Centre universitaire Abdelhafid Boussouf - Mila
Institut des sciences et de technologie
Département de Mathématiques et informatique

Polycopié de la matière :

Gestion de projets informatiques

Cours dispensé en Deuxième année Master Sciences et Technologie de l'Information et de la Communication (STIC)

2022

Réalisé par : Dr. Sadek Benhammada
s.benhammada@centre-univ-mila.dz

Table de matière

Liste des tableaux

Liste des figures

INTRODUCTION GENERALE

Le développement de logiciels informatiques est une entreprise complexe, soumise à une série de contraintes de calendrier, de budget et d'organisation, surtout si elle implique de nombreuses personnes travaillant sur une période relativement longue. C'est pourquoi les projets logiciels doivent être gérés. La gestion de projet logiciel est la discipline de la planification et de la direction de projets logiciels. C'est une sous-discipline du génie logiciel dans laquelle les projets logiciels sont planifiés, mis en œuvre, surveillés et contrôlés.

Ce cours vise à fournir aux étudiants une connaissance suffisante des principes de base, et des méthodes, techniques et outils de la gestion de projet logiciel, pour qu'ils puissent aider un chef de projet à exercer les activités de gestion de projet, et comprendre le contexte dans lequel la gestion de projet est menée. Bien que l'objectif principal est la gestion de projets logiciels, bon nombre des compétences acquises sont utiles pour la gestion de projets en général.

Ce cours est organisé en six chapitres qui couvrent des techniques, des méthodes et des outils des principales activités de gestion de projets:

1. Le premier chapitre met en relief les concepts et les principes de base de la gestion de projets, les principaux acteurs d'un projet, et les principales activités de gestion de projets.
2. Le deuxième chapitre se focalise sur l'activité de découpage de projets logiciels, les critères de découpage, les modèles de processus logiciels, et les organigrammes de découpage.
3. Le troisième chapitre introduit les techniques d'estimation de coûts et de durée des projets logiciels. Après avoir passé en revue les différentes techniques d'estimation de projets logiciel, l'accent est mis sur le modèle COCOMO et la méthode de points de fonction.
4. Le quatrième chapitre couvre les concepts et les techniques de planification de projets, particulièrement, la méthode des antécédents comme technique d'ordonnancement, et le diagramme de Gantt pour élaborer le planning du projet qui prend en compte les contraintes de ressources et de calendrier.

5. Le cinquième chapitre aborde les concepts et techniques de suivi et de pilotage de projets, notamment le tableau de bord et ses indicateurs de suivi individuel et au niveau du projet. Le chapitre couvre aussi la méthode de points de la valeur acquise particulièrement adaptée au grands projets.
6. Le chapitre six est un tutoriel de l'outil de planification et de suivi de projets Microsoft Project.

Chapitre 1

INTRODUCTION A LA GESTION DE PROJETS

1 Définition

Pour mettre en évidence les concepts de base de la gestion de projet, nous avons besoin d'une définition comme point de départ pour comprendre ce qu'est un projet. Souvent, les gens appellent tout travail qu'ils ont à faire un projet, cependant, les projets ont en fait une définition spécifique. Si un ensemble de tâches ou de travaux à effectuer ne répond pas à la définition stricte, alors il ne peut pas être appelé un projet.

Un projet est un **processus unique**, qui consiste en un **ensemble d'activités** coordonnées et maitrisées comportant des dates de début et de fin, entrepris dans le but d'atteindre un **objectif** défini avec des **moyens** adaptés et dans un **délai** donné[1]

La définition d'un projet comporte ses caractéristiques clés :

- **Un projet est un processus unique.** L'unicité du processus projet doit être comprise de deux façons. D'une part, les activités sont définies de façon à prendre en compte les particularités du projet, et sont différents des activités des autres projets même si l'on réutilise des trames générales. D'autre part, les activités ne seront exécutées qu'une seule fois, à la différence de processus constitués d'activités répétitives (Exemple : La production en série). Il y a donc unicité au niveau du type et au niveau de l'instance.
- **Un projet a un objectif unique.** Un projet doit avoir un objectif à caractère novateur et n'est pas répétitif. Par exemple, de nombreuses entreprise réalisent des projets de construction d'immeubles, cependant, chaque immeuble est unique. Généralement, un

[1] ISO 10006 (2003) : « Systèmes de management de la qualité - Lignes directrices pour le management de la qualité dans les projets »,

projet doit avoir un seul objectif, cependant, les projets importants ou complexes peuvent être divisés en plusieurs sous-projets, chacun ayant son propre objectif.

- **Un projet comporte un ensemble d'activités.** Un projet comprend un certain nombre d'activités qui doivent être réalisées dans un ordre ou une séquence spécifiée. Pour déterminer la séquence, il est utile de penser en termes d'entrées et de sorties. La sortie d'une activité ou d'un ensemble d'activités devient l'entrée d'une autre activité ou d'un autre ensemble d'activités.
- **Un projet est temporaire.** Les projets sont temporaires, à la différence des processus qui sont continus, et se répètent. Les projets ont un délai, une date de début et une date de fin spécifiées. Ces dates peuvent être auto-imposées par la direction du projet ou spécifiée de l'extérieur par un client ou un organisme.
- **Un projet nécessite des ressources.** La réalisation d'un projet nécessite des ressources humaines et matériels. Les ressources affectées à un projet sont limitées, elles peuvent être exceptionnellement ajustées à la hausse ou à la baisse par la direction.

2 Contraintes du projet

Les projets doivent être réalisés et livrés sous certaines contraintes. Ces contraintes ont été répertoriées en termes de **l'objectif**, de **temps** et de **coût**.

- La contrainte de temps fait référence au temps disponible pour terminer un projet.
- La contrainte de coût fait référence au montant budgété disponible pour le projet.
- La contrainte de l'objectif fait référence à ce qui doit être fait pour produire le résultat final du projet.

Ces trois variables sont interdépendantes, i.e., si l'une d'entre elles change, au moins une autre variable doit également changer pour rétablir l'équilibre du projet. Pour exprimer l'interdépendance entre les trois contraintes, elles sont souvent représentées par un triangle dit « Triangle projet », où chaque côté représente une contrainte (Figure 1.1).

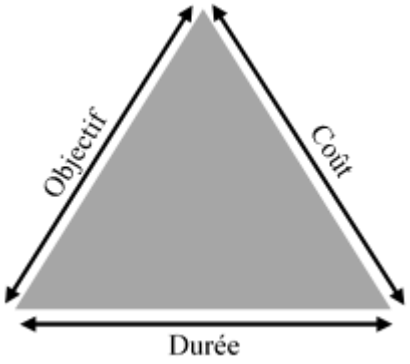


Figure 1.1. Triangle du projet

Ainsi, une extension de l'objectif signifie généralement une augmentation des coûts et des délais nécessaires pour réaliser le projet, une augmentation de la durée peut signifier une augmentation des coûts (Figure 1.2), une réduction du budget peut entraîner une augmentation du temps et une réduction de l'objectif, etc.

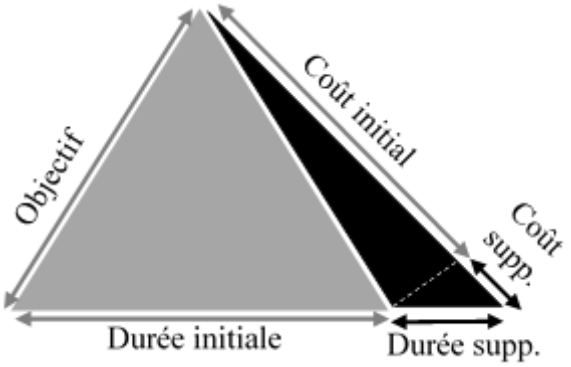


Figure 1.2. Impact d'un changement de la durée du projet

3 Gestion de projet

Le **Project Management Institute** (PMI)[2], définit formellement la gestion de projet comme suit :

« La gestion de projet est l'application de connaissances, de compétences, d'outils et de techniques aux activités du projet pour répondre aux exigences du projet ».

[2] Le Project Management Institute est une association professionnelle qui propose des méthodes de Gestion de projet, et publie des standards relatifs à la gestion de projet.

L'IPMA[3] donne la définition de management de projet :

La gestion de projet consiste à planifier, organiser, suivre et maîtriser tous les aspects d'un 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. Cela inclut les tâches de direction nécessaires aux performances du projet. »

La discipline de la gestion de projet consiste à fournir les outils et les techniques qui permettent à l'équipe de projet de s'assurer que les objectifs du projet seront atteints suivant les contraintes fixées (Section 2). Les trois contraintes de projets (Objectif, durée, moyens) représentées par le triangle Projet (Figure 1.1) doivent être mis sous contrôle. Chacune fait l'objet d'une gestion spécifique, qui prend en compte l'existence des deux autres, ainsi, aux trois contraintes de projets corresponds trois types de gestion (Figure 1.3):

3.1 La gestion du temps.

La contrainte de délai donne lieu à une gestion du temps. La gestion du temps comprend les processus et les techniques utilisés pour maîtriser la consommation de l'enveloppe temps, afin d'assurer l'achèvement du projet dans les délais, via l'élaboration et la gestion du calendrier du projet qui doit indiquer les dates de début, la date de fin, et les ressources requises pour chaque tâche.

3.2 La gestion des ressources.

Les contraintes du coût de projet donnent lieu à une gestion de ressources, en fait, le budget du projet est transformé en ressources humaines et matérielles. Dans les projets logiciels, les ressources humaines occupent une place primordiale. La fonction de la gestion de ressources consiste à optimiser leur utilisation, constituer des équipes efficaces, affecter les personnes au moment adéquat en fonction de leurs compétences, coordonner les travaux, assurer la communication d'équipe, résoudre les conflits, l'évaluation du rendement et la formation.

[3] The International Practice Management Association (IPMA)

3.3 La gestion de la production.

L'objectif du projet donne lieu à une gestion de la production, qui a pour but de suivre et diriger l'avancement vers l'objectif tout au long du projet.

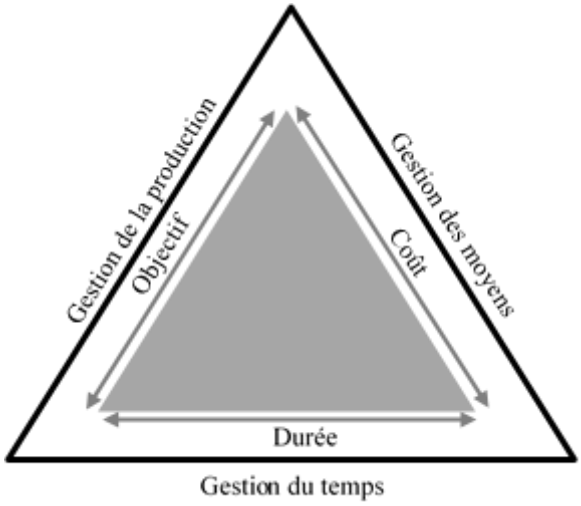


Figure 1.3. Triangle de gestion de projet

4 Acteurs d'un projet

Un acteur du projet est une personne physique ou morale qui prend en charge des responsabilités et des activités à effectuer dans le cadre de la réalisation du projet. L'ensemble des acteurs impliqués dans un projet s'appelle les parties prenantes (En anglais : Project stakeholders). Quand on organise un projet, on commence par déterminer les besoins et les attentes des principales parties prenantes. Les parties prenantes du projet comprennent généralement le maitre d'ouvrage le maitre d'œuvre, le chef du projet et les membres de l'équipe au sein du maitre d'œuvre, les utilisateurs du produit final, et les partenaires externes (Figure 1.4).

4.1 Le maître d'ouvrage (MOA)

Le maître d'ouvrage (MOA) (En anglais: Owner), est la personne physique ou morale (entreprise, direction etc.) qui sera le propriétaire de **l'ouvrage**. **L'ouvrage** est le résultat concret d'un projet. Me MOA fixe les objectifs, l'enveloppe budgétaire et les délais souhaités pour le projet.

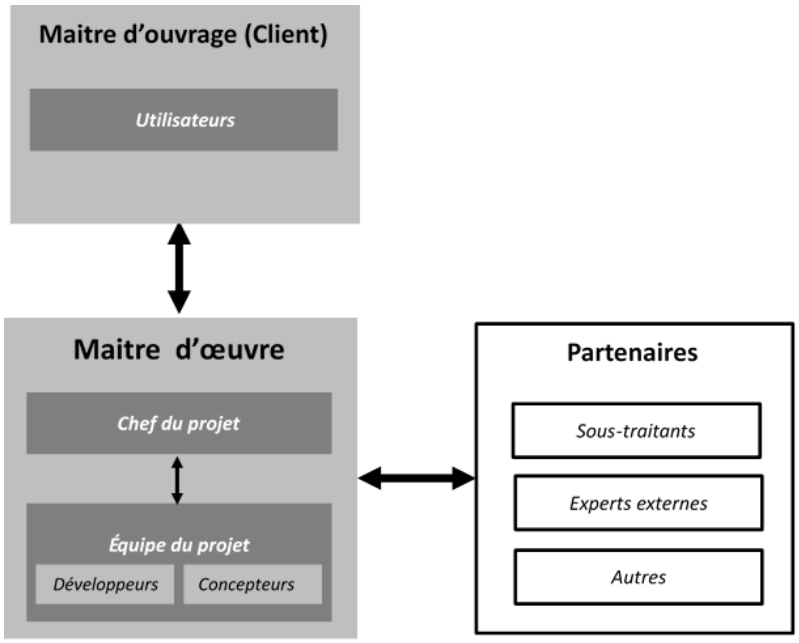


Figure 1.4. Acteurs d'un projet

4.2 Le maître d'œuvre (MOE)

Le maître d'œuvre (MOE) est la personne physique ou morale (entreprise, direction, etc.) qui réalise l'ouvrage pour le compte du maître d'ouvrage et qui assure la responsabilité globale de la qualité technique, du délai et du coût. **L'œuvre** est le processus de réalisation de l'ouvrage.

Le maître d'œuvre et maître d'ouvrage d'un projet sont liés par une relation contractuelle qu'on appelle **cahier des charges**. Le cahier des charges d'un projet est un document construit, compréhensible pour tous les acteurs du projet, exhaustif au sujet de fonctionnalités et les caractéristiques attendues du système à développer.

Le maître d'ouvrage représente le client, il établit un **cahier des charges** qui peut servir de base à un appel d'offres. Après discussions sur une ou plusieurs propositions reçues, il passe contrat avec un fournisseur qui deviendra le maître d'œuvre. Celui-ci est responsable de la conduite du projet. Le maître d'ouvrage assure un suivi de l'avancement du projet, selon des modalités contractuellement définies. À chaque fourniture contractuelle par le maître d'œuvre, il procède à la recette et prononce l'acceptation ou le refus. Il pilote la mise en œuvre du projet.

4.3 Utilisateurs

Un utilisateur ((En anglais :User) est la personne qui utilisera les livrables du projet, Leurs rôle est l'expression des besoins et des contraintes lors de l'initialisation du projet, et la validation du produit final (s'assurer qu'il est conforme à son cahier des charges).

4.4 Equipe de projet

L'équipe de projet est l'ensemble des personnes placées sous l'autorité directe ou indirect du chef de projet. Mais, on peut parfois considérer que l'équipe de projet s'étend à toutes les personnes participant à la réalisation du projet. Dans les projets logiciels, on distingue ainsi trois types d'acteurs dans l'équipe de projet :

- Le ***chef de projet***, est responsable devant le maître d'œuvre de l'avancement du projet.
- Le ***concepteur***, qui peut être tenu par un informaticien, et qui joue le rôle d'un organisateur ou un gestionnaire selon le stade d'avancement : sa responsabilité est de concevoir le futur système aux étapes étude préalable et étude détaillée.
- Le ***développeur***, qui est tenu par un informaticien : sa responsabilité est d'écrire les programmes ou de réaliser un prototype. Pour certains développements réalisés en langage de 4e génération, le rôle peut être tenu par un gestionnaire.

4.5 Les acteurs externes

Dans le cadre d'un projet, il est possible de faire recours à des acteurs externes, qui peuvent être des sous-traitants ou des experts :

- **Sous-traitant.** Une entreprise tierce qui prend en charge une partie du projet (étude, réalisation, formation, etc.), pour le compte du maitre d'œuvre, et sur la base d'un contrat.
- **Expert**. Des experts externes peuvent être sollicité pour traiter des problèmes précis (estimation de charge, planification, etc.). L'expert intervient ponctuellement pour donner des recommandations.

4.6 Le chef de projet

Le chef du projet est la personne physique chargée par le maître d'œuvre, dans le cadre d'une mission définie, d'assurer la maîtrise du projet, c'est-à-dire de veiller à sa bonne réalisation dans les objectifs de technique, de coût et de délai.

Bien que les responsabilités précises d'un chef de projet varient d'une entreprise à l'autre et d'un projet à l'autre, elles doivent toujours inclure la planification et la prévision. Parmi les tâches dont se charge le chef de projet:

Les responsabilités interpersonnelles, qui comprennent :

- Animer et en maintenir la cohésion de l'équipe. ;
- Assurer la communication avec les membres de l'équipe du projet et les parties prenantes;
- Donner l'exemple à l'équipe de projet et représenter le projet lors d'occasions formelles.

Les responsabilités en matière d'information, qui comprennent :

- Le suivi de la performance du personnel et de la mise en œuvre du plannning du projet ;
- La diffusion des informations sur les tâches à l'équipe de projet ;
- La diffusion des informations sur l'état d'avancement du projet à la direction générale ;
- Agir en tant que porte-parole de l'équipe de projet.

Les responsabilités décisionnelles, qui comprennent :

- Allocation des ressources conformément au plan du projet et ajustement de ces allocations lorsque les circonstances l'exigent (le chef de projet est responsable du budget) ;
- Négociations avec les responsables concernant la gestion optimale des ressources, et avec le personnel du projet au sujet de leurs tâches ;
- Gestion des perturbations au bon déroulement du projet telles que les défaillances des équipements et de personnel.

5 Gestion de Projets logiciel

Le développement de logiciels est une activité dont l'objectif est la création ou la personnalisation (maintenance) d'applications, de frameworks ou d'autres composants logiciels. Le processus développement de logiciel implique les activités de de spécification, de conception, de programmation, de documentation, et de test.

Les projets de développement de logiciels nécessitent une gestion dans le but d'atteindre les objectifs du projet, tout en respectant les contraintes de budget et de calendrier.

La gestion de projets logiciels est le processus de planification, d'organisation, de surveillance, de contrôle et de direction d'un projet logiciels. Il s'agit d'une sous-discipline de gestion de projets et du génie logiciel.

Ainsi, pour comprendre l'utilité de la gestion de projets logiciels, nous allons d'abord revenir sur la crise logiciel, qui a conduit à la naissance de la discipline du génie logiciel.

5.1 Crise logiciel et génie logiciel

La « crise du logiciel » des années 1960 et 1970 a été appelée ainsi en raison d'une série d'échecs de projets logiciels. La crise s'est manifestée de plusieurs manières :

- Le budget de et le délai de livraison sont souvent dépassés dans les projets de développement de logiciels.
- Les logiciels livrés sont souvent de qualité insatisfaite, et ne répondent pas aux exigences des utilisateurs.
- La maintenance de logiciels est difficile est souvent coûteuse et à l'origine de nouvelles erreurs.

En 1995, une étude du Standish Group dressait un tableau accablant de la conduite des projets informatiques. Reposant sur un échantillon représentatif de 365 entreprises, totalisant 8380 applications, cette étude établissait que :

- 16,2% seulement des projets étaient conformes aux prévisions initiales,
- 52,7% avaient subi des dépassements en coût et délai d'un facteur 2 à 3 avec diminution du nombre des fonctions offertes,
- 31,11% ont été purement abandonnés durant leur développement.

Pour les grandes entreprises (qui lancent proportionnellement davantage de gros projets), le taux de succès est de 9% seulement, 37% des projets sont arrêtés en cours de réalisation, 50% aboutissent hors délai et hors budget.

La crise a donné lieu à l'avènement d'une nouvelle discipline : le génie logiciel qui cherche à établir et à utiliser des méthodes et des outils dans le but de développer économiquement du logiciel qui est fiable et qui fonctionne efficacement sur des machines réelles.

5.2 Génie logiciel et gestion de projets

Le génie logiciel peut se définir comme un ensemble d'activités permettant à un maître d'œuvre de produire un logiciel répondant aux besoins exprimés par un maître d'ouvrage dans les délais et les coût prévus.

Les activités du génie logiciel peuvent être subdivisées en trois groupes : les activités d'ingénierie du produit, les activités de vérification et de validation, et les activités de gestion technique[4]. Ces trois groupes contiennent les activités majeures se déroulant en parallèle, visant à produire, vérifier et piloter, sans être concentrées dans une seule phase du cycle de vie :

1. **Les activités d'ingénierie du produit *(product engineering function)* :** Fait référence aux activités qui sont nécessaires pour spécifier, concevoir et réaliser le logiciel qui constitue le produit final. (Analyse des besoins, Conception, Codage, Intégration, Maintenance, etc.).
2. **Les activités de vérification et de validation *(verification and validation function)*:** ce sont les activités techniques déployées pour déterminer si les produits de chaque phase de développement sont conforme aux exigences et si le système ou composant final est conforme aux exigences spécifiées (revue, audit, test, etc.).
3. **Les activités de gestion technique *(technical management function)* :** ce sont celles qui planifient, structurent et pilotent les activités d'ingénierie (L'estimation, la planification, le pilotage, etc.). Les activités de gestion sont déployées pour gérer le processus, le produit et les ressources.
   - **La gestion du processus** *(process management)* comprend la direction, le suivi et la coordination des travaux entrepris pour développer un produit ou fournir un service. L'assurance de qualité est un exemple de gestion d'un processus.
   - **La gestion du produit** *(product management)* comprend la définition, la coordination et le suivi des caractéristiques d'un produit pendant son développement. La gestion de la configuration en est un exemple.
   - **La gestion des ressources** (*resource management*) comprend l'identification, l'estimation, l'allocation et la surveillance de ressources utilisées pour développer un produit ou fournir un service.

