses conséquences.
4. **Atténuation des risques** : C'est une stratégie pro-active pour tenter de diminuer la probabilité ou d'en réduire l'impact. La démarche couramment utilisée consiste à traiter les points potentiellement problématiques le plus tôt possible.
5. **Plan de gestion des risques** : Il doit contenir , pour chacun des risques, un identifiant, une description, une estimation de sa probabilité et de son impact, une liste des stratégies d'atténuation envisageables, des plans de secours, des événements déclencheurs qui détermineront l'activation des plans de secours et enfin la liste des personnes responsables.

3.8 Conclusion

- Une bonne gestion des projets est essentielle car ceux - ci doivent être développés en respectant certaines contraintes (coût, délai par exemple).
- Le plus important rôle du chef de projet est la planification, l'estimation du coût et la programmation du projet.
- La programmation d'un projet implique l'établissement d'un diagramme des antécédents déterminant les activités et leurs relations ainsi qu'un diagramme de Gantt.
- Les risques associés à un projet doivent être identifiés, leurs probabilités sont à estimer ainsi que leurs conséquences. Des plans sont à établir afin de les éviter ou de réduire leurs conséquences.

Méthode 1

La méthode de planning PERT

1.1 Généralités

- Dans le cas de fabrications non répétitives et très complexes, (prototypes, avions, grands ensembles en construction, etc), on utilisera la méthode dite du chemin critique dont la plus utilisée est la méthode P.E.R.T.
- Cette méthode a été mise au point en 1957 aux Etats-Unis, lors du développement du missile POLARIS. Ce projet mobilisait 250 fournisseurs principaux et environ 9000 sous-traitants. Le délai initial prévu de 6 ans a pu être ramené à 2 années et demie.
- **P.E.R.T** (Program Evaluation Review Technique) : technique d'élaboration et de mise à jour de programme.

1.2 Objectifs & Domaines d'application

La méthode de planning PERT sert à :

- Définir le délai total d'accomplissement de l'œuvre et éventuellement proposer des moyens pour le réduire.
- Connaître les conséquences du changement de la durée d'une tâche partielle.
- Evaluer les moyens à mettre en œuvre.
- Etablir une relation entre les délais et les coûts.

Exemples de domaines d'application de cette méthode :

- Dans le bâtiment (grands ensembles, hôpitaux, etc.....)
- Dans les travaux public (routes, ponts, etc.....)
- Pour l'ordonnancement de prototypes.
- En maintenance pour coordonner les tâches de plusieurs équipes de spécialités différentes.

1.3 Conditions de mise en œuvre

Les conditions de mise en œuvre de la méthode PERT sont :

- L'œuvre doit être **divisée** en tâches partielles.
- La durée de chaque tâche doit être **connue**.


- L'étude technique doit **préciser** si certaines tâches doivent être impérativement effectuées avant certaines autres tâches.

Principe → basé sur le **graphique** ou **diagramme** apparaitront les liaisons entre les différentes tâches de l'œuvre à réaliser.

1.4 Définitions & Représentations

1.4.1 Tâche

Une tâche fait évoluer l'œuvre vers son état final, elle consomme donc du temps, de l'énergie, de la matière et de ce fait coûte. Chaque tâche est représentée par une flèche (segment orienté dans le sens de l'écoulement du temps) dont la longueur est indépendante de la durée de la tâche.

Symbole : 

A = Identification de la tâche
5 = Durée de la tâche

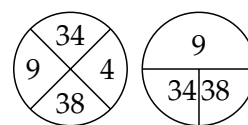
1.4.2 Tâche fictive

Une tâche fictive représente une contrainte entre tâches non indépendantes. Chaque tâche fictive est représentée par une flèche pointillée, sa durée est nulle, elle ne consomme aucune ressource, elle ne coûte donc rien.

Symbole : 

1.4.3 Étape

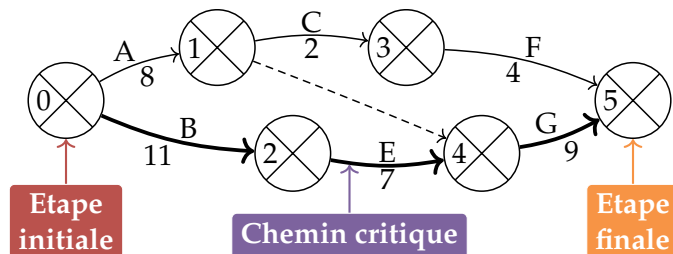
On appelle étape le début ou la fin d'une tâche. Une étape sera représentée par un cercle ou une autre figure géométrique. Une étape est de durée nulle, elle ne coûte donc rien.

Symbole : 

9 = Identification de l'étape
34 = Date au plus tôt
38 = Date au plus tard
4 = Marge totale

1.4.4 Réseau

Un réseau PERT est l'ensemble des tâches et des étapes qui représente l'œuvre. Le réseau met en évidence les relations entre les tâches et les étapes.



Remarque : un réseau PERT est :

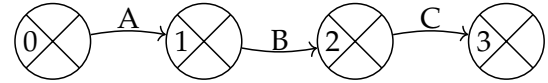
- Non cyclique;
- A une seule tâche entre deux étapes;
- A une seule étape initiale : c'est l'étape qui n'a aucune tâche entrante;
- A une seule étape finale : c'est l'étape qui n'a aucune tâche sortante.

1.5 Conventions

- Des tâches **consécutives** sont des tâches qui se suivent.
- Des tâches **antérieures** sont des tâches qui, par rapport à une autre, doivent être réalisées avant.
- Les antériorités immédiates sont appelées **antécédentes**.

Exemple :

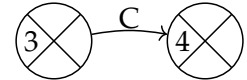
- A et B sont antérieures de C
- B est antécédente de C.



1^{ère} Convention

Toute tâche a une étape début pour origine et une étape fin pour extrémité.

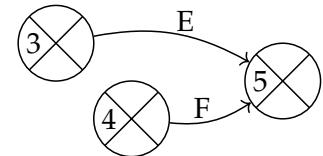
Exemple : La tâche C commence à l'étape 3 et se termine à l'étape 4.



2^{ème} Convention

Une étape ne peut être atteinte que lorsque les tâches qui la précèdent sont toutes terminées.

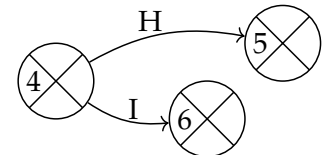
Exemple : L'étape 5 ne sera atteinte que lorsque les tâches E et F seront terminées.



3^{ème} Convention

Aucune tâche ne peut commencer tant que l'étape située à son origine n'est pas atteinte.

Exemple : Les tâches H et I ne pourront commencer que lorsque l'étape 4 sera atteinte.

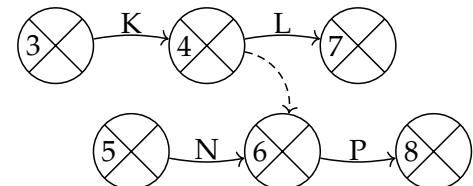


4^{ème} Convention

L'étape située à l'extrémité d'une fictive ne peut être atteinte qu'après l'étape située à son origine.

Exemple :

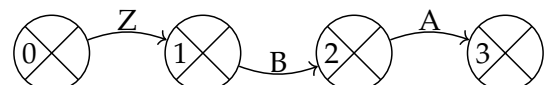
- L'étape 6 ne peut être atteinte que : lorsque l'étape 4 est atteinte, et lorsque la tâche N est terminée.
- Lecture : La tâche L a pour antécédente K. La tâche P a pour antécédente N et K.



1.6 Exercices

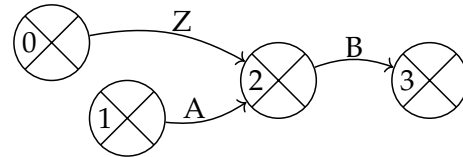
Exercice 01

- A a pour antériorité B.
- B a pour antériorité Z.

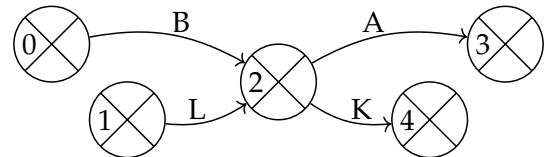


Exercice 02

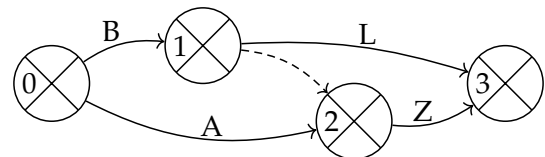
- La tâche B a pour antécédents A et Z.
-

**Exercice 03**

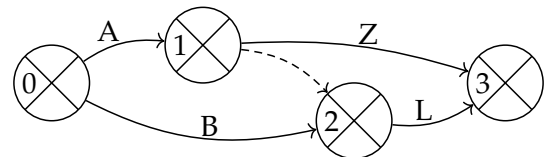
- Les tâches K et A ont pour antécédents L et B.
-

**Exercice 04**

- La tâche Z a pour antécédents A et B.
- La tâche L a pour antécédent B.

**Exercice 05**

- La tâche Z a pour antécédent A.
- La tâche L a pour antécédents A et B.

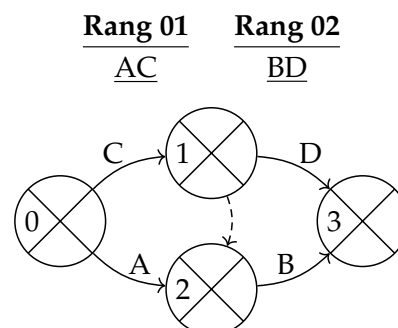
**1.7 Rangs ou Niveaux**

- **Sont de RANG 1** : Les tâches qui n'ont pas de tâches antérieures.
- **Sont de RANG 2** : Les tâches qui ont pour antécédentes les tâches de rang 1.
- **Sont de RANG 3** : Les tâches qui ont pour antécédentes les tâches : de rang 2.
- La même logique est à appliquer jusqu'aux dernières tâches.
-
- **Cas particulier** : Lorsqu'une tâche a **plusieurs antécédentes**, on prend la tâche de **rang le plus élevé**.

Exercice 01

Classer les tâches dans leurs différents rangs.

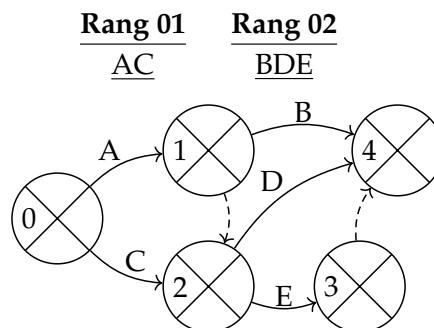
Antécédents	Opération	Rang
Rien	A	1
A C	B	2
Rien	C	1
C	D	2



Exercice 02

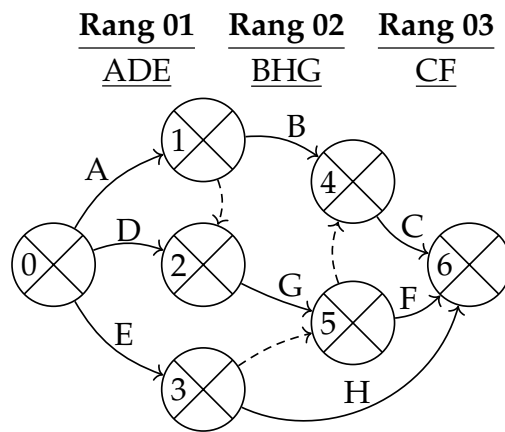
Classer les taches dans leurs différents rangs.

Antériorités	Opération	Rang
Rien	A	1
A	B	2
Rien	C	1
A C	D	2
A C	E	2

**Exercice 03**

Classer les taches dans leurs différents rangs.

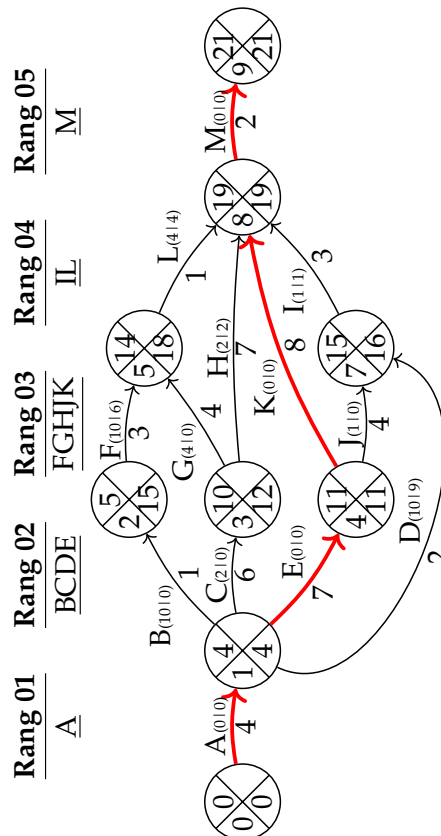
Antériorités	Opération	Rang
Rien	A	1
A	B	2
B E G	C	3
Rien	D	1
Rien	E	1
E G	F	3
A D	G	2
E	H	2

**Exercice 04**

Soit le tableau suivant :

Opération	Taches	Durée	Antériorités	Rang
A	X	4	Rien	1
B	X	1	A	2
C	X	6	A	2
D	X	2	A	2
E	X	7	A	2
F	X	3	B	3
G	X	4	C	3
H	X	7	C	3
I	X	3	DJ	4
J	X	4	E	3
K	X	8	E	3
L	X	1	FG	4
M	X	2	HIKL	5

- Déterminer les niveaux des différentes tâches.
- Tracer le réseau P.E.R.T.
- Calculer les dates au plus tôt, les dates au plus tard, les marges totales et les marges libres.
- Mettre en évidence le chemin critique



1.8 Calcul des dates et des marges

1.8.1 Calcul des dates au plus tôt

$$\text{Date au plus tôt d'une étape} \stackrel{\text{Max}}{=} \text{Date au plus tôt de l'étape précédente} + \text{Durée de la tâche comprise entre les 2 étapes}$$

- Au dessous, à gauche du symbole de chaque étape, porter la date au plus tôt de l'étape; c'est la date à laquelle l'étape peut être atteinte au plus tôt.
- On procède dans l'ordre croissant des étapes.
- Quand il y a plusieurs tâches convergentes, on ne retient que la valeur la plus grande.
- La date de la dernière étape représente le temps normal d'exécution.

1.8.2 Calcul des dates au plus tard

$$\text{Date au plus tard d'une étape} \stackrel{\text{Min}}{=} \text{Date au plus tard de l'étape suivante} - \text{Durée de la tâche comprise entre les 2 étapes}$$

- Au dessous, à droite du symbole de chaque étape, porter la date au plus tard de l'étape; c'est la date à laquelle l'étape peut être atteinte au plus tard.
- On procède dans l'ordre décroissant des étapes.
- Quand il y a plusieurs dates au plus tard à une étape, on ne retient que la valeur la plus petite.
- A l'étape 0 la date au plus tard doit être 0.
- **NB** : Toujours la date au plus tôt \leq la date au plus tard.

1.8.3 Calculs des marges

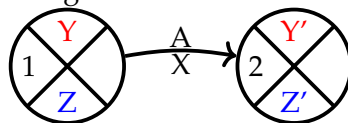
1. **La marge totale** : C'est le retard maximum que peut prendre une tâche sans porter atteinte au plus tard de la tâche suivante (donc sans retarder la fin des travaux). C'est la différence entre la date au plus tard et la date au plus tôt.

$$\text{Marge totale d'une tâche} = \text{Date au plus tard de l'étape suivante} - \text{Durée de la tâche} - \text{Date au plus tôt de l'étape précédente}$$

2. **La marge libre** : C'est le retard maximum que peut prendre une tâche sans porter atteinte au plus tôt de la tâche suivante.

$$\text{Marge libre d'une tâche} = \text{Date au plus tôt de l'étape suivante} - \text{Durée de la tâche} - \text{Date au plus tôt de l'étape précédente}$$

- **NB** : Toujours la marge libre \leq la marge totale.
- **Résumé** : Afin de faciliter le calcul de différents paramètres de réseau PERT (dates ou marges), on peut utiliser la figure suivante :



- La date au plus tard(1) = $Z = Z' - X$
- La date au plus tôt(2) = $Y' = X + Y$
- La marge totale(A) = $Z' - (X + Y)$
- La marge libre (A) = $Y' - (X + Y)$

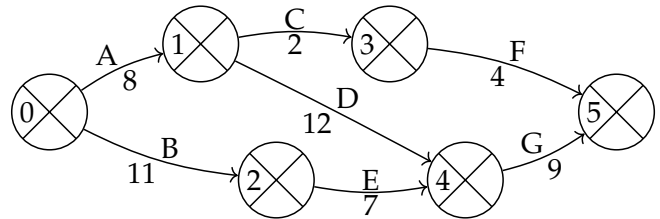
1.9 Exploitation du réseau

- On peut remarquer que certaines étapes présentent des dates au plus tôt **différentes** des dates au plus tard, cela traduit une **marge** qui autorise une certaine souplesse dans la réalisation des tâches.
- **Exemple** : (de l'exemple suivant)
 - L'étape 5 sera atteinte au plus tôt 10 jours après le début du projet, mais pourrait être atteinte au plus tard 25 jours après le début du projet.
- Quand la date au plus tôt est **identique** à la date au plus tard, la marge (totale) est **nulle** et on dit que l'étape est **critique**.
- Le **chemin critique** devra être tracé de l'étape initiale jusqu'à l'étape finale du projet, en reliant les étapes critiques → Il est formé par les tâches critiques.
- Une **tâche** est dite **critique** si sa marge totale est **NULLE** alors que une **étape** est dite **critique** si sa date au plus tôt **égale** à sa date au plus tard.
- Pour un même projet, il peut y avoir plusieurs chemins critiques.
- **Par Convention** : on représente le chemin critique par une **flèche orientée rouge barrée de 2 traits**.

1.10 Exemple de calcul

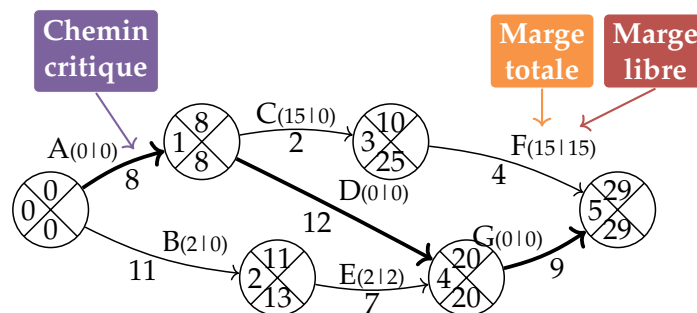
Questions :

- Calculer les dates au plus tôt et les dates au plus tard.
- Calculer les marges totales et les marges libres.
- Mettre en évidence le chemin critique.



Etape	Date au plus tôt Le calcul part de l'étape initiale au temps 0 vers l'étape finale	Date au plus tard Le calcul part de l'étape finale et remonte vers l'étape initiale
0	0	Min(8 - 8 = 0, 13 - 11 = 2) = 0
1	0 + 8 = 8	Min(20 - 12 = 8, 25 - 2 = 23) = 8
2	0 + 11 = 11	20 - 7 = 13
3	8 + 2 = 10	29 - 4 = 25
4	Max(8 + 12 = 20, 11 + 7 = 18) = 20	29 - 9 = 20
5	Max(20 + 9 = 29, 10 + 4 = 14) = 29	29

Tâche	Marge totale	Marge libre
A	8 - (8 + 0) = 0	8 - (8 + 0) = 0
B	13 - (11 + 0) = 2	11 - (11 + 0) = 0
C	25 - (8 + 2) = 15	10 - (8 + 2) = 0
D	20 - (12 + 8) = 0	20 - (12 + 8) = 0
E	20 - (7 + 11) = 2	20 - (7 + 11) = 2
F	29 - (4 + 10) = 15	29 - (4 + 10) = 15
G	29 - (9 + 20) = 0	29 - (9 + 20) = 0

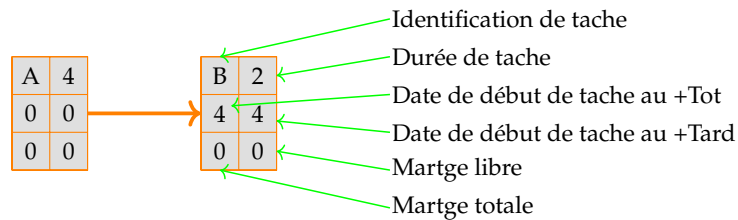


1.11 La méthode MPM

- Le MPM (méthode des potentiels métra) est méthode proche du PERT qui sert également à déterminer et diminuer la durée de réalisation d'un projet.
- La différence entre ces deux méthodes sera un changement au niveau de la représentation du graphe.
- Ici, les tâches seront représentées par des sommets et les flèches définiront les liens d'antériorités.
- On notera également la disparition des tâches fictives.
- On trouvera les mêmes paramètres que pour le PERT avec les dates au plus tôt et au plus tard, les marges libres et totales et le chemin critique.