[4] IEEE Std 610.12-1990; IEEE Standard Glossary of Software Engineering Terminology. 1990
Strohmeier, Alfred. "Cycle de vie du logiciel." Ecole Polytechnique Fédérale de Lausanne 28 (2000)

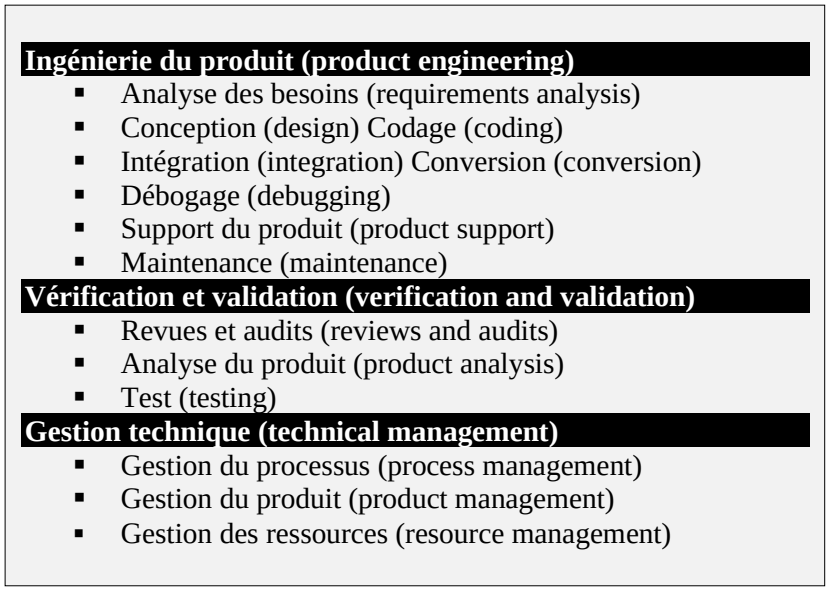


Figure 1.5. Classification des activités du génie logiciel

La gestion de projet logiciel est donc, une partie essentielle du génie logiciel qui implique les activités de gestion technique. Les projets logiciels doivent être gérés car le génie logicielle professionnel est toujours soumise à des contraintes de budget et de calendrier organisationnels. L'objectif des activités de gestion est de s'assurer que le projet logiciel respecte et surmonte ces contraintes, et de fournir un logiciel de haute qualité.

Une bonne gestion ne peut garantir le succès du projet. Cependant, une mauvaise gestion entraîne généralement l'échec du projet : le logiciel peut être livré en retard, coûter plus cher que prévu à l'origine ou ne pas répondre aux attentes des clients. Les critères de réussite pour la gestion de projet varient évidemment d'un projet à l'autre, mais, pour la plupart des projets, les objectifs importants sont :

- de livrer le logiciel au client à la date prévue ;
- maintenir les coûts globaux dans les limites du budget ;
- livrer un logiciel répondant aux attentes du client ;
- maintenir une équipe de développement cohérente et performante.

5.3 Activités de gestion de projets

La gestion de projet logiciel comprend de nombreuses activités, notamment l'estimation, la planification, le pilotage, la gestion des risques, et la gestion des ressources :

Le découpage de projets. Implique le découpage temporel du projet en phases et tâche, et le découpage structurel en modules.

L'estimation de la charge et de la durée nécessaires à la réalisation des du projet, ainsi que la charge et la durée nécessaire pour réaliser chaque tâche.

La planification du projet : l'élaboration d'un calendrier qui définit les dates de début et de fin du projet, ainsi que la date de début et de fin de chaque tâche.

Le pilotage de projet : comprend le suivi de l'avancement du projet, en quantité et en qualité, ainsi que l'analyse et le traitement des écarts entre l'avancement réel et l'avancement prévu.

La gestion des ressources : Les ressources nécessaires à la réalisation d'un projet peuvent être des ressources humaines ou matérielles. La gestion des ressources comprend la création d'une équipe de projet, l'attribution des responsabilités à chaque membre de l'équipe, et l'ajustement des ressources.

Gestion des risques. La gestion des risques comprend l'identification et l'évaluation des risques qui peuvent affecter le projet, la surveillance des risques tout au long du projet, et la préparation d'un plan pour éviter ou minimiser les effets des risques.

Chapitre 2.

DECOUPAGE DE PROJETS LOGICIELS

1 Introduction

Nous avons expliqué dans le chapitre précédent que la gestion du projet concerne les trois contraintes : Objectif, coût et délai. L'une des tâches principales du chef d'un projet logiciel consiste à découper le projet en phases et tâches pour pouvoir repartir dans le temps la production et les ressources. L'origine du découpage de projets provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation. Le découpage du projet permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés.

2 Définition

Le découpage d'un projet est le processus qui consiste à subdiviser les livrables du projet et le travail du projet en parties plus petites et plus gérables. Ce processus est effectué une ou plusieurs fois au cours de réalisation d'un projet. Une partie se définit par les caractéristiques suivantes :

- Chaque partie du projet donne lieu à un résultat défini ;
- La charge et la durée nécessaire à la réalisation de chaque partie peut être évaluée ;
- Les contraintes d'enchaînement entre les parties sont identifiables : certaines parties peuvent être réalisés parallèlement, d'autres sont liés par des contraintes d'enchainement (une partie ne peut commencer qu'après la fin de ses prédécesseurs).
- Une partie peut éventuellement être découpée elle-même en sous-parties.

3 Critères de découpage de projet

Deux critères sont utilisé pour découpage un projet : temporel (succession d'étapes et de phases) ou structurelle (modularisation).

3.1 Découpage temporel

Il permet de répartir le travail dans le temps. Le projet peut être découpé en étapes ou phases, chaque étape est découpée en phases ; et chaque phase est découpée en tâches. Chaque étape, phase ou tâche comporte une date de début et une date de fin prévues, et un résultat défini (Figure 2.1). L'ensemble des phases d'un projet s'appelle le cycle de vie du projet (Section 4).

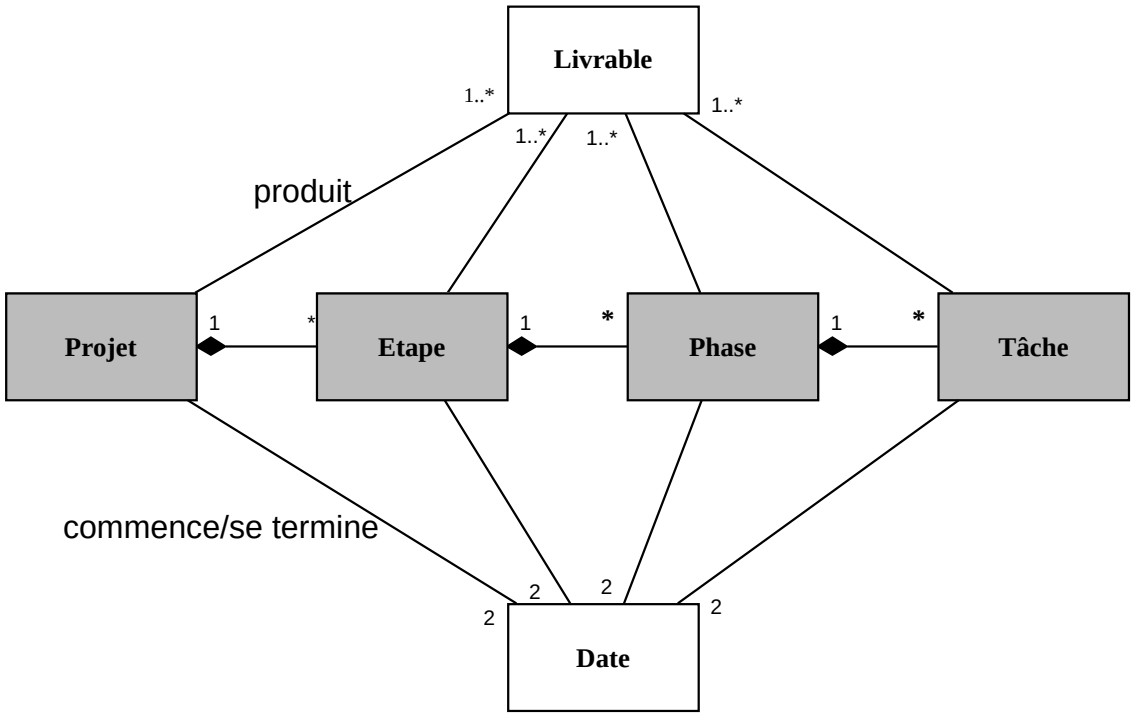


Figure 2.1. Découpage temporel d'un projet

3.2 Découpage structurel

Le Découpage structurel permet d'organiser le travail en se basant sur la structure du produit final. Le projet est découpé en modules ; un module peut être, à son tour, découpé en d'autres modules (Figure 2.2).

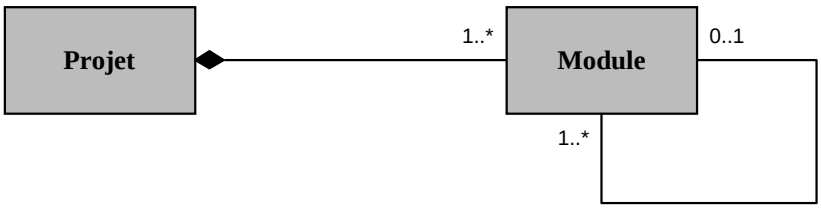


Figure 2.2. Découpage structurel du projet

4 Processus logiciel

Un processus logiciel (software process) ou « cycle de vie logiciel » (software life cycle), définit l'enchainement des étapes à suivre pour le développement d'un logiciel (découpage temporel). L'objectif d'un tel processus est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel et la vérification du processus de développement, ainsi, le processus apporte stabilité, contrôle et organisation au développement de logiciel, qui peut, si elle n'est pas contrôlée, devenir assez chaotique[5]. Les processus garantissent donc que les efforts déployés par les concepteurs, les développeurs et les testeurs seront bien orientés vers les buts communément convenus d'un projet[6].

4.1 Phases de processus logiciels

Le processus logiciel est découpé en phases discrètes, chaque phase a des entrées et des sorties, qui sont généralement des documents ou produits (code). Le découpage classique du cycle de développement comprend les phases : **analyse**, **conception**, **implémentation** et **test unitaire**, **intégration et test d'intégration**, et éventuellement phase **d'installation**. D'autres noms sont parfois donnés à ces phases, aussi, deux phases consécutives peuvent être fusionnées en une seule phase, également, un découpage plus fin est souvent utilisé.

Les phases du processus logiciel sont organisées différemment dans différents processus de développement. Dans le modèle en cascade, elles sont organisées en séquence, alors que dans le développement incrémental, elles sont imbriquées (Section 4.2).

[5] Roger, S.P. and R.M. Bruce, *Software engineering: a practitioner's approach*. 2015: McGraw-Hill Education
[6] Unhelkar, B., *Software engineering with uml*. 2017: CRC Press.

4.1.1 Phase d'analyse (en anglais requirements phase, analysis phase)

Lors de cette phase on analyse les besoins fonctionnels et les contraintes en consultation avec les utilisateurs du logiciel à développer. Le résultat de la phase d'analyse est consigné dans un document appelé *cahier des charges du logiciel* (ne pas confondre avec le cahier des charges du projet) ou spécification du logiciel, (en anglais: *software requirements, software specification* ou *requirements specification*). Une spécification comporte les éléments suivants:

- Description de l'environnement du logiciel;
- Spécification fonctionnelle (functional specification), qui définit toutes les fonctions que le logiciel doit offrir;
- Performances requises *(performance requirements)*, par exemple: temps de réponse, encombrement en mémoire, sécurité de fonctionnement;
- Interfaces avec l'utilisateur *(user interface)* ;
- Interfaces avec d'autres logiciels;
- Contraintes de réalisation, telles que l'environnement de développement, le langage de programmation à utiliser, etc.

4.1.2 Phase de conception *(design phase)*

A partir de spécification élaborée lors de la phase d'analyse, la phase de conception doit fournir une architecture détaillée du système à développer. La phase de conception est souvent découpée en deux phases : phase conception générale ou architecturale (*preliminary design* ou *architectural design*), et la phase de conception détaillée *(detailed design)*.

Un système est souvent découpé en modules pour faciliter son développement, ainsi, l'objectif de la phase conception générale est l'élaboration d'une architecture globale qui identifie les modules qui constituent le système, et les interactions entre ces modules.

Lors de la phase de conception détaillée, les modules qui constituent le système seront décomposés en modules plus élémentaires. La décomposition est poursuivie jusqu'au niveau où les modules sont faciles à implémenter et à tester. Il faut ensuite décrire chaque module logiciel en détail.

Pendant la conception détaillée, il faut également préparer la vérification des composants logiciels élémentaires qui fera l'objet de la phase des tests unitaires. Le résultat est consigné dans un document appelé *plan de tests unitaires*.

4.1.3 Phase d'implémentation et test unitaire (*implementation/coding and unit testing*)

Au cours de cette phase, les modules identifiés et décrits lors de la conception du logiciel sont implémentés sous la forme d'un ensemble de programmes ou d'unités de programmes. L'implémentation est suivie du test unitaire. Lors des tests unitaires *(unit test)*, on vérifie chaque composant pour s'assurer qu'il est conforme à la conception détaillée.

4.1.4 Intégration et test du d'integration

A cours de cette phase, les unités de programme sont intégrées et testées progressivement pour s'assurer que les exigences du logiciel ont été satisfaites. D'abord, on construit des composants par intégration de composants plus petits, ensuite ont test les nouveaux composants assemblés, et on répète la procédure jusqu'à l'assemblage de tous les composants qui constituent le système.

4.1.5 Phase d'installation (installation phase)

Après avoir intégré le logiciel, on peut l'installer dans son environnement d'exploitation, ou dans un environnement qui simule cet environnement d'exploitation, et le tester pour s'assurer qu'il est conforme à la spécification élaborée lors de la phase d'analyse (Test de validation ou de réception).

4.2 Les modèles de processus logiciels

Un modèle de processus logiciel est une description abstraite de haut niveau des étapes d'un processus logiciel, qui peuvent être étendus et adaptés pour créer des processus de génie logiciel plus spécifiques[7].

Il existe de nombreux modèles de processus, vu qu'il n'est pas possible d'appliquer un processus unique pour tous les projets logiciels. Le choix du processus approprié du type de logiciel développé, de l'expérience et de la compétence des développeurs et du type d'organisation développant le logiciel.

4.3 Le modèle code-and-fix

Ce modèle est basé sur l'hypothèse de la possibilité d'une détermination facile des besoins : Le développement commence avec peu de planification initiale, suivie immédiatement d'une phase

[7] Sommerville, I., 2016. *Software engineering*. Harlow, England: Pearson Education Limited

de Programmation ; puis une phase de Mise au point, parfois en collaboration avec l'utilisateur du futur système, si le résultat est non satisfaisant on revient à la phase de programmation, jusqu'à ce que le résultat soit satisfaisant (Figure 2.3).

Le modèle code-and-fix est un choix tentant lorsqu'on est confronté à un calendrier de développement serré, car on commence directement à coder la solution, cependant, si on rencontre des problèmes architecturaux majeurs tard dans le processus, on doit généralement réécrire de grandes parties du code. Le modèle code-and-fix n'est donc approprié que pour les petits projets qui ne sont pas destinés à servir de base à un développement futur.

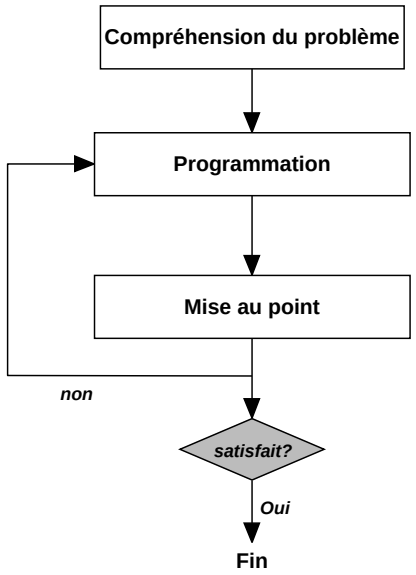


Figure 2.3. Modèle Code-and-fix

4.4 Le modèle en cascade (waterfall)

Le modèle en cascade est le modèle classique du génie logiciel, et il est totalement opposé au modèle code-and-fix, étant donné que le modèle met l'accent sur la planification dès les premières étapes. Le modèle en cascade se déroule en une succession de phases (Figure 2.4), ainsi, il commence par la spécification du système et se poursuit par la conception architecturale, la conception détaillée, le codage, l'intégration et l'installation. Les principes de ce modèle sont les suivants :

- Toutes les phases sont exécutées sans exception dans l'ordre indiqué.
- Chaque phase se termine par une tâche de vérification et validation des résultats (afin d'éliminer les anomalies et les incohérences).
- Passage à la phase suivante si les résultats sont validés.

- Retour uniquement à la phase précédente : on ne peut changer que les décisions prises à la phase précédente.

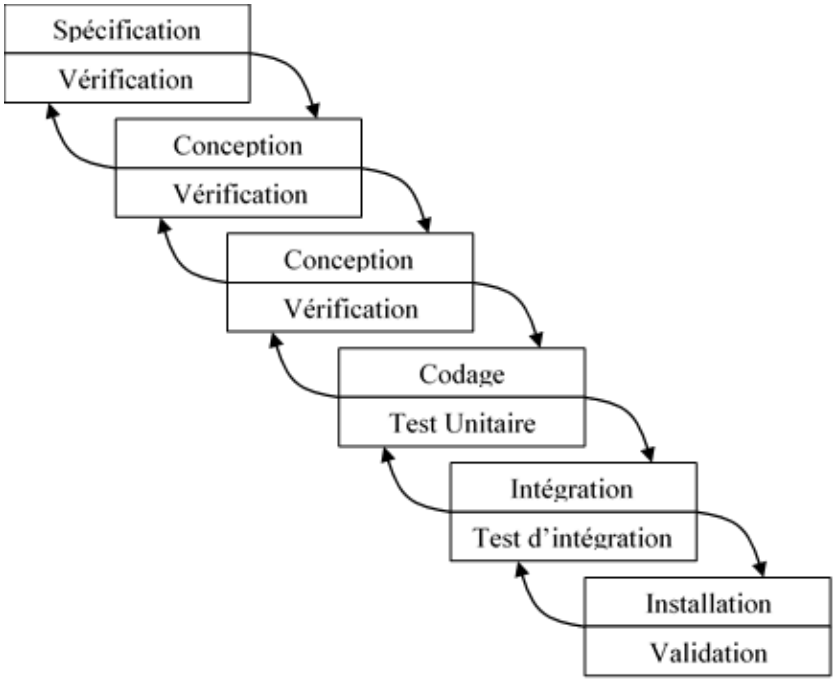


Figure 2.4. Modèle en cascade

Étant donné que le modèle en cascade utilise beaucoup de documents et de planification, il fonctionne bien pour les projets dans lesquels le contrôle de la qualité est une préoccupation majeure. Cependant, ce modèle est fondé sur l'hypothèse souvent irréaliste que l'on peut dès le départ identifier tous les besoins. La pratique montre qu'il est difficile d'obtenir de la part de l'utilisateur un énoncé complet et cohérent de ses besoins.

4.5 Modèle en V

Le modèle en V est une amélioration du modèle en cascade. Il repose sur l'association d'une phase de test à chaque phase de développement, où, chaque phase de test est préparée dès la phase de développement correspondant (Figure 2.5.).

Ceci rend explicite la préparation des phases de test par les phases de développement, et permet de mieux approfondir le développement et de mieux planifier les tests. Malgré l'amélioration par rapport au modèle précédent, l'inconvénient de ce modèle reste que le client ne voit le produit que lorsque celui-ci est terminé, avec tous les risques que cela comporte.

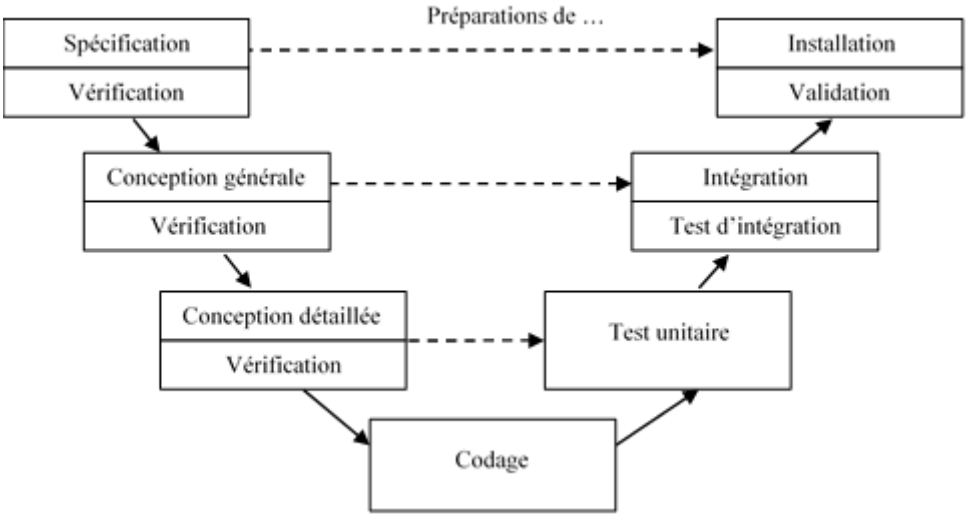


Figure 2.5. Modèle en V

4.6 Modèle incrémental

Dans les modèles précédents un logiciel est décomposé en composants développés séparément et intégrés à la fin du processus. Le développement incrémental est basé sur l'idée de développer un ensemble de composants initial appelé « incrément », et de faire évoluer le logiciel à travers le développement de nouveaux incréments qui viennent s'intégrer aux précédents jusqu'à ce que le système requis ait été développé (Figure 2.6). Chaque incrément peut donner lieu à un cycle de vie classique plus ou moins complet, ainsi, les phases de spécification, de développement et de test sont entrelacées plutôt que séparées, avec un retour d'information rapide entre les phases. Chaque incrément ou version du système intègre certaines des fonctionnalités dont le client a besoin. Généralement, les premiers incréments du système incluent la fonctionnalité la plus importante ou la plus urgente. Cela signifie que l'utilisateur peut évaluer le système à un stade relativement précoce du développement.

Le modèle incrémental est adapté aux situations dans lesquelles les exigences logicielles initiales sont raisonnablement bien définies, mais l'effort de développement n'est pas suffisant pour développer l'ensemble des fonctionnalités du système en appliquant un processus purement linéaire[8].

Le développement incrémental sous une forme ou une autre est maintenant l'approche la plus couramment appliquée pour le développement de systèmes d'application et de produits

[8] Roger, S.P. and R.M. Bruce, *Software engineering: a practitioner's approach*. 2015: McGraw-Hill Education

logiciels. Cette approche peut être soit pilotée par un plan, agile ou, plus généralement, un mélange de ces deux approches. Dans une approche axée sur le plan (classique), tous les incréments du système sont identifiés à l'avance ; dans une approche agile, les premiers incréments sont identifiés, mais le développement des incréments ultérieurs dépend des progrès et des priorités du client[9].