Symbolisation :

- Date de début de tache B au +Tôt = Date de fin de tache A au +Tôt.
- Date de début de tache B au +Tard = Date de fin de tache A au +Tard.



Méthode 2

La méthode de planning PERT Probabiliste

2.1 Généralités

- Cette technique s'appuie sur le graphe des antécédents. Elle permet :
 - d'inclure dans la planification le risque et l'incertitude attachés à chaque tâche ;
 - d'en déduire une durée du projet assortie d'un niveau de probabilité ;
- La durée de chaque tâche peut être considérée comme une variable aléatoire, c'est-à-dire que l'on peut faire plusieurs estimations (\pm probables) de la durée de la tâche.
- Alors la durée de tout chemin dans le graphe PERT est également considérée une variable aléatoire ; puisque c'est une somme de variables aléatoires.
- Sous réserves des conditions suivantes :
 - un nombre suffisamment élevé de tâches (min 4 tâches)
 - un ordre de grandeur semblable pour toutes les tâches
 - l'indépendance entre les durées des tâches.
- La durée probable du chemin obéit à une loi de distribution proche de loi normale (Laplace-gauss), dont la représentation graphique de la courbe en cloche dite de gauss. Le calcul des paramètres du PERT probabiliste se fait en trois étapes.

2.2 Calcul des paramètres

1. **1^{ière} étape** : consiste à déterminer la loi de probabilité attachée à chaque tâche. En pratique, on retient souvent une loi de distribution bêta c'est dire qu'on est capable de donner trois estimations :
 - T_{opt} **estimation optimiste de la durée** : c'est le temps minimal possible, tout se déroule que prévu
 - T_{pes} **estimation pessimiste de la durée** : c'est le temps maximal possible, si tout déroule au plus mal (hors catastrophe)
 - T_{vrai} **durée vraisemblable de la durée** : c'est que l'on donnerait si on devait n'en donner qu'une seule.
2. **2^{ième} étape** : consiste à calculer trois valeurs pour chaque tâche i :
 - Sa **durée probable** $T_{prob}(i)$: c'est le temps moyen de la tâche si elle est répétée un grand nombre de fois ;
 - Sa **variance** $v(i)$ et son **écart -type** $e(i)$: plus les estimations optimistes et pessimistes sont éloignées ; plus elles présentent d'incertitude. C'est la variance qui mesure cette incertitude.

C'est la variance qui mesure cette incertitude. Si elle est faible, l'estimation de la durée probable de la tâche sera plus précise.

- $t_{prob}(i) = (T_{opt}(i) + 4 * T_{vrai}(i) + T_{pes}(i))/6$
- $e(i) = (T_{pes}(i) - T_{opt}(i))/6$
- $v(i) = e(i)^2$

3. **3^{ième} étape** : consiste à calculer pour chaque chemin :

- $D_{est} = \text{Somme}(t_{prob}(i))$: la durée estimée de chemin pour toutes les tâches i du chemin
- $V_{est} = \text{somme}(e(i)^2)$: la variance estimée de chemin pour toutes les tâches i du chemin
- $E_{est} = \sqrt{V_{est}}$: l'écart-type estimé de chemin

Comme la durée du chemin obéit à une loi normale; on peut utiliser la table de gauss pour obtenir la durée D_p : $D_p = D_{est} + G(p) \times E_{est}$

La durée estimée du projet est probable à 50% (car $G(50) = 0$).

2.2.1 Exemple

- Soit un chemin de durée estimée $D_{est} = 100$ et d'écart-type estimé $E_{est} = 15$. On peut en déduire soit une durée probable pour une probabilité donnée; soit une probabilité d'achèvement dans un délai donné. Ainsi,
- La durée probable à 90 % est : $D_{90} = 100 + G(90) \times 15 = 100 + 1.28 \times 15 = 119$ jours.
- La durée probable à 60 % est : $D_{60} = 100 + G(60) \times 15 = 100 + 0.26 \times 15 = 104$ jours.
- La probabilité de terminer en 80 jours est : $80 = 100 + G(P) \times 15$ d'où $G(p) = -1,33$
- La probabilité est 0,0918. car $P(-1,33) = 1 - P(1,33) = 1 - 0,9082$

2.2.2 Calcul des risques

- On peut donner une mesure du ratio : $R = (T_{pes} - T_{opt})/T_{pes}$
 - $R < 0,25$ le risque est faible
 - $0,25 < R < 0,50$ le risque est moyen
 - $R > 0,5$ le risque est fort
- Le PERT probabiliste est intéressante pour les projets à forte incertitude, particulièrement sur le chemin critique, ainsi que sur les chemins ayant les plus forts risques.
- Si l'incertitude est faible, la différence $(T_{pes} - T_{opt})$ est faible par rapport à la durée estimée de la tâche, dans ce cas, les variances des tâches sont faibles de même que la variance du chemin, donc E_{est} on n'a pas donc besoin d'un PERT probabiliste (On peut se contenter d'un graphe de la méthode des antécédents).

2.3 Exercice «Recette»

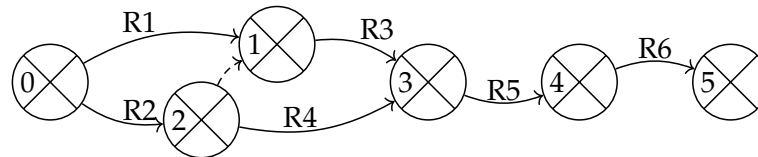
- Soit à estimer la durée d'une recette de gâteau au chocolat. Une incertitude pèse sur la durée de la préparation. Elle provient de différents facteurs : la qualité du four, la disponibilité de la certains outillages (batteur électrique...) et la rapidité. Le tableau ci-dessous donne la liste des tâches, ainsi les estimations optimiste, pessimiste et vraisemblable de chaque tâche.
 1. Calculer le risque, la durée probable, l'écart-type et la variance de chaque tâche.
 2. Quel est le chemin critique.
 3. Quelle est la durée estimée de préparation du gâteau $P=95\%$ et $P=90\%$,
 4. Quelle est la probabilité que l'on termine en 37 minutes.

TABLE 2.1 – Liste des tâches de l'exercice «Recette».

Code	Tâches	T_{opt}	T_{pes}	T_{vrai}
R1	Faire fondre le beurre et le chocolat	6	9	7,5
R2	Séparer les œufs en jaunes et blancs	1	4,5	3
R3	Ajuster les jaunes au mélange fondu et faire cuire en 2 mn	6	8	7
R4	Monter les blancs en neige	2	12	5
R5	Retirer le mélange du feu et incorporer les blancs	2	6	3
R6	Faire cuire au four	16	22	18

Corrigé

1. Le réseau PERT :



2. Les paramètres probabilistes :

Tâches	T_{opt}	T_{pes}	T_{vrai}	Risque	$t_{prob}(i)$	$e(i)$	$v(i)$
R1	6	9	7,5	0,33	7,5	1/2	0,25
R2	1	4,5	3	0,78	2,92	7/12	0,34
R3	6	8	7	0,25	7	1/3	0,11
R4	2	12	5	0,83	5,67	5/3	2,78
R5	2	6	3	0,67	3,33	2/3	0,44
R6	16	22	18	0,27	18,33	1	1

3. Les durées probables des différents chemins sont données sur le tableau suivant

- $D_{95} = D_{est} + 1,65 \times E_{est}$
- $D_{90} = D_{est} + 1,28 \times E_{est}$

Chemins	D_{est}	V_{est}	E_{est}	D_{95}	D_{90}
R1R3R5R6	36,17	1,81	1,34	38,38	37,89
R2R3R5R6	31,58	1,9	1,38	33,86	33,35
R2R4R5R6	30,25	4,56	2,14	33,77	32,98

4. La probabilité d'une durée de 37 minutes se calcule sur le chemin critique :

- $D_p = D_{est} + G(p) \times E_{est}$ soit $37 = 36,17 + G(p) \times 1,34$.
- $G(p) = 0,619$ d'où la probabilité est environ 0,72.

2.4 Fonction de répartition de loi normale standard

μ_α	0,00	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08	0,09
0,0	0,5000	0,5040	0,5080	0,5120	0,5160	0,5199	0,5239	0,5279	0,5319	0,5359
0,1	0,5398	0,5438	0,5478	0,5517	0,5557	0,5596	0,5636	0,5675	0,5714	0,5753
0,2	0,5793	0,5832	0,5871	0,5910	0,5948	0,5987	0,6026	0,6064	0,6103	0,6141
0,3	0,6179	0,6217	0,6255	0,6293	0,6331	0,6368	0,6406	0,6443	0,6480	0,6517
0,4	0,6554	0,6591	0,6628	0,6664	0,6700	0,6736	0,6772	0,6808	0,6844	0,6879
0,5	0,6915	0,6950	0,6985	0,7019	0,7054	0,7088	0,7123	0,7157	0,7190	0,7224
0,6	0,7257	0,7291	0,7324	0,7357	0,7389	0,7422	0,7454	0,7486	0,7517	0,7549
0,7	0,7580	0,7611	0,7642	0,7673	0,7704	0,7734	0,7764	0,7794	0,7823	0,7852
0,8	0,7881	0,7910	0,7939	0,7967	0,7995	0,8023	0,8051	0,8078	0,8106	0,8133
0,9	0,8159	0,8186	0,8212	0,8238	0,8264	0,8289	0,8315	0,8340	0,8365	0,8389
1,0	0,8413	0,8438	0,8461	0,8485	0,8508	0,8531	0,8554	0,8577	0,8599	0,8621
1,1	0,8643	0,8665	0,8686	0,8708	0,8729	0,8749	0,8770	0,8790	0,8810	0,8830
1,2	0,8849	0,8869	0,8888	0,8907	0,8925	0,8944	0,8962	0,8980	0,8997	0,9015
1,3	0,9032	0,9049	0,9066	0,9082	0,9099	0,9115	0,9131	0,9147	0,9162	0,9177
1,4	0,9192	0,9207	0,9222	0,9236	0,9251	0,9265	0,9279	0,9292	0,9306	0,9319
1,5	0,9332	0,9345	0,9357	0,9370	0,9382	0,9394	0,9406	0,9418	0,9429	0,9441
1,6	0,9452	0,9463	0,9474	0,9484	0,9495	0,9505	0,9515	0,9525	0,9535	0,9545
1,7	0,9554	0,9564	0,9573	0,9582	0,9591	0,9599	0,9608	0,9616	0,9625	0,9633
1,8	0,9641	0,9649	0,9656	0,9664	0,9671	0,9678	0,9686	0,9693	0,9699	0,9706
1,9	0,9713	0,9719	0,9726	0,9732	0,9738	0,9744	0,9750	0,9756	0,9761	0,9767
2,0	0,9772	0,9778	0,9783	0,9788	0,9793	0,9798	0,9803	0,9808	0,9812	0,9817
2,1	0,9821	0,9826	0,9830	0,9834	0,9838	0,9842	0,9846	0,9850	0,9854	0,9857
2,2	0,9861	0,9864	0,9868	0,9871	0,9875	0,9878	0,9881	0,9884	0,9887	0,9890
2,3	0,9893	0,9896	0,9898	0,9901	0,9904	0,9906	0,9909	0,9911	0,9913	0,9916
2,4	0,9918	0,9920	0,9922	0,9925	0,9927	0,9929	0,9931	0,9932	0,9934	0,9936
2,5	0,9938	0,9940	0,9941	0,9943	0,9945	0,9946	0,9948	0,9949	0,9951	0,9952
2,6	0,9953	0,9955	0,9956	0,9957	0,9959	0,9960	0,9961	0,9962	0,9963	0,9964
2,7	0,9965	0,9966	0,9967	0,9968	0,9969	0,9970	0,9971	0,9972	0,9973	0,9974
2,8	0,9974	0,9975	0,9976	0,9977	0,9977	0,9978	0,9979	0,9979	0,9980	0,9981
2,9	0,9981	0,9982	0,9982	0,9983	0,9984	0,9984	0,9985	0,9985	0,9986	0,9986
3,0	0,9987	0,9987	0,9987	0,9988	0,9988	0,9989	0,9989	0,9989	0,9990	0,9990
3,1	0,9990	0,9991	0,9991	0,9991	0,9992	0,9992	0,9992	0,9992	0,9993	0,9993
3,2	0,9993	0,9993	0,9994	0,9994	0,9994	0,9994	0,9994	0,9995	0,9995	0,9995
3,3	0,9995	0,9995	0,9995	0,9996	0,9996	0,9996	0,9996	0,9996	0,9996	0,9997
3,4	0,9997	0,9997	0,9997	0,9997	0,9997	0,9997	0,9997	0,9997	0,9997	0,9998
3,5	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998	0,9998
3,6	0,9998	0,9998	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999	0,9999
>	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Méthode 3

La méthode de planning GANTT

3.1 Généralités

- Le suivi de l'avancement de l'œuvre peut se faire directement sur le réseau P.E.R.T. Ce n'est pas toujours très facile car les longueurs des tâches ne sont pas proportionnelles à leurs durées.
- On peut avantageusement **transformer le réseau en un graphique de GANTT** qui lui tient compte de la durée des tâches (la tâche est représentée par segment de droite dont la longueur est proportionnelle au temps). On l'appelle également **planning à bandes**.
- Cette méthode est proposée par Henry Laurence Gantt en 1910 et encore très répandue. Elle consiste à déterminer la manière de positionner les différentes tâches d'un projet à exécuter, sur une période déterminée.
- **Objectifs** : la Méthode GANTT permet de :
 - déterminer la charge des postes de travail.
 - répartir chronologiquement les postes de travail.
 - suivre l'avancement de la fabrication, par pièce et par ensemble.
 - vérifier la disponibilité de la main d'œuvre.
 - établir la situation des stocks.
 - établir le calendrier des opérations ou tâches à mettre en œuvre pour réaliser un projet.
- **Les domaines d'application** de la méthode GANTT sont les mêmes domaines que PERT.
- **Conditions de mise en œuvre** : son application est indispensable après le graphique PERT si le projet est complexe, ou seule si le projet est simple.

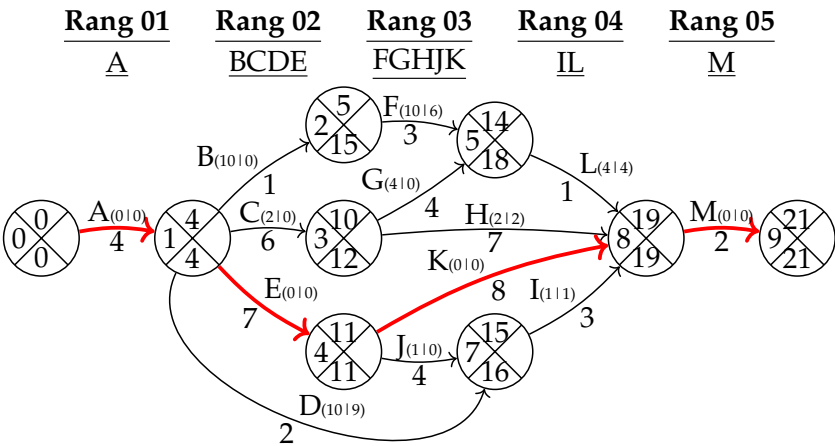
Exercice type

Soit le tableau suivant :

- Déterminer les niveaux des différentes tâches.
- Tracer le diagramme GANTT.
- Calculer les dates au plus tôt, les dates au plus tard.
- Mettre en évidence le chemin critique.
- Proposer un ajustement des tâches au plus tard.

Opération	Tâches	Durée	Antériorités	Rang
A	X	4	Rien	1
B	X	1	A	2
C	X	6	A	2
D	X	2	A	2
E	X	7	A	2
F	X	3	B	3
G	X	4	C	3
H	X	7	C	3
I	X	3	DJ	4
J	X	4	E	3
K	X	8	E	3
L	X	1	FG	4
M	X	2	HIKL	5

Réseau PERT : avec calcul des dates au plus tôt, date au plus tard, marges totales, marge libres et détermination du chemin critique.



Complétez le diagramme GANTT suivant :

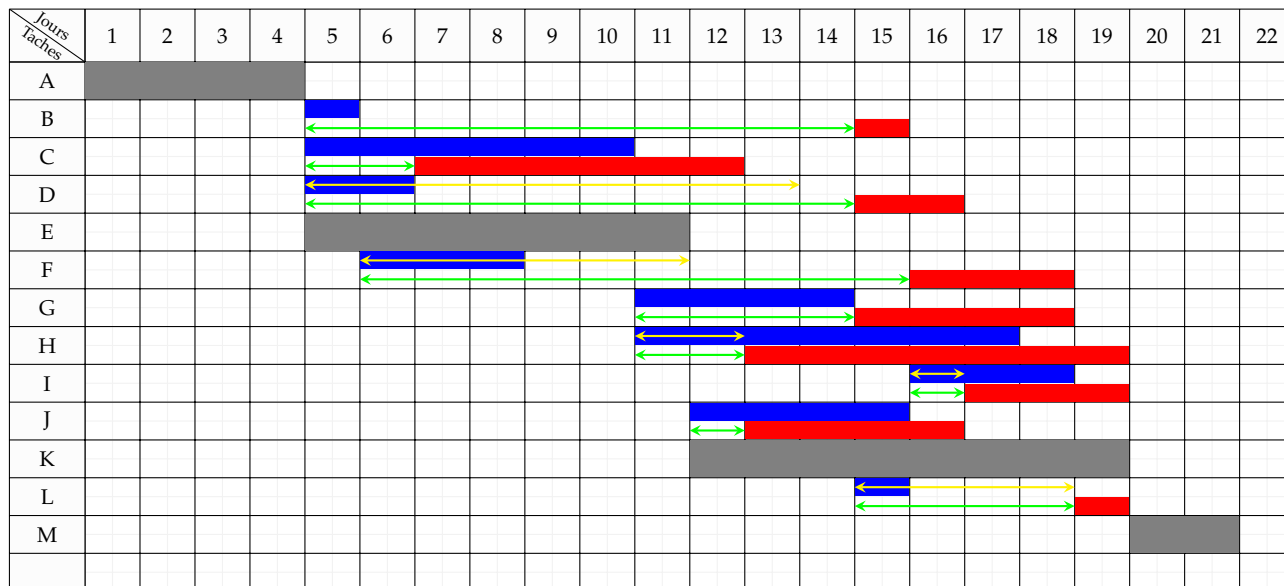
Jours tâches	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A																						
B																						
C																						
D																						
E																						
F																						
G																						
H																						
I																						
J																						
K																						
L																						
M																						

3.1.1 Utilisation de la matrice

1. Remplir la 1^{ière} colonne : inscrire les identifications des tâches.
2. Remplir la 1^{ière} ligne : inscrire les dénominations temps à lesquelles l’œuvre est exécuté.
3. Remplir le **tableau** : pour chaque tâche répertoriée, composée ainsi de 2 lignes :
 - Tracer **sur la ligne supérieure** un segment de longueur proportionnelle à **la durée** de la tâche, en se basant **sur les dates au plus tôt**. La même procédure **sur la ligne inférieure** mais en se basant **sur les dates au plus tard**.
 - Si le diagramme n’est pas composé de deux lignes pour chaque tâche, il faudra effectuer deux diagrammes de GANTT. Un diagramme au plus tôt et un au plus tard.

- Chaque tâche devra être ainsi reportée sur le graphique en tenant compte des antériorités qui les lient entre elles.
- Le segment représentant une durée de tâche appartenant au **chemin critique** devra être symbolisé par un **trait fort**.

Le diagramme GANTT :



3.2 Calcul des marges

3.2.1 Calcul des marges totales

Une marge totale d'une tâche est un retard possible au démarrage de cette tâche par rapport à l'instant de départ au plus tôt qui n'entraîne aucun recul de la date d'achèvement de l'œuvre, mais qui imposera que les tâches suivantes soient commencées aux dates au plus tard.

$$\begin{aligned}
 \text{Marge totale d'une tâche} &= \text{Date de fin au plus tard de la tâche} - \text{Date de début au plus tôt de la tâche} - \text{Durée de la tâche} \\
 &= \text{Date de fin au plus tard de la tâche} - \text{Date de fin au plus tôt de la tâche}
 \end{aligned}$$

3.2.2 Calcul des marges libres

Une marge libre d'une tâche est un retard toléré par cette tâche par rapport à l'instant de départ au plus tôt (ou un temps d'interruption en cours d'exécution, ou un allongement de la durée prévue) qui n'entraîne aucune modification du calendrier des tâches en aval et notamment aucun recul de la date d'achèvement de l'œuvre.

$$\text{Marge libre d'une tâche} = \text{Date de fin au plus tôt de la tâche} - \text{Date de début au plus tôt de la tâche} - \text{Durée de la tâche}$$

Selon les circonstances, les marges pourront être utilisées :

- Pour conserver une sécurité de temps sur les tâches qui ne sont pas situées sur le chemin critique. Les tâches sont déclenchées au plus tôt.
- Pour réduire le coût des en-cours. Les tâches sont alors déclenchées au plus tard au risque de dépasser la durée impartie.