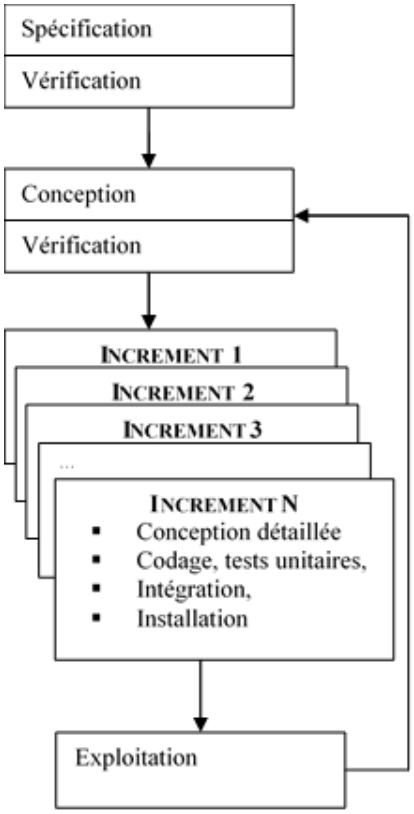


Figure 2.6. Modèle incrémental

Le modèle incrémental est la base des méthodes de développement agiles, préférables à une approche en cascade pour les systèmes dont les exigences sont susceptibles de changer au cours du processus de développement. C'est le cas pour la plupart des systèmes d'entreprise et des produits logiciels. Le développement incrémental reflète la façon dont nous résolvons les problèmes. Nous élaborons rarement une solution complète au problème à l'avance, mais nous nous dirigeons vers une solution en une série d'étapes, en revenant en arrière lorsque nous réalisons que nous avons fait une erreur. En développant le logiciel de manière incrémentale, il est moins coûteux et plus facile d'apporter des modifications au logiciel au fur et à mesure de son développement.

4.7 Modèle par prototypage

Souvent, un client définit un ensemble d'objectifs généraux pour le logiciel, mais n'identifie pas les exigences détaillées. Dans d'autres cas, le développeur peut être incertain de l'efficacité d'un

[9] Sommerville, I., 2016. *Software engineering*. Harlow, England: Pearson Education Limited

algorithme. Dans ces situations, et bien d'autres, un paradigme de prototypage peut offrir la meilleure approche.

Le paradigme du prototypage commence par la définition des objectifs généraux du logiciel, et la réalisation d'une première spécification des exigences, puis, une itération de prototypage est planifiée rapidement pour la construction d'un prototype qui se concentre sur une représentation des aspects visibles du logiciel pour les utilisateurs finaux (l'IHM). Le prototype est déployé et évalué par les utilisateurs, qui fournissent des commentaires qui sont utilisés pour affiner davantage les exigences. L'itération se produit au fur et à mesure que le prototype est réglé pour satisfaire les exigences des utilisateurs[10].

On distingue deux types de prototypage : jetable et évolutif :

Le prototypage jetable est utilisé comme une technique pouvant être mise en œuvre dans un autre modèle de processus pour aider à mieux identifier les exigences lorsque ces dernières sont floues (Figure 2.7).

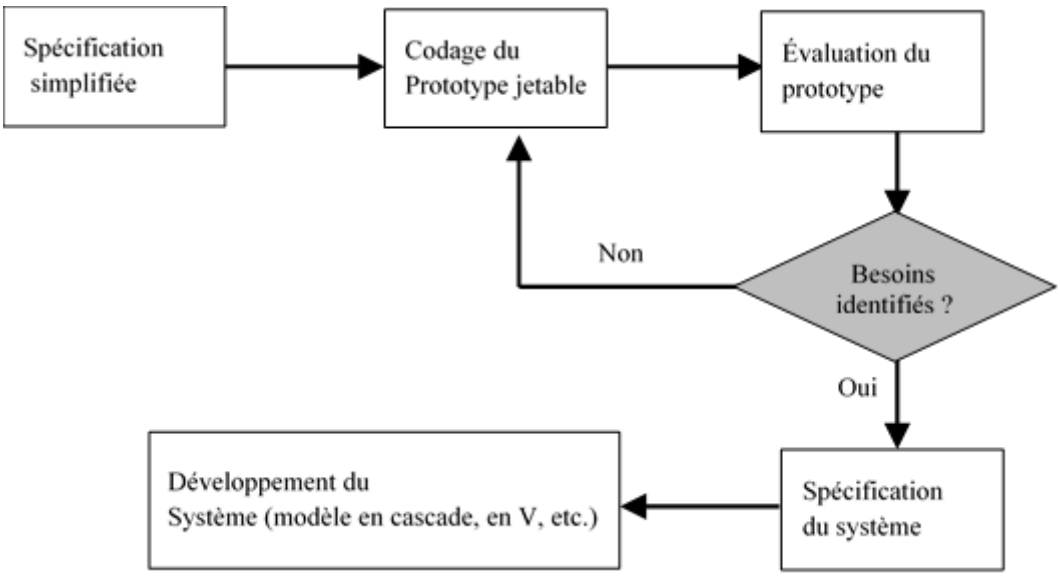


Figure 2.7. Prototypage jetable

Tandis que l'objectif du prototypage évolutif est de faire évoluer le prototype progressivement vers le système réel (Figure 2.8).

[10] Roger, S.P. and R.M. Bruce, Software engineering: a practitioner's approach. 2015: McGraw-Hill Education

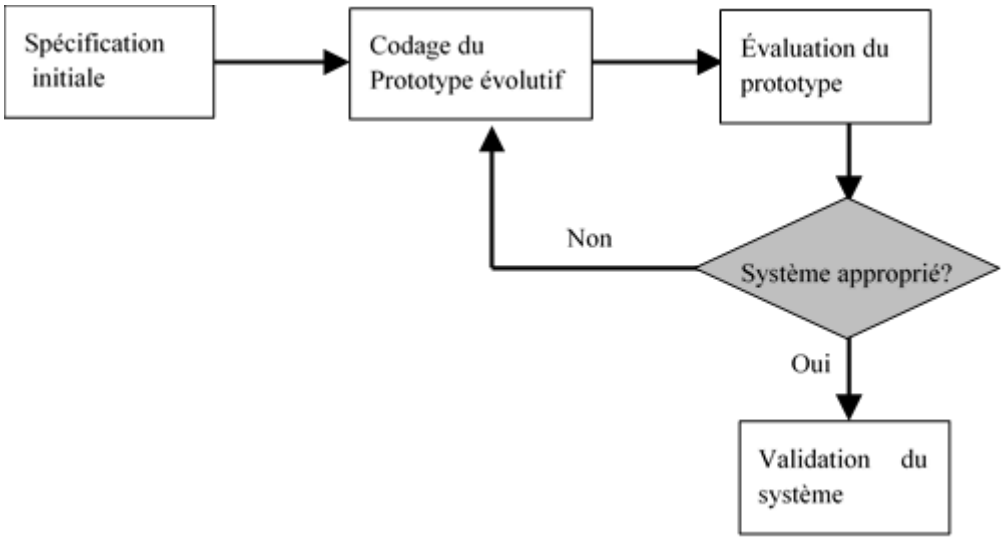


Figure 2.8. Prototypage évolutif

4.8 Modèle en spirale

Proposé par B. Boehm en 1988, le modèle en spirale met l'accent sur l'activité d'analyse des risques afin d'identifier les problèmes majeurs tôt dans le cycle de développement, et associe la nature itérative du prototypage aux aspects contrôlés et systématiques du modèle en cascade. Ce qui offre la possibilité de développer rapidement des versions de plus en plus complètes du logiciel. En utilisant le modèle en spirale, le logiciel est développé dans une série de versions évolutives. Au cours des premières itérations, la version est un simple prototype. Au cours des itérations ultérieures, des versions deviennent de plus en plus complètes du logiciel sont produites.

Chaque cycle de la spirale se déroule en quatre phases (Figure 2.9) :

1. Détermination, à partir des résultats des cycles précédents, ou de l'analyse préliminaire des besoins, les objectifs du cycle (fonctionnalité, performances, etc.), des alternatives pour les atteindre (conception, réutilisation, achat, etc.) et les contraintes imposées à l'application de ces alternatives (coût, durée, etc.);
2. Analyse des risques, et élaboration d'une stratégie pour résoudre les sources de risque.
3. Développement et vérification de la solution retenue, un modèle « classique » (cascade ou en V) peut être utilisé ;
4. Revue des résultats du cycle précédent, toutes les personnes et les organisations concernés par le produit sont impliqués. L'objectif est de décider d'aborder le cycle suivant ou arrêter la spirale.

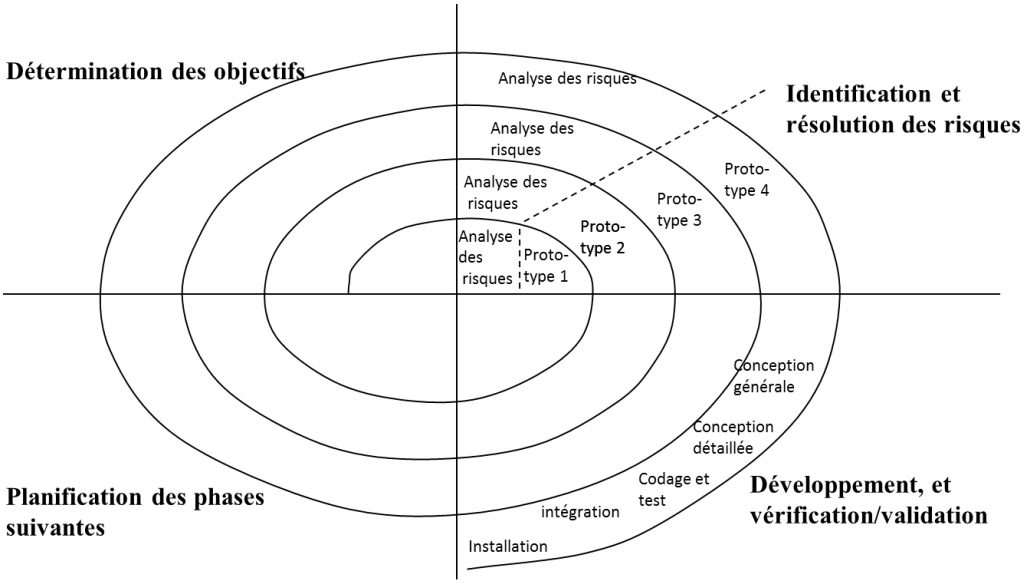


Figure 2.9. Modèle en spirale

5 Organigrammes de découpage

Dans la pratique, les chefs de projets utilisent des organigrammes de découpage qui définissent en détail le travail nécessaire pour atteindre les objectifs d'un projet, et qui fournissent ainsi, un outil de base pour une variété d'activités telles que l'estimation des coûts et des durées, l'élaboration de calendriers, et l'affectation des ressources. Ces organigrammes sont également un outil de communication précieux entre les parties prenantes du projet, car ils fournissent une image claire de ce qui doit être accompli et de la manière dont le travail sera effectué. On distingue trois structures de découpage : PBS, WBS, et OBS.

5.1 Product Breakdown Structure (PBS) (Structure de Décomposition du Produit)

Le PBS s'appuie sur un découpage purement structurel (Section 3.2). Il représente le découpage structurel du produit final sous une forme hiérarchique de composants (Figure 2.10). Les composants sont découpés à des niveaux successifs avec des composants plus petits jusqu'à ce que le travail soit suffisamment adapté à la gestion. Le nombre de niveaux varie d'un projet à l'autre et dépend de la taille et de la complexité du projet.

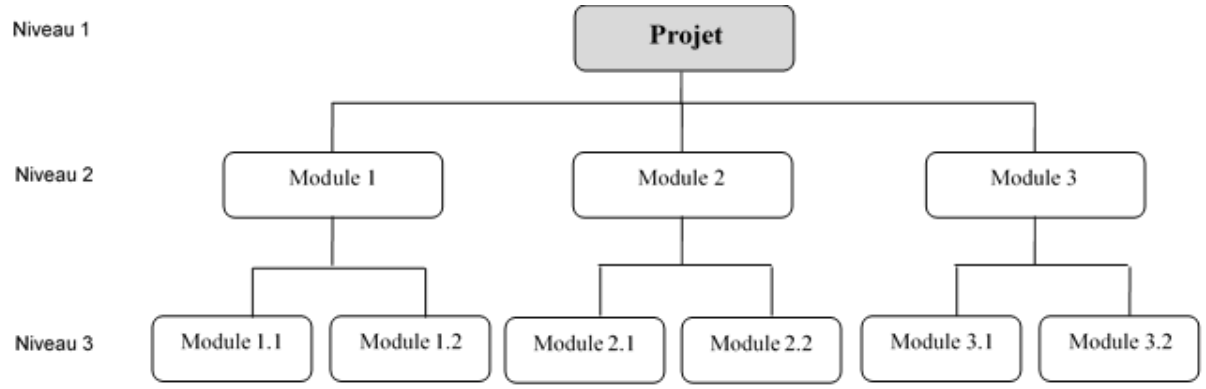


Figure 2.10. Exemple d'un PBS

5.2 Work Breakdown Structure (WBS) (structure de décomposition du travail)

Le WBS s'appuis sur un découpage à la fois structurel et temporel. Il représente sous forme d'une arborescence, les tâches nécessaires pour parvenir au résultat tel qu'il est décrit dans le PBS (Figure 2.11).

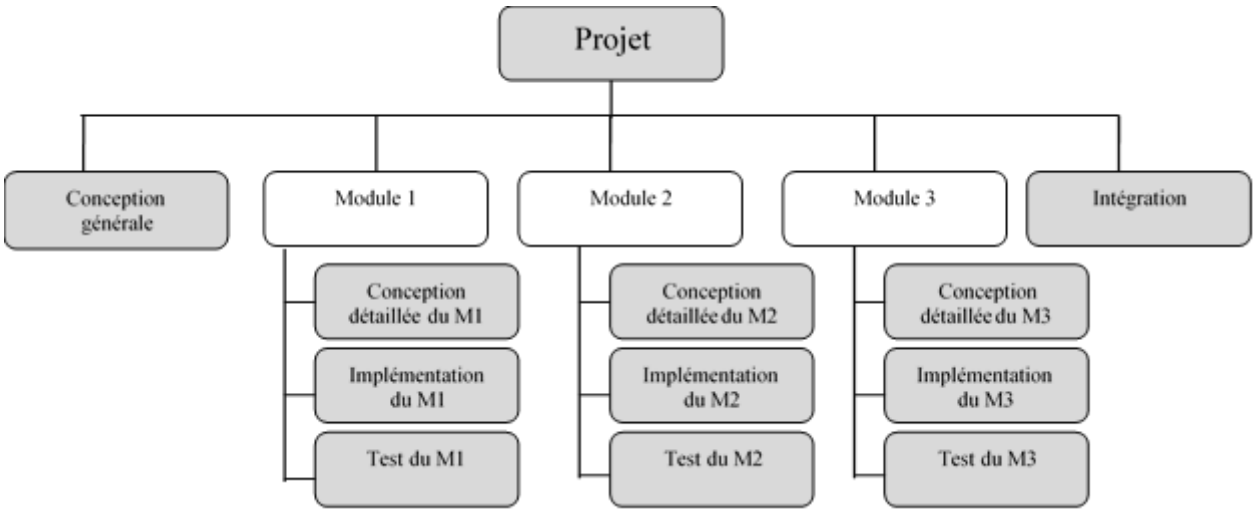


Figure 2.11. Exemple d'un WBS qui représentent les modules et les tâches correspondante

5.3 Organization Breakdown Structure (OBS) (structure de décomposition organisationnelle)

L'OBS représente les ressources qui mèneront les tâches du WBS pour développer les composants du PBS (Figure 2.12).

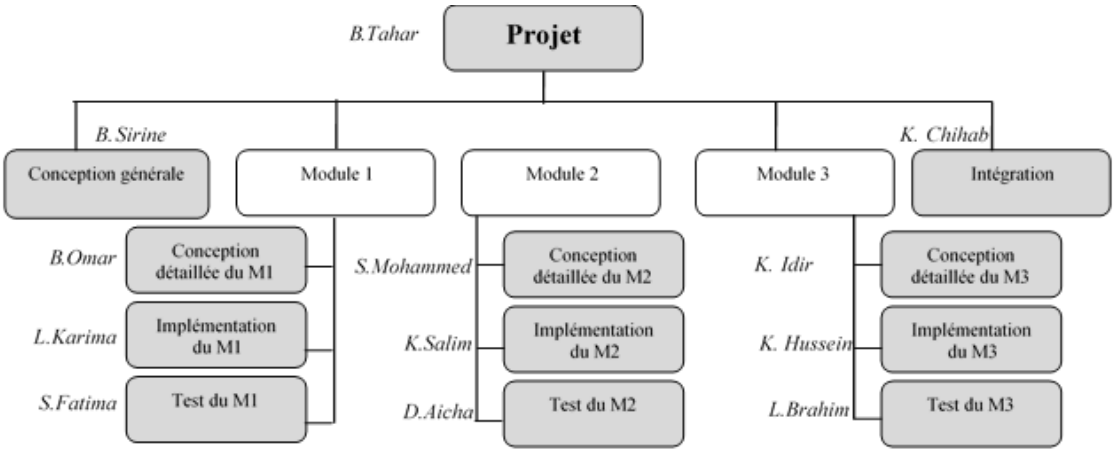


Figure 2.12. Exemple d'un OBS

Chapitre 3 .

ESTIMATION DE LA CHARGE ET DE LA DUREE

1 Introduction

La gestion d'un projet vise à atteindre ses objectifs tout en respectant les contraintes du temps et de ressources, ainsi, l'estimation des coûts et des durées est l'une des activités les plus importantes du développement logiciel. Une planification et un pilotage corrects du projet ne sont pas possibles sans une estimation fiable. La sous-estimation d'un projet conduit à une allocation de ressources insuffisante, et à un calendrier trop court, ce qui peut nuire à la qualité des livrables, et conduire à des dépassements de délais. La surestimation d'un projet entraîne un coût plus élevé qu'il ne le devrait et un retard dans l'utilisation des ressources sur d'autres projets.

2 Processus d'estimation

Généralement, l'estimation d'un projet logiciel est effectuée et révisée plusieurs fois durant le processus de développement. L'estimation d'un projet informatique comprend les quatre étapes suivantes (Figure 3.1)[11]:

1) Estimation de la taille du logiciel
2) Estimation de l'effort en Homme-mois ou Homme-jours, etc.
3) Estimation de la durée en mois.
4) Estimation du coût du projet en monnaie locale.

2.1 Estimation de la taille

La taille d'un logiciel est la quantification des exigences fonctionnelles exprimées par les utilisateurs. Elle peut être exprimée par plusieurs unités : Points de fonction (Function Points:

[11] de Barcelos Tronto, I.F., J.D.S. da Silva, and N. Sant'Anna, An investigation of artificial neural networks based prediction systems in software project management. Journal of Systems and Software, 2008. 81(3): p. 356-367

FP), Lignes de code source (Line Of Code: LOC) , etc. Une estimation précise de la taille du logiciel à construire est la première étape d'une estimation efficace de l'effort, du coût et de la durée du projet.

Au début de projet, une spécification initiale de la taille peut être sur la base de la spécification des exigences dans le cahier des charges. Dans les phases ultérieures du processus de développement, les documents de conception peuvent être utilisés comme source d'informations supplémentaires pour une ré-estimation plus précise.

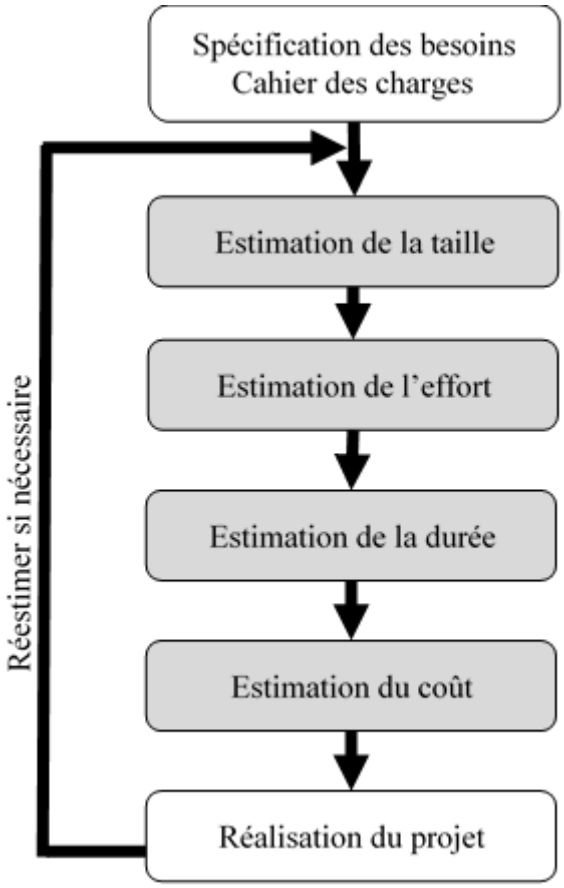


Figure 3.1. Processus d'estimation

2.2 Estimation de l'effort (charge)

Une fois la taille du projet est estimée, il est possible d'en déduire l'estimation de l'effort nécessaire pour réaliser le projet. L'effort (charge) d'un logiciel est la quantité du travail nécessaire indépendamment du nombre de personnes qui vont réaliser le travail. Il s'exprime généralement en Homme-mois (person-months), ce qui représente le travail d'une personne

pendant un mois. Si par exemple 10 personnes ont travaillé pendant 5 mois dans un projet, l'effort de développement de ce projet est alors 50 Homme-mois.

2.3 Estimation de la durée

La troisième étape dans l'estimation d'un projet logiciel consiste à déterminer la durée du projet à partir de l'estimation de l'effort. Cela implique généralement l'estimation du nombre de personnes qui travailleront sur le projet.

2.4 Estimation du coût

Le coût total d'un projet logiciel comprend le coût de développement du logiciel (coût de la main-d'œuvre), les achats ou la location de matériel et de logiciels, le coût des déplacements à des fins de réunions par exemple, les coûts des formations, etc. Dans les projets logiciels, le coût de développement dépasse 80% du coût total du projet. Une estimation simple du coût de développement d'un logiciel peut être obtenu en multipliant l'estimation de l'effort par le coût salarial d'un développeur.

Exemple

L'effort estimé d'un projet est 40 Homme-mois, et le coût d'un Homme-mois est de 50000 DA, alors le coût de développement sera estimé à 2 Million DA.

Un calcul plus précis de coût de la charge résulte de l'utilisation d'un taux pour chaque catégorie de personnel (technicien, qualiticien, encadrement, documentation, support, etc.). Vous devrez déterminer quel pourcentage de la charge totale du projet doit être affecté à chaque profil.

3 Besoins d'estimation

Les besoins d'estimation de projets logiciels sont nombreux, que ce soit du côté du maitrise d'ouvrage ou du maitrise d'œuvre, et se situent à plusieurs niveaux : du projet, des phases et des tâches.

En amont du projet, la maitrise d'ouvrage et la maitrise d'œuvre font chacune une estimation de la durée et du coût du projet. Côté maitre d'ouvrage (MOA), une estimation est effectuée dans le cadre de l'étude de faisabilité. A ce stade, la maitrise d'ouvrage s'appuie sur l'estimation pour les raisons suivantes :

- Décider de la faisabilité du projet ;
- Définir une enveloppe budgétaire pour le projet ;
- Évaluer les propositions reçues de différents fournisseurs pour le développement du logiciel ;
- Parvenir à un accord avec le fournisseur sélectionné sur les frais et la durée de développement.