3.3 Utilisation du tableau

- Complétez la colonne des supériorités.
- Calculez les dates de début au plus tôt et au plus tard pour chaque tâche.
- Calculez les dates de fin au plus tard et au plus tôt pour chaque tâche.
- Déterminez les marges et le chemin critique.

Exercice type :

Complétez le tableau suivant :

Tâches	Antériorités	Supériorités	Durée	Début		Fin		Marges		Chemin Critique
				+Tôt	+Tard	+Tôt	+Tard	Totale	Libre	
A	Rein		4							
B	A		1							
C	A		6							
D	A		2							
E	A		7							
F	B		3							
G	C		4							
H	C		7							
I	DJ		3							
J	E		4							
K	E		8							
L	FG		1							
M	HIKL		2							

Corrigé : le tableau de calcul des marges - PERT analytique.

Tâches	Antériorités	Supériorités	Durée	Début		Fin		Marges		Chemin Critique
				+Tôt	+Tard	+Tôt	+Tard	Totale	Libre	
A	Rein	BCDE	4	0	0	4	4	0	0	X
B	A	F	1	4	14	5	15	10	0	
C	A	GH	6	4	6	10	12	2	0	
D	A	I	2	4	14	6	16	10	9	
E	A	JK	7	4	4	11	11	0	0	X
F	B	L	3	5	15	8	18	10	6	
G	C	L	4	10	14	14	18	4	0	
H	C	M	7	10	12	17	19	2	2	
I	DJ	M	3	15	16	18	19	1	1	
J	E	I	4	11	12	15	16	1	0	
K	E	M	8	11	11	19	19	0	0	X
L	FG	M	1	14	18	15	19	4	4	
M	HIKL	Rien	2	19	19	21	21	0	0	X

Exemple 01

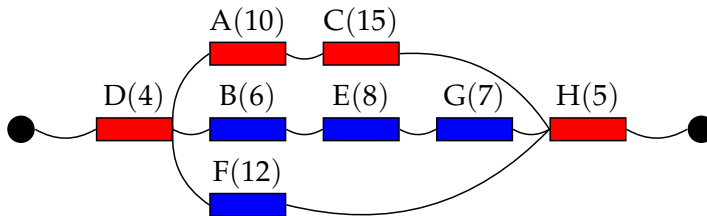
Soit le tableau suivant :

Tâches	Antériorités	Durée
A	D	10
B	D	6
C	A	15
D	/	4
E	B	8
F	D	12
G	E	7
H	FCG	5

- Tracer le réseau de tâches en fonction de contraintes.
- Calculer les marges libres, les marges totales.
- Tracer le réseau GANTT à jalonnement au plus tôt et le réseau GANTT à jalonnement au plus tard.

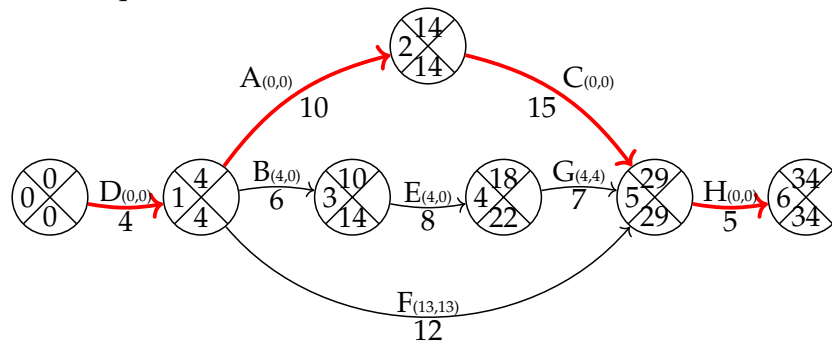
Corrigé :

- Réseau de tâches en fonction de contraintes :

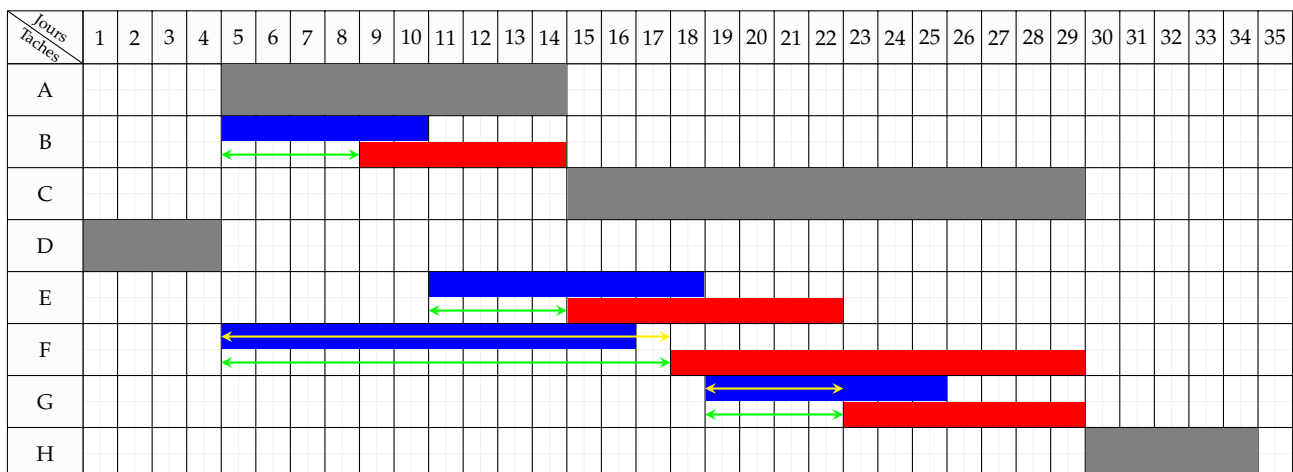


- En **rouge**, le chemin critique est composé des tâches dites critiques pour lesquelles un retard éventuel de réalisation entraînerait une augmentation globale de la durée de projet (34 jours)

- Le diagramme PERT avec calcul des dates au plus tard, marges totales et libres avec détermination du chemin critique.

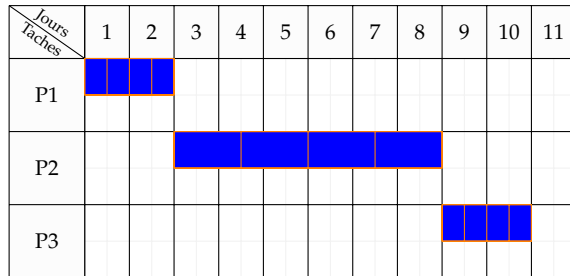


- Le diagramme GANTT (jalonnement au plus tôt et jalonnement au plus tard).

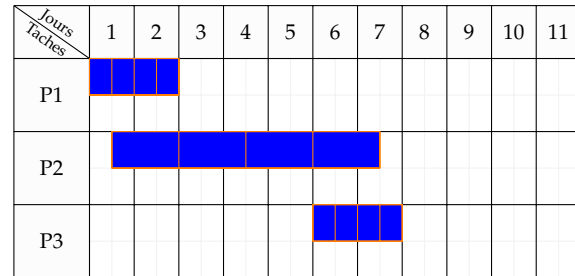


Exemple 02

Soit l'ordonnancement de la production de 100 pièces référencées ZCC et devant subir des opérations sur les postes P1, P2 et P3. Dans le premier cas la production se termine au bout de 10 heures. Si les lots sont fractionnés en 4, il est possible d'effectuer un chevauchement. Cela va se traduire par un transfert au poste suivant toutes les 25 pièces. La production se termine maintenant au bout de 7 heures, on a gagné 3 heures.



Jalonnement au plus tôt



Jalonnement au plus tôt avec chevauchement

3.4 Optimisation des ressources

- L'optimisation des ressources est une technique d'analyse de réseau de planification appliquée à une planification qui a déjà été analysée par une méthode du chemin critique.
- L'optimisation des ressources est nécessaire lorsque les ressources ont été sur-utilisées.

Exemple : lorsqu'une ressource a été affectée à deux tâches ou plus au cours de la même période. Il peut également être nécessaire lorsque certaines ressources sont disponibles en quantités limitées, alors que le calendrier exige plus que les quantités disponibles.

- En bref, l'optimisation des ressources est utilisée lorsqu'il y a un conflit de ressources (c'est-à-dire lorsque le calendrier nécessite plus que les ressources disponibles) ou lorsqu'il est nécessaire de maintenir l'utilisation des ressources à un niveau constant.
- Deux exemples de techniques d'optimisation des ressources :

1. Nivellement des ressources (resource leveling)

Ré-ordonnancement des activités d'un planning tenant compte de la logique séquentielle et des contraintes de disponibilité de ressources. Lorsqu'une activité ne peut être réalisée à une certaine date parce qu'on manque de ressources, elle est reculée jusqu'au moment où les ressources nécessaires seront disponibles, ce qui conduit souvent à reculer la ou les dates objectif souhaitées du projet.

2. Lissage des ressources (resource smoothing)

Le lissage des ressources est une procédé d'utilisation des marges permettant une mobilisation des ressources aussi uniforme que possible (ne pas confondre avec nivellement), sans décaler les dates objectifs du projet. Le lissage admet une augmentation éventuelle des moyens prévus initialement.

Exercice type

- Considérez un projet avec 7 tâches comme indiqué dans le tableau ci-dessous. L'organisation n'a que 6 ressources disponibles avec eux.
- La première étape consiste à tracer le schéma du réseau PERT et à déterminer le chemin critique.

Tâche	Antériorités	Durée	Nombre de ressources utilisées
A	Rien	3	6
B	A	2	1
C	B	5	5
D	B	4	2
E	C	9	4
F	CD	2	4
G	EF	1	6

- Le chemin critique est ABCEG et la durée totale du projet est de 20 jours.

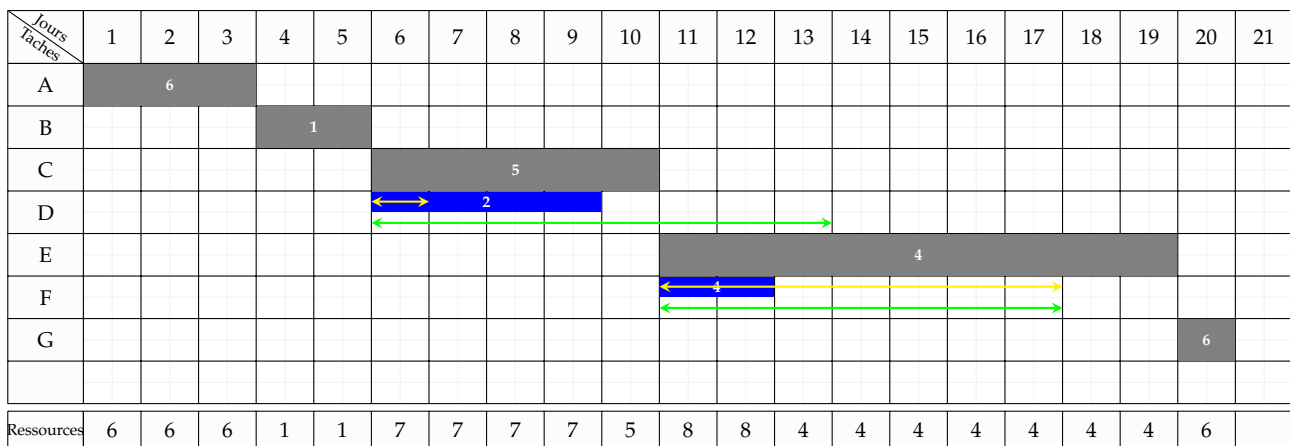
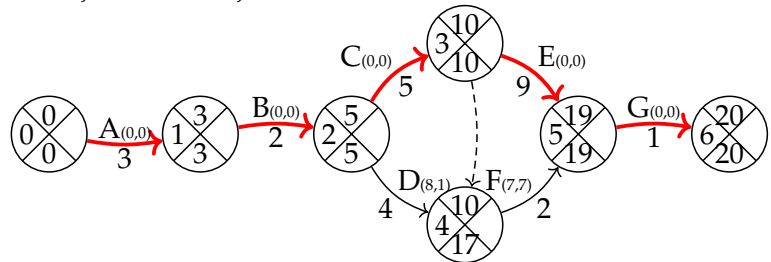


FIGURE 3.1 – Le diagramme GANTT avant l'optimisation des ressources.

En traçant les ressources nécessaires sous forme d'histogramme et une ligne pour montrer la limite des ressources. Le conflit de ressources se produit clairement aux jours 6, 7, 8, 9, 11 et 12.

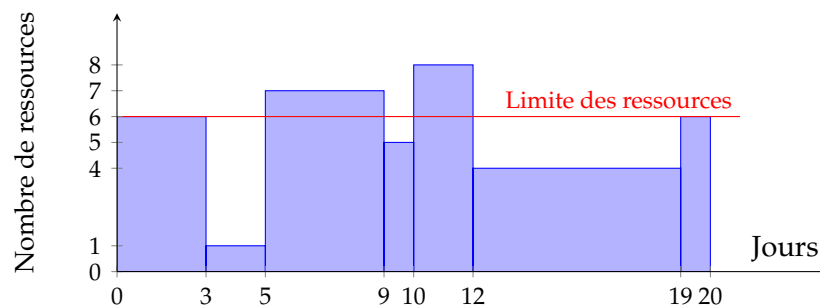


FIGURE 3.2 – L'histogramme des ressources nécessaires avant l'optimisation.

3.4.1 Nivellement des ressources

- Le nivellement des ressources vise à supprimer tous les conflits de ressources sans trop se soucier de l'allongement de la durée du projet.
- Le premier conflit se produit entre les tâches C et D. Pour éviter ce conflit, nous devons retarder la tâche C ou D. Puisque la tâche C est sur le chemin critique, il est prudent de retarder D jusqu'à ce que C se termine. Cela signifie que la tâche D doit être décalée pour démarrer le jour 11. Comme il y a une marge dans la tâche D, tout va bien.
- Mais, lorsque nous déplaçons une tâche, nous devons voir comment cela affecte les autres tâches qui lui succèdent. Dans ce cas, étant donné que F dépend de l'achèvement de D, elle doit être décalée.
- Maintenant, il y a un conflit aux jours 15 et 16 ; la tâche E a besoin de 4 ressources et la tâche F a également besoin de 4 ressources.
- Ainsi, nous devons retarder la tâche E ou F pour éviter ce conflit. nous retardons F jusqu'à la fin de E. Cela signifie que la tâche F ne peut commencer qu'au jour 20. Cependant, cela conduit également au déplacement de G puisque F est son prédécesseur.

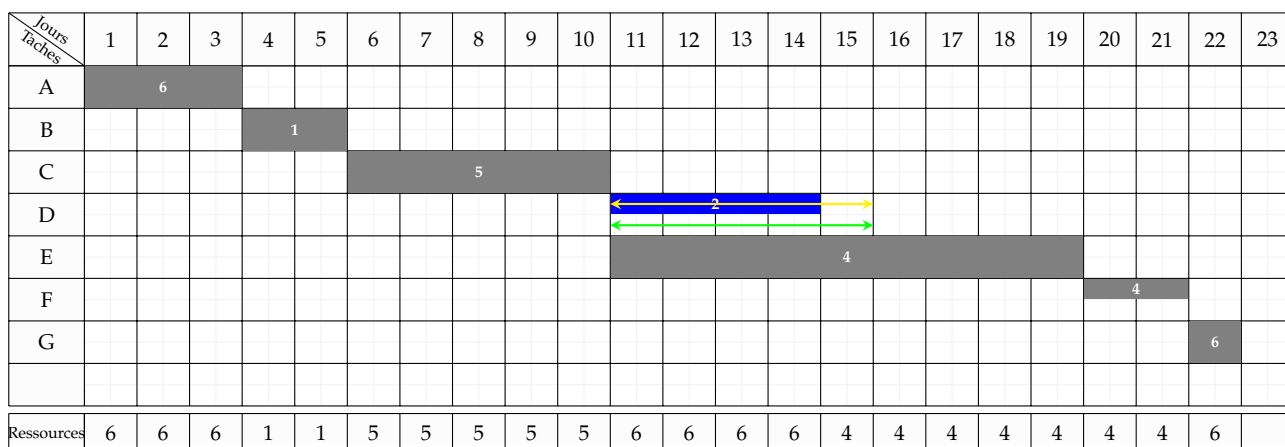


FIGURE 3.3 – Le diagramme GANTT après nivellement des ressources.

C'est bien. Tous les conflits de ressources sont supprimés. Mais le projet est retardé de 2 jours. La durée totale du projet est maintenant de 22 jours. Le chemin critique a également changé. Donc, le nouveau chemin critique est ABCEFG.

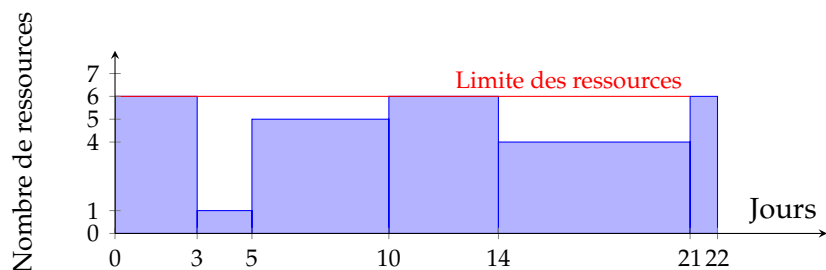


FIGURE 3.4 – L'histogramme des ressources nécessaires après nivellement.

3.4.2 Lissage des ressources

- Le lissage des ressources cherche à éliminer autant de conflits de ressources que possible sans retarder la durée totale du projet.
- Commençons à nouveau par ajuster le calendrier d’origine pour éviter les pics de ressources.
- La tâche D est décalée du jour 6 au jour 11. Cela supprime les pics de ressources du jour 6 au jour 9. Cependant, il existe toujours des conflits de ressources au jour 15 et au jour 16. Ce conflit ne peut pas être supprimé sans retarder la durée totale du projet. Ainsi, le lissage des ressources s’arrêtera ici.

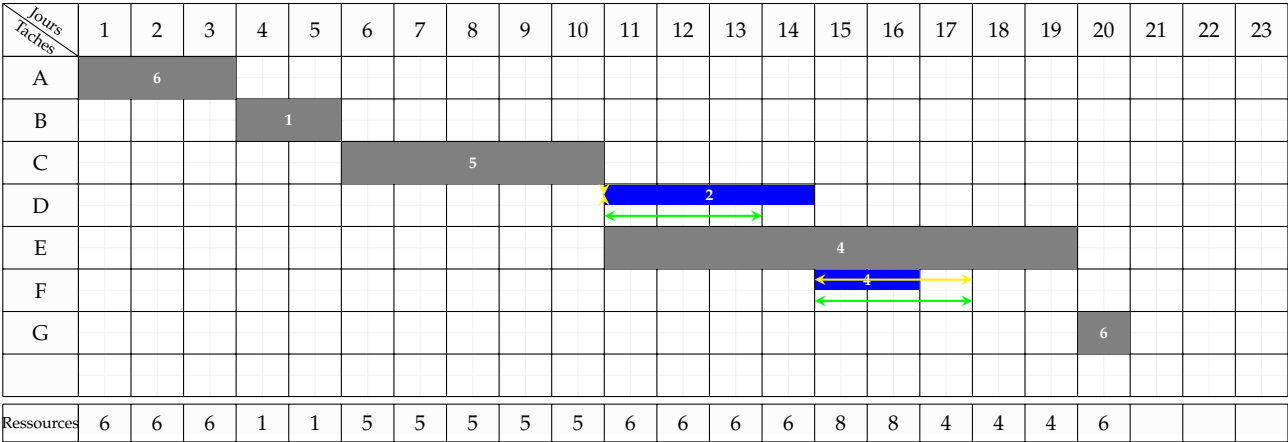


FIGURE 3.5 – Le diagramme GANTT après lissage des ressources.

Sur les 6 jours de conflit, le lissage des ressources a réussi à supprimer 4 jours de conflit. Cependant, si l’organisation souhaite s’en tenir au calendrier d’origine, elle doit apporter des ressources supplémentaires les jours 15 et 16.

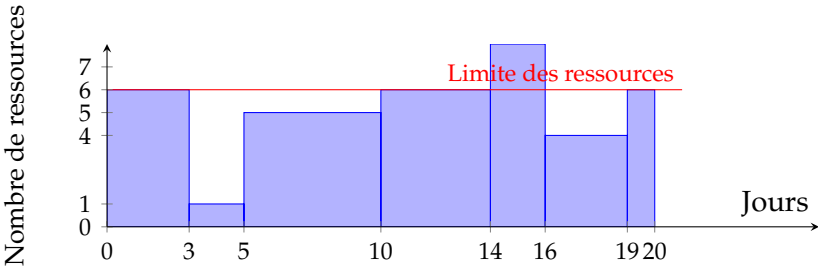


FIGURE 3.6 – L’histogramme des ressources nécessaires après lissage.