Coté maitre d'œuvre (MOE), une estimation de la durée et du coût d'un projet est effectué lors d'une procédure d'appel d'offre pour :

- Assister les décideurs à déterminer si l'organisation possède les capacités et les ressources nécessaires à la réalisation du projet ;
- Définir une enveloppe budgétaire et une durée vraisemblable pour faire une proposition à l'appel d'offre afin d'acquérir le projet ;
- Faire une estimation de la rentabilité de l'investissement.
- Assister les décideurs à parvenir à des engagements sur les frais et la durée avec le client.

Au niveau des phases et tâches, l'estimation est indispensable pour différentes raisons :

- Réaliser une planification précise des différentes tâches du projet ;
- Effectuer un suivi du projet, et un suivi individuel des intervenants pour surveiller les écarts ;
- Fixer un calendrier de remise des différents résultats intermédiaires ;
- Prévoir l'affectation des ressources, car il peut y avoir une montée en charge ou une diminution au cours de réalisation du projet.

4 Incertitude des estimations

Les estimations initiales au début du projet sont souvent faites sur la base d'une spécification sont souvent incomplètes et imprécises des besoins des utilisateurs, et les personnes impliquées dans le projet et leurs compétences ne seront probablement pas connues, par conséquent, il est impossible d'estimer avec précision les coûts de développement du système pendant les premières étapes d'un projet. Au fur et à mesure qu'on avance dans le projet, les spécifications deviennent plus détaillées, et les connaissances concernant la complexité du système et le personnel augmentent, ainsi, la marge d'erreur diminue et les estimations se rapprochent du

coût réel. McConnell[12] illustre l'évolution de l'incertitude des estimations durant le processus développent à travers le « Cône d'incertitude » (Figure 3.2). Sur le graphique, on constate que les estimations au début du projet sont susceptibles à un degré élevé d'erreur, les estimations lors de la conception initiale varient entre 25% et 400% du couts réel du projet. À la phase de conception détaillée, les estimations se précisent et passent entre 80% 120% du couts réel.

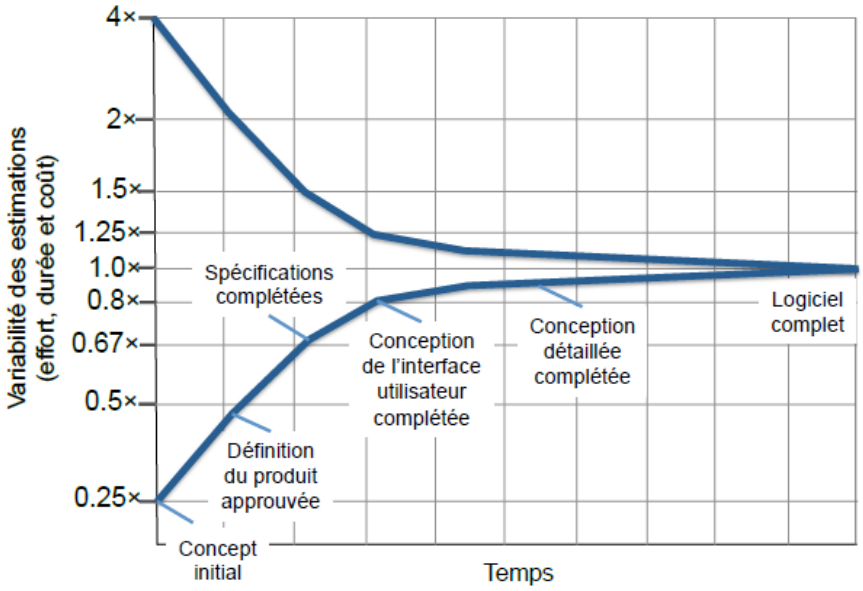


Figure 3.2. Cône d'incertitude des estimations

5 Techniques d'estimation

Il existe de nombreuses techniques d'estimation de projets logiciels, qui ne sont pas nécessairement exclusives, mais qui peuvent être utilisées conjointement, ou à des moments différents du processus de développement. Ces techniques peuvent être classées en trois grandes catégories : les techniques informelles, les modèles paramétriques, et les modèles non paramétriques.

[12] McConnell, S., *Software estimation: demystifying the black art*. 2006: Microsoft press

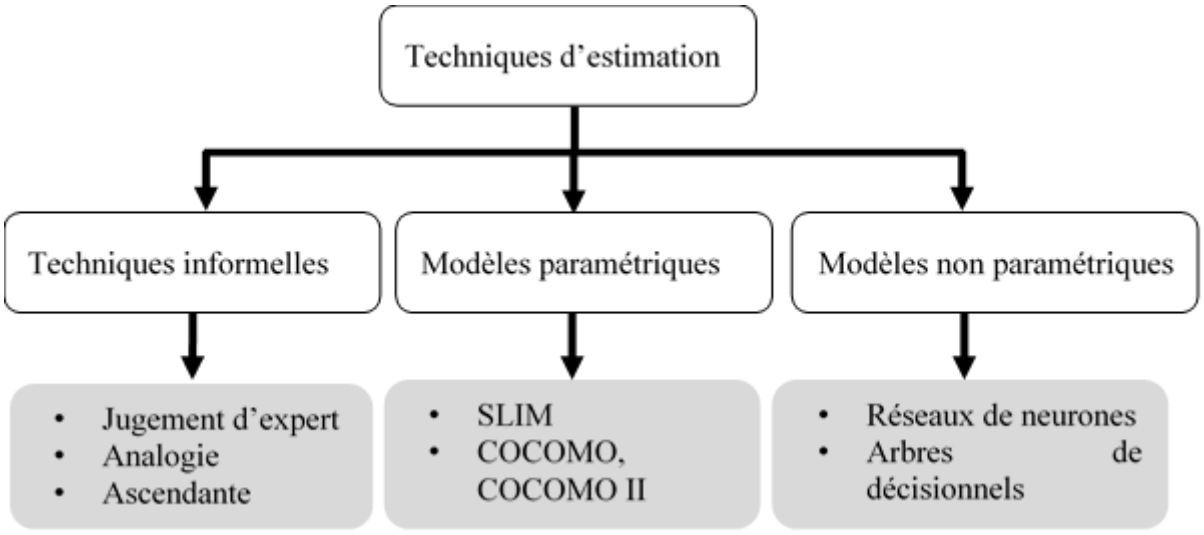


Figure 3.3. Catégories des techniques d'estimation

5.1 Les techniques informelles

Les techniques informelles sont conduites par une ou plusieurs personnes dites expertes dans le domaine de l'estimation, qui s'appuient sur l'intuition ainsi que l'expérience de l'historique des projets similaires. Ces techniques d'estimation sont les plus utilisées selon plusieurs revues sur les pratiques d'estimation.

Les techniques informelles sont rapides à appliquer, et nécessitent peu de documents, ainsi, elles peuvent être utilisées assez tôt dans le processus de développement logiciel. Cependant, l'application de telles techniques nécessitent la disponibilité d'experts, en plus, les estimations obtenues sont subjectives, et manquent d'argumentation analytique.

Parmi ces techniques, nous pouvons citer : la méthode Delphi, l'analogie, et l'estimation ascendante,

5.1.1 La méthode Delphi

Cette méthode nécessite la disponibilité d'experts en estimation qui utilisent leurs expériences ainsi que leur compréhension du projet afin de fournir une estimation de la taille du logiciel ou de l'effort de développement logiciel, ou les deux. La **méthode Delphi** propose une démarche pour combiner les différents jugements d'experts, qui peut être résumée comme suit :

1) Chaque expert propose une estimation en utilisant sa propre expérience ;
2) Tous les jugements son rendus publics, mais restent anonymes. Chaque expert peut alors modifier sa propre estimation ou la confirmer : Un expert doit se poser des questions si ses estimations sont très éloignées de celles des autres.
3) Les estimations sont dévoilées et chacun peut justifier son propre jugement.

4) Chacun propose une révision de son estimation.
5) Ce processus est réitéré jusqu'à l'adoption d'une estimation par tout le groupe.

5.1.2 Estimation par analogie

Un projet similaire antérieur similaire dont on connait la taille, est utilisé comme base à partir de laquelle on peut dériver l'estimation pour le projet en cours, il est possible d'estimerez chaque partie principale du nouveau projet comme un pourcentage de la taille de la partie similaire du précédent projet. Cette technique est rapide à mettre en œuvre et peut être utilisée durant tout le processus de développement. Cependant, elle nécessite d'avoir un historique sur lequel on peut se baser afin d'établir des analogies.

5.1.3 Estimation ascendante (bottom-up)

L'estimation ascendante (bottom-up), également appelée l'estimation analytique peut etre résumée aux étapes suivantes :

1) Découpage du projet logiciel à en plusieurs tâches constituant une arborescence (WBS) ;
2) Estimation de l'effort nécessaire pour chaque tâche du plus bas niveau dans l'arborescence (généralement par jugement d'expert);
3) Estimation progressive de l'effort des autres tâches, se retrouvant dans un niveau supérieur dans l'arborescence, en combinant les efforts nécessaires associés aux sous-tâches. ;
4) L'effort de tout le projet est alors la somme de l'effort des estimations de chaque tâche du projet, éventuellement avec l'ajout d'une quantité d'effort pour couvrir les activités et les événements imprévus.

Le découpage du projet en tâches nécessite une connaissance détaillée du projet, par conséquent, l'estimation ascendante ne peut être appliquée que dans les phases en aval du projet.

5.2 Modèles paramétriques

Développés pour pallier aux inconvénients des techniques informelles se basant essentiellement sur la disponibilité des experts, les modèles paramétriques s'appuient sur la modélisation statistique pour exprimer la relation qui existent entre l'effort et les variables considérées déterminantes de l'effort et appelées « facteurs d'effort » ou " (en anglais : *cost drivers*).

L'élaboration des modèles d'estimation se fait en deux étapes majeures :

1) **Identification des facteurs du coût « cost drivers »** : Ce sont les facteurs qui influencent l'effort de développement de logiciels. Leur identification s'appuie essentiellement sur l'analyse statistique de données qui proviennent de la réalisation de projet antérieurs. La taille du logiciel est souvent considéré le facteur principal pour l'estimation de l'effort de développement, cependant, d'autres facteurs sont souvent prise en compte, entre autres, l'expérience des développeurs, la complexité du logiciel à développer, etc.

2) **Identification de la relation exprimant l'effort en fonction des facteurs du coût** : La relation entre l'effort et les facteurs peut être exprimée par une fonction *f*:

$$\boldsymbol{Effort = f\ (X_1\ , X_2, \ldots, X_M)}$$

Où $\boldsymbol{X_1\ , X_2, \ldots, X_M}$ sont les facteurs du coût; $\boldsymbol{f}$ est la relation exprimant l'effort en fonction des facteurs $\boldsymbol{X_i}$, et $\boldsymbol{M}$ est le nombre de facteur de cout considérés.

La construction de la fonction $\boldsymbol{f}$ est souvent basée sur l'analyse des données historiques collectées sur les projets logiciels déjà achevés. Ces données historiques peuvent être représentées par une matrice $N \times (M + 1)$

	Effort réel	X_1	X_2	…	X_M
Projet 1	E_1	x_{11}	x_{12}	…	x_{1M}
Projet 2	E_2	x_{21}	x_{22}	…	x_{2M}
Projet 3	E_3	x_{31}	x_{32}	…	x_{3M}
…	…	…	…	…	…
Projet N	E_N	x_{N1}	x_{N2}	…	x_{NM}

N représente le nombre de projets logiciels déjà achevés. La colonne *Effort réel* ($\boldsymbol{E_1\ , E_2, \ldots, E_N}$) représente l'effort réel de chaque projet logiciel déjà achevé.

Le défi est de reconstruire la fonction $\boldsymbol{f}$ à partir de ces données afin qu'elle soit utilisée pour prédire le coût des nouveaux projets logiciels. Les techniques souvent utilisées construire la fonction *f* sont la régression linéaire simple ou multiple si on fait l'hypothèse d'une relation linéaire entre le les facteurs de coût et l'effort de développement.

5.2.1 Les modèles linéaires

Ces modèles s'appuient sur l'hypothèse d'une relation linéaire entre les facteurs de coût et l'effort de développement, ainsi, l'équation de l'effort en fonction des facteurs est de la forme :

$$\boldsymbol{Effort = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_M x_M}$$

Où les $\boldsymbol{x_i}$ sont les facteurs affectant l'effort et les $\boldsymbol{a_i}$ sont des coefficients choisis pour fournir le meilleur ajustement à l'ensemble des données historiques observées. La mise au point de tels modèles est souvent accomplie à l'aide de régression linéaire simple ou multiple selon le nombre de facteurs $\boldsymbol{x_i}$.

Régression linéaire simple

L'application de la régression linéaire simple, considère la taille du logiciel comme étant le facteur le plus significatif pour la prédiction de l'effort :

$$\text{Effort} = A + B \times \text{taille}$$

Où *A* et *B* sont des constantes. La taille d'un logiciel est souvent mesurée par le nombre de lignes de son code source (Kilo Line Of Code -KLOC-) ou le nombre de points de fonctions (Function points –FP-).

Régression linéaire multiple

Dans le cas où le nombre de facteurs est supérieur ou égal à deux, la mise au point du modèle fait appel à la régression linéaire multiple. Des exemples de facteurs affectant le coût, autres que la taille, sont l'expérience du personnel impliqué dans le développement, la complexité de l'application et la méthodologie de développement. La fonction de l'effort est de la forme :

$$\text{Effort} = a_0 + a_1 x_1 + a_2 x_2 + \cdots a_m x_m$$

5.2.2 Transformation logarithmique

La majorité des modèles paramétriques font l'hypothèse d'une relation exponentielle entre l'effort et les facteurs du coût. Cela reflète le fait que l'effort de développement n'augmentent généralement pas de manière linéaire avec la taille du projet. Au fur et à mesure que la taille et la complexité du logiciel augmentent, des coûts supplémentaires sont encourus en raison de la

surcharge de communication d'équipes plus importantes, d'une intégration du système plus difficile, etc.[13]. Ainsi, l'équation de l'effort est de la forme :

$$\text{Effort} = B \times \text{taille}^{C}$$

La transformation logarithmique est appliquée pour rendre linéaire ce genre de modèles, ainsi, on transforme les deux variables *effort* et *taille* en *log(effort)* et *log(taille)*. Ainsi, l'équation centrale transformée du modèle s'écrit :

$$\log(\text{Effort}) = \log(\text{B}) + C \times \log(\text{taille})$$

On utilise ensuite la régression linéaire pour déterminer les deux constantes $log(B)$ et C .

5.3 Modèles non paramétriques

Les modèles non paramétriques s'appuient généralement sur des algorithmes d'apprentissage automatique, tels que les réseaux de neurones, les arbres de décision, etc. Ces algorithmes utilisent des données de projets réalisés dans une phase d'apprentissage qui vise à optimiser l'estimation de l'effort, après quoi, le modèle donne en sortie une estimation de l'effort à partir es facteurs du coût en entrée (taille du logiciel, complexité du projet, etc). L'avantage des modèles non-paramétriques c'est qu'ils s'appuient sur moins d'hypothèses concernant la distribution des données et supposent que la forme de la fonction n'est pas définie *à priori*, contrairement aux modèles paramétriques qui sont construits sur des hypothèses peu irréalistes d'une relation linéaire ou exponentielle entre les facteurs du coût et l'effort de développement.

6 Le modèle COCOMO (COnstructive COst MOdel)

COCOMO est un acronyme pour COnstructive COst MOdel (Modèle de construction de coût). Il a été développé à l'origine par le Dr Barry Boehm[14], 1981 à partir d'une analyse des données par régression pratiquée sur 63 projets logiciels de 2.000 à 100.000 lignes de codes. Ce modèle général se compose de trois modèles différents : le *modèle de base*, le *modèle intermédiaire*, et le *modèle détaillé.* Une nouvelle version du modèle appelé COCOMO II a été développé en

[13] Sommerville, I., 2016. *Software engineering*. Harlow, England: Pearson Education Limited
[14] Boehm, B.W., Software engineering economics. IEEE transactions on Software Engineering, 1984(1): p. 4-21

2000. COCOMO II est réputé être mieux adapté pour estimer des projets de développement de logiciels modernes.

6.1 Le modèle COCOMO de base

Le modèle COCOMO s'appuient sur une estimation de la charge basée sur les formules:

$$Charge = a \times KLOC^{b}$$

$$Durée = c \times Charge^{d}$$

- **Charge** (ou effort): est l'effort de développement exprimé en *homme-mois* (HM). Un homme-mois correspond à 152 heures (19 jours) de travail effectif. Ce chiffre tient compte des absences pour formation, vacances, arrêts maladie, etc.
- ***KLOC*** représente la taille du logiciel en milliers de lignes de code (instructions) (KLOC = Kilo Lines Of Code).
- ***Durée*** : est la durée de développement en *Mois.*
- Les paramètres ***a***,***b***,***c*** et ***d*** prennent des valeurs différentes selon le type du projet. Les 3 types de projets dans COCOMO sont:
  - **Simple** *(Organique)*: *projet qui* peut être réalisé par une équipe de petite taille (2 à 8 personnes), travaillant dans un domaine qu'ils connaissent (ex: petite gestion, système de notes dans une école, traducteurs).
  - **Moyen** *(Semi-détaché)*: projet qui présente un degré de difficulté moyen, l'équipe a une expérience limitée du type d'application (ex: Compilateurs, système bancaire interactif).
  - **Complexe** *(Intégré)* : projet complexe qui présente des contraintes fortes (contraintes de type temps réel, sécurité, support matériel et logiciel complexes). Le coût de changement d'une contrainte est très élevé. (Exemples : Gros système d'exploitation, système de contrôle aérospatial, etc.).

Type de projet	a	b	c	d
Simple	2.4	1.05	2,5	0,38
Moyen	3.0	1.12	2,5	0,35
Complexe	3.6	1.20	2,5	0,32

Tableau 3.1. Paramètres du modèle COCOMO de base

A partir des valeurs obtenues pour la charge et la durée, on peut déduire :

- **Le Staffing Moyen** : Le nombre de personnes requises pour réaliser le projet dans la durée estimée, exprimé en FSP (Full Time Equivalent Software Personnel):

$$\boldsymbol{Staffing\ Moyen}\ (\mathbf{FSP}) = \frac{\boldsymbol{Charge}}{\boldsymbol{Dur\acute{e}e}}$$

- **La productivité moyenne** de l'équipe, exprimée en LOC/HM (lignes de code par homme-mois):

$$\boldsymbol{Productivit\acute{e}} = \frac{\boldsymbol{Taille\ (LOC)}}{\boldsymbol{Charge}}$$

On peut ensuite calculer la distribution de la charge et de la durée de développement par phases (en %), selon le type et la taille du logiciel.

		Taille en KLOC				
Type de projet	Phase	2	8	32	128	512
Simple	Conception général	16	16	16	16	
	Conception détaillée	26	25	24	23	
	Programmation et tests unitaires	42	40	38	36	
	Intégration et test d'intégration	16	19	22	25	
Moyen	Conception général	17	17	17	17	17
	Conception détaillée	27	26	25	24	23
	Programmation et tests unitaires	37	35	33	31	29
	Intégration et test d'intégration	19	22	25	28	31
Complexe	Conception général	18	18	18	18	18
	Conception détaillée	28	27	26	25	24
	Programmation et tests unitaires	32	30	28	26	24
	Intégration et test d'intégration	22	25	28	31	34

Tableau 3.2. Distribution de la charge par phase en pourcentage

		Taille en KLOC				
Type du projet	Phase	2	8	32	128	512
Simple	Conception générale	19	19	19	19	
	Conception détaillée et Programmation	63	59	55	51	
	Tests et intégration	18	22	26	30	
Moyen	Conception générale	24	25	26	27	28
	Conception détaillée et Programmation	56	52	48	44	40
	Tests et intégration	20	23	26	29	32
Complexe	Conception générale	30	32	34	36	38
	Conception détaillée et Programmation	48	44	40	36	32
	Tests et intégration	22	24	26	28	30

Tableau 3.3. Distribution de la durée par phase en pourcentage

Exemple

Application du modele COCOMO de base pour estimer un projet de type simple ayant une taille estimée à 32000 lignes de code.

- Charge = 2.4 * $(32)^{1.05}$ = 91 HM (Homme-Mois)
- Durée = 2.5 * $(91)^{0.38}$ = 14 Mois
- Productivité= 32000 LOC/91 HM = 352 LOC/HM
- Staffing Moyen = 91 HM / 14 MOIS = 6.5 FSP
- Distribution de la charge :
  - Phase de conception générale : 0.16 * 91 = 14.6 HM
  - Phase de conception détaillée : 0.24 * 91 = 21.9 HM
  - Phase de programmation : 0.38* 91=34.6 HM
  - Phase d'intégration : 0.22*91= 20HM
- Distribution de la durée :
  - Phase de conception : 0.19*14=2.6 Mois
  - Phase de conception et de la programmation : 0.55 * 14 = 7.7 Mois
  - Phase d'intégration : 0.26 * 14 = 3.7 Mois

6.2 Le modèle COCOMO intermédiaire

Le modèle COCOMO Intermédiaire est une extension du modèle COCOMO de base. Il en diffère principalement par la prise en compte de 15 facteurs qui influence l'effort, en plus de la taille et du type de logiciel. Ces 15 facteurs justifient la variation entre les coûts de deux logiciels de même taille. Cette variation n'est pas révélée ni expliquée par le modèle COCOMO de base.

L'équation d'estimation du modèle de l'effort du modèle COCOMO Intermédiaire est:

$$\boldsymbol{Charge} = \boldsymbol{a} \times \boldsymbol{KLOC^b} \times \prod_{i=1}^{15} \boldsymbol{C_i}$$

L'équation de l'estimation de la durée est :

$$\boldsymbol{Durée} = \boldsymbol{b} \times \boldsymbol{Charge}^{\boldsymbol{c}}$$

Les paramètres *a*, *b*, *c*, et *d* du modèle COCOMO intermédiaire, en fonction du type du projet, sont présenté dans le tableau suivant :

Type de projet	Simple	Moyen	Complexe
a	3,20	3,00	2,80
b	1,05	1,12	1,20
c	2,50	2,50	2,50
d	0,38	0,35	0,32

Tableau 3.4. Paramètres du modèle COCOMO intermédiaire

C_i représente la valeur du facteur d'effort i. Il existe 15 facteurs d'effort qui sont représentés dans le Tableau 3.5.