3.4.3 Comparaison entre nivellement et lissage

Nivellement des ressources	Lissage des ressources
Une technique de planification à ressources limitées c.-à-d. les ressources sont la principale contrainte.	Une technique de planification à durée limitée c.-à-d. la date de fin du projet est la contrainte principale.
Supprime tous les conflits de ressources.	Supprime les conflits de ressources en retardant les tâches dans leurs marges.
Utilisé lorsque les ressources sont sur-affectées	Utilisé lorsque les ressources sont utilisées de manière inégale.
Peut être appliqué aux tâches de chemin critique.	Les tâches de chemin critique sont intouchables.
La date de fin du projet peut changer.	La date de fin du projet ne change pas.
Le chemin critique change (augmente).	Le chemin critique ne change pas.

TABLE 3.1 – Comparaison entre nivellement et lissage des ressources.

Exemple 1

Prenons l'exemple représenté par le tableau suivant et supposons que le nombre des ressources soit limité à 10.

Code	Tâche	Durée	Tâches antérieures	Ressources
A	Définition du budget	4	Rien	5
B	Selection thème,date, lieu	3	A	5
C	Embauche traiteur	3	B	2
D	Annonce interne	3	B	4
E	Annonce de presse	4	D	6
F	Selection menu	2	C	7
G	Location des équipements	4	CE	3
H	Embauche personnel	4	G	4
I	Préparatifs	5	G	4
J	Evènement	1	IHF	10

— On commence par la construction de réseau PERT, puis le calcul des dates au plus tôt, des dates au plus tard, des marges libres et des marges totales. Ensuite la construction de diagramme GANTT et on termine par un lissage/nivellement à la limite 10.

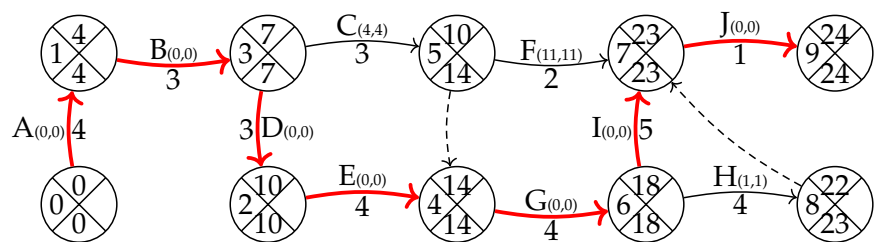


FIGURE 3.7 – Le réseau PERT.

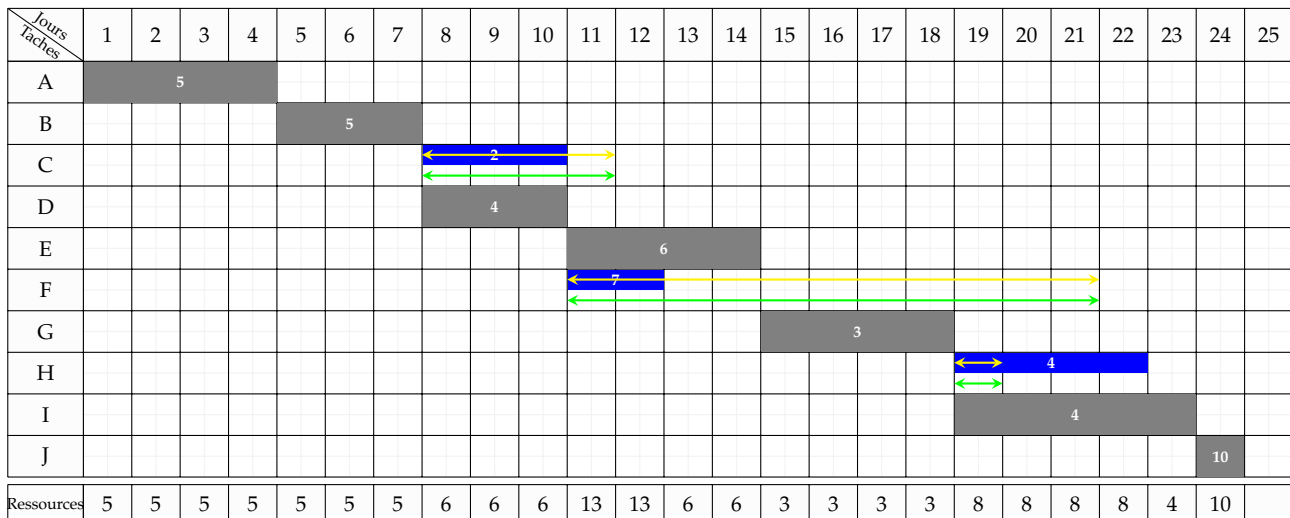


FIGURE 3.8 – Le diagramme GANTT avant l’optimisation des ressources.

L’histogramme des ressources nécessaires avant l’optimisation est le suivant :

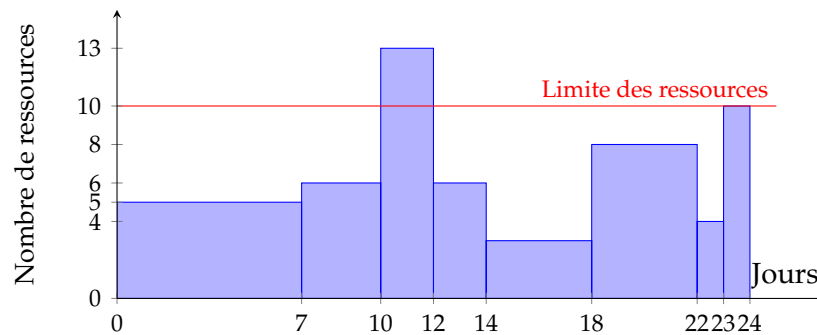


FIGURE 3.9 – L’histogramme des ressources nécessaires avant optimisation.

- En réalisant la somme des besoins en personnel sur le diagramme de GANTT, on s’aperçoit qu’il existe une période entre le 11^{me} et le 12^{me} jour où deux tâches (E,F) se déroulent simultanément et nécessitent plus de dix personnes.
- Pour éviter de dépasser la ressource maximum autorisée nous allons effectuer un retardement. Les deux tâches impliquées sont E et F avec E étant une tâche du chemin critique c-à-d n’ayant pas de marge. Nous pouvons donc décaler la tâche F.
- En décalant la tâche F de 4 journées (E et F ne sont plus simultanées) on ne dépasse plus la limite autorisée des ressources et on ne décale pas non plus la fin du projet, mais bien sûr on diminue la marge totale et libre de F.

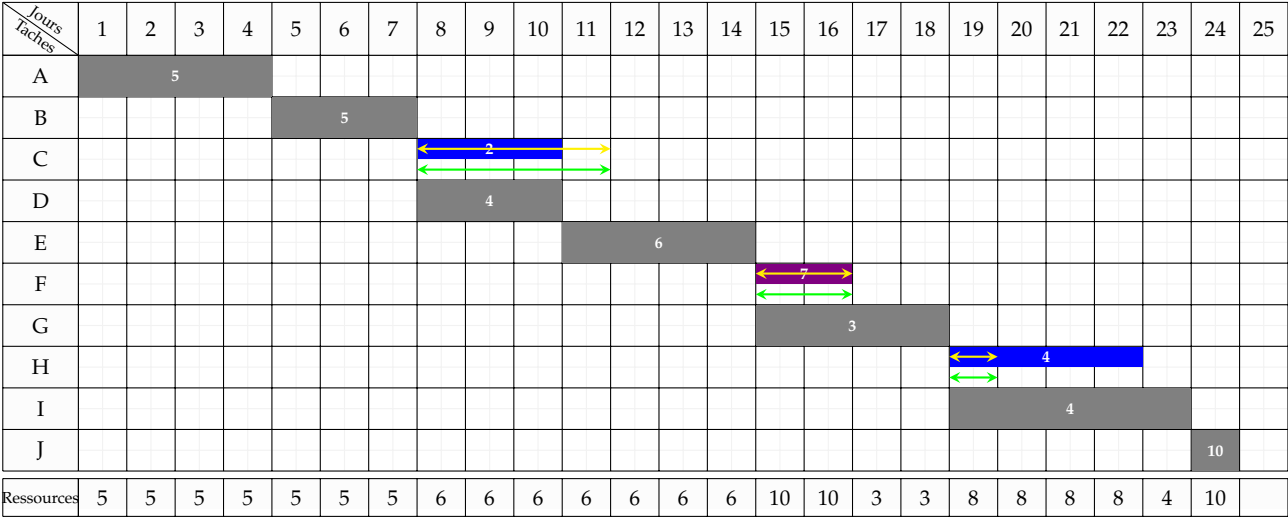


FIGURE 3.10 – Le diagramme GANTT après optimisation des ressources.

Dans cet exemple le résultat d’optimisation par nivellement des ressources est identique à celle obtenue par lissage des ressources.

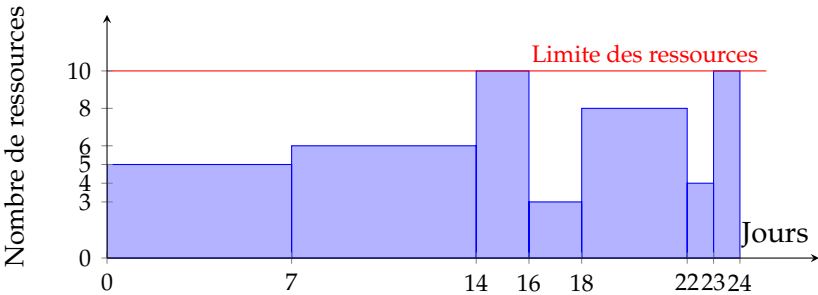


FIGURE 3.11 – L’histogramme des ressources nécessaires après optimisation.

Exemple 2

Une entreprise met à l’étude le lancement d’une nouvelle gamme de produits. Ce lancement nécessite la réalisation de tâches repérées par les lettres A à I et dont les caractéristiques sont les suivantes :

1. Réalisez le PERT potentiel tâches en calculant les dates au plus tard, au plus tôt, les marges libres et totales et en déterminant le chemin critique.

2. Réalisez le diagramme de GANTT au plus tôt relatif au PERT potentiel tâches.

3. Effectuez un lissage sachant qu’on ne peut utiliser au maximum que 6 ressources mais que les ressources sont polyvalentes et peuvent être affectées à n’importe quelle tâche.
- | Tâche | Durée | Tâches antérieures | Ressources |
|-------|-------|--------------------|------------|
| A | 5 | D | 2 |
| B | 2 | GH | 3 |
| C | 5 | B | 3 |
| D | 4 | Rien | 3 |
| E | 2 | GH | 3 |
| F | 4 | EI | 3 |
| G | 3 | Rien | 3 |
| H | 2 | D | 3 |
| I | 6 | A | 3 |

Corrigé

— On commence par la construction de réseau PERT, puis la construction de diagramme GANTT et on termine par un lissage à la limite 6.

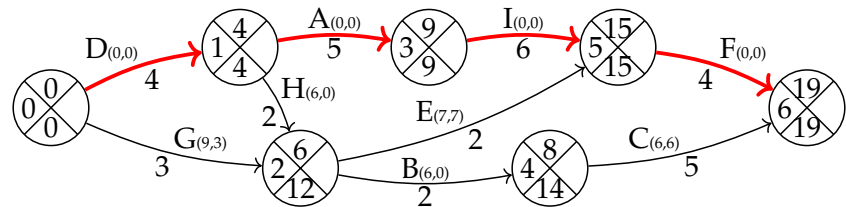


FIGURE 3.12 – Le réseau PERT.

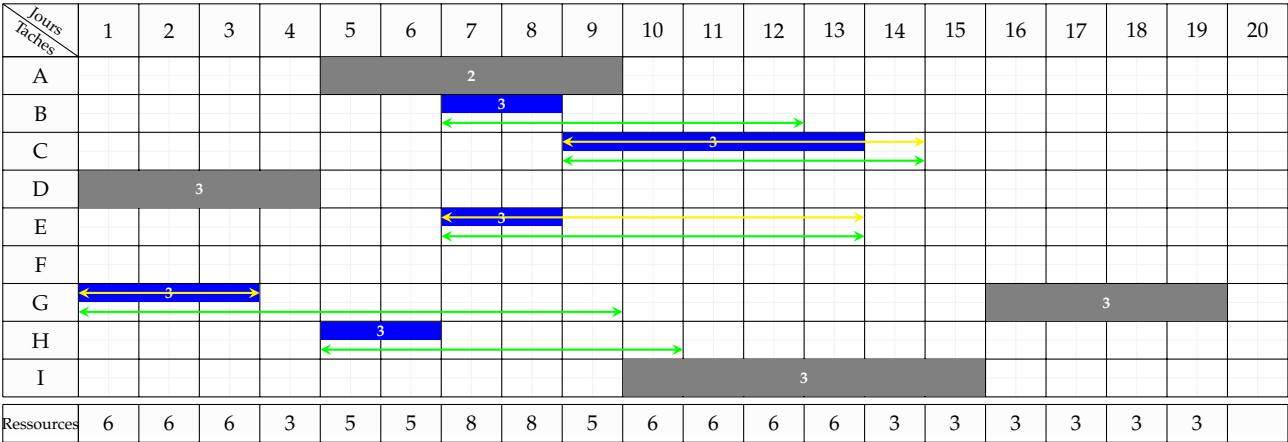


FIGURE 3.13 – Le diagramme GANTT avant lissage.

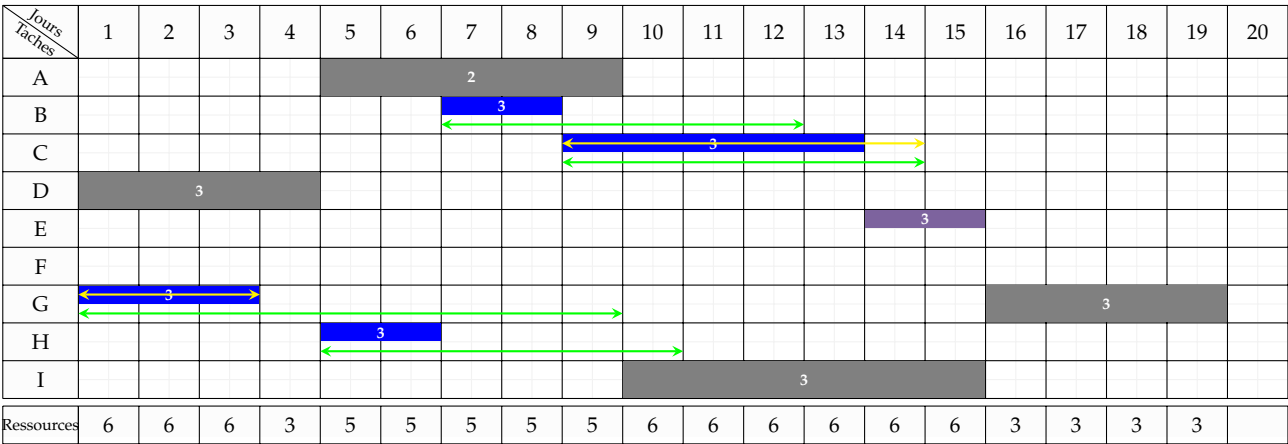


FIGURE 3.14 – Le diagramme GANTT après lissage.

Chapitre 4

Gestion de la qualité

4.1 Objectif

L'objectif de ce chapitre est d'introduire la gestion de la qualité du logiciel et les mesures du logiciel. Cela permettra de :

- Connaître et comprendre le processus de gestion de la qualité et les activités associées : l'assurance qualité, la planification de la qualité et le contrôle de la qualité.
- Comprendre l'importance des standards dans le processus de gestion de la qualité.
- Connaître Certaines métriques du logiciel.
- Comprendre comment les mesures peuvent aider à estimer certains attributs de qualité du logiciel.

4.2 Introduction

La qualité du logiciel est un concept complexe et qui n'est pas directement comparable avec la qualité dans le domaine de la production. Cela est dû à :

- La spécification vise à définir les caractéristiques du produit demandé par l'utilisateur (besoins fonctionnels) alors que l'organisation de développement peut être intéressée par certains besoins non fonctionnels et qui ne figurent pas parmi ces spécifications.
- La spécification de certaines caractéristiques non fonctionnelles (maintenabilité, par exemple) est difficile.
- La spécification d'un produit logiciel n'est pas toujours (ou n'est jamais) complète. La gestion de la qualité se base sur des standards qui encapsulent certaines bonnes pratiques.
- D'une manière générale, les bonnes pratiques peuvent être considérées comme des solutions (voire meilleures solutions) à un problème posé dans un contexte donné.
- Elles permettent donc d'indiquer ce qui a lieu de faire et ce qui a lieu d'éviter.
- La gestion de la qualité formalisée est particulièrement importante pour l'équipe développeur des systèmes complexes.
- Elle peut être structurée en trois activités :
 - Assurance qualité;
 - Planification de la qualité;
 - Contrôle de la qualité.
- Pour les petits systèmes, une approche informelle à la gestion de la qualité est adoptée à travers l'établissement d'une culture de qualité qui sera partagée par tous les membres de l'équipe.

4.3 Processus basé sur la qualité

- Le processus de gestion de la qualité a pour but de vérifier les livrables du projet avec les standards de l'organisation. La gestion de la qualité est affectée à une équipe indépendante de l'équipe de développement.
- La qualité du processus de développement affecte directement la qualité du produit. Mais, la relation entre la qualité du processus de développement et la qualité du produit est plus complexe. Il est difficile de mesurer les attributs de qualité du logiciel même après avoir exploité longuement le logiciel → **difficulté de savoir comment les caractéristiques du processus influent sur les attributs de qualité.**
- La gestion de la qualité d'un processus consiste à :
 - définir les standards : comment et quand les revues sont conduites.
 - contrôler le processus de développement afin de s'assurer que les standards sont bien suivis.
 - reporter le processus logiciel à la gestion de projet.

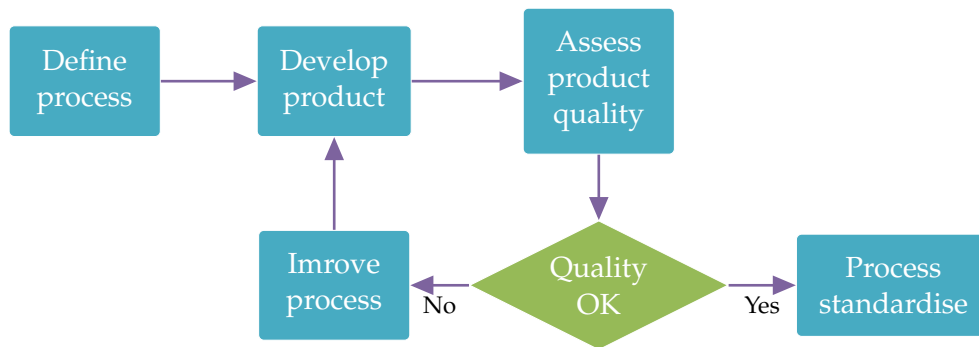


FIGURE 4.1 – La qualité du produit et la qualité du processus.

4.4 Assurance qualité et standards

- L'assurance qualité est le processus de définition complète de la qualité du logiciel et comment l'organisation de développement connaît le niveau de qualité requis d'un logiciel. Le processus d'assurance qualité consiste à définir ou à sélectionner les standards qui seront appliqués aux processus de développement ou au produit logiciel.
- Le processus d'assurance qualité regroupe deux standards :
 - **Standards du produit** : applicables aux produits logiciels, inclut les standards des documents (structure des documents des requis)
 - **Standards de processus** : définition du processus à suivre durant le développement du logiciel, inclut la définition des besoins, la description des documents générés durant ces processus.
 - **Les standards du logiciel** : sont importants car ils sont basées sur la connaissance des meilleures pratiques d'une organisation; Fournissent un Framework pour implémenter un processus d'assurance qualité; Assurent la continuité des travaux car les membres se partagent les mêmes pratiques.

Exemples de standards :

1. **Standards du produit :**

- forme de revues de conception.
- structure du document des requis (besoins ou exigences).
- format d'un plan de projet sont compatibles.

2. **Standards du processus :**

- processus des livrables d'une version.
- processus de contrôle de changement.

3. **Les standards de documentation :** La documentation est l'unique moyen tangible pour représenter le logiciel et le processus logiciel. Ils sont répartis en trois types :

- (a) *Standards du processus de documentation* : définissent le processus suivi pour produire une documentation
- (b) *Standards des documents* : définissent la structure et la présentation des documents.
- (c) *Standards d'échange de documentation* : assurent que toutes les copies électroniques des documents sont compatibles.

4.5 Planification de la qualité

- C'est le processus de développement du plan de qualité d'un projet. Le plan de qualité formule les qualités du logiciel et comment elles sont évaluées.
- Le plan de qualité diffère selon la taille et le type du système à développer. Il doit être le plus court possible.
- Le plan de qualité doit définir les attributs de qualité les plus importants d'un logiciel (maintenabilité, robustesse). Il doit aussi inclure le processus d'estimation de la qualité.