Attributs du Produit
RELY (Required Software Reliability): Fiabilité requise
DATA (*Data Base Size*): Taille de la base de données
CPLX (*Product Complexity*): Complexité du logiciel
Attributs du Matériel
TIME (*Execution Time Contraint*): Contrainte du temps d'exécution
STOR (*Main storage Contraint*) Contrainte de la taille mémoire
VIRT (*Virtual machine Volatility*): Instabilité de la plateforme
TURN (*Computer Turnaround Time*): Temps de restitution de l'ordinateur
Attributs du Personnel
ACAP (*Analyst Capability*): Compétence des analystes
PCAP (*Programmer Capability*): Compétence des programmeurs
AEXP (*Application Experience*): Expérience du domaine d'application
VEXP (*Virtual Machine Experience*): Expérience dans la plateforme
LEXP (*Programming Language Experience*): Expérience du langage de programmation
Attributs du Projet
MODP (*Modern Programming Practices*): Pratiques des méthodes de programmation
TOOL (*Use of Software Tools*): Utilisation d'outils logiciels
SCED (*Required Development Schedule*): Contraintes de planification

Tableau 3.5. Les 15 facteurs d'effort du modèle COCOMO intermédiaire

Chaque facteur est évalué avec une *note* (valeurs linguistiques), ensuite, la note est convertie à une *valeur* (*facteur d'ajustement*). Les notes qu'un facteur d'effort peut prendre sont :

Très faible, Faible, Moyen, Élevé, Très élevé, et Extra-élevé

Les évaluations possibles des 15 facteurs et les valeurs correspondantes sont représentées dans le Tableau 3.6.

Facteur	Evaluation					
	Très faible	Faible	Moyen	Elevé	Très élevé	Extra-élevé
RELY	0,75	0,88	1	1,15	1,4	
DATA		0,94	1	1,08	1,16	
CPLX	0,7	0,85	1	1,15	1,3	1,65
TIME			1	1,11	1,3	1,66
STOR			1	1,06	1,21	1,56
VIRT		0,87	1	1,15	1,3	
TURN		0,87	1	1,07	1,15	
ACAP	1,46	1,19	1	0,86	0,71	
AEXP	1,29	1,13	1	0,91	0,82	
PCAP	1,42	1,17	1	0,86	0,7	
VEXP	1,21	1,1	1	0,9		
LEXP	1,14	1,07	1	0,95		
MODP	1,24	1,1	1	0,91	0,82	
TOOL	1,24	1,1	1	0,91	0,83	
SCED	1,23	1,08	1	1,04	1,1	

Tableau 3.6. Les évaluations possibles des 15 facteurs et les valeurs correspondantes du modèle COCOMO intermédiaire

Exemple

Considérons un projet de type simple ayant une taille estimée à 32000 lignes de code.
1ier Cas: Tous les facteurs ont tous la valeur « Moyen »:
Charge = $3.2 * (32)^{1.05} *1$= 122 H-M
2ème Cas: La fiabilité est «Très élevée », et le reste des facteurs ont la valeur « Moyen » :
Charge = $3.2 * (32)^{1.05}*1.4$ = 170.5 H-M
3ème Cas: La fiabilité est «Très faible », et le reste des facteurs ont la valeur « Moyen » :
Charge = $3.2 * (32)^{1.05}*0.75$ = 91.3 H-M

7 Méthode de points de fonction

Le point de fonction (Function point : FP) est une mesure de taille de logiciels, c'est de loin la mesure de taille la plus populaire utilisée à ce jour, car il est accepté comme norme plus que toute autre mesure de taille de logiciel[15]. La méthode de de points de fonction a été développées en 1979 par Allan J. Albrecht chez IBM, dans l'objectif d'estimer la taille du logiciel en points de fonction à partir d'une description externe du futur logiciel. En 1986, a été fondé l'International Function Point Users Group (IFPUG)[16], qui supervise l'utilisation du FP et met à niveau la

[15] Chemuturi, Murali. Software estimation best practices, tools & techniques: A complete guide for software project estimators. J. Ross Publishing, 2009.
[16] https://www.ifpug.org/

méthodologie FP en incluant les développements les plus récents dans la technologie de développement de logiciels. La méthode de points de fonction est décrite en détail dans "Function Point Counting Practices " publié par l'IFPUG.

La comptabilisation des points de fonction exige d'abord la disponibilité d'une documentation suffisante. La liste de documentation suivante est utile lors de l'identification des composants: Cahier des charges, Spécification des besoins, Maquettes IHM, Prototypes, Descriptions des fonctionnalités (Scenarios des cas d'utilisation, user story, etc.), Modèles entité association, Diagrammes de classes, Diagrammes de séquences, d'activités, etc., Schéma de la base de données, Scénarios de test , Manuels d'utilisation, etc.

7.1 Etapes de la méthode de points de fonction

La méthode se déroule en six étapes, dont l'objectif consiste à estimer la taille à partir d'une description externe du futur système (Figure 3.4). Cette approche permet d'obtenir tôt dans le cycle de vie du logiciel une estimation de sa taille qui sert à estimer l'effort, le coût de développement et la durée.

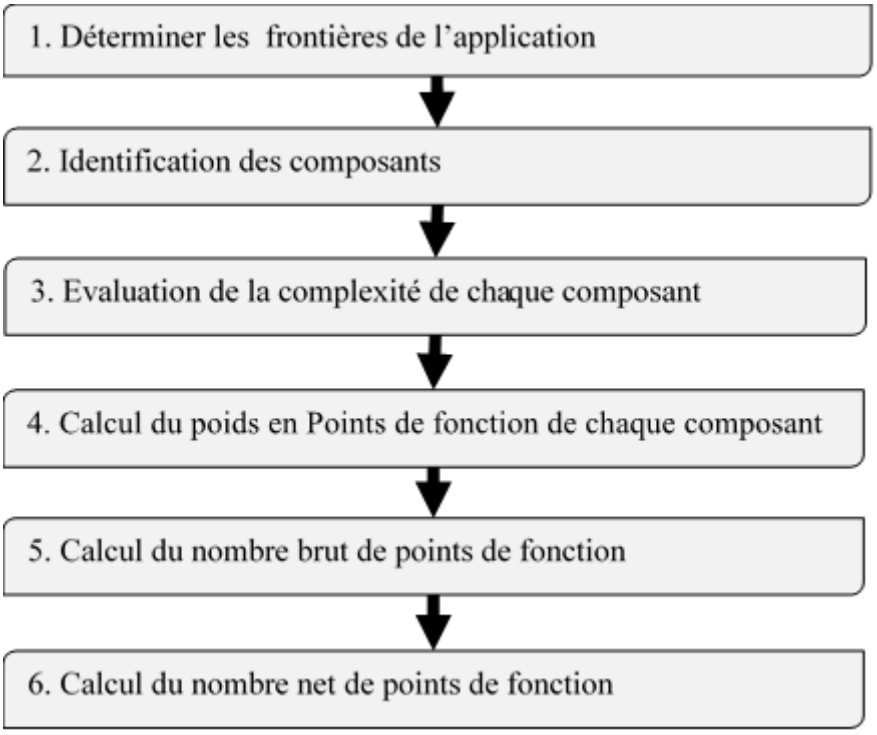


Figure 3.4. Etapes de la méthode de points de fonction

7.1.1 Etape 1 : Déterminer les frontières de l'application

Les frontières de l'application sont la limite entre l'application mesurée et les applications externes ou le domaine utilisateur. Pour définir les frontières, il faut identifier les utilisateurs de l'application mesurée, et identifier les applications externes qui interagissent avec l'application. Au niveau conception, une frontière est une interface conceptuelle qui sépare l'application mesurée et ses données (Groupes de données internes : GDI), des utilisateurs, et des autres applications qui interagissent et leurs données (Groupes de données externes : GDE), et agit comme une membrane à travers laquelle les données traitées par les transactions (entrées, sorties, et interrogations) entrent et sortent de l'application (Figure 3.5).

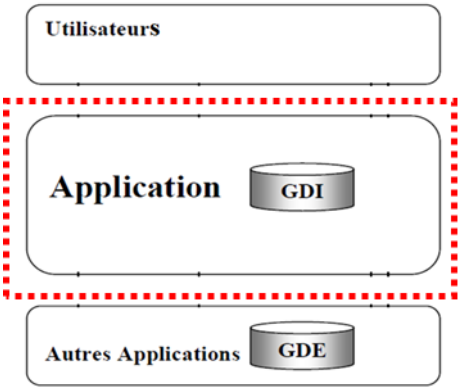


Figure 3.5. Frontières de l'application

7.1.2 Etape 2 : Identification des composants

Le comptage des points de fonction passe par l'identification des composants du système et leurs complexités. Nous allons d'abord présenter les 5 types de composants définis par la méthode points de fonction, puis, nous allons détailler les règles d'identification de ces composants.

7.1.2.1 Les composants

La méthode de point de points de fonction définit 5 types de composants, deux sont des données (GDI et GDE), et trois des traitements (ENT, SOR et INT) (Figure 3.6).

A. ***Les données (Data functions)***

Les données font référence aux données logiques qui sont stockées et mises en disposition de l'application pour la mise à jour et la consultation. Les données peuvent être identifiées comme des groupes de données internes (GDI) ou des groupes de données externes (GDE).

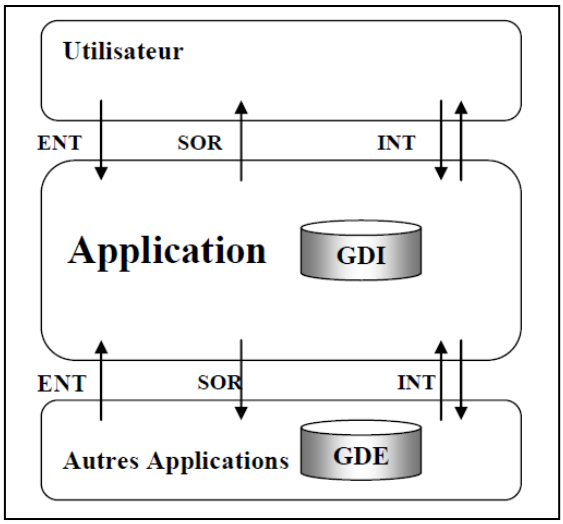


Figure 3.6. Composants de la méthode de points de fonction

Les GDI et les GDE sont des groupes de données liées logiquement et identifiables par l'utilisateur.

Groupe logique de données internes (GDI) (Internal Logical Files : ILF)

Un GDI fait référence à un groupe de données :

- Logiquement liées, identifiables par l'utilisateur, et maintenu (créé, consulté, modifié, et supprimé) à l'intérieur de la frontière de l'application.
- L'objectif principal d'un GDI est de conserver les données maintenues à travers un ou plusieurs traitements de l'application mesurée.

Groupe de données externes (GDE) (EIF: External Interface Files)

Un GDE fait référence aux groupes de données :

- Liées logiquement, identifiables par l'utilisateur, utilisés par l'application, et mis à jour par une autre application.
- L'objectif principal d'un GDE est de conserver les données référencées à travers un ou plusieurs traitements dans les limites de l'application mesurée. Cela signifie qu'un GDE compté pour une application doit être un GDI d'une autre application.

Le diagramme de classe est bien adapté à l'identification des GDI et GDE. Un GDI (GDE) correspond à une classe ou une classe-association.

B. *Les traitements (Transactional Functions)*

Un traitement est une activité identifiable par l'utilisateur qui fournit des fonctionnalités pour traiter les données. Un traitement peut être une entrée, une sortie ou une interrogation.

Entrées (ENT) (External Inputs : EI)

Une entrée fait référence à un traitement qui :

- Fait l'objet d'un traitement unique ;
- Traite des données ou des informations de contrôle envoyées depuis l'extérieur de la frontière ;
- Comprenant la mise à jour d'un ou plusieurs GDI ou la modification le comportement du système.

L'objectif principal d'une ENT est de maintenir un ou plusieurs groupes de données internes (GDI) et/ou de modifier le comportement du système.
Par simplification, une Entrée correspond à un écran de saisie de l'utilisateur, ou à une réception de données provenant d'une autre application, et provoquant une mise à jour.

Sorties (SOR) (External Outputs : EO)

Une sortie fait référence à un traitement qui :

- Fait l'objet d'un traitement unique ;
- Envoie des données ou des informations de contrôle en dehors des frontières de l'application ;
- Comprend un traitement supplémentaire différent d'une simple extraction de données : combinaison des extractions avec des calculs et traitements supplémentaires.

L'objectif principal d'une sortie externe est de présenter des informations à l'utilisateur via un traitement autre que ou plus qu'une extraction de données ou d'informations de contrôle. La logique de traitement doit contenir au moins une formule mathématique ou un calcul, ou créer des données dérivées.
Par simplification, une sortie correspond à la génération d'un écran de visualisation ou d'un message à destination d'une autre application, avec des données calculées à partir d'autres données.

Interrogations (INT) (External Inquiry :EO)

Une interrogation fait référence à un traitement qui:

- Fait l'objet d'un traitement unique ;
- Envoie des données ou des informations de contrôle à l'extérieur de la frontière ;

- Ne fait pas de mise à jour de GDI
- Ne résulte pas d'un traitement autre que des extractions de données
- Ne contient pas de données dérivées (calculées en sortie)

L'objectif principal d'une interrogation est de présenter des informations à un utilisateur par le biais de l'extraction de données ou d'informations de contrôle. Le traitement ne contient aucune formule mathématique ni aucun calcul et ne crée aucune donnée dérivée. Aucun GDI n'est maintenu pendant le traitement, et le comportement du système n'est pas modifié.

7.1.2.2 Identification des GDI et GDE à l'aide de diagrammes de classes

Les règles suivantes permettent d'identifier les groupes logiques de données (GDI et GDE) à partir de diagrammes de classes[17,18].

Regle 1 : Classe

Pour une classe séparée on comptabilise un groupe logique de données (1 GD) et un sous-groupe logique de données (1 SLD). Pour chaque attribut d'une classe, on compte une données élémentaires (DE).

Regle 2 : Composition

Pour une structure d'agrégation complète, on compte un seul groupe de données, on compte aussi 1 SLD pour chaque classe, et une DE pour chaque attribut.

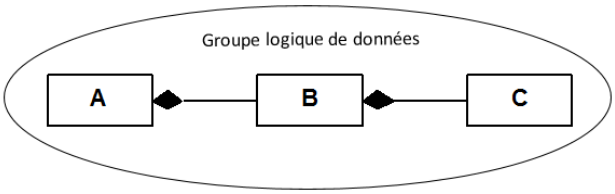


Figure 3.7. Diagramme de classes de la règle 2

[17] Pow-Sang, J. A., Gasco, L., & Nakasone, A. (2010). Evaluating the applicability of a function point logic file identification technique through controlled experiments. International Journal of Software Engineering and Its Applications, 4(3), 29-42.

[18] Pow-Sang, J. A., Villanueva, D., Flores, L., & Rusu, C. (2013, October). A Conversion Model and a Tool to Identify Function Point Logic Files using UML Analysis Class Diagrams. In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on* (pp. 126-134). IEEE.

Regle 3 : Association et agrégation

S'il existe deux classes, A et B, qui sont connectées par une relation d'association ou d'agrégation, les indications indiquées dans le Tableau 3.8 doivent être suivies.

Comptez une DE pour chaque attribut de chaque classe, et une autre DE pour la relation d'association/agrégation du côté qui a une multiplicité (*), uniquement si elle est considérée comme indépendante.

Multiplicité A	Multiplicité B	Condition vérifiée	Comptabilisation des GDs, et SLDs
0..1	0..*	A et B sont indépendants	2 **GDs**, chacun avec un SLD
1	1..*	Si B est indépendant de A	2 GDs, chacun avec un SLD
		Si B dépend de A	1 GD, 2 SLD
1	0..*	Si B est indépendant de A	2 GDs, chacun avec un SLD
		Si B dépend de A	1 GD, 2 SLD
0..1	1..*	Si A est indépendant de B	2 GDs, chacun avec un SLD
		Si A dépend de B	1 GD, 2 SLD
0..1	0..1	A et B sont indépendants	2 GDs, chacun avec un SLD
1	1	A et B sont dépendants	1 GD, 1 SLD
1	0..1	Si B est indépendant de A	2 GDs, chacun avec un SLD
		Si B dépend de A	1 GD, 2 SLD
0..*	0..*	A et B sont indépendants	2 GDs, chacun avec un SLD
1..*	1..*	Si B est indépendant de A	2 GDs, chacun avec un SLD
		Si B dépend de A	1 GD, 2 SLD
1..*	0..*	Si B est indépendant de A	2 GD. , chacun avec un SLD
		Si B dépend de A	1 GD, 2 SLD

Figure 3.8. Règles pour identifier les groupes de données à partir de classes sans associations de composition

La dépendance/indépendance de deux classe A et B peut être déterminée comme suit :

1. Si les multiplicités minimales des deux terminaisons de l'association entre A et B égales à 0, alors A et B sont indépendantes. (Les instances de A et B peuvent exister indépendamment). L'analyse des points de fonction compte les classes A et B comme deux groupes logiques distincts, comme indiqué dans le tableau 3.8.
2. Si la multiplicité minimale coté A est supérieur à 0 (Une instance de A doit être associées à 1 ou plusieurs instances de B, dans ce cas, posez la question : « Quand vous supprimez une instance de A, que deviennent les instances de B qui y sont liées? »
   - 1er Cas : Si une instance de A est supprimée, alors tous les B qui y sont liés doivent également être supprimées, car l'entreprise n'est plus intéressée par les instances de B. Par conséquent, B est une entité dépendante de A, et A et B ensemble sont identifiés comme un seul groupe de données.

- 2ième Cas: Si la suppression de A n'est pas autorisée tant que les B y sont toujours liés parce que l'entreprise est toujours intéressée par les B, alors B est une entité indépendante de A, et A et B sont identifiés comme des fichiers logiques distincts.

3. Si la multiplicité minimale coté B est supérieure à 0 (Une instance de B doit être associées à 1 ou plusieurs instances de A, dans ce cas, posez la question inverse: « Quand vous supprimez une instance de B, que deviennent les instances de A qui y sont liées? »

Regle 4 : Généralisation/Spécialisation

Dans une hiérarchie d'héritage, on considère un seul groupe de données pour chaque chemin complet de classes allant de la superclasse racine à une sous-classe feuille. Pour chaque classe d'un groupe de données on compte un SLD. Pour chaque attribut d'une classe, on compte une données élémentaires (DE).

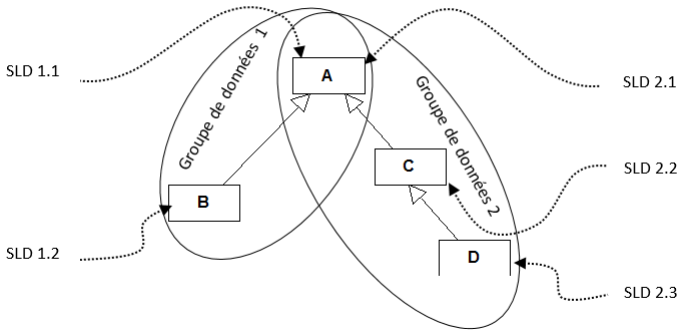


Figure 3.9. Diagramme de classes de la règle 4

Regle 5 : Classe-association

Pour une classe-association on compte un groupe logique de données et un SLD. On compte une DE pour chaque attribut de la classe-association avec deux autres DE en raison de ses relation avec les autres classes.

7.1.2.3 Identification des traitements[19]

[19] Garmus, David, Janet Russac, and Royce Edwards. Certified Function Point Specialist Examination Guide. CRC Press, 2010

Les types de traitements (ENT, SOR, INT) sont déterminés en identifiant l'intention principal de chaque traitement élémentaire. L'intention principale du traitement élémentaire doit être identifiée comme l'une des suivantes :

- Modification du comportement de l'application
- Maintenir un ou plusieurs GDI
- Présentation des informations à l'utilisateur

Classer le processus comme une ENT s'il a pour objectif principal l'un des éléments suivants :

- Il maintient un ou plusieurs GDI.
- Il modifie le comportement de l'application.
- Il comprend une logique de traitement pour accepter les données ou les informations de contrôle qui entrent dans le périmètre de l'application.

Classer le processus comme une SOR s'il a pour objectif principal de présenter des informations à l'utilisateur et qu'il comprend au moins l'une des formes de logique de traitement suivantes :

- Des calculs mathématiques sont effectués.
- Un ou plusieurs GDI sont mis à jour.
- Les données dérivées sont créées.
- Le comportement de l'application est modifié.

Classer le processus comme une INT s'il a pour objectif principal de présenter des informations à l'utilisateur, et :

- Il référence un groupe de données.
- Il ne satisfait pas aux critères pour être classé en SOR.

D'autres différences entre les traitements (ENT, SOR, INT) sont identifiées par les formes de **logiques de traitement** utilisées par chaque traitement.

La logique de traitement est définie comme l'une des exigences spécifiquement demandées par l'utilisateur pour accomplir un traitement élémentaire (ENT, SOR, INT), tel que des validations, des algorithmes ou des calculs, des consultations et des mises à jour de groupes de données. Ces exigences peuvent inclure les actions suivantes :

1. **Les validations sont effectuées**. Par exemple, lors de l'ajout d'un nouvel employé à une organisation, le traitement d'ajout d'employé a une logique de traitement qui valide les informations ajoutées.

2. **Des formules mathématiques et des calculs sont effectués**. Par exemple, lors de la création de rapports sur tous les employés d'une organisation, le traitement comprend le calcul du nombre total d'employés salariés, d'employés horaires et de tous les employés.
3. **Les valeurs équivalentes sont converties**. Par exemple, l'âge de l'employé est converti en une tranche d'âge à l'aide d'un tableau. Les tranches d'âges sont retrouvées en récupérant les valeurs dans une table, de sorte qu'il n'est pas nécessaire d'effectuer des calculs. Un autre exemple, un traitement élémentaire référence les taux de conversion de devises. La conversion est effectuée en récupérant les valeurs dans des tables.
4. **Les données sont filtrées et sélectionnées à l'aide de critères spécifiés**. Par exemple, pour générer une liste d'employés par service, un traitement élémentaire compare le numéro de service pour sélectionner et répertorier les employés de ce service.
5. **Des conditions sont analysées pour déterminer lesquelles sont applicables**. Par exemple, la logique de traitement exercée par le traitement élémentaire lorsqu'un employé est ajouté dépendra du fait qu'un employé est payé en fonction du salaire ou des heures travaillées.
6. **Un ou plusieurs GDI sont mis à jour**. Par exemple, lors de l'ajout d'un employé, le traitement élémentaire met à jour le GDI Employé pour sauvegarder les données de l'employé.
7. **Un ou plusieurs fichiers GDI ou GDE sont référencés**. Par exemple, lors de l'ajout d'un employé, le GDI Service est référencée afin de déterminer le code du Service de l'employé.
8. **Des données ou des informations de contrôle sont récupérées**. Par exemple, pour afficher une liste d'employés, les informations sur les employés sont extraites d'un groupe de données.
9. **Des données dérivées sont créées en transformant les données existantes pour créer des données supplémentaires**. Par exemple, pour déterminer (déduire) le numéro d'enregistrement d'un employé (par exemple, BELDJ01), les données suivantes sont concaténées :
    - Les trois premières lettres du nom de famille de l'employé (par exemple, BEL pour Belmadi)
    - Les deux premières lettres du prénom de l'employé (ex : DJ pour Djamel)
    - Un numéro de séquence unique à deux chiffres (commençant par 01)
10. **Le comportement de l'application est modifié**. Par exemple, le comportement du traitement élémentaire de paiement des employés est modifié lorsqu'un changement est

apporté pour payer les employés chaque 15 jour au lieu de les payer chaque mois, ce qui donne 24 périodes de paiement par an au lieu de 12.