Exemple de structure d'un plan de qualité proposé par Humphrey :

- **Introduction du produit** : description du produit, le marché visé, les prévisions de qualité du produit.
- **Planning du produit** : dates de livraison critiques et les responsabilités ainsi que les plannings de distribution et de service du produit.
- **Description du processus** : processus de développement et de service qui seront utilisés pour le développement du produit et pour sa gestion.
- **Objectifs de qualité** : les objectifs et les plans de qualité du produit incluant l'identification et la justification des attributs critiques de qualité du produit.
- **Risques et la gestion des risques** : Les risques importants qui peuvent affecter la qualité du produit et les actions à entreprendre pour traiter ces risques.

4.6 Contrôle de qualité

- Le contrôle de la qualité implique le suivi du processus de développement du logiciel afin de s'assurer que les procédures de qualité et les standards sont bien suivis. Deux approches complémentaires peuvent être utilisées pour vérifier la qualité des livrables.
 - 1. **Les revues de qualité** : le logiciel, sa documentation et le processus utilisé pour produire ce logiciel sont révisés par une équipe afin de vérifier le suivi des standards du projet et la conformité du logiciel et de sa documentation aux standards.

2. **L'évaluation automatisée du logiciel** : le logiciel et ses documents sont traités par certains programmes et comparés aux standards mesurer certains attributs du logiciel et les comparer avec certains niveaux désirables.
3. **Les revues de qualité** : sont couteuses et consomment du temps. Pour accélérer le processus de revues, il est nécessaire d'utiliser des outils dans le but d'évaluer la qualité du logiciel ou du processus.

TABLE 4.1 – Types des revues.

textbfType de revue	But principal
Inspections de programmes et de conception	Détecter les erreurs au niveau de la conception ou du code
Revue de progression	Fournir l'information sur le progrès du projet : revues du produit et du processus concernant le coût, la programmation.

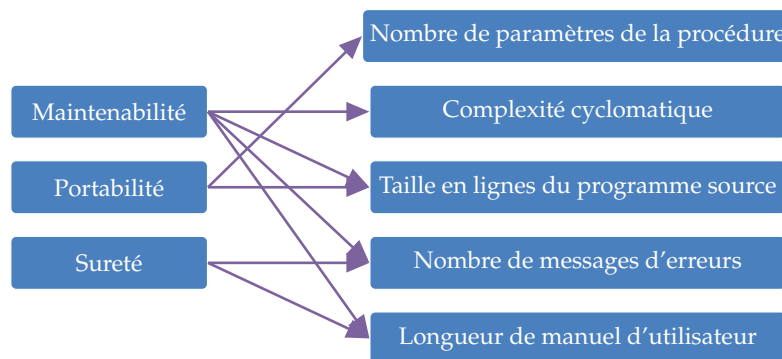
4.7 Mesures du logiciel

Les mesures du produit logiciel peuvent être utilisées pour :

- **Faire des prédictions sur le système** : dériver une estimation concernant certains attributs du système (nombre d'erreurs dans le système)
- **Identifier les composants défectueux** : identifier certains composants qui violent certaines normes (composants ayant une complexité importante).

Certains attributs externes ne sont pas directement mesurables (maintenabilité, compréhensibilité). Ils sont affectés par plusieurs attributs internes (taille) → il faut donc établir des relations entre attributs internes et externes ;

Exemple



- Cependant, la nature de la relation n'est pas bien définie. Pour que les attributs internes soient des prédicateurs utiles , il faut :

1. Mesurer précisément les attributs internes ;
2. Définir la relation entre ce qui a été mesuré et l'attribut externe d'intérêt ;
3. Établir la formule ou le modèle exprimant la relation déjà définie.

— **La fiabilité d'un logiciel :**

Elle se définit par la probabilité qu'un logiciel ne pas connaître des défaillances pendant une période donnée. Elle est notée $F(n)$ où n est le nombre d'unités temporelles.

Exemple : Si l'unité choisie est le jour, $F(1)$ désigne la probabilité que le programme ne connaisse pas de défaillances pendant une journée d'où la probabilité qu'un programme connaisse une défaillance pendant une durée donnée correspond à : $D(n) = 1 - F(n)$

4.8 Processus de mesure

- Le processus de mesure représente une partie importante du processus de qualité.
- Chaque composant du logiciel est analysé séparément et les valeurs obtenues sont comparées aux données collectées à partir des projets précédents.
- Les mesures anormales servent à déterminer les composants présentant des problèmes de qualité.
- Le processus de mesure du produit est illustré par :



4.9 Métriques du logiciel

Les métriques du logiciel se divisent en deux catégories :

1. **Les métriques statiques** : collectées à partir des mesures établis sur la représentation du système (conception, programme, documentation). Elles permettent de calculer la complexité, la taille d'un programme, des prédicateurs pour certains attributs de qualité (maintenabilité, compréhensibilité)
2. **Les métriques dynamiques** : collectées à partir des mesures effectuées sur un programme en exécution. Elles servent à estimer la sureté et l'efficacité d'un programme.

Exemple :

- *Métriques génériques* : Fan-in / Fan-out, complexité cyclomatique, la longueur du code, la longueur des identificateurs.
- *Métriques spécifiques (orientées objet)* : profondeur de l'arbre d'héritage, méthode fan-in / fan-out.

4.10 Conclusion

- La gestion de la qualité du logiciel a pour but de réduire le nombre de défauts.
- Les activités de gestion de la qualité incluent l'assurance qualité, la planification de la qualité et le contrôle de la qualité.
- Les standards du logiciel sont importants à l'assurance qualité car ils représentent une identification des bonnes pratiques.

- Le processus de contrôle de la qualité consiste à vérifier la conformité de la qualité du produit et du processus avec les standards.
- Les revues représentent les techniques les plus utilisées d'estimation de la qualité.
- Les métriques de qualité du produit permettent de déterminer les composants anormaux. Ces composants doivent être analysés en détail.
- Il n'y a pas de métriques du logiciel applicables à tout logiciel. Les organisations doivent sélectionner les métriques et analyser les mesures sur la base des connaissances locales.

Chapitre 5

Inspection du logiciel

5.1 Objectif

L'objectif de ce chapitre est d'introduire la notion d'inspection du logiciel dans le but d'améliorer la qualité du logiciel. Ce chapitre permet de :

- Fournir les avantages des inspections du logiciel.
- Expliciter ce qu'est une inspection et plus particulièrement l'inspection de Fagan.
- Expliciter l'inspection des documents du logiciel.

5.2 Introduction

- Les programmes, dont on dispose, présentent souvent certaines erreurs telles que variables inutilisées, tests booléens incorrects, commentaires qui ne correspondent pas au code, etc. Ce qui nécessite souvent des correction/améliorations. Les inspections du logiciel, les critiques, sont des techniques prouvées pour réduire le nombre de défauts dans un programme.
- Dans une organisation de développement des logiciels, une activité d'inspection devrait faire partie du processus de développement du logiciel.
- L'inspection de Fagan correspond à un processus structuré destiné à trouver des défauts dans des documents du logiciel tels que le code, la (les) conception (s) etc. et évidemment pendant les diverses phases du procédé de développement du logiciel.

5.3 Avantages des inspections

Les inspections présentent certains avantages tels que :

1. **Le coût de la correction d'un défaut augmente** considérablement s'il est rencontré tardivement, dans le cycle de développement du logiciel. Pour cela, s'il est possible de détecter et De corriger les erreurs plus tôt, cela permettra d'économiser du temps ainsi que de l'argent tout le long du cycle de vie du logiciel.
2. **L'inspection ne porte pas sur le code source uniquement** mais, tout artefact lisible produit pendant le développement du logiciel peut être consulté à savoir les spécifications des exigences, des documents de conception et de modèles, les plans de test, la documentation du système, etc.

3. **Les inspections permettent de raccourcir les délais de livraison**, en réduisant le temps passé dans les phases de test. Une meilleure qualité du produit fini permet aussi de réduire le temps de maintenance. Ainsi, ce dernier peut être utilisé pour le développement de nouveaux projets.
4. Dans une organisation à un niveau élevé de maturité des processus logiciels, les données recueillies lors d'inspections peuvent être utilisées pour améliorer davantage le processus. Les données provenant des rapports de synthèse d'inspection peuvent être utilisées pour identifier les types les plus communs ou les plus coûteux de bugs, de déterminer leurs causes profondes, et changer la façon dont le travail est effectué afin d'éviter ces types d'erreurs.
5. De plus, Il ya quelques avantages secondaires à savoir :
 - Tous les participants peuvent apprendre de chaque inspection. Les inspections offrent une occasion de voir comment les autres membres de l'équipe travaillent. Une inspection fournit un forum efficace pour les membres de l'équipe les moins expérimentés d'en apprendre beaucoup tout en continuant à faire des contributions utiles.
 - Les inspections ont tendance à révéler différents types d'erreurs qui ne sont pas détectées par les activités de test. La combinaison d'inspections formelles et structurées et des tests systématiques fournit un outil puissant pour créer des programmes sans défaut.

5.4 Inspections

- **L'inspection formelle** (inspection de Fagan, par exemple) : correspond au fait que quelqu'un d'autre que le créateur d'un produit logiciel examine le produit avec l'intention spécifique de trouver des erreurs. C'est une lecture critique d'un document (spécification, conception, code, plan d'intégration...); Elle est destinée à améliorer la qualité d'un document.
- D'une manière générale, l'inspection est faite par une équipe indépendante de l'équipe de développement du projet. Pour qu'elle puisse être bénéfique, une inspection doit donner lieu à la rédaction de fiches de défauts avec une échelle de gravité et la définition des responsabilités concernant la correction des défauts. L'inspection de Fagan est une méthode de revue de groupe employée pour évaluer le rendement d'un processus donné.
- Les exemples des activités pour lesquelles l'inspection de Fagan peut être employée sont : la **spécifications**, l'**architecture** de logiciel/système d'information, la **programmation** et l'**essai de logiciel** (par exemple en créant des jeux d'essai) «Un défaut est un exemple dans lequel une condition n'est pas satisfaite».
- **Inspection de logiciel** : en cours d'inspection les défauts qui sont trouvés sont classés en deux catégories :
 1. **Les défauts principaux** : concernent une fonctionnalité incorrecte ou même absente peuvent être classifiés en tant que défauts principaux : le logiciel ne fonctionnera pas correctement quand ces défauts ne sont pas résolus.
 2. **Les défauts mineurs** : elles ne menacent pas le fonctionnement correct du logiciel, mais sont la plupart du temps de petites erreurs comme des erreurs d'appellation dans les documents ou erreurs du type positionnement incorrect des commandes dans une interface de programme.

5.5 Inspection de logiciel

Les inspections sont à la base des décisions prises en revues. Une revue est une réunion permettant de valider une des phases du cycle de vie. On distingue :

- **Les revues des produits** : état d'un projet sous ses différents aspects : Techniques, Financiers, commerciaux, Calendrier, ...
- **Les revues techniques** : elles permettent de fournir au marketing et à l'unité de développement une évaluation des aspects techniques du projet et des coûts de réalisation .
- **Les réunions de décision** : elles valident le passage à la phase suivante et font bien souvent suite à l'une des deux précédentes.

Le processus d'inspection formelle de logiciels développé par Michael Fagan; nécessite une équipe d'inspection composée de 3 à 8 membres ayant les rôles suivants :

1. **Le modérateur** : conduit l'inspection, les horaires des réunions, le contrôle des réunions, les résultats des rapports d'inspection et le suivi des questions à reprendre. Les modérateurs devraient être formés sur la manière de mener les inspections, y compris la façon de s'entretenir avec les participants ayant de fortes compétences techniques.
2. **L'auteur** : créé ou maintient le produit du travail à inspecter. L'auteur ne peut répondre aux questions posées sur le produit pendant l'inspection, et il examine aussi des défauts. L'auteur ne peut pas servir en tant que modérateur, lecteur ou secrétaire.
3. **Le lecteur** : décrit les sections du produit du travail à l'équipe lors de l'inspection. Il peut paraphraser ce qui se passe dans le produit, tel que la description d'une section de code, mais il n'a pas l'habitude de lire mot à mot le produit.
4. **Le secrétaire** : classe et enregistre les défauts et les questions soulevées lors de l'inspection. Le modérateur pourrait remplir ce rôle dans une petite équipe d'inspection.
5. **L'inspecteur** : tente de trouver des erreurs dans le produit. Tous les participants sont réellement considérés comme des inspecteurs, en plus de toute autre responsabilité. Les inspecteurs comprennent : la personne qui a créé la spécification du produit du travail inspecté (une inspection de code), ceux chargés de l'application, des tests, ou du maintien du produit, un représentant d'assurance qualité pour agir selon les normes.

5.6 Le processus d'inspection

Le processus d'inspection se compose d'activités suivantes (d'après Fagan) dont chacune a un objectif spécifique :

1. **Planification** : Le modérateur sélectionne l'équipe d'inspection, prépare les matériaux qui doivent être inspectés, les distribue ainsi que d'autres documents pertinents à l'équipe d'inspection à l'avance. Le matériel devrait être distribué au moins deux ou trois jours avant l'inspection.
2. **Vue d'ensemble** : Cette réunion donne à l'auteur l'occasion de décrire les caractéristiques importantes du produit à l'équipe d'inspection. Il peut ne pas la faire si cette information est déjà connue par les autres participants.
3. **Préparation** : Chaque participant est responsable de l'examen des artefacts de travail avant la réunion de l'inspection réelle, en notant tous les défauts trouvés ou questions qui seront soulevées. Peut-être 75% des erreurs constatées lors des inspections sont identifiées lors de l'étape de préparation. Le produit doit être comparé à ses prédécesseurs (spécification) des documents pour évaluer l'exhaustivité et l'exactitude.

4. **Reprise** : L'auteur est chargé de résoudre toutes les questions soulevées pendant l'inspection. Cela ne signifie pas nécessairement faire tout changement qui a été suggéré, mais une décision explicite doit être faite sur la manière dont chaque défaut sera traité.
5. **Réunion d'inspection** : Lors de cette session, l'équipe se réunit et est dirigé par le produit du travail par le modérateur et le lecteur. Si le modérateur détermine au début de la réunion que le manque de temps a été consacré à la préparation par les participants, la réunion devrait être reportée. Pendant la discussion, tous les inspecteurs peuvent signaler des défauts ou soulever d'autres questions, qui sont documentés sur un formulaire par le secrétaire. La réunion ne devrait pas durer plus de deux heures. Lors de sa conclusion, le groupe s'engage sur une évaluation du produit : Accepté comme tel ; accepté avec révisions mineures ; révisions majeures nécessaires et une deuxième inspection requise ; ou reconstruire le produit.
6. **Suivi** : Pour vérifier que la reprise nécessaire a été effectuée correctement, le modérateur est responsable du suivi avec l'auteur. Si une fraction importante (soit, 10%) du produit du travail a été modifiée, une inspection supplémentaire peut être nécessaire.
Ce travail semble être important et même fastidieux, mais l'expérience a montré que ce processus d'inspection formelle est le moyen le plus efficace permettant de trouver des défauts dans un produit logiciel.

5.7 Inspection des documents

- Le document est l'élément d'examen. Il ya trois aspects à cette tâche : les défauts enregistrés lors de la réunion d'inspection, collecte de données auprès de multiples inspections, et l'analyse des tendances d'un défaut pour évaluer l'efficacité d'inspection et d'identifier la façon d'améliorer le processus de développement logiciel afin de prévenir les types courants de défauts.
- Chaque erreur trouvée est décrite avec suffisamment de détails afin de permettre à l'auteur de se référer à cette liste (de détails) au cours de l'étape de reprise et de comprendre le problème qui a été soulevé. Les références à des numéros de ligne où les erreurs ont été trouvées sont importantes pour des défauts particuliers, mais des observations plus générales sur les violations des normes ou des suggestions de style peuvent s'appliquer à l'ensemble du produit.
- La liste des défauts d'une inspection devrait aboutir à un rapport de synthèse avec un comptage des défauts de chaque catégorie. Le but est de permettre aux gestionnaires de savoir comment le projet avance et de chercher des domaines où des améliorations doivent être apportées. Le modérateur est habituellement responsable de la préparation de ces rapports post-inspection.
- Un processus d'inspection efficace permet à une organisation de comparer les données provenant de multiples inspections afin de mieux comprendre la qualité à la fois du processus d'examen et les produits en cours de révision.
- L'objectif ultime est d'avoir une base de données d'inspection afin que des conclusions quantitatives puissent être tirées des tendances des défauts ainsi que des informations sur les processus d'inspection.

5.8 Conclusion

- L'inspection du logiciel est considérée comme une méthode de test statique permettant de vérifier que le logiciel répond aux besoins.

- La mise en œuvre d’inspection du logiciel est importante car elle permet d’aboutir à un processus de développement logiciel plus mature. L’amélioration continue des processus mène à l’amélioration simultanée de la qualité du produit logiciel et de la productivité et les inspections peuvent y jouer un rôle majeur.
- L’inspection permet de détecter un taux de défauts entre 60% et 90%.
- Un important avantage à long terme d’un programme d’inspection est l’idée qu’il peut fournir des types de défauts créés et des changements de processus dans le but de les prévenir.
- Enfin, la qualité de l’inspection a un effet direct sur la qualité du produit.

Chapitre 6

Estimation du coût du logiciel

6.1 Objectif

L'objectif de ce chapitre est d'introduire les techniques d'estimation du coût et de l'effort nécessaires à la production d'un logiciel. Il s'agit de

- Comprendre que le coût du logiciel n'est pas directement lié au coût du développement.
- Introduire l'estimation de la productivité du logiciel.
- Introduire les techniques d'estimation du coût.
- Introduire le modèle d'estimation du coût COCOMO.
- Introduire la notion des points de fonction et de la valeur acquise.

6.2 Introduction

- Dans la planification d'un projet logiciel, l'estimation consiste à déterminer :
 - L'effort nécessaire pour compléter chaque activité.
 - Le temps nécessaire pour compléter chaque activité.
 - Le coût total pour chaque activité.
- L'estimation du coût est effectuée en même temps que la programmation du projet.
- Trois paramètres interviennent dans le calcul du coût d'un projet de développement du logiciel :
 1. coût du matériel et du logiciel ainsi que le coût de la maintenance ;
 2. coût de formation et de déplacement ;
 3. coût de l'effort.
- Pour la majorité des projets, le coût de l'effort est le plus dominant relativement aux différents coûts qui s'y ajoutent tels que le coût des charges des bureaux, le coût de l'équipe support (administrateur), le coût de communication, etc.

6.3 Productivité du logiciel

- La productivité correspond aux nombres d'unités produites par le nombre de heures-hommes (ou mois-hommes) nécessaires à leur production.
- L'estimation de la productivité des ingénieurs est une tâche difficile pour le chef de projet.
- L'estimation de la productivité est basée sur les mesures des attributs du logiciel par l'effort total nécessaire au développement. On dispose de **deux types** de métriques :

1. **Métriques liées à la taille** : c'est la métrique la plus utilisée. La taille peut être mesurée en :
 - Nombre de lignes du code source livré;
 - Nombre d'instructions du code objet livré;
 - Nombre de pages de documentation du système.
2. **Métriques liées aux fonctions** : liées aux fonctionnalités générales du logiciel livré. La productivité est exprimée en termes de total des fonctionnalités utiles produites pendant un certain temps donné. Les points fonctions et les points objets sont les métriques les plus utilisés. La détermination du nombre de points de fonction est donnée dans la section suivante. Les points objet représentent une alternative des points de fonction qui peuvent être utilisés avec les langages de programmation des bases de données.

Exemple :

Le nombre d'écran séparé qui est affiché représente un point objet ayant comme coefficient : Simple écran → 1, modérément complexe → 2, très complexe → 3.

6.4 L'estimation du coût d'un logiciel

- L'estimation du coût du logiciel consiste à déterminer les ressources nécessaires à sa réalisation. Cette estimation est faite généralement en **MH (Mois-Hommes)**.
- Il existe deux approches très différentes pour estimer le coût d'un logiciel.
 1. **La ancienne** : est appelée estimation **LOC (Lines Of Code)** car elle s'appuie sur l'estimation du nombre de lignes de code qui devront être écrites pour le projet.
 2. **La moderne** : s'appuie sur le nombre de points de fonction dans la description du projet.
- La première étape dans l'estimation fondée sur le nombre de ligne de code consiste à estimer leur nombre dans le code du projet terminé.
- Cette estimation peut s'appuyer sur la taille des précédents projets, celle d'un produit concurrent ou bien diviser le projet en plusieurs parties, plus simples à estimer.
- La démarche standard consiste, pour chaque partie p_i du projet, à estimer sa taille maximale max_i , sa taille minimale min_i et une approximation au plus juste j_i .
- L'estimation de la taille des différentes parties est la suivante :

$$Taille_{estimee} = \sum ((Taille_{min} + 4 \times Approximation + Taille_{max}) / 6) \text{ LOC}$$

Exemple

L'équipe WRT(West River Telecom) a décomposé son projet en 7 parties, dont l'estimation des tailles respectives figure dans le tableau suivant :

<i>Partie</i>	<i>Taille_{min}</i>	<i>Approximation</i>	<i>Taille_{max}</i>	<i>Taille_{estimee}</i>
P1	20	30	50	31.67 LOC
P2	10	15	25	15.83 LOC
P3	25	30	45	31.67 LOC
P4	30	35	40	35.00 LOC
P5	15	20	25	20.00 LOC
P6	10	12	14	12.00 LOC
P7	20	22	25	22.17 LOC

6.4.1 Fondé sur des lignes de code

L'approche LOC repose sur une formule qui utilise des données historiques. La version la plus simple comprend trois paramètres : $Cout = a \times KLOC^b + c$

- a : le coût marginal par KLOC (Kilo LOC), c'est-à-dire le coût de chaque millier de lignes de code supplémentaires.
- b : est un exposant qui reflète le caractère non linéaire de cette relation. Selon des études menées sur ce sujet, la valeur de b n'est pas bien définie. Si $b > 1$ le coût du KLOC augmente avec la taille du projet → pas d'économie d'échelle. Si $b < 1$ il y a économie d'échelle.
- c : représente le coût fixe du projet, sa valeur peut être positive ou nulle.

6.4.2 Le modèle COCOMO

- COCOMO (Constructive Cost Model) est la formule classique d'estimation du coût fondée sur le nombre de lignes de code.
- Introduite en 1981 par Barry Boehm, il utilise le millier d'instructions source livrées (KDSI : Kilo Delivered Source Instruction) comme unité de taille ce qui est équivalent au KLOC et Mois-Homme comme unité de coût.
- Ce modèle existe en trois versions : **simple**, **intermédiaire** et **détaillé**.
- Trois types de projet ont été identifiés :
 1. **Projets de mode organique** : ces projets sont réalisés par une équipe relativement petite, dans un environnement familier et dans un domaine d'application connu de l'équipe.
 2. **Projets de mode semi-détaché** : ce mode représente un intermédiaire entre le mode organique et le mode embarqué. L'équipe peut être composée de programmeurs de divers niveaux d'expérience. Les membres de l'équipe ont une expérience limitée.
 3. **Les projets de mode embarqué** : la caractéristique principale d'un tel projet est que le système doit fonctionner sous des contraintes particulièrement fortes. Le système à développer est une partie d'un système complexe et fortement connecté de matériels, de logiciels, de normes et de procédures opérationnelles.
- Pour chaque catégorie, il attribue différentes valeurs aux paramètres de la formule :
 1. Mode organique : la charge = $MH = 2.4 \times KDSI^{1.05}$
 2. Mode semi-détaché : la charge = $MH = 3.0 \times KDSI^{1.12}$
 3. Mode embarqué : la charge = $MH = 3.6 \times KDSI^{1.20}$

- Boehm a également proposé des formules pour calculer **le temps de développement** ($TDEV$) standard en fonction des ressources de personnel disponible et du type de projet :
 1. Mode organique : la durée = $TDEV = 2.5 \times MH^{0.38}$
 2. Mode semi-détaché : la durée = $TDEV = 2.5 \times MH^{0.35}$
 3. Mode embarqué : la durée = $TDEV = 2.5 \times MH^{0.32}$
- La taille moyenne de l'équipe : $Taille = MH/TDEV$ hommes.
- La catégorie du projet dépend de la taille t du logiciel, si :
 - $t < 50 KDSI$: le projet est organique (simple);
 - $50 KDSI \leq t \leq 300 KDSI$: le projet est semi-détaché (moyen);
 - $300 KDSI < t$: le projet est embarqué (complexe).
- Relativement au modèle COCOMO simple, vu ci-dessus, le modèle intermédiaire prend en considération d'autres facteurs, autre que la taille et le type de projet. Les facteurs introduits sont au nombre de 15 répartis en 4 classes :
 1. **Les attributs du produit** : sont fiabilité requise du logiciel (FIAB), taille de la base de données (DONN), complexité du produit (CPLX).
 2. **Les attributs de l'environnement matériel et logiciel** : sont contraintes de temps d'exécution (TEMP), contraintes d'espace mémoire (ESPA), volatilité de la machine virtuelle (VIRT), contraintes du système de développement (CSYS).
 3. **Les attributs du personnel** : sont aptitude à l'analyse (APTA), l'expérience dans le domaine d'application (EXPA), expérience de la machine virtuelle (EXPV), aptitude à la programmation (APTP), expérience du langage de programmation (EXPL).
 4. **Les attributs du projet** : sont méthodes de programmation modernes (PMOD) outils logiciels (OLOG), durée requise du développement (DREQ).

Les multiplicateurs associés à ces attributs sont décrits dans le tableau suivant avec **TB** signifie : très bas, **B** : bas, **M** : moyen, **E** : élevé, **TE** : très élevé et **TTE** : très très élevé.

Attributs	TB	B	M	E	TE	TTE
FIAB	0.75	0.88	1.00	1.15	1.4	
DONN		0.94	1.00	1.08	1.16	
PLX	0.70	0.85	1.00	1.15	1.30	1.65
TEMP		1.00	1.11	1.30	1.66	
ESPA		1.00	1.06	1.21	1.56	
VIRT		0.87	1.00	1.15	1.30	
CSYS		0.87	1.00	1.07	1.15	
APTA	1.46	1.19	1.00	0.86	0.71	
EXPA	1.29	1.13	1.00	0.91	0.82	
APTP	1.42	1.17	1.00	0.86	0.70	
EXPV	1.21	1.10	1.00	0.90		
EXPL	1.14	1.07	1.00	0.95		
PMOD	1.24	1.10	1.00	0.91	0.82	
OLOG	1.24	1.10	1.00	0.91	0.83	
DREQ	1.23	1.08	1.00	1.04	1.10	

6.4.3 Analyse des points de fonction

- Le principe des points de fonction consiste à identifier et à quantifier les fonctionnalités réclamées par le projet, en comptant les éléments extérieurs qui devront être traités. Les éléments les plus couramment pris en compte sont les suivants :
 1. **Les interrogations** : ce sont les paires requête-réponse qui ne modifient pas les données internes. Par exemple : une requête pour connaître l'adresse d'un employé.
 2. **Les entrées** : ce sont les données fournies au programme. Par exemple : l'entrée du nom, du prénom d'un employé.
 3. **Les sorties** : les données qu'affiche le programme. Par exemple : afficher une liste des employés de l'entreprise, l'ensemble de ces lignes sera compté comme un élément unique.
 4. **Les fichiers internes** : ce sont les fichiers que le client comprend que le système doit gérer. Par exemple : un fichier qui comprend 1000 entrées de données Personnelles sera compté comme un seul fichier mais, s'il contient d'une part les données personnelles et d'autre part des données relatives au département, il sera compté comme deux fichiers différents.
 5. **Les interfaces externes** : ce sont les données partagées avec d'autres programmes. Par exemple : le fichier personnel qui peut être utilisé par le service paie pour calculer les payes sera considéré comme une interface dans les deux systèmes.

Les points de fonction recensés sont classés en trois catégories : simples, moyens ou complexes.

Les coefficients sont ensuite appliqués aux différents éléments et la somme de ces produits s'appelle les points de fonctions bruts :

Type	Simple	Moyen	Complexe
Sorties	4	5	7
Interrogations	3	4	6
Entrées	3	4	6
Fichiers	7	10	15
Interfaces	5	7	10

Cas où le point de fonction correspond aux interrogations avec deux catégories simples et moyennes.

Type	Simple	Moyen	Complexe	Total
Interrogations	Requête par nom	Rendez-vous	-	7

6.5 Analyse de la valeur acquise

Elle consiste à calculer ce qui a été accompli pour mesurer le progrès effectué. Mesures de base :

- **CBT** (Coût Budgété du Travail) : est la quantité de travail estimée pour une tâche donnée.
- **CBA** (Coût Budgété à l'Achèvement) : est la somme de tous les quantités de travail estimées.
- **CBTE** (Coût Budgété du Travail Effectué) : est la somme de tous les quantités de travail estimées qui sont effectivement achevées.
- **CBTP** (Coût Budgété du Travail Prévu) : est la somme de tous les quantités de travail estimées qui sont effectivement achevées à une date donnée.
- **CRTE** (Coût Réel du Travail Effectué) : est la somme de tous les quantités de travail réelles qui sont effectivement achevées.
- **VP** (Valeur Prévue) : est la proportion de la quantité de travail totale estimée attribuée à une tâche donnée $VP = CBT/CBA$.

6.6 Indicateurs d'avancement

- **VA** (Valeur Acquise) ou **PA** (Pourcentage achevé) : est la somme des VP pour les tâches achevées :
 $VA = PA = CBTE/CBA$.
- **IPT** (Indicateur de Performance Temporel) :
 $IPT = CBTE/CBTP$.
- **VE** (Variance à l'Echéancier) : variance par rapport à l'échéancier :
 $VE = CBTE - CBTP$.
- **IC** (Indicateur des Coûts) : indicateur d'écart sur les coûts :
 $IC = CBTE/CRTE$.
- **VC** (Variance des Coûts) : variance par rapport aux coûts :
 $VC = CBTE - CRTE$.

Problème 1

Tache	Travail estimé (jh)	Travail réel aujourd'hui	Date d'achèvement estimée	Date d'achèvement effective
1	5	10	25-janv	01-févr
2	25	20	15-févr	15-févr
3	120	80 (140 au problème 2)	15-mai	- (01-juil au problème 2)
4	40	50	15-avr	15-avr
5	60	50	01-juil	-
6	80	70	01-sept	-

- Calculer les indicateurs d'avancement au 01/04 ?

Corrigé

- Le CBA : somme des estimations des quantités de travail
 - $CBA = 5 + 25 + 120 + 40 + 60 + 80 = 330\text{jh}$
- Les tâches 1, 2 et 4 sont achevées
 - Le CBTE est la somme des CBT pour ces tâches
 - $CBTE = 5 + 25 + 40 = 70\text{jh}$
 - $VA = CBTE/CBA = 70/330 = 21,2\%$
- Les tâches 1 et 2 devraient être achevées pour le 01/04, et pas 1, 2 et 4 donc :
 - $CBTP = 5 + 25 = 30\text{jh}$
 - $IPT = CBTE/CBTP = 70/30 = 233\%$
 - $VE = CBTE - CBTP = +40\text{jh}$
- La CRTE est la somme des quantités de travail réelles pour les tâches 1, 2 et 4
 - $CRTE = 10 + 20 + 50 = 80\text{jh}$
 - $IC = CBTE/CRTE = 70/80 = 87,5\%$
 - $VC = CBTE - CRTE = 70 - 80 = -10\text{jh}$

Problème 2

Que deviennent ces indicateurs, à supposer que la tâche 3 a également été achevée avec 140jh de travail et que nous sommes le 01/07?

- CBTE = 190jh
- CBTP = 250jh
- CRTE = 220jh
- VA = $190/330 = 57.5\%$
- IPT = $190/250 = 76\%$
- VE = $190 - 250 = -60\text{jh}$

Seules les tâches 1–4 sont réalisées, au lieu de 1–5

- IC = $190/220 = 86.6\%$
- VC = $190 - 220 = -30\text{jh}$

6.7 Conclusion

- Les facteurs qui affectent la productivité incluent l'aptitude individuelle (facteur dominant), expérience du domaine, processus de développement, la taille du projet, le support outil et l'environnement de travail.
- Le modèle COCOMO est le modèle le mieux développé, il prend en considération les attributs du projet, du produit, du matériel et du personnel dans la formulation de l'estimation du coût.
- Le modèle COCOMO 81 suppose que le logiciel est développé selon un modèle de processus en cascade et utilisant un langage de programmation impératif (C ou FORTRAN).
- Cependant, les changements effectués dans le développement du logiciel à savoir les processus logiciels adoptés (processus de développement incrémental et itératif), le développement de logiciel par assemblage de composants réutilisables, la technique de ré-ingénierie utilisée pour créer un nouveau logiciel à partir d'un logiciel existant, etc. ont Contribué à donner lieu au modèle COCOMO II (COCOMO 2) qui prend en compte ces changements.
- Le temps nécessaire pour compléter un projet n'est pas simplement proportionnel à l'effort requis. L'ajout du personnel en fin de projet peut augmenter plutôt que diminuer le temps prévu pour terminer le projet. Enfin, l'analyse de la valeur acquise permet de mesurer les progrès du développement réalisés pour mieux contrôler le projet.

Chapitre 7

Gestion des ressources humaines

7.1 Objectifs

L'objectif de ce chapitre est de présenter l'importance du personnel (des hommes) dans le processus logiciel. Cela consiste à :

- Comprendre comment effectuer la sélection et la conservation du personnel dans une organisation
- Comprendre comment constituer une équipe et y attribuer des rôles
- Comprendre la gestion de groupe incluant la composition du groupe, la cohésion, la communication et l'organisation de l'équipe.
- Être informé de la structure du P-CMM, un modèle permettant l'amélioration des capacités des ingénieurs du logiciel dans une organisation.

7.2 Introduction

- Le personnel travaillant dans une organisation de développement de logiciels représente un capital intellectuel. Dans une organisation, une gestion effective est donc liée à la gestion des ressources humaines.
- Une mauvaise gestion des ressources humaines est l'une des plus significatives contributions à l'échec d'un projet. Quatre facteurs critiques dans la gestion des ressources humaines sont à considérer :
 1. **Consistance** : dans une équipe de projet, l'ensemble des membres doivent être traité de manière comparable.
 2. **Respect** : les différents membres ont différentes compétence et le chef de projet doit tenir compte de cette différence.
 3. **Inclusion** : le personnel contribue effectivement quand il sent qu'il est écouté et ses propositions sont prises en compte.
 4. **Honnêteté** : le chef doit être honnête sur son jugement et doit donner de manière fidèle son point de vue sur l'équipe. La gestion est une question d'apprentissage à travers les expériences vécues durant les projets précédents.

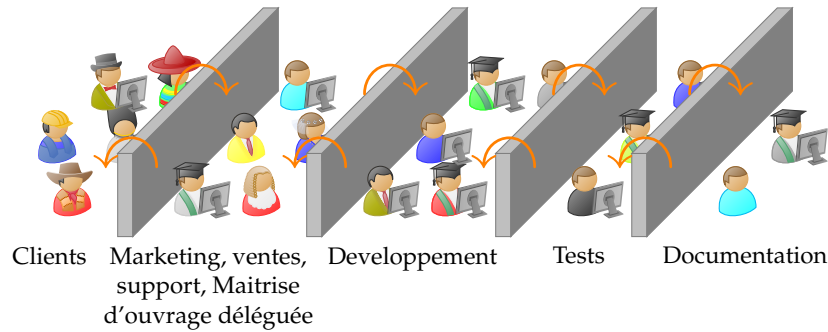


FIGURE 7.1 – Organisation de développement de logiciels.

7.3 Sélection du personnel

- La sélection du personnel est une des tâches les plus importantes dans la gestion d'un projet. Parfois, le chef de projet peut nommer le personnel qui convient au travail sans tenir compte de leurs responsabilités et des considérations du budget. Le plus souvent il n'a pas le libre choix. La décision sur laquelle il se base pour désigner une équipe s'appuie sur trois types d'information :
 1. **L'information fournie par les candidats** concernant leur formation et leurs expériences (généralement, il s'agit du CV).
 2. **L'interview avec les candidats** dans le but de savoir s'ils sont de bons communicateurs.
 3. **Les recommandations des personnes** qui ont déjà travaillé avec ces candidats.
 4. **Certaines compagnies procèdent à des tests** pour la sélection des candidats. Ces tests incluent les tests d'aptitude à la programmation et les tests psychométriques où il s'agit de compléter certains petits exercices dans un délai relativement court. Les tests psychométriques permettent de fournir le profil psychologique du candidat indiquant leur convenance à certains types de tâches.

7.4 Constitution de l'équipe

La première responsabilité qui incombe au chef de projet est de constituer son équipe. Idéalement, il aura préalablement listé les compétences nécessaires en fonction des objectifs du projet et du contexte fonctionnel et technologique, et il aura défini les rôles et responsabilités nécessaires à la bonne réalisation du projet.

7.4.1 Définir les rôles et responsabilités

La description des activités et des rôles du projet peut être formalisée en amont, de façon générique, dans les référentiels méthodologiques des organisations ou par Un Project Management Office (PMO). En fonction des spécificités du projet, telle ou telle activité ou tel ou tel rôle se révélera nécessaire ou superflu.

- Une équipe traditionnelle se caractérise par une séparation claire des rôles et responsabilités (figure ci-dessous). Le déroulement séquentiel des activités induit cette répartition, chaque acteur intervenant l'un après l'autre (successivement, l'analyste, le concepteur, le développeur et le testeur).

- Chaque acteur travaille généralement seul sur des tâches bien spécifiques et des parties distinctes de l'application. Mais cette division du travail présente certains inconvénients sur la productivité de l'équipe, les coûts et le délai du projet... Et sur le syndrome du « c'est pas moi, c'est lui ! »
- En effet, cette division suppose que la connaissance autour du projet soit justement formalisée afin que ces informations plus ou moins formelles soient transmises aux autres collaborateurs qui prennent le relais; ce temps consacré à la formalisation a un coût, d'autant qu'une partie de l'information initiale est de toute façon perdue et que le(s) collaborateur(s) suivant(s) dans la chaîne aura à recouvrer cette information partiellement perdue.
- Cela risque de générer des retards et des erreurs qui ne pourront être détectées qu'à la fin du cycle de fabrication, une fois toutes les activités achevées. À ce moment-là, les personnes présentes lors des premières étapes et responsables des tâches auront peut-être été affectées à d'autres projets. Par ailleurs, si chacun travaille isolément, le sentiment de contribuer à un travail collectif pour un résultat concret n'est pas réel puisque le résultat final est lointain, hors de portée, ce qui crée une frustration et une déresponsabilisation néfastes au succès du projet.

7.4.2 Déterminer la composition de l'équipe

- La structure de l'équipe dépend étroitement de facteurs d'environnement : l'organisation (organisation hiérarchique ou en mode projet), le coût des ressources, les départements susceptibles de fournir les ressources, l'éloignement géographique, la disponibilité des compétences, les différences culturelles, les dépendances hiérarchiques, la qualité des relations interpersonnelles...
- Le chef de projet devra mettre en oeuvre ses capacités à négocier pour obtenir les ressources nécessaires. La question qui se pose : il y a un travail à réaliser pour servir la vision et atteindre l'objectif, l'équipe a des besoins spécifiques pour devenir collectivement performante, mais chaque collaborateur a des préférences individuelles. Il ne sert à rien d'affecter un collaborateur à un projet qui ne présente aucune perspective de montée en compétence ou de motivation personnelle.
- Le chef de projet doit veiller à ce que tous les besoins soient satisfaits (figure ci-dessous). Si l'une des dimensions échoue, l'efficacité de l'équipe est compromise.
- Ce qui est important, c'est de tirer profit de la richesse de la diversité des membres de l'équipe : homme/femme, expérimenté/non expérimenté, technique/fonctionnel, personnalités fortes/effacées.
- Différentes structures existent dont les deux suivantes :

1. Structure hiérarchique :

Elle est basée sur des relations d'autorité. Le principe est qu'un subordonné ne reçoit d'ordre que d'un seul chef auquel il doit rendre compte.

Avantage :

- Simplicité (tout est écrit ou presque);
- Répartition claire des responsabilités;
- Maintien d'une discipline.

Inconvénient :

- Lenteur d'information;
- Pas de prise d'initiative;
- Non adaptation à un environnement changeant ou évolutif.

2. Structure mode projet :

Pour chaque projet, un chef est nommé compte tenu de ses compétences. Il a pour mission de coordonner les activités nécessaires à l'aboutissement de son projet en sollicitant l'expertise

des personnes avec lesquelles il n'a aucune relation d'autorité.

Avantage :

- efficace pour développer les activités nouvelles et coordonner les activités multiples dans un environnement complexe et évolutif.

Inconvénient :

- source de conflits générés par les problèmes d'encadrement et de communication.