11. **Des données sont préparées et présentées en dehors des frontières de l'application**. Par exemple, une liste d'employés est formatée et affichée pour l'utilisateur.
12. **Des données ou des informations de contrôle entrant dans l'application sont acceptées**. Par exemple, un utilisateur saisit des informations pour ajouter une commande client à l'application.
13. **Un ensemble de données est trié ou organisé.** Par exemple, une liste d'employés peut être triée par ordre alphabétique ou par lieu de travail, ou, sur un écran de saisie de commande, les informations d'en-tête de commande sont disposées en haut de l'écran et les détails de la commande sont placés en dessous. Cette forme de logique de traitement n'impacte pas l'identification du type ni ne contribue à l'unicité d'un traitement élémentaire.

Forme de logique de traitement		**Type du traitement**		
		ENT	**SOR**	**INT**
1	Les validations sont effectuées	*P*	*P*	*P*
2	Des formules mathématiques et des calculs sont effectués	*P*	*O**	*N*
3	Les valeurs équivalentes sont converties	*P*	*P*	*P*
4	Les données sont filtrées et sélectionnées à l'aide de critères spécifiés pour comparer plusieurs ensembles de données	*P*	*P*	*P*
5	Les conditions sont analysées pour déterminer lesquelles sont applicables	*P*	*P*	*P*
6	Un ou plusieurs GDI sont mis à jour	*O**	*O**	*N*
7	Un ou plusieurs GDI ou GDE sont référencés	*P*	*P*	*O*
8	Les données ou les informations de contrôle sont récupérées	*P*	*P*	*O*
9	Les données dérivées sont créées en transformant les données existantes pour créer des données supplémentaires	*P*	*O**	*N*
10	Le comportement de l'application est modifié	*O**	*O**	*N*
11	Des données sont préparé et présenté en dehors des frontières de l'application	*P*	*O*	*O*
12	Des données ou des informations de contrôle entrant dans l'application sont acceptées.	*O*	*P*	*P*
13	Un ensemble de données est trié ou organisé	*P*	*P*	*P*

Figure 3.10. Résumé des logiques de traitement utilisée par les ENT, les SOR et les INT

Légende :
P (Possible) : le type de traitement peut effectuer la forme de traitement, mais ce n'est pas obligatoire ;
O :(Obligatoire) : il est obligatoire que le type traitement effectuer la forme de traitement ;
O* : il est obligatoire que le type de traitement exécute au moins l'un des (O*) formes de traitement ;
N : le type de traitement ne peut pas exécuter la forme de traitement.

Le Tableau 3.10 résume les formes de logiques de traitement pouvant être exécutées par les ENT, les SOR et les INT. Pour chaque type de traitement, certains types de logique de traitement doivent être exécutés pour accomplir l'objectif principal de ce type, et certains types de traitement peuvent ou ne peuvent pas être exécutés.

7.1.3 Etape 3 : Evaluation de la complexité de chaque composant

Après l'identification des composants du système étudié (données et traitements), l'étape suivante consiste à niveau de complexité des composants. Trois niveaux de complexité des composants : *faible, moyenne,* et *élevée*.

7.1.3.1 Complexité des groupes logiques de données (GDI et GDE)

Un GDI (GDE) est composé de Données Elémentaires (DE). Une DE correspond à un champ de données (Une DE est un attribut au sens diagramme de classes UML).

On peut parfois identifier plusieurs Sous-ensemble Logique de Données (SLD) dans un groupe de données interne ou externe (Dans un diagramme de classe, un SLD peut être assimilé à une classe spécialisée, on compte un SLD pour la classe mère, et un SLD pour chaque classe fille).

Le niveau de complexité d'un GDI (GDE) est déterminé par le nombre de Sous-ensembles Logique de Données (SLD) et de Données Elémentaires (DE) du groupe logique de données (Tableau 3.7).

SLD	DE		
	1-19	**20-50**	**51 et plus**
1	Faible	Faible	Moyen
2 à 5	Faible	Moyen	Elevé
6 et plus	Moyen	Elevé	Elevé

Tableau 3.7. Niveaux de complexité GDE/GDI

7.1.3.2 Complexité d'une Entrée (ENT)

Le niveau de complexité d'une Entrée est déterminé par le nombre de Groupe de Données Référencées (GDR) et de DE de l'Entrée (Tableau 3.8).

Les groupes de données référencés (GDR) sont:

- Les Groupe logiques de Données Internes mis à jour par l'Entrée ;
- Les Groupe logiques de Données Internes consultés par l'Entrée ;
- Le Groupes logiques de Données Externes consultés par l'Entrée.

GDR	DE		
	1-4	5-15	16 et plus
0 ou 1	Faible	Faible	Moyen
2	Faible	Moyen	Elevé
3 et plus	Moyen	Elevé	Elevé

Tableau 3.8. Niveaux de complexité des ENT

Concernant les Données élémentaires, seules les DE mis à jour par l'Entrée sont comptabilisées, les DE consultés ne sont pas comptabilisés.

7.1.3.3 Complexité d'une sortie (SOR) ou d'une interrogation (INT)

Le niveau de complexité d'une Sortie ou d'une interrogation est déterminé par le nombre de Groupes de Données Référencées (GDR) et de DE (Tableau 3.9). Les groupes de données référencés (GDR) sont:

- Les Groupes logiques de Données Internes consultés.
- Les Groupes logiques de Données Externes consultés.

Le nombre de DE est celui de DE utilisées par la Sortie ou l'interrogation.

GDR	DE		
	1-5	5-19	20 et plus
0 ou 1	Faible	Faible	Moyen
2 à 3	Faible	Moyen	Elevé
4 et plus	Moyen	Elevé	Elevé

Tableau 3.9. Niveaux de complexité des INT et SOR

7.1.4 Etape 4 : Calcul du poids en Points de fonction de chaque composant

Après l'évaluation de la complexité des composants, on peut déterminer le nombre de point de fonction qui correspond à chaque composant, en fonction de son type et de sa complexité (Tableau 3.10).

Degré de complexité du composant	Faible	Moyen	Elevé
Nombre de point de fonction du GDI	7	10	15
Nombre de point de fonction du GDE	5	7	10
Nombre de point de l'ENT	3	4	6
Nombre de point de fonction de SOR	4	5	7
Nombre de point de fonction de l'INT	3	4	6

Tableau 3.10. Nombre de points de fonction correspondant à chaque composant en fonction de son niveau de complexité

Type de composant	Complexité	Nombre	Poids	Totaux par types de composant
GDI	Faible	___	x7=	___
	Moyen	___	x10=	___
	Elevé	___	x15=	___
GDE	Faible	___	x5=	___
	Moyen	___	x7=	___
	Elevé	___	x10=	___
ENT	Faible	___	x3=	___
	Moyen	___	x4=	___
	Elevé	___	x6=	___
SOR	Faible	___	x4=	___
	Moyen	___	x5=	___
	Elevé	___	x7=	___
INT	Faible	___	x3=	___
	Moyen	___	x4=	___
	Elevé	___	x6=	___
Nombre de points de fonction bruts : UFP				=

Tableau 3.11. Comptage des points de fonction bruts

7.1.5 Etape 5 : Calcul du nombre brut de points de fonction (*Unadjusted Function Point*: UFP)

Le calcul se fait en sommant pour chaque type de composant, les composants ayant le même niveau de complexité. Chaque nombre est ensuite multiplié par le poids correspondant (Tableau 3.11). On réalise ensuite le total général des points de fonction brut *UFP* (*Unadjusted Function Point*).

7.1.6 Etape 6: Calcule du nombre de points de fonctions ajustés

Le nombre de points de fonction brut est calculé sur la base de composants fonctionnels et de leur complexité. L'étape suivante (facultative) consiste à ajuster le nombre de points de fonction brut (UFP) par le facteur d'ajustement de valeur (*value adjustment factor :VAF*).

Le facteur d'ajustement de valeur (VAF) est basé sur 14 caractéristiques générales du système (CGS). Les CGS décrivent l'environnement dans lequel l'application est déployée et l'influence qu'elle peut avoir sur le travail de développement (Tableau 3.12) Chaque CGS peut avoir une degré d'influence allant de 0 (aucune influence) à un maximum de 5 (pleine influence) :

- 0 = aucune influence
- 1 = influence mineure
- 2 = influence modérée
- 3 = influence moyenne
- 4 = influence significative
- 5 = forte influence.

▪ Caractéristique		Description
1	Communication de données	Le système en cours de développement aurait-il besoin de gérer les aspects de communication de données ? (saisie et impression à distance, protocoles de communication, etc.)
2	Traitement de données distribué	Le système en cours de développement devra-t-il prévoir un traitement distribué des données ? (Les données et les traitement sont-t-ils distribués)
3	Critères de performance	L'utilisateur a-t-il des exigences en matière de temps de réponse ?
4	Configuration matérielle très chargée	Le système en cours de développement aurait-il besoin d'optimiser l'utilisation du matériel ? (pour s'adapter aux limites du matériels)
5	Fréquences de transactions	Quelle est la fréquence des pointes d'exécution des transactions (quotidien, hebdomadaire, mensuel…).
6	Saisie de données en interactive	Le pourcentage des transactions qui sont des saisies de données interactives par rapport aux transactions traité par lots ? (forte influence à partir de 30%)
7	Efficacité des interfaces utilisateur	Les interfaces du système en cours de développement devrait-il être conçu de manière à être efficace pour les utilisateurs finaux ? (convivialité, multilinguisme, etc.)
8	Mise à jour en temps réel des GDI	Combien d'ILF (GDI) seraient mis à jour en ligne par le système ?
9	Traitement complexe	Le système aurait-il besoin de gérer des équations mathématiques ou une logique complexes ? (contrôle, traitement logique, algorithme, traitement des interruptions, traitement d'entrées/sorties multiples)
10	Réutilisabilité	Le système en cours de développement devrait-il être conçu de manière à pouvoir répondre aux besoins de réutilisation ?
11	Facilité d'installation	L'application en cours de développement aurait-elle besoin d'être conçue pour que l'installation soit plus facile ?
12	Facilité opérationnelle	L'application contiendrait-elle des modules pour faciliter la tâche du personnel d'exploitation ?
13	Portabilité	L'application est-elle spécifiquement conçue, développée maintenue pour être installée sur de multiples des environnements matériel et logiciel différents?
14	Evolutivité	L'application est-elle spécifiquement conçue, développée maintenue pour faciliter le changement ?

Tableau 3.12. Caractéristiques générales du système

Soit DI_i, degré d'influence du $i^{\text{ième}}$ caractéristique générale du système. Une fois que les caractéristiques générales du système ont été évaluées avec leur degré d'influence respectif, on peut calculer le Degré d'Influence Total **(**DIT**)** qui est la somme des degrés d'influence de chaque caractéristique ($0 \leq DIT \leq 70$) :

$$DIT = \sum_{i=1}^{14} DI_i$$

Le facteur d'ajustement se calcule par :

$$\boldsymbol{VAF = 0.65 + DIT * 0.01}$$

Le facteur d'ajustement prendra donc des valeur entre 0.65 et 1.35 ($0.65 \leq \boldsymbol{VAF} \leq 1.35$)

Le passage du nombre de points de fonction brut (UFC) au nombre ajusté (FP), se fait par la multiplication de l'UFC par le Facteur d'Ajustement (VAF) :

$$\boldsymbol{FP = VAF * UFC}$$

Le Facteur d'Ajustement ajuste donc le nombre brut de points de fonction avec un taux allant de -35% à $+35\%$.

7.1 Transformation de points de fonction en charge

Deux solutions possibles pour calculer la charge à partir du nombre de points de fonction, la Conversion des FP en lignes de code, et la transformation directe des points de fonctions en charge.

7.1.1 Transformation directe des points de fonctions en charge

Le coefficient de transformation de points de fonction en effort est variable selon l'environnement matériel et humain. Il est recommandé que chaque entreprise établisse sa base de projets pour déterminer ses propres coefficients. Les valeurs couramment admises dépendent de la taille du projet (Tableau 3.13).

Taille du projet	Charge (Homme-jour) par 1 FP
Petit	2
Moyen	3
Grand	4

Tableau 3.13. Coefficients de transformation de points de fonction en charge

7.1.2 Transformation des points de fonctions en lignes de code

Les points de fonction non ajustés peuvent être convertis en lignes de code source dans le langage de programmation utilisée. Après la conversion, il est possible d'appliquer une technique d'estimation de l'effort à partir du nombre de lignes de code (COCOMO intermédiaire, COCOMO II, etc.). Les facteurs de l'environnement (compétence des

programmeurs, etc.) peuvent entrainer une variation significative du coefficient de conversion. De ce fait, il est recommandé de aux organismes d'établir leurs propres coefficients de conversion, en s'appuyant sur leur historique des projets terminés. Lorsqu'il n'y a pas de données de projet terminées disponibles pour l'estimation, il est possible de faire référence aux tables de conversion disponible (Tableau 3.14)[20].

Langage	Coefficient de conversion			
	Moyen	Médiane	Bas	Elevé
C	97	99	39	333
C++	50	53	25	80
C#	54	59	29	70
HTML	34	40	14	48
J2EE	46	49	15	67
Java	53	53	14	134
JavaScript	47	53	31	63
.NET	57	60	53	60
Oracle	37	40	17	60
PL/SQL	37	35	13	60
SQL	21	21	13	37
VB.NET	52	60	26	60
Visual Basic	42	44	20	60

Tableau 3.14. Conversion de points de fonction en lignes de code

7.2 Avantages et les inconvénients de la méthode de point de fonction

Avantages

- L'unité de point de fonction utilisée par la méthode, est la mesure de taille de logiciels la plus utilisée.
- La méthode est supervisé par l'IFPUG qui met à niveau la méthode, et prend en charge la certification de personne pour l'application de la méthode.
- La méthode de points de fonction est bien documentée, l'IFPUG publie périodiquement, le manuel d'utilisation « «Function Point Counting Practices », aussi, de nombreux livres et publications scientifiques sont consacré à la méthodes.
- Elle peut être réalisée très en amont dans le cycle de développement de l'application.
- La méthode de points de fonction est indépendante du langage programmation utilisé.

Inconvénients

- Difficulté d'identifier les composants et leurs types : les GDI et GDE ne sont pas toujours facilement identifiables au début du projet, aussi, certains traitements ne correspondent

[20] http://www.qsm.com/resources/function-point-languages-table

pas aux trois types définis par la méthode (ENT, SOR, INT).

- L'application de cette méthode demande encore un travail important, la présence d'un spécialiste de cette méthode et ne peut pas encore être automatisée
- Le comptage d'une personne à une autre peut varier de 10% contrairement à ce que la méthode présente théoriquement.
- Cette méthode est conçue pour les systèmes d'information de gestion. Elle n'est pas applicable pour les autres types de logiciel (industrielles, temps réel, scientifique). En effet, dans ces cas précis la méthode des points de fonction ne permet pas d'apprécier la complexité de développement d'algorithmes.

7.3 Exemple

On considère les composants suivants identifiés à partir de la spécification d'un système à développer :

- 4 GDI (1 simples, 1 moyennes et 2 complexes).
- 2 GDE (1 moyennes et 1 complexes).
- 10 ENT (3 simples, 3 moyennes et 4 complexes).
- 10 SOR (4 simples, 4 moyennes et 2 complexes).
- 5 INT (3 simples, 2 moyennes et 0 complexes).

Le responsable du comptage a apprécié l'application suivant les 14 Caractéristiques Générales du Système. L'addition des 14 Degrés d'Influence obtenues a donné DIT=55.

Type de composant	Complexité	Nombre	Poids	Totaux par types de composant
GDI	Faible	1	x7=	7
	Moyen	1	x10=	10
	Elevé	2	x15=	30
GDE	Faible	0	x5=	0
	Moyen	1	x7=	7
	Elevé	1	x10=	10
ENT	Faible	3	x3=	9
	Moyen	3	x4=	12
	Elevé	4	x6=	24
SOR	Faible	4	x4=	16
	Moyen	4	x5=	20
	Elevé	2	x7=	14
INT	Faible	3	x3=	9
	Moyen	2	x4=	8
	Elevé	0	x6=	0
Nombre de points de fonction bruts : UFP=				176

VAF=(DIT*0.01)+0.65=1.2 ,
FP=TCF*UFC=176*1.2=211 FP

Chapitre 4 .

LA PLANIFICATION

1 Généralités

La planification d'un projet est le processus consistant à décider comment le travail d'un projet sera organisé en tâches distinctes, et quand et comment ces tâches seront exécutées. Autrement dit, la planification d'un projet consiste à organiser le projet en tâches distinctes, et prévoir l'ordonnancement des tâches du projet sur le plan des délais et sur le plan de l'utilisation des ressources (élaboration d'un calendrier).

Lors de la planification d'un projet, il est essentiel de comprendre les aspects d'un projet :

- Un projet consiste en un certain nombre de tâches. L'exécution de ces tâches entraîne l'exécution du projet.
- Un projet comporte un certain nombre de jalons. Atteindre les jalons signifie l'achèvement d'un certain groupe de tâches.
- Un projet a un point de départ, qui est le premier jalon du projet.
- Un projet a un point de fin, qui est le dernier jalon ou jalon de fin du projet.
- Toutes les tâches d'un projet doivent être réalisées entre les jalons de début et de fin.
- Certaines tâches d'un projet peuvent être exécutées en parallèle.
- Certaines tâches doivent être exécutées séquentiellement (l'une après l'autre).
- Les réalisation d'une tâche nécessite une ou plusieurs ressources (ressources humaines, matériel, équipement, outils logiciel).
- Il existe une limite au nombre de ressources pouvant être déployées pour une tâche donnée.

La planification permet au chef du projet d'effectuer les opérations suivantes :

- Surveiller et contrôler les activités du projet.
- Déterminer la meilleure façon d'allouer les ressources afin d'atteindre l'objectif du projet.
- Évaluer l'impact des retards sur le projet.
- Optimiser l'allocation de ressources
- Fournir une base pour suivre l'avancement du projet.

2 Etapes de planification

La planification d'un projet passe par plusieurs étapes (Figure 4.1):

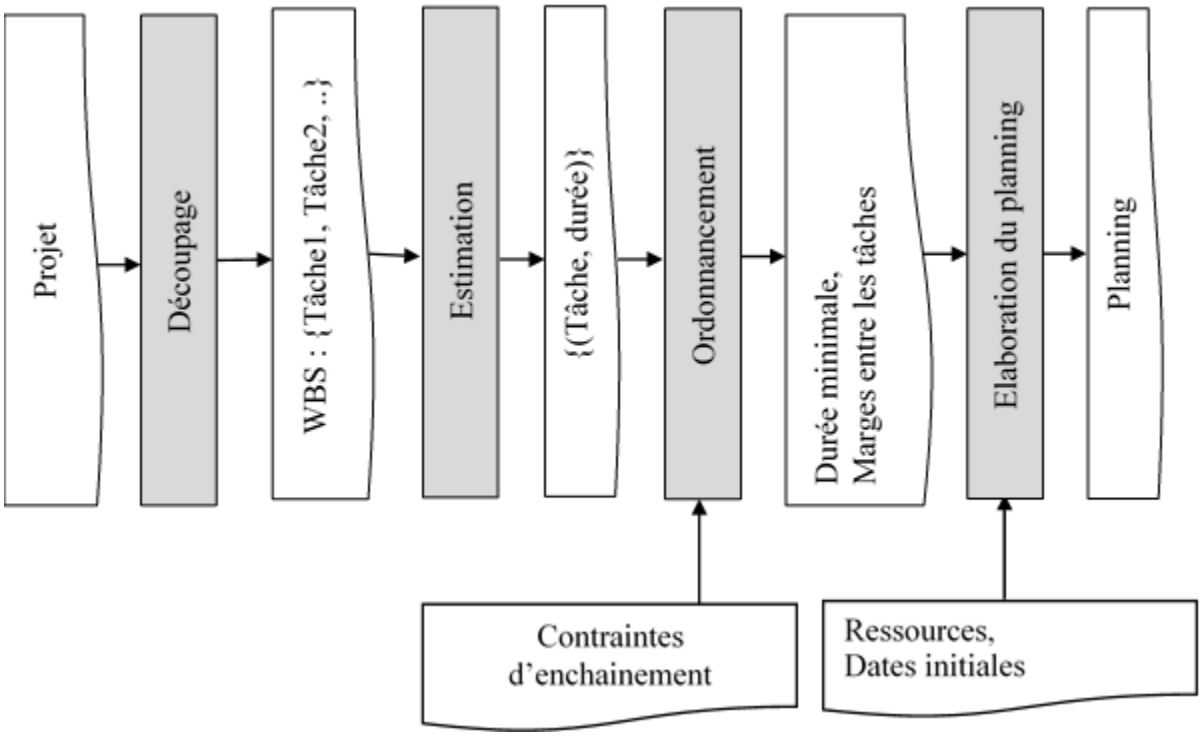


Figure 4.1. Processus de planification

2.1 Découpage du projet en tâches

La première étape de la planification d'un projet consiste à préparer un WBS (Work Breakdown Structure) (Section Chapitre 25.1), contenant toutes les tâches à planifier. Le premier élément d'un WBS est le jalon de début du projet. Le dernier élément d'un WBS est le jalon de fin, qui signifie l'achèvement du projet. Le projet se déroule entre ces deux jalons.

2.2 Estimation de la durée de chaque tâche

Pour réaliser une planification du projet, il est nécessaire d'avoir une estimation des durées des différentes tâches représenté dans le WBS du projet.

2.3 L'ordonnancement

Après avoir préparé le WBS et estimer les durées des tâches, l'étape suivante consiste à déterminer *les contraintes d'ordonnancement* et les possibilités de parallélisme des tâches répertoriées dans le WBS. En fait, certaines tâches peuvent se dérouler en parallèle, tandis que d'autres ne peuvent pas commencer qu'après la fin de leurs tâches prédécesseurs.

Par définition, la tâche jalon de début n'a pas de prédécesseur. Les autres tâches du projet, devraient avoir au moins un prédécesseur (certaines tâches peuvent en avoir plusieurs). Par définition, le jalon de fin n'a pas de successeur.

La détermination des prédécesseurs consiste en un processus qui comprend l'examen de chaque tâche pour déterminer les tâches qui doivent être terminées avant que cette tâche ne puisse commencer. Les contraintes d'enchainement entre les tâches sont souvent représentées à l'aide du graphe des antécédents (Section 3.2). Le graphe des antécédents permettra de définir des dates initiales de début et de fin des tâches.

2.4 Le planning

L'ordonnancement permet de définir des dates initiales de début et de fin pour les tâches et le projet, cependant, ces dates ne prennent en considération que les contraintes d'enchainement entre les tâches, sans se soucier des contraintes liées au ressources utiliser pour réaliser les différentes tâches, par exemple, même si un ensemble de tâches peuvent être réalisées en parallèle, les ressources nécessaires pour exécuter ces tâches en concurrence ne sont pas toujours disponibles. L'objectif du planning consiste à prendre en compte les contraintes d'enchainement et les contraintes de ressources pour déterminer les dates de début et de fin pour chaque tâche, ainsi que la date de début et de fin du projet.