7.5 Gestion du groupe

- La majorité des logiciels professionnels sont développés par des équipes de projet allant de 2 à plusieurs centaines de personnes. Chaque groupe est responsable d'une partie du système entier. L'effectif d'un groupe ne dépasse pas 10 ce qui permet de réduire les problèmes de communication. Le succès d'un groupe réside dans leur esprit de groupe. Il existe un nombre de facteurs qui influence sur le travail de groupe.
- **Composition du groupe** : un groupe est souvent composé de personnes ayant leur propre façon de résoudre les problèmes. Un groupe avec des personnalités complémentaires peut travailler de façon meilleure qu'un groupe sélectionné selon des capacités techniques.
- **Cohésion du groupe** : l'intérêt du groupe est important relativement à l'intérêt individuel. Les avantages d'une cohésion sont :
 - les standards de qualité du groupe peuvent être développés au lieu d'être imposés au groupe (cas des standards externes).
 - les membres du groupe travaillent ensemble → apprentissage collectif
 - les membres du groupe connaissent le travail les uns des autres.
 - Les programmes sont plutôt considérés comme une propriété du groupe et non individuelle.
- Cependant certains problèmes peuvent se poser dans le cas des groupes fortement cohésives à savoir : le changement de dirigeant qui peut provoquer quelques perturbations telles que le refus du nouveau dirigeant, perte de temps pour s'adapter avec le nouveau dirigeant et par conséquent une diminution de la productivité.

7.6 Motivation du personnel

La motivation consiste à organiser le travail et l'environnement de travail de façon que le personnel puisse travailler efficacement. Selon Maslow, le personnel est motivé pour travailler en fonction de la satisfaction de ses besoins. Ces derniers sont rangés selon une série de niveaux.

1. **Besoins Physiologiques** (Psychological needs) : Besoins liés au fonctionnement du corps humain ;
2. **Besoins de sécurité** (Safety needs) : Besoin de se protéger contre les agressions d'ordre physique, psychologique et économique ;
3. **Besoins sociaux** (Social needs) : Besoin d'amitié de la part des autres. Acceptation et appartenance à un groupe ;
4. **Besoins de reconnaissance** (Esteem needs) : Besoin d'avoir du pouvoir sur les autres, de se faire respecter et d'être capable d'influencer et de convaincre ses amis ou ses collègues de travail ;
5. **Besoins d'auto-réalisation** (Self-realisation needs) : Les besoins les plus élevés. La personne cherche à s'accomplir.

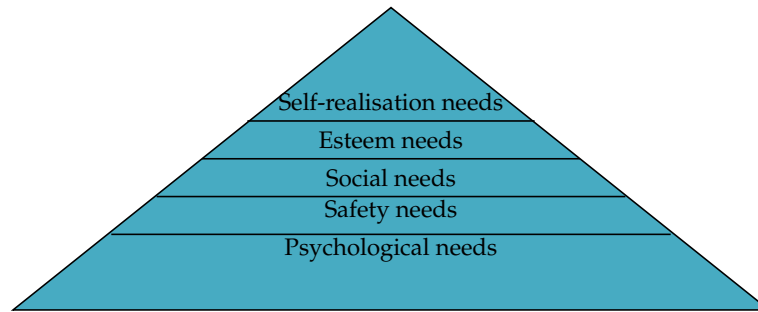


FIGURE 7.2 – La pyramide de Maslow.

- En fait, les besoins les plus significatifs du point de vue gestion sont : les besoins sociaux, les besoins de reconnaissance et les besoins d'auto-réalisation.
- Chaque niveau de la hiérarchie est un pré requis pour le niveau suivant.

7.7 Modèle de maturité des capacités du personnel

L'institut de génie logiciel SEI (Software Engineering Institut) a proposé un modèle de maturité des capacités du personnel P-CMM (People Capability Maturity Model) qui est utilisé comme Framework pour améliorer le moyen selon lequel une organisation gère ses biens humains. Ce modèle regroupe 5 niveaux :

1. **Initial** : les pratiques de gestion du personnel sont informelles
2. **Répétable** : établissement des politiques pour développer les capacités du personnel.
3. **Défini** : standardisation des bonnes pratiques de gestion du personnel à travers l'organisation
4. **Géré** : les buts quantitatifs pour la gestion du personnel sont établis.
5. **Optimisé** : accent continu sur l'amélioration des compétences individuelles et la motivation des mains d'œuvre.

Selon Curtis et al, les objectifs stratégiques du P-CMM sont :

- améliorer les capacités des organisations de développement des logiciels en augmentant les capacités de la main d'œuvre.
- Garantir que les capacités de développement des logiciels sont liées à l'organisation.

7.8 Conclusion

- La sélection du personnel dans un projet est une tâche importante des chefs de projets. Les facteurs qui peuvent être utilisés pour une telle sélection sont ; l'expérience du domaine d'application, l'adaptabilité et la personnalité.
- Les groupes de développement des logiciels doivent être petits et cohésifs. Les dirigeants des groupes doivent être techniquement compétents.
- La communication à l'intérieur d'un groupe peut être influencée par la taille du groupe, le genre de composition du groupe, le status des membres du groupe, les personnalités et les moyens de communication disponibles.

- Le P-CMM fournit un Framework et une aide pour améliorer les capacités du personnel dans une organisation et améliorer les capacités de l'organisation pour tirer profit des biens humains.

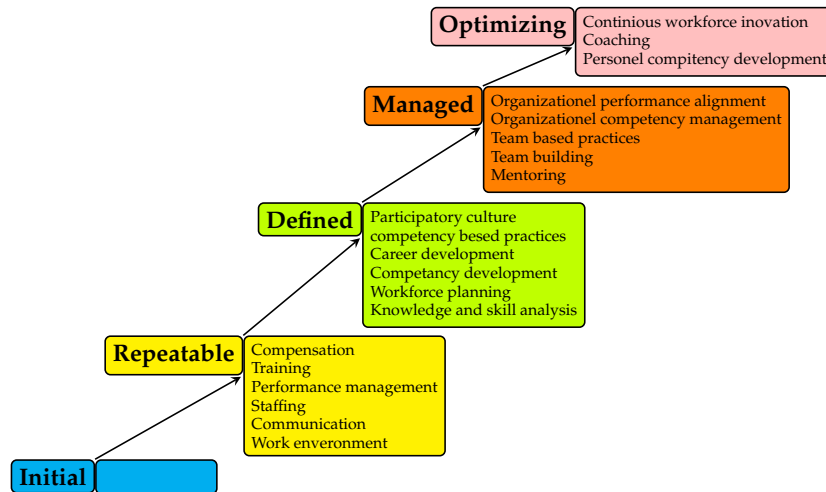


FIGURE 7.3 – P-CMM.

Chapitre 8

Gestion de la configuration

8.1 Objectifs

L'objectif de ce chapitre est d'introduire un aperçu sur la gestion de la configuration. Cela consiste à :

- Comprendre l'inévitabilité du changement du logiciel.
- Définir la notion de gestion de configuration et tous les termes et les concepts associés.
- Détailler les fonctions de gestion de configuration.
- Montrer le rôle de la gestion de configuration dans le cycle de développement.
- D'introduire certains outils de gestion de configuration et leur fonctionnement.

8.2 Introduction

- La gestion de configuration a pour but de gérer l'évolution des systèmes logiciels. La gestion de configuration est une partie de la gestion de la qualité. Les procédures de gestion de configuration définissent comment enregistrer et traiter les changements de systèmes.
- Il existe plusieurs raisons qui font que le système possède différentes configurations les configurations peuvent être produites pour différents ordinateurs, pour différents systèmes d'exploitation La définition et l'utilisation des standards de gestion de configuration est essentielle pour la certification de la qualité dans ISO9000 et les standards CMM / CMMI.
- **A propos de changements** : un logiciel évolue en subissant une suite de changements qui sont inévitables. Voici quatre lois de Lehman qui sont particulièrement importantes :
 1. **Loi du changement continu** : un logiciel doit être continuellement adapté, faute de quoi il devient progressivement moins satisfaisant à l'usage.
 2. **Loi de la complexité croissante** : un logiciel a tendance à augmenter en complexité, à moins que des actions spécifiques ne soient menées pour maintenir sa complexité ou la réduire.
 3. **Loi de la croissance constante** : un logiciel doit se doter constamment de nouvelles fonctionnalités afin de maintenir la satisfaction des utilisateurs tout au long de sa vie.
 4. **Loi de la qualité déclinante** : la qualité d'un logiciel tend à diminuer, à moins qu'il ne soit rigoureusement adapté pour faire face aux changements.
- Un changement peut influencer : **les délais** de livraison, **les coûts** de la réalisation, **la cohérence** et **la complétude** du système.

8.3 Définitions

- **Définition** : La gestion de configuration est une discipline de l'ingénierie logicielle permettant de gérer l'évolution d'un système informatique pendant tout son cycle de vie, c.-à-d., depuis l'expression des besoins jusqu'à son retrait de service.
- **Selon ISO** : La gestion de configuration est une discipline de management de projet qui permet de définir, d'identifier, de gérer et de contrôler les articles de configuration tout au long du cycle de développement d'un logiciel (y compris la phase de maintenance).
- **Selon IEEE 1983** : Configuration management is the process of identifying and defining the items in the system, controlling the changes to these items throughout their life cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items.

8.3.1 Les avantages de la gestion de configuration

- **Mieux satisfaire l'utilisateur** : délai de réalisation est respecté, informer les nouveautés dans la nouvelle version, les fonctionnalités fournies par cette nouvelle livraison correspondent à celles qu'il attendait, absence de bogues.
- **Assurer la qualité interne du logiciel** : respect des exigences telles que l'intégrité (protection des composants contre les accès non autorisés), l'intégration (facilité d'assemblage), la traçabilité (savoir qui fait quoi, où et comment), la maintenabilité (facilité de changement), vérifiabilité (facilité de préparation des procédures de test), etc.
- **Gérer efficacement le travail d'équipe** : support pour définir les rôles, les activités, le partage des ressources, la communication des informations et la coordination intra et inter-équipes.
- **Obtenir une certification** : la gestion de configuration est un préalable pour mettre en place un système de qualité certifié ISO ou SEI-CMM.

8.3.2 Les définition des termes et concepts

- **Un élément de configuration (ou article de configuration)** : est considéré tout élément fonctionnel ou physique concernant le système que l'on a décidé de gérer en configuration.
Exemple : un code, des exécutables, des documents de spécification, etc. A chaque élément de configuration sont rattachés des caractéristiques telles que : identification, auteur date de création, date de la dernière modification, version, droit d'accès et de localisation.
- **Une configuration de référence (Baseline)** : est la photographie d'un état stable de la configuration du système. Elle est utilisée comme une référence de départ pour le développement d'une nouvelle release.
 Un référentiel regroupe toutes les configurations du système. D'autres appellations sont aussi attribuées à ce terme (repository, dépôt, etc.). Le référentiel de gestion de configuration n'est pas seulement un outil de stockage mais c'est aussi le cycle de vie et les évolutions de tous les éléments de configuration.
- **Une évolution** : est un changement dans la définition du système. Le changement peut correspondre à un ajout de nouvelles fonctionnalités ou à une optimisation de la performance ou peut être une réaction à une anomalie constatée.
- **Une version** : est attribut formé d'une suite de caractères alphanumérique permettant d'identifier de manière unique un élément de configuration dans l'histoire de son évolution. Exemple : le fichier X a comme version 3.1.
- **Une release** : est une évolution à l'échelle du système livrable aux utilisateurs.

- **Une version** s'applique à un élément de configuration comme un fichier. Cependant, une release désigne une version significative apportant des évolutions majeures livrable aux utilisateurs. Le développement d'une release peut donner lieu à plusieurs versions intermédiaires dont seule la dernière est livrable.

8.4 Les fonctions de la gestion de configuration

La gestion de la configuration recouvre plusieurs fonctions. Ces fonctions peuvent être classées en deux catégories principales :

- les fonctions liées à la mise en place et à la gestion du référentiel ;
- les fonctions liées à l'activité courante des développeurs.

8.4.1 Les fonctions liées à la mise en place et à la gestion du référentiel

Les fonctions liées à la mise en place et à la gestion du référentiel sont :

1. **La gestion des demandes de changements** : consiste à
 - Gérer le cycle de vie d'une demande de changement depuis sa formulation jusqu'à sa clôture. En standard, les états d'une demande de changements sont : entrée, vérifiée assignée, résolue, intégrée, testée et fermée.
 - Analyser les impacts d'une demande de changement sur les éléments de configuration et sur les exigences.
 - Proposer une étude de faisabilité des demandes de modification.
2. **L'identification et l'enregistrement de la configuration** : consiste d'une part à identifier les éléments de configuration fonctionnels et physiques qui composent un système et à les désigner de façon unique pendant le cycle de vie de ce dernier. D'autre part cela consiste à saisir ensuite la configuration de référence dans le référentiel.
3. **Le contrôle de la configuration** : consiste à maîtriser les modifications apportées à la configuration de référence :
 - Identifier, enregistrer et documenter les nouveaux éléments de configuration.
 - Vérifier la cohérence globale de la configuration
 - Vérifier et fournir les informations concernant toute modification (auteur, impact, date et les raisons)
4. **La documentation sur l'état** : consiste à documenter à l'aide de rapports et de graphiques les différentes informations se rapportant aux modifications apportées aux éléments de configuration.
5. **L'audit et la revue** : consiste à vérifier si la release livrée a été élaborée conformément aux exigences fonctionnelles et techniques et respecte les clauses du contrat.

8.4.2 Les fonctions liées à l'activité courante des développeurs

1. **La gestion du travail d'équipe** : consiste à gérer les interactions entre les multiples développeurs travaillant ensemble sur une release ou en parallèle sur plusieurs. Cela revient à définir les rôles et les responsabilités de chaque développeur et à leur assigner des tâches.
2. **La gestion des fabrications** : consiste à gérer les fabrications d'une release d'une façon optimale et automatisée.

3. **La gestion du processus de développement** : consiste à assurer l'exécution correcte du processus d'évolution du logiciel et par conséquent le respect de l'enchaînement des étapes du processus.

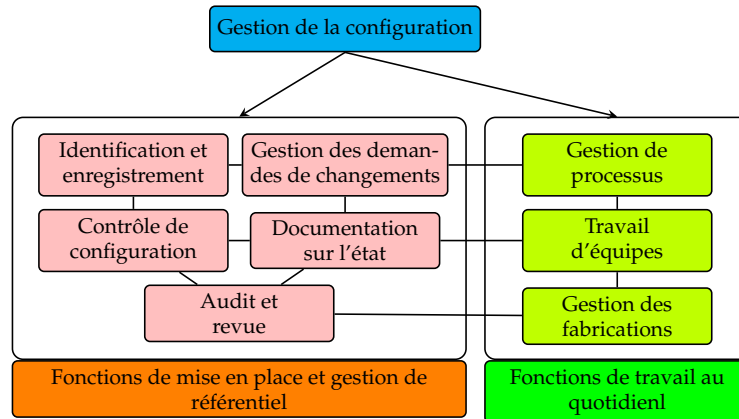


FIGURE 8.1 – Les fonctions principales de la gestion de configuration.

8.5 La gestion de configuration et le processus de développement

- La gestion de configuration peut s'appliquer à n'importe quel modèle choisi pour le processus de développement (modèle en V ou en cascade).
- Il existe plusieurs modèles de développement mais le modèle qui s'avère le plus efficace est le modèle itératif car il permet une réalisation progressive du système, en plusieurs itérations. Chaque itération fournit une version intermédiaire, exécutable du système.

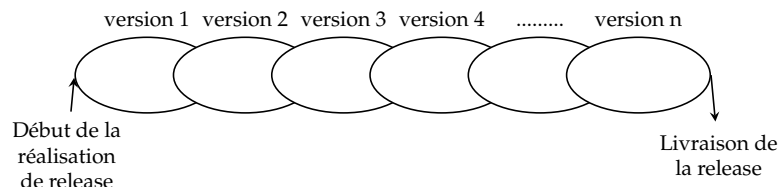


FIGURE 8.2 – Le modèle itératif.

Chaque itération regroupe une succession des activités traditionnelles : analyse, conception codage, intégration, tests de validation, livraison. Les itérations peuvent être définies comme suit :

- Une première itération qui définit une maquette de faisabilité ;
- Plusieurs itérations de mise au point de l'architecture du système ;
- Plusieurs itérations fonctionnelles pour implémenter de manière incrémentale les fonctionnalités du système ;
- Plusieurs itérations pour corriger des problèmes et améliorer les performances du produit en vue de la livraison finale.

8.6 Rôle de la gestion de configuration dans le cycle de développement

Pour comprendre le rôle de la gestion de configuration dans le cycle de développement, prenons en considération les problèmes suivants :

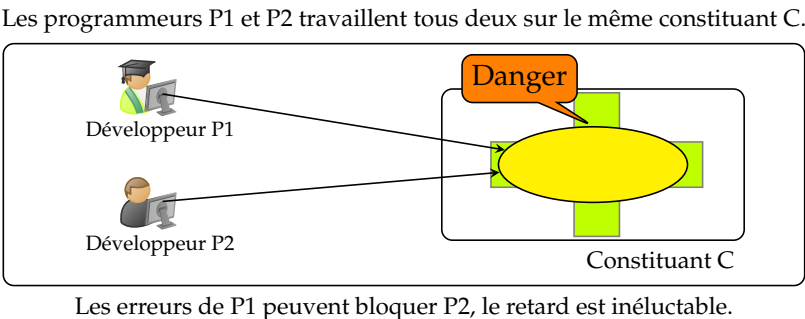


FIGURE 8.3 – Le rôle de la gestion de configuration dans le cycle de développement.

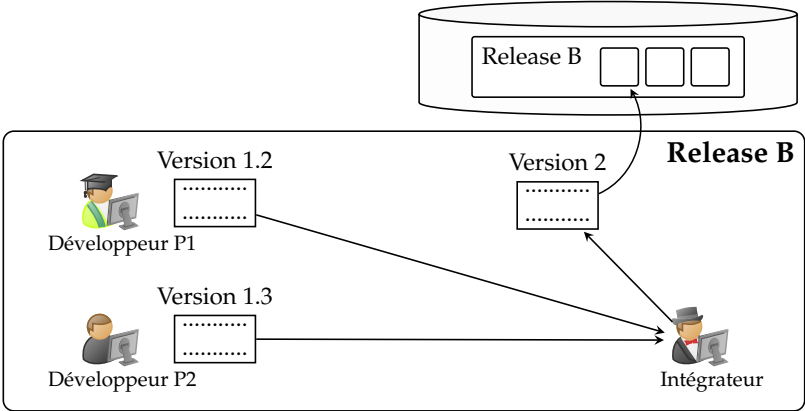
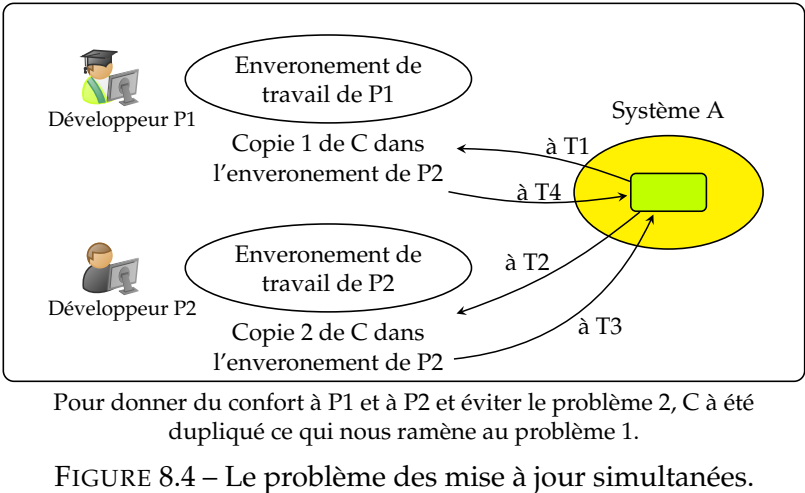


FIGURE 8.5 – Développement contrôlé par la gestion de configuration.

8.7 Les outils de gestion de version et de gestion de configuration

- Les gestionnaires de configuration (CVS, par exemple) sont peut être les outils les plus importants pour le développement des gros logiciels. Ils permettent à plusieurs développeurs de stocker et de partager les mêmes objets en gardant la trace de leurs évolutions : programmes sources, documentations.
- Les outils de base comme RCS s'appuient sur le modèle du « check in – check out ». Le modèle du check in – check out exploite un référentiel partagé qui stocke les fichiers multi versionnés et des espaces de travail privés qui sont des sous ensembles mono versionnés du référentiel (voir la figure suivante).

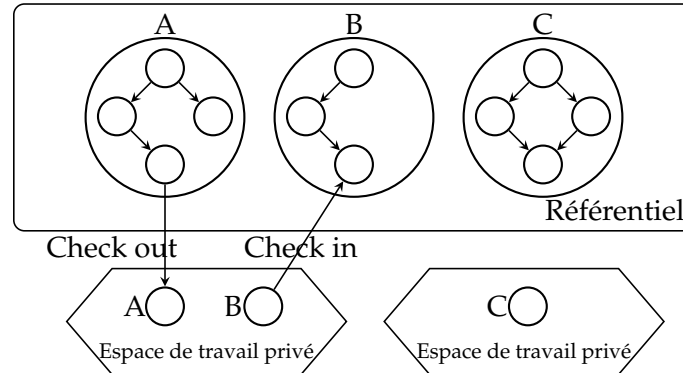


FIGURE 8.6 – Le modèle du check in check out.