3 Les techniques d'ordonnancement

L'objectif de des techniques d'ordonnancement consiste à modéliser les tâches du projet et leurs contraintes d'enchainement, afin de calculer les dates prévues pour accomplir les tâches, et déterminer le chemin critique qui définit le temps minimum requis pour terminer le projet.

Il existe deux techniques de représentation de l'ordonnancement des activités : la méthode des antécédents et la méthode PERT.

3.1 Le réseau PERT (Program Evaluation and Review Technique)

Dans le réseau PERT, les tâche sont représenté sur les arcs (Figure 4.2), les nœuds représentent des événements de fin et/ou de début d'une ou plusieurs tâches.

Pour représenter de façon correcte les contraintes d'enchaînement, il est nécessaire parfois d'introduire des tâches fictives, représentées par des flèches en pointillés.

Exemple

Dans l'exemple de la Figure 4.2, une tâche fictive a été introduite pour représenter la contrainte entre les tâches B et C. Une autre tâche fictive est introduite pour représenter la contrainte entre les tâches C et F.

Tâche	Prédécesseurs
A	-
B	-
C	A,B
D	B
E	C
F	C,D
G	F

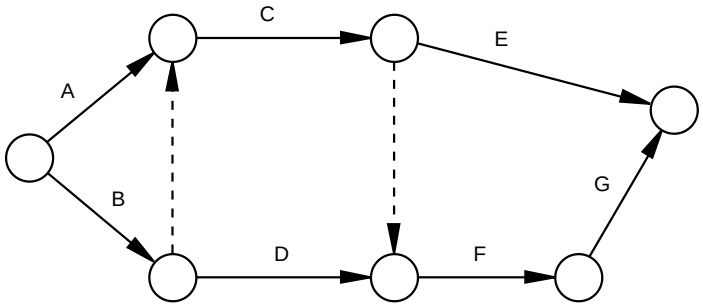


Figure 4.2. Exemple d'un réseau PERT

3.2 La méthode des antécédents

Le principe de la méthode des antécédents consiste à représenter les tâches par des nœuds, tandis que les arcs représentent les contraintes d'ordonnancement.

Exemple

La Figure 4.3 montre la représentation du même projet par un graphe des antécédents.

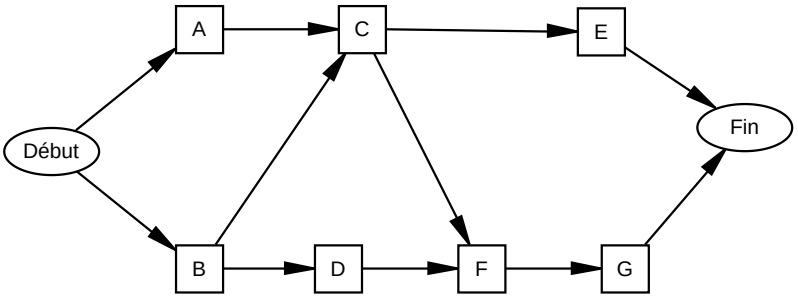


Figure 4.3. Un exemple d'un graphe des antécédents

Le réseau PERT et le graphe des antécédents sont équivalents. Dans ce cours, nous utilisons le formalisme du graphe des antécédents, qui est plus simple et qui est retenu par la majorité des progiciels de planification (comme MS Project).

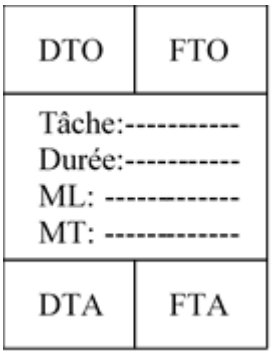
3.3 Méthode du chemin critique (Critical Path Method)

La méthode du chemin critique consistant à examiner toutes les tâches qui doivent être réalisées pour déterminer le temps d'achèvement du projet, et identifier le chemin critique (les tâches qui ne doivent pas être retardées pour terminer le projet dans un minimum de temps).

En fait, parmi tous les chemins d'un graphe des antécédents allant de début à la fin, il en existe un appelé chemin critique qui relie les tâches "critiques". Les tâches critiques sont les tâches dont le retard impliquera un retard effectif du projet.

On détermine le chemin critique avec les paramètres suivants :

- Les dates au plus tôt
  - Date de début au plus tôt (DTO)
  - Date de fin au plus tôt (FTO)
- Les dates au plus tard
  - Date de début au plus tard (DTA)
  - Date de fin au plus tard (FTA)
- Les marges
  - Marge Libre (ML)
  - Marge Totale (MT)



3.3.1 Calcul des dates au plus tôt (*DTO* et *FTO*)

Compte tenu des *contraintes d'enchaînement*, de la durée des tâches et *de la date de début de projet*, une tâche T_i ne peut pas commencer avant *DTO* et ne peut se terminer avant *FTO*.

Compte tenu des contraintes d'enchaînement, de la durée des tâches et de la date de fin de projet, une tâche T_i ne doit pas commencer après *DTA* et ne doit pas se terminer après *FTA*, sinon la date de fin du projet serait dépassée.

Pour calculer ces dates, nous devons avoir la durée d_i de chaque tâche T_i. On va faire l'hypothèse d'une date de début de projet (t_0) et on va parcourir le graphe vers l'avant en respectant les liens.

1er cas : Si la tâche T_i se situe en début de projet:

$$DTO(T_i) = t_0$$

$$FTO(T_i) = t_0 + d_i - 1$$

Exemple

Soit la tâche A située au début du projet

DTO(A)= t_0=2

FTO(A)= $t_0 + d_A - 1 = 2 + 5 - 1 = 6$

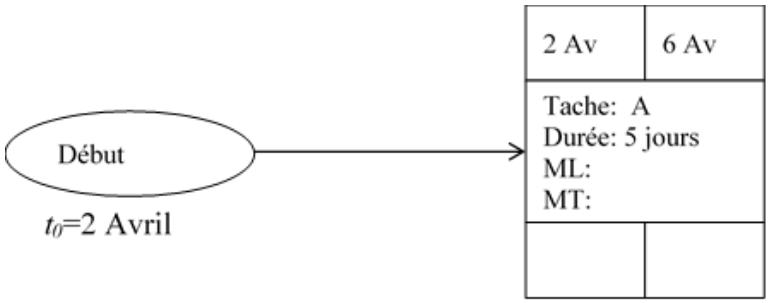


Figure 4.4. Exemple de calcul des dates au plus tôt d'une tâche située au début

2ième cas : Si la tâche T_i ne se situe pas en début de projet, elle a des prédécesseurs:

$$DTO(T_i) = sup\ \{FTO\ (prédécesseurs)\} + 1$$
$$FTO(T_i) = DTO(T_i) + d_i - 1$$

Exemple

Soit la tâche D ayant 3 prédécesseurs : A, B, C :

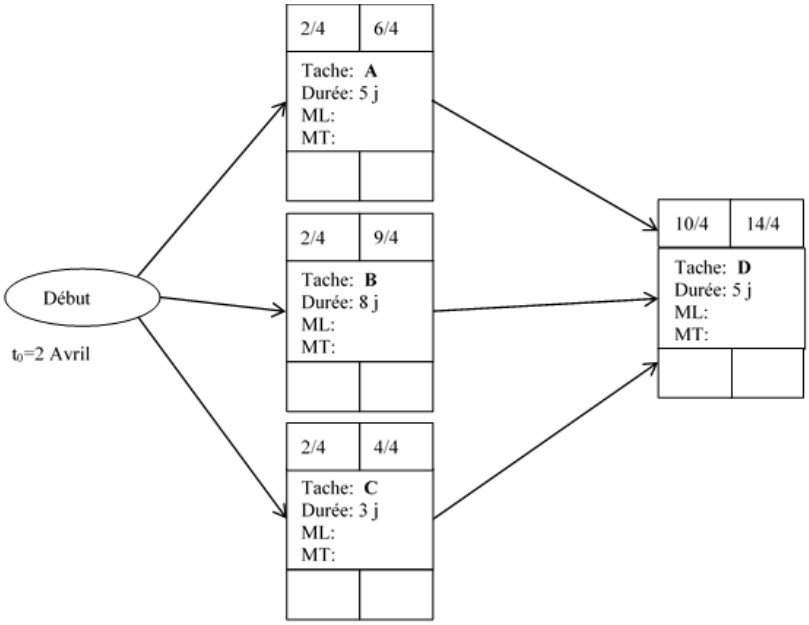


Figure 4.5. Exemple de calcul des dates au plus tôt d'une tâche qui n'est pas située au début

$DTO(D)$ = sup {FTO (prédécesseurs)} + 1
= sup{FTO (A),FTO (B),FTO (C)} + 1
= sup{6,9,4} + 1=9+1=10
FTO(D) = DTO(D)+$d_D - 1 = 10 + 5 - 1 = 14$

3.3.2 Calcul des dates au plus tard (*DTA* et *FTA*)

Pour calculer les dates au plus tard de chacune des tâches, on va faire l'hypothèse d'une date de fin de projet t_f et on va parcourir le graphe vers l'arrière en respectant les liens.

1er cas : Si la tâche T_i se situe en fin de projet:

$$\boldsymbol{FTA(T_i) = t_f}$$
$$\boldsymbol{DTA(T_i) = FTA(T_i) - d_i + 1}$$

Exemple

Soit la tâche située en fin du projet :

Figure 4.6. Exemple de calcul des dates au plus tard d'une tâche située à la fin

FTA(K) = $t_f = 15$
DTA(K) = FTA(K) $- d_k + 1 = 15 - 5 + 1 = 11$

2ième cas : Si la tâche $\boldsymbol{T_i}$ ne se situe pas en fin de projet, elle a des successeurs:

$$\text{FTA}(\boldsymbol{T_i}) = \boldsymbol{inf}\{\boldsymbol{DTA}(\boldsymbol{successeurs})\} - \mathbf{1}$$
$$\text{DTA}(\boldsymbol{T_i}) = \text{FTA}(\boldsymbol{T_i}) - \boldsymbol{d_i} + \mathbf{1}$$

Exemple

Soit la tâche D ayant des successeurs A, B, C

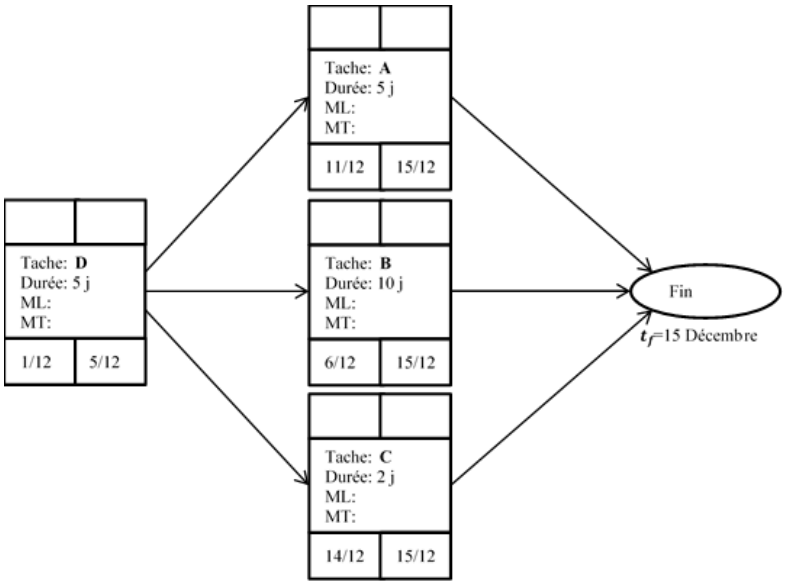


Figure 4.7. Exemple de calcul des dates au plus tard d'une tâche qui n'est pas située à la fin

$$\text{FTA(D)} = inf\{DTA(successeurs)\} - 1$$
$$= inf\{DTA(A), DTA(B), DTA(C)\} - 1$$
$$= inf\{11,6,14\} - 1 = 6 - 1 = 5$$
$$\text{DTA}(T_i) = \text{FTA(D)} - d_D + 1 = 5 - 5 + 1 = 1$$

3.3.3 Les marges

La marge est la durée pendant laquelle une tâche peut être retardée une tâche sans impacter le projet. On distingue deux types de marges : la marge totale et la marge libre.

3.3.3.1 La marge totale (MT) (Total slack)

On dispose d'une marge totale $\boldsymbol{MT(T_i)}$ sur une tâche $\boldsymbol{T_i}$ si on peut planifier $\boldsymbol{T_i}$ à la date $(\boldsymbol{DTO(T_i)} + \boldsymbol{MT(T_i)})$ sans retarder la date de fin du projet. Autrement dit, la marge total est le retard maximum que peut atteindre une tâche avant qu'elle ne retarde la fin du projet.

$$\boldsymbol{MT(T_i) = FTA(T_i) - FTO(T_i)}$$
$$\boldsymbol{= DTA(T_i) - DTO(T_i)}$$

3.3.3.2 La marge libre (ML) (Free slack)

On dispose d'une marge libre $\boldsymbol{ML(T_i)}$ *sur* une tâche $\boldsymbol{T_i}$ *si on peut planifier* $\boldsymbol{T_i}$ *à la date* $(\boldsymbol{DTO(T_i)} + \boldsymbol{ML(T_i)})$ *sans que* cela ait un impact sur ses successeurs. Autrement dit, la marge libre est le retard maximum que peut atteindre une tâche avant qu'elle ne retarde ses successeurs

(c'est-à-dire qu'on peut toujours les planifier au plus tôt).

$$\text{ML}(T_i) = inf\{DTO(Successeurs(T_i)\} - FTO(T_i) - 1$$

Exemple

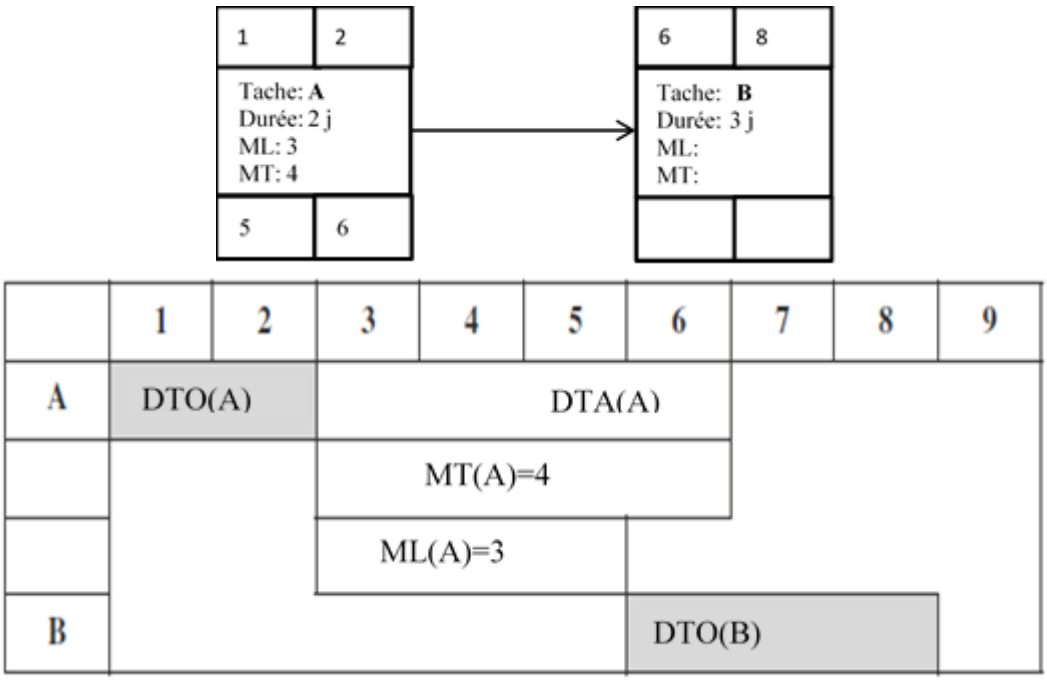


Figure 4.8. Exemple de calcul des marges totales et libre

Durée de A= 2 jours
DTO(A)=1, FTO(A)=2, DTA(A)=5, FTA(A)=6
DTO(B)=6 (La tâche B commence au plus tôt en période 6).
MT(A)=6-2=5-1=4
ML(A)=inf(DTO(successeurs))-FTO(A)-1=DTO(B)- FTO(A)-1=6-2-1=3

3.3.4 Chemin critique

Une tâche avec une marge totale nulle (MT=0) est considérée comme une **tâche critique**. Si une tâche critique est retardée, la date de fin du projet est également retardée.

Le chemin critique de la planification de projet est la séquence des tâches critiques qui commence au début du projet et se terminent à la fin du projet. La date de début de la première tâche du chemin critique correspond à la date de début du projet, et la date de fin de la dernière tâche du chemin critique correspond à la date de fin du projet.

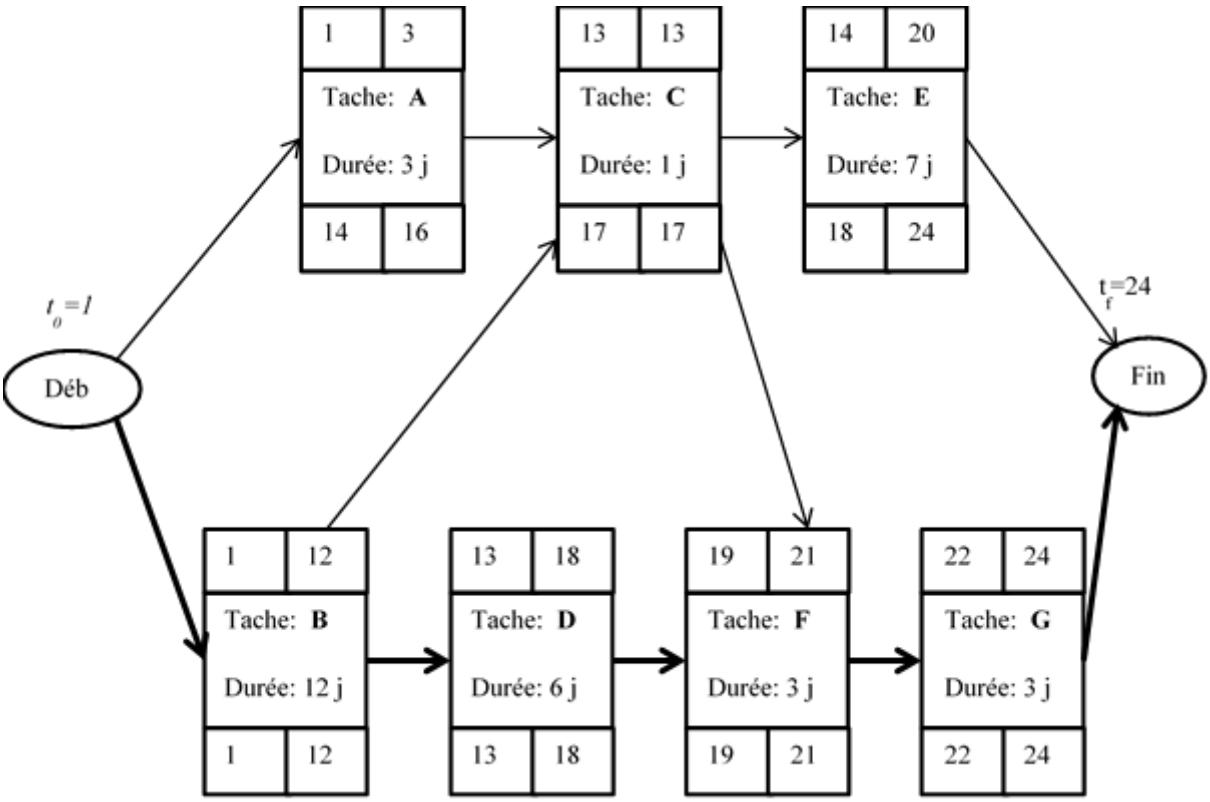
Exemple

Les tâches nécessaires à la réalisation d'un projet, leurs durées, ainsi que les contraintes d'enchainement qui les relient sont données au tableau suivant :

Tâche	Durée	Prédécesseurs
A	3	-
B	12	-
C	1	A,B
D	6	B
E	7	C
F	3	C,D
G	3	F

Tableau 4.1. Exemple d'un projet à planifier

Construire le graphe des antécédents du projet. Calculer les paramètres clés et fait les figurer sur le réseau (Pour chaque tâche indiquer : DTA, FTO, DTA, FTA, MT, ML) et déterminer le chemin critique.



Chemin critique : B→ D→ F→ G

Figure 4.9. Exemple de la méthode du chemin critique

3.4 Diagramme de Gantt

Le graphe des antécédents donne les dates de réalisation des différentes taches, sans prendre en considération :

1. Les contraintes de ressources (Les personnes à affecter au tâches, ressources matériels)
2. Les contraintes de calendrier (jours non ouvrables, jours fériés, etc.).

Le planning doit prendre en considération les contraintes des ressources et de calendrier.

Le diagramme de Gantt est utilisé pour représenter le planning, et se construit de la façon suivante :

- Chaque colonne représente une unité du temps. (Heure, Jour, Semaine, etc.)
- Les ressources sont représentées sur les lignes.
- Chaque tâche est représentée par une barre dont la longueur correspond à sa durée.
- Les contraintes d'enchainement entre les tâches sont représentées par des flèches.
- Les jalons de début et de fin et des évènements importants peuvent être représentés par des losange.

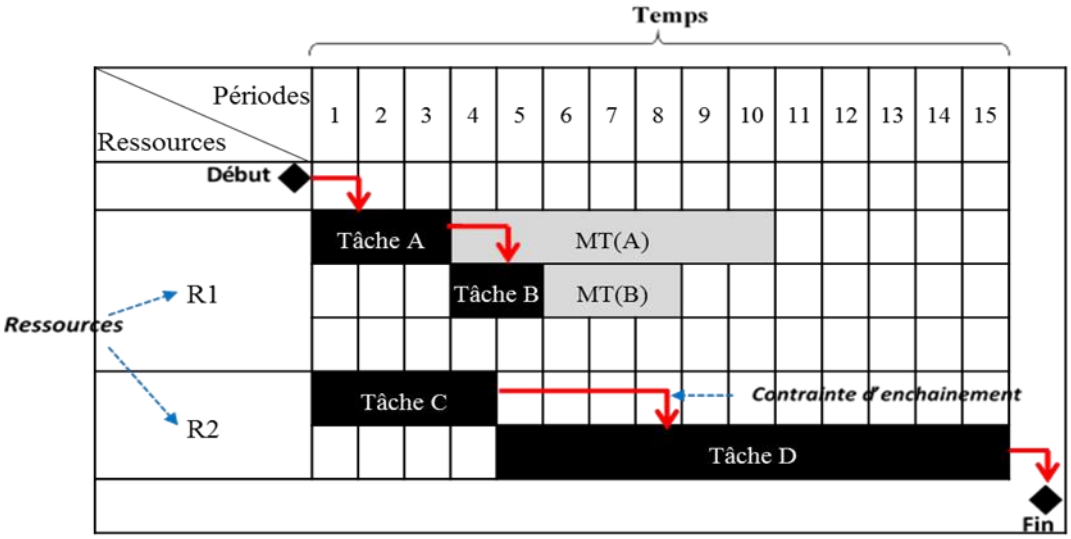


Figure 4.10. Diagramme de Gantt

3.4.1 Prise en compte des contraintes d'enchainement

Souvent, il est possible de planifier un projet de différentes manières, une planification satisfaisante est celle qui permettra de terminer le projet dans le délai minimum. Ainsi, lors de la planification d'un projet, on commence d'abord par les tâches du chemin critique, ensuite, on planifie les tâches qui sont des prédécesseurs des tâches critiques.