Les versions sont organisées selon un graphe direct acyclique, avec une branche principale et des branches annexes.

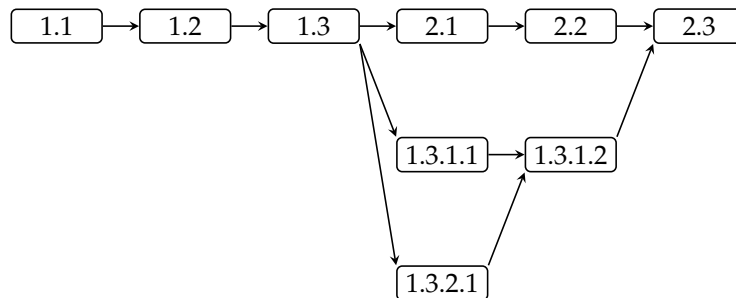


FIGURE 8.7 – Le graphe des versions.

- L'utilisateur dispose d'opérateurs de transfert entre le référentiel et son espace de travail :
 1. **Check out** : création d'un fichier dans l'espace de travail à partir d'un nom de fichier et d'un numéro de version.
 2. **Check in** : création d'une nouvelle version dans le référentiel à partir d'un nom de fichier dans l'espace de travail.
- Les accès concurrents sont gérés par des verrous en lecture et en écriture. Ce mode de fonctionnement permet :
 1. **Le développement parallèle** : un fichier peut être développé de plusieurs manières différentes dans des branches de versions séparées. Une branche de versions peut être ultérieurement fusionnée avec une autre ou abandonnée.

2. **La gestion des mises à jour conflictuelles** : si un fichier est en cours de modification, il est verrouillé. Un second utilisateur peut effectuer une mise à jour en créant une nouvelle version dans une autre branche. Elle sera fusionnée à la version principale ultérieurement. Ce mode de fonctionnement permet toujours la lecture des objets. Par contre, il n'assure pas la correction des exécutions concurrentes. Un utilisateur peut lire des objets liés incohérents si l'un a été modifié et remis dans le référentiel et l'autre est encore en cours de modification. L'utilisateur risque d'accéder à la version mise à jour du premier objet et à l'ancienne version du second.

8.8 Conclusion

- La gestion de la configuration est la gestion des changements d'un système. Lors de l'évolution, d'un système, le rôle de l'équipe de gestion de configuration est de garantir que les changements sont effectués de manière contrôlée.
- La gestion de configuration aide les différents intervenants dans le processus de développement à mieux s'approcher de leur objectif qui est de réaliser un système de qualité conforme aux attentes des utilisateurs.
- Les développeurs gagnent en temps et en productivité car leurs efforts sont concentrés sur leurs métiers réels (analyse, conception, etc.) et non sur les opérations récurrentes de gestion de contrôle et de fabrication.
- Les outils de gestion de configuration servent à supporter toutes les activités de gestion de changement, de gestion des versions et de la construction du système.

Chapitre 9

Métriques du logiciel

9.1 Objectifs

Le but de ce chapitre est d'introduire certaines métriques utilisées dans l'estimation de certains attributs du logiciel/processus. Cela consiste à :

- Présenter certaines métriques du produit logiciel connues selon l'approche classique et l'approche orientée objet.
- Présenter certaines métriques du processus logiciel.
- Expliquer et de détailler l'approche GQM d'un projet.

9.2 Introduction

- La mesure en génie logiciel consiste à faire correspondre des symboles à des objets, afin de pouvoir quantifier certains attributs de ces objets.
- Il peut s'agir de mesurer la taille d'un projet logiciel ou d'évaluer des attributs qui ne sont pas directement mesurables (ressources nécessaires au développement d'un projet). Le domaine d'application des métriques est le coût et les techniques d'estimation de taille.
- La prédiction des niveaux de qualité pour les logiciels, souvent en termes de fiabilité, est aussi un domaine où les métriques logicielles ont un rôle important à jouer. L'utilisation des métriques du logiciel sert à fournir des contrôles quantitatifs sur la conception de logiciels. En effet, "You cannot control what you cannot measure".[De Marcos, 1982]

9.3 Définitions

- Une métrique est définie comme : "a quantitative measure of the degree to which a system component or process possesses a given attribute" IEEE.
- Software metrics can be defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products".

9.4 Classification des métriques

Les métriques peuvent être classées en trois catégories :

1. **Les métriques du produit** : concernent les résultats explicites des activités de développement du logiciel (livrables, de documentation,).
2. **Les métriques du processus** : concernent les activités liées à la production de logiciels.
3. **Les métriques des ressources** : concernent les entrées des activités de développement logiciel (Le matériel, les connaissances, les gens).

9.5 Les métriques du produit

Les métriques du produit sont calculées directement à partir du document, indépendamment de la façon dont il a été produit. Elle s'applique à la structure du code source.

9.5.1 Les métriques orientées taille

Concernent la taille des logiciels produits

- LOC : Lignes de code. KLOC : 1000 lignes de code.
- SLOC : Lignes de code source (ignorer les espaces).
- Mesures typiques : Erreurs/KLOC, Défauts/KLOC, Coût/LOC, Pages de documentation/KLOC.

Les métriques LOC sont, en général :

- Faciles à utiliser
- Faciles à calculer
- Dépendants du programmeur et du langage de programmation.

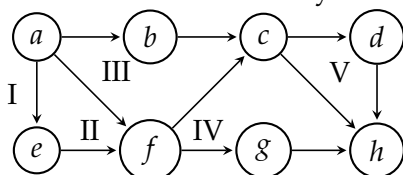
9.5.2 Métriques de Mac Cabe

La complexité structurelle selon Mc Cabe, introduite en 1976, constitue, après les lignes de code, la métrique la plus employée dans le cadre du développement logiciel. Elle met en évidence la complexité structurelle du code.

- On produit un graphe de contrôle qui représente un code
- Le nombre de faces du graphe donne la complexité structurelle du code
- Nombre cyclomatique de Mc Cabe : $C = a - n + 2p$
avec :
 - a = nombre d'arcs du graphe de contrôle
 - n = nombre de nœuds du graphe de contrôle
 - p = nombre de composantes connexes (1 le plus souvent)

Exemple (Métriques de Mac Cabe)

Déterminer le nombre cyclomatique pour le graphe de contrôle suivant :



- Ici, $n = 8$, $a = 11$ et $p = 1$ donc $C = 11 - 8 + 2 = 5$. C correspond au nombre de faces (en comptant la face extérieure)
- Le graphe de contrôle doit posséder un nœud de départ et un nœud d'arrivée distincts pour que la formule précédente soit valide.

Calcul direct du nombre de Mc Cabe

- Produire un graphe de contrôle et l'analyser peut s'avérer long dans le cas de programmes complexes \Rightarrow Mc Cabe a introduit une nouvelle manière de calculer la complexité structurale : $C = D + 1$ avec D le nombre de décisions du code.
- Dans le code source, une instruction IF compte pour 1 décision, une boucle FOR ou WHILE compte pour 1 décision. Une instruction CASE ou tout autre embranchement multiple compte pour 1 décision de moins que le nombre d'alternatives proposées.
- **Exemple** : Dans la figure précédente, il y a 3 arcs qui partent du nœud a ce qui correspond à deux décisions notées A et B. depuis les nœuds c et f partent 2 arcs ce qui correspond à une décision pour chacun. Les autres arcs ne possèdent qu'un seul arc de sortie d'où pas de décision. Au total, on a quatre décisions et $C = 4 + 1 = 5$.

Valeur seuil

- Pour chaque métrique, il est important d'avoir une idée des valeurs raisonnables de celles qui ne le sont pas.
- Après une analyse des projets de grande envergure, Mc Cabe a constaté que les modules dont le nombre cyclomatique dépassait 10 avaient posé plus de difficultés de maintenance que ceux dont ce nombre était inférieur à cette valeur.
- La valeur 10 est donc considérée comme la valeur seuil pour le nombre cyclomatique d'un module.

9.5.3 Métriques de Halstead

- Maurice Halstead était l'un des premiers à faire des recherches sur les métriques logicielles, entre la fin des années 60 et le début des années 70. Son objectif était de déterminer ce qui contribuait à la complexité des logiciels.
- Il a cherché, de manière empirique, des mesures pour leur taille. Il a ensuite tenté de mettre au point une théorie cohérente.
- Ses métriques sont considérées comme valides, contrairement à ses formules complexes de prédiction. Les entités de base :
 - Opérandes : jetons qui contiennent une valeur
 - Opérateurs : tout le reste (virgules, parenthèses, opérateurs arithmétiques...)
- Mesures de base N_1 et N_2
 - N_1 : nombre d'opérateurs distincts
 - N_2 : nombre d'opérandes distincts
 - $N_1 + N_2$: nombre total de jetons distincts

Exemple

```

z = 0;
while x > 0
    z = z + y;
    x = x - 1;
end-while;
print (z);

```

Opérandes	=	;	>	+	-	()	print	while/end-while
Nombre	3	5	1	1	1	1	1	1
Opérateurs	z	0	x	y	1			
Nombre	4	2	3	1	1			

- Le nombre d'opérateurs distincts = 8.
- Le nombre d'opérandes distincts = 5.

Longueur d'un programme en nombre de jetons

- N_1 : Nombre total d'opérateurs (égale 14 dans l'exemple)
- N_2 : Nombre total d'opérandes (égale 12 dans l'exemple)
- $N = N_1 + N_2$: Nombre total de jetons (égale 26 dans l'exemple)

1. Estimation de la longueur :

L'estimation de la longueur est la plus simple des formules de prédiction de Halstead. Elle permet une estimation de la longueur totale du programme en nombre de jetons à partir de N_1 et de N_2 : $N = N_1 \times \log_2(N_1) + N_2 \times \log_2(N_2)$

Dans notre exemple : $N = 8 \times \log_2(8) + 5 \times \log_2(5) = 8 \times 3 + 5 \times 2,32 = 35,6$

Ici, N qui était 26 n'est pas en fait une mauvaise approximation. En pratique, Si la différence entre N (qui était égale à 26) et N (qui est égal à 35,6) est supérieure à 30%, il vaut mieux renoncer à utiliser les autres mesures de Halstead.

2. Volume :

L'estimation du nombre de bits nécessaires pour coder le programme mesuré est :

$$V = N \times \log_2(N_1 + N_2)$$

Dans notre exemple : $V = 25 \times \log_2(13) = 25 \times 3,7 = 92,5$

3. Autres mesures de Halstead

- **Volume potentiel V^*** : Il correspond à la taille minimale d'une solution au problème. Halstead suppose que dans l'implémentation minimale, il n'y aurait que deux opérateurs : le nom de la fonction et un opérateur de groupement. Le nombre minimal d'opérandes est noté N_2^* : $V^* = (2 + N_2^*) \times \log_2(2 + N_2^*)$
- **Niveau d'implémentation L** : Il exprime dans quelle mesure l'implémentation actuelle est proche de l'implémentation minimale : $L = V^*/V$
- Il a aussi proposé des formules pour mesurer l'effort intellectuel nécessaire à l'implémentation d'un algorithme et le temps nécessaire pour l'implémenter.

9.5.4 Métriques de Henry-Kafura

Salie Henry et Denis Kafura ont développé une métrique pour mesurer la complexité des modules d'un programme en fonction des liens qu'ils entretiennent, On utilise pour chaque module i :

- Le nombre de flux d'information entrant noté in_i
- Le nombre de flux d'information sortant noté out_i
- Le poids du module noté $poids_i$ calculé en fonction de son nombre de LOC et de sa complexité : $HK_i = poids_i \times (out_i \times in_i)^2$

La mesure totale HK correspond à la somme des HK_i

Exemple : A partir des in_i et out_i ci-dessous, calculer les métriques HK en supposant que le poids de chaque module vaut 1. $HK = 1110$

Module	a	b	c	d	e	f	g	h
in_i	4	3	1	5	2	5	6	1
out_i	3	3	4	3	4	4	2	6
HK_i	144	81	16	225	64	400	144	36

9.6 Les métriques Objet

L'ensemble des métriques pour la conception orientée objet a été conçu pour permettre d'effectuer une évaluation des classes d'un système. La plupart des métriques sont calculées classe par classe : aucune n'évalue le système comme un tout et le passage au système global n'est pas clair car il n'est généralement pas approprié de calculer une moyenne pour l'ensemble des classes.

1. **Méthodes pondérées par classe (WMC : Weighted Methods per Class) :**
 $WMC = 1/n \times \sum_{i=0}^n C_i \times M_i$ avec un ensemble de n classes comportant chacune M_i méthodes dont la complexité (poids) est noté C_i .
2. **Profondeur de l'arbre d'héritage (DIC : Depth of Inheritance Tree) :** distance maximale entre un noeud et la racine de l'arbre d'héritage de la classe concernée. Cette métrique est calculée pour chaque classe.
3. **Nombre d'enfants (NOC : Number Of Children) :** nombre de sous-classes dépendant immédiatement d'une classe donnée, par une relation d'héritage. Elle est calculée pour chaque classe.
4. **Couplage entre classes :** dans un contexte OO, le couplage est l'utilisation de méthodes ou d'attributs d'une autre classe. Deux classes sont couplées si les méthodes déclarées dans l'une utilisent des méthodes ou instancie des variables définies dans l'autre. La relation est symétrique : si la classe A est couplée à B, alors B l'est à A.
CBO : Coupling Between Object classes : nombre de classes couplées. Elle est calculée pour chaque classe.
5. **Réponses pour une classe (RFC) RS :** ensemble des méthodes qui peuvent être exécutées en réponse à un message reçu par un objet de la classe considérée.
 Il est constitué par la réunion de toutes les méthodes de la classe avec toutes les méthodes appelées directement par celles-ci. Elle est calculée pour chaque classe. $RFC = |RS|$
6. **Manque de cohésion des méthodes :** un module (ou classe) est cohésif lorsque tous ses éléments sont étroitement liés.
 La métrique LCOM (Lack of COhesion in Methods) tente de mesurer l'absence de ce facteur. Posons :
 - I_i l'ensemble des variables d'instance utilisées par la méthode i .
 - P l'ensemble les paires de (I_i, I_j) ayant une intersection vide.
 - Q l'ensemble des paires de (I_i, I_j) ayant une intersection non vide.
 - $LCOM = \max(|P| - |Q|, 0)$
 - LCOM peut être visualisé comme un graphe bipartite.
 - Le premier ensemble de nœuds correspond aux n différents attributs et le second aux m différentes méthodes.
 - Un attribut est lié à une fonction si elle y accède ou modifie sa valeur.
 - L'ensemble des arcs est Q . S'il y a n attributs et m fonctions alors il y a $n \times m$ arcs possibles.
 - Le cardinal de P vaut $n \times m$ moins le cardinal de Q .

9.7 Les métriques du processus

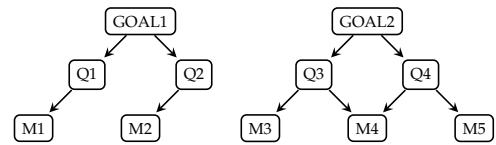
La productivité

- La productivité est une des métriques de processus les plus élémentaires.

- Elle se calcule en divisant le nombre de total de lignes de code source livrées par le nombre de jours-homme qu'a coûté le projet (une manière possible de la calculer parmi d'autres).
- Dans les années 60, la productivité de nombreux projets était de 1 LOC/jour-homme. Maintenant, ce chiffre varie entre 2 et 20 pour les projets importants et peut même être plus élevé pour les projets individuels, de petite taille.

9.7.1 L'approche GQM

- L'approche GQM (Goals, Questions and Metrics) a été mise au point par Vic Basili et Dieter Rombach à l'université de Maryland.
- L'idée est de commencer par identifier les objectifs, puis de formuler les questions liées à ces objectifs puis ensuite de développer des métriques pour mesurer des attributs liés à ces questions, L'arbre GQM comporte 3 niveaux :
 1. **Conceptuel** : Objectif (Goal) ce qu'il faut accomplir?
 2. **Opérationnel** : Question (Question) comment atteindre l'objectif?
 3. **Quantitatif** : Métriques (Metrics) métriques pour répondre aux questions.



Mécanisme :

- Il s'articule autour des points suivants :
 - Déterminer les objectifs,
 - Créer les questions,
 - Définir les métriques/mesures.
- 1. **Détermination des objectifs** : Chaque objectif vise :
 - **Objet** : Quel est l'objet que nous voulons analyser ? (par exemple, processus, produits, ressources).
 - **But** : Pour quelle raison nous analysons l'objet ? (dans le but de l'analyser, de le caractériser, de prédire, ...).
 - **Attribut de qualité** : Quel est l'attribut de qualité de l'objet à analyser ?
 - **Perspective** : De quel point de vue nous analysons l'objet ? Qui devons-nous demander ? Qui est le propriétaire des données ?
 - **Contexte/environnement** : Dans quel contexte peut-on analyser l'objet, Un moyen de créer l'objectif consiste à construire une phrase regroupant « objet, but, attribut de qualité, point de vue/perspective, environnement ».
- 2. **Créer les questions** :
- 3. **Définition des métriques** :
 - Déplacement du niveau opérationnel au niveau quantitatif
 - **Objectif** :
 - définir quelles sont les données à collecter ;
 - Créer les définitions opérationnelles ;
 - Raffiner les questions et les objectifs :
 - Les métriques peuvent être d'un des types suivants :
 - **Objectif** : compte de choses ou d'événements ;
 - **Absolu** : taille de quelque chose indépendamment d'autres choses ;

- **Explicite** : obtenu directement ;
- **Dérivé** : calculé à partir explicites et/ou dérivés ;
- **Dynamique** : lié au temps Statique : indépendante du temps.
- D'où **Goal1** : Analyser le processus de test unitaire pour comprendre l'impact d'ajouter des tests supplémentaires au projet K, selon le point de vue du chef de projet.

Caractéristiques :

- Les objectifs peuvent être définis de manière exhaustive.
- Les objectifs peuvent être affinés en questions et des questions en mesures.
- Ce raffinement (top-down) permet une explicite et complète définition des objectifs.
- Ce raffinement peut être fourni avec des hypothèses d'interprétation (bottom-up)

Exemple :

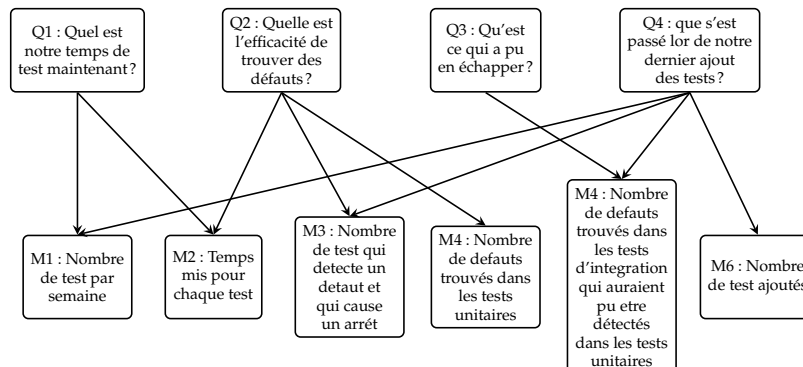
Analyser le processus de test unitaire pour comprendre l'impact d'ajouter des tests supplémentaires au projet K, selon le point de vue du chef de projet.

1. Détermination des objectifs :

- Objet : processus de test unitaire ;
- But : comprendre ;
- Attribut de qualité : l'impact d'ajouter des tests supplémentaires ;
- Point de vue : chef de projet ;
- Environnement : projet K ;

2. Création des questions :

- Déplacement du niveau conceptuel au niveau opérationnel ;
- Clarification de l'objectif ;
- Implication de tous les acteurs ;
- Les questions possibles à se poser sont :
 - Q1 : Quel est notre temps de test maintenant ?
 - Q2 : Quelle est l'efficacité de trouver des défauts ?
 - Q3 : Qu'est ce qui a pu en échapper ?
 - Q4 : que s'est passé lors de notre dernier ajout des tests ?



9.8 Conclusion

- Les mesures du logiciel consistent à assigner des nombres ou des symboles aux attributs des entités du logiciel (produit, processus). Les métriques proposées concernent aussi bien l'approche classique que l'approche orientée objet conçue pour le développement du logiciel.
- Les mesures du logiciel sont importantes et visent à contrôler certains attributs du logiciel dans le but d'améliorer la qualité des logiciels ainsi que d'anticiper et réduire les besoins futurs de maintenance.
- L'approche GQM est un mécanisme permettant de définir et d'interpréter un logiciel mesurable et opérationnel. Elle peut être utilisée de manière isolée ou encore mieux dans le contexte plus général visant à l'amélioration de la qualité du logiciel.
- L'approche GQM combine en soi la plupart des méthodes actuelles de mesure et les généralise de façon à intégrer les processus et les ressources ainsi que les produits. Ce qui la rend adaptable à différents environnements, tel que confirmé par le fait qu'elle a été appliquée dans plusieurs organisations (Nasa, Hewlett Packard, Motorola).