Si les ressources disponibles sont suffisantes, la planification prendra en compte uniquement les contraintes d'enchainement. Ainsi, la planification des tâches peut être au plus tôt ou au plus tard.

Exemple

Prenons l'exemple précédent (Tableau 4.1, Figure 4.9). Il est possible de planifier ce projet de différentes manières :

Planification au plus tôt : on planifie les tâches en s'appuyant sur les dates au plus tôt

Deux ressources sont suffisantes pour une planification au plus tôt : Ressource 1 (R1) et Ressource 2 (R2).

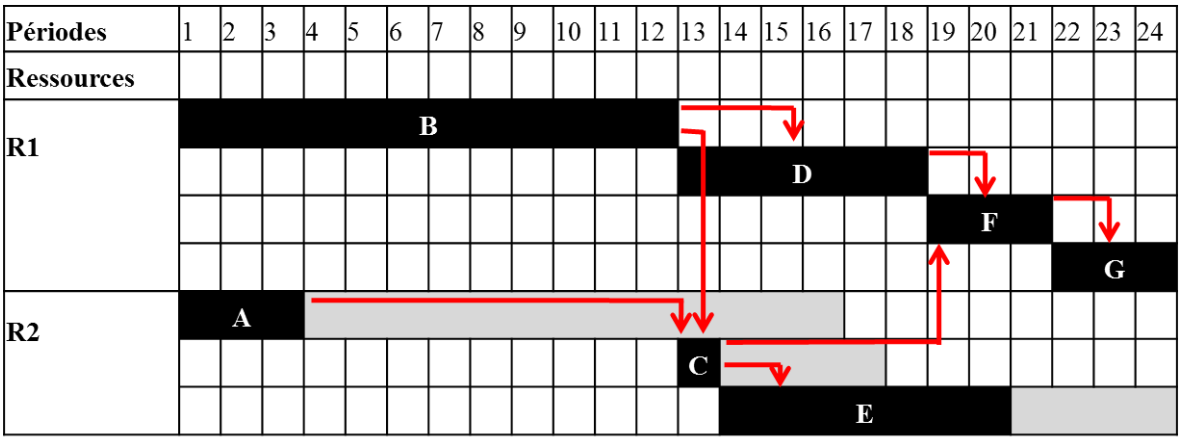


Figure 4.11. Exemple de planification au plus tôt

Problème : R2 doit attendre pendant 9 périodes, car la tâche C ne peut démarrer avant la fin de la tâche B.

Planification au plus tard. On planifie les tâches en s'appuyant sur les dates au plus tard.

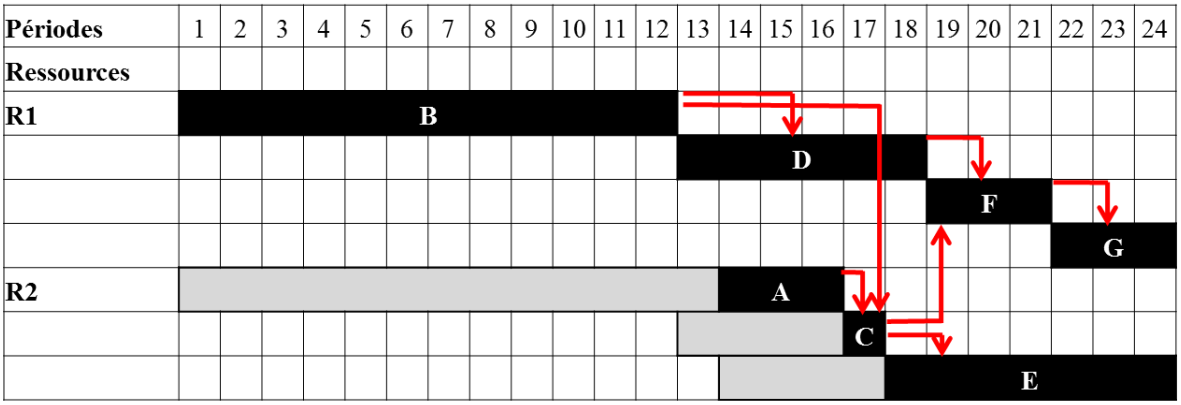


Figure 4.12. Exemple de planification au plus tard

Problème: Aucune tâche ne peut prendre de retard, sinon le projet ne pourra s'achever dans le délai minimum.

Planification améliorée. On place la tâche A de façon à éviter une attente pour la ressource R2.

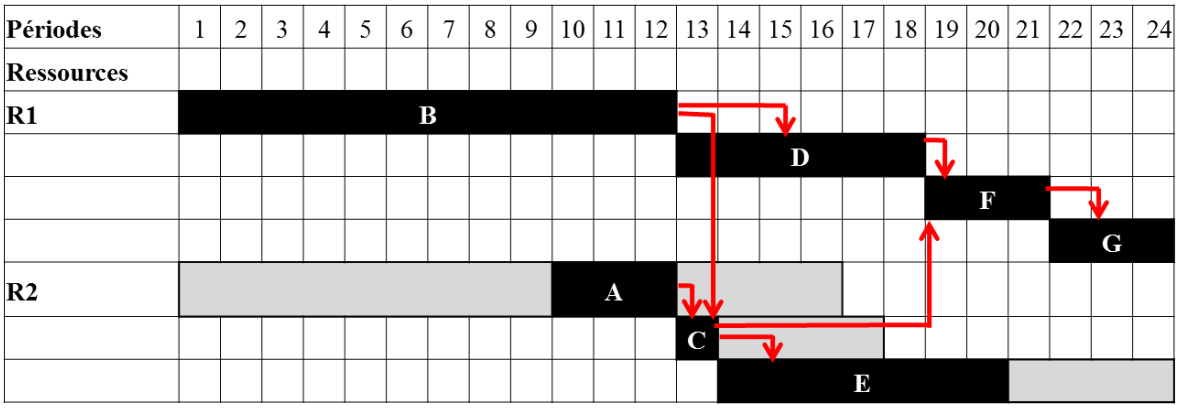


Figure 4.13. Planification améliorée

3.4.2 Prise en compte des contraintes de ressources

Dans les planification précédentes (au plus tôt, au plus tard, et la planification améliorée), l'hypothèse est que toutes les ressources nécessaires sont disponibles pour exécuter les tâches dans les dates prévues, cependant, en réalité les ressources d'un projet ne sont pas illimitées, et sont soumises à des contraintes, particulièrement les ressources spécialisées qui peuvent être affectées simultanément de nombreuses tâche au sein du même projet, ou même à d'autres projets. On peut aussi vouloir répartir de façon relativement homogène les ressources dans le temps.

Cela peut conduire à revoir la planification pour prendre en compte les contraintes de ressources. Deux techniques sont souvent utilisées pour l'optimisation des ressources : Le nivellement et le lissage.

Le nivellement

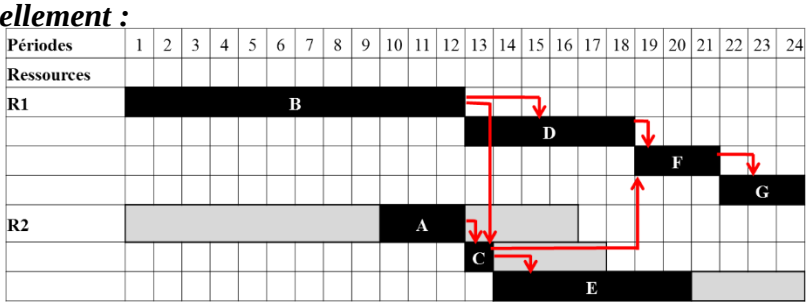
Le nivellement des ressources consiste à limiter le nombre de ressources utilisées simultanément sur un projet, dans le but de prendre en compte les contraintes de ressources, et de maintenir la durée du projet. Si cela n'est pas possible, alors l'objectif est de retarder le projet le minimum possible.

La technique de nivellement peur être appliqué pour différentes raisons :

- La disponibilité des ressources (personnes, matériel, locaux…) peut être telle que l'on doit renoncer à utiliser toutes les possibilités d'exécuter des tâches en parallèle, telles qu'elles figurent sur le graphe des antécédents.
- Le nivellement évite d'avoir une taille d'équipe de projet trop importante par rapport à la durée totale du projet : Une taille d'équipe risquant de générer des surcharges de coordination.
- Le nivellement permet d'étaler dans le temps les dépenses liées au projet.

Exemple. Le passage de deux ressources à une seule :

Avant le nivellement :



Après le nivellement :

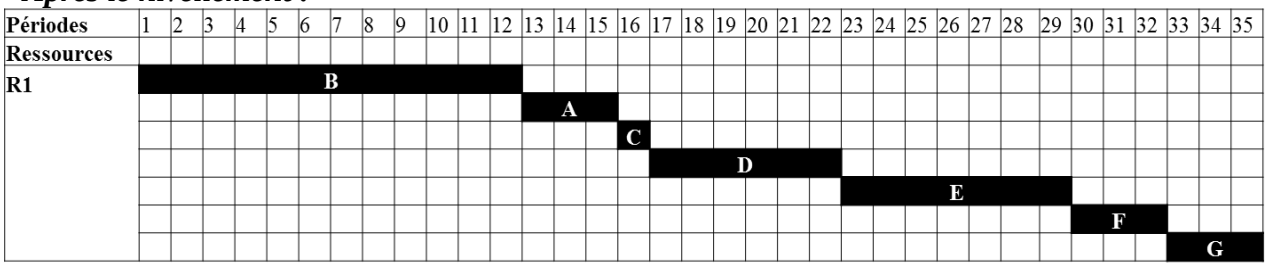


Figure 4.14. Exemple de nivellement

Lissage

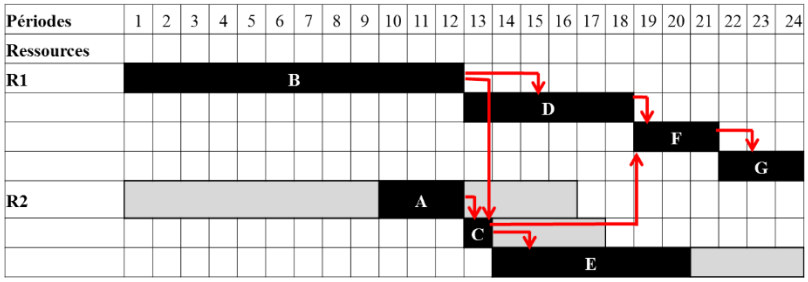
Le lissage consiste à répartir pour chaque ressource sa charge de travail de telle façon qu'elle ne se trouve à aucun moment en surcharge ou en sous-charge. Le lissage exploite les marges pour décaler les tâches, dans l'objectif d'éviter d'allonger la durée du projet, ou du moins retarder le projet le minimum possible.

Les raisons du lissage sont le plus souvent des contraintes liées à l'utilisation des personnes, ou à la disponibilité réduite du matériel.

Exemple

Si la ressource R2 travaille à mi-temps (à 50%) et la ressource R1 à temps complet.

Avant le lissage :



Après le lissage :

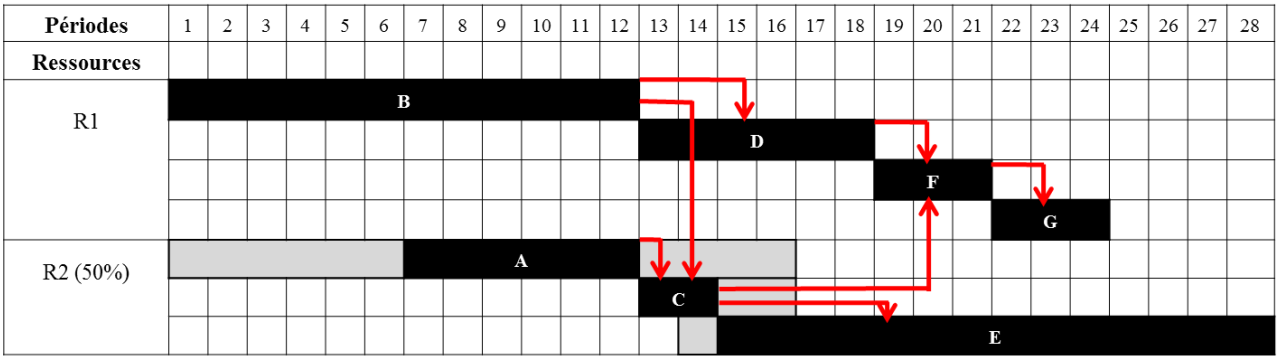


Figure 4.15. Exemple de lissage

Chapitre 5

Le pilotage

1 Généralités

Le pilotage d'un système est l'ensemble de processus qui permettent de maîtriser et de guider son fonctionnement et son évolution. Le principe de base du pilotage est la causalité circulaire ou la rétroaction (Feedback), autrement dit, pour obtenir les résultats désirés d'un système, on exploite les résultats des actions antérieurs pour déterminer les actions ultérieures (Figure…).

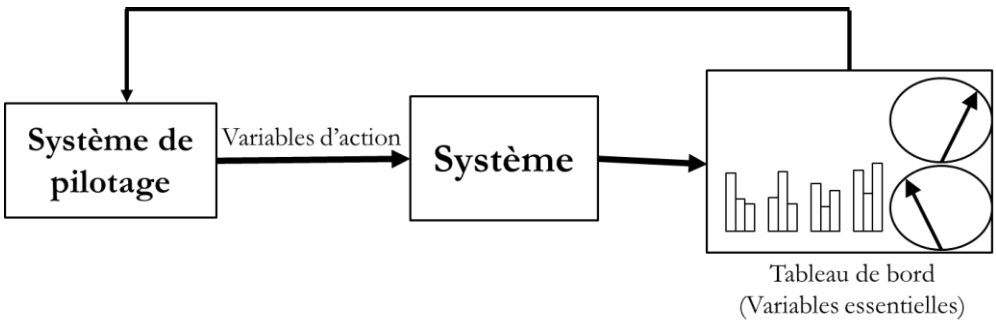


Figure 5.1. Pilotage d'un système

Les deux concepts clés du pilotage sont le **contrôle** et la **régulation**.

Le contrôle comprend :

1. L'Etablissement de variables essentielles (indicateurs du **Tableau de bord**) pour contrôler l'évolution du système, et les plages admissibles pour chaque indicateur.
2. La détermination de variables d'action, c.à.d., les moyens d'actions pouvant faire varier les résultats.

La régulation vise à maintenir le système dans les limites de fonctionnement que le système de contrôle a désignées. Il comprend :

1. Le suivi : examen du tableau de bord,
2. Prise en compte des écarts : utilisation des variables d'action.

2 Pilotage de projets logiciels

Malgré tous les efforts consacrés à l'élaboration d'un planning pour un projet, il y a peu de chances que le projet se déroule précisément selon ce planning, et ceci pour différentes raisons, e.g., besoins mal identifiés, sous-estimation des charges et des durées, changement de l'environnement, etc.

Le pilotage d'un projet consiste aux activités accomplies périodiquement pendant l'exécution du projet, afin de mesurer l'avancement du projet et identifier les écarts entre l'avancement réel et l'avancement planifié (le suivi), et prendre les mesures correctives pour faire disparaitre ou atténuer les écarts (la régulation).

Examinons quelques concept clés de cette définition :

Exécution du projet. Le pilotage du projet se fait pendant la phase d'exécution du projet.

Mesurer l'avancement. L'avancement du projet est déterminé en mesurant périodiquement les indicateurs du tableau de bord (section 3)

Identifier les écarts. Deux aspects sont mis en évidence :

- *Réalisations planifiées* : les réalisations planifiées sont les objectifs fixées et acceptées lors de l'estimation et la planification du projet. Ces objectifs sont fixés par la direction du projet en concertation avec les parties prenantes du projet (le client, la direction de l'organisation et l'équipe du projet).
- *Réalisations réelles* : correspond à l'avancement réel du projet.

La comparaison peut révéler : l'absence d'écart entre le planifié et le réel, un écart positif est lorsque l'avancement réel est meilleur que l'avancement planifié, ou un écart négatif lorsque l'avancement réel est inférieur à l'avancement planifié.

Mesures correctives. Lorsqu'un écart négatif est révélé, il devient nécessaire d'agir pour combler l'écart négatif (utilisation des variables d'action). Les mesures correctives peuvent prendre plusieurs formes, notamment injecter plus de ressources dans le projet afin que l'écart négatif disparaisse.

Périodiquement : Les mesures, les comparaisons et les actions correctives ont lieu régulièrement au cours de l'exécution du projet à des intervalles de temps prédéfinis.

3 Tableau de bord

Le **tableau de bord** est l'outil de suivi et de pilotage du projet, il est constitué de plusieurs indicateurs qui permettent de mesurer l'avancement réel du projet, et les écarts avec l'avancement planifié, puis de déterminer si une intervention corrective est nécessaire.

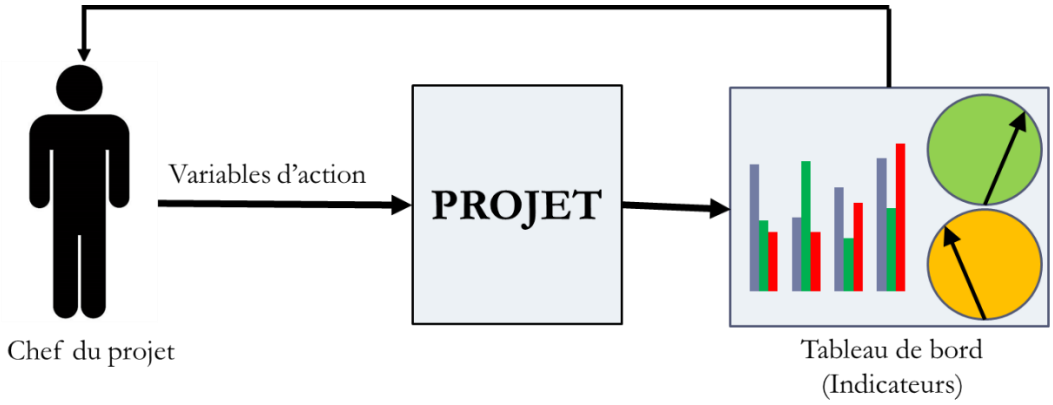


Figure 5.2. Pilotage et tableau de bord

Le tableau de bord doit permettre pour un chef de projet, de pouvoir répondre à n'importe quel moment aux questions sur :

- L'avancement réel aux niveaux du projet et des tâches (Ce qui a été produit)
- Les ressources utilisées (Ce qui a été consommé);
- Les écarts entre le planifié et le réalisé ;
- L'origine des écarts, que ce soit une cause ayant des effets sur plusieurs tâches, par exemple l'indisponibilité d'une machine, ou un problème ponctuel lié à une tâche ou à une personne ;
- Ce qu'il reste à faire.

Le tableau de bord est alimenté par des rapports d'avancement qui sont rédigés sur une base périodique (chaque semaine ou chaque mois) par les intervenants du projet, et permettent :

1. un *suivi individuel* de chaque intervenant et de chaque tâche, et un suivi au niveau du projet ;
2. Le *suivi du projet*, qui sert de base à un point d'avancement périodique avec le maître d'ouvrage.

3.1 Suivi individuel

Le projet est découpé en tâches qui sont affectées aux intervenants qui travaillent sur le projet. Le suivi individuel permet de mesurer l'avancement des tâches et les performances des

intervenants, et de détecter ainsi d'éventuelles difficultés pour un intervenant (ressource) ou sur une tâche.

3.1.1 Comptes rendu d'activité

Les comptes rendu d'activité sont la base pour calculer les indicateurs du tableau de bord. Un compte rendu est rédigé en fin de chaque semaine, par chaque intervenant affecté au projet. Les données du compte rendu doivent être précises, sinon tous les indicateurs calculés du tableau de bord seront faussés. Il comprend, par intervenant et par tâche :

La charge initiale : c'est la charge initialement estimée et qui a servi pour l'élaboration du planning.

La charge affectée : c'est un ajustement de la charge initiale en fonction de l'expérience et de la compétence de l'intervenant chargée de la réalisation de la tâche. Elle peut être supérieure ou inférieure à la charge initiale. Cette valeur représente le contrat entre le chef de projet et l'intervenant concerné.

La charge actualisée : C'est la charge réajustée au cours du projet pour prendre en compte des éléments nouveaux qui peuvent conduire à revoir l'estimation initiale.

L'activité périodique :

- **Le temps passé T:** C'est le temps consommé par l'intervenant sur la tâche pendant la semaine (consommation de ressource humaine).
- **Le reste à faire R**: C'est temps nécessaire pour achever la tâche à la fin de la semaine :
- R=charge affectée ou actualisée – T

Déc. 2022 Semaine 1	**Tâches**	**Charge initiale**	**Charge affectée**	**Charge actualisée**	**Temps passé T**	**Reste à faire R**
Omar	Codage du module TRAITEMENT	24	22	23	3	20
	Maladie				1	
Aicha	Codage du module MISE À JOUR	4	4	4	4	0
Ahmed	Codage du module SORTIES IMPRIMÉES	12	12	11	2	9
	Congé				2	
Idir	Codage du module RECHERCHE	15	15	17	4	13

Tableau 5.1. Exemple de comptes rendu d'activité de 4 intervenants

3.1.2 Récapitulatif individuel mensuel

Le *récapitulatif mensuel* donne un suivi mensuel pour chaque intervenant. Il comprend pour chaque tache et pour chaque intervenant :

- Le *temps passé* ($T(n)$) de chaque semaine *n*;
- Le *reste à faire* à la fin de chaque semaine *n* ($R(n)$) ;
- L'avancement de chaque semaine n : $A(n) = R(n-1) - R(n)$
- Le temps passé, le reste à faire et l'avancement du mois.

	Tâche	**Charge affectée**	**Semaine 1**			**Semaine 2**			**Semaine 3**			**Semaine 4**			**Total du mois**		
			T	R	A	T	R	A	T	R	A	T	R	A	T	R	A
Décembre 2022	Codage du module **Traitement**	23	3	20	3	5	16	4	3	13	3	2	11	2	13	11	12
	Codage du module **Interface**	16							2	14	2	3	11	3	5	11	5
	Total	39													18	22	17

Tableau 5.2. Exemple d'un récapitulatif individuel mensuel

L'attention du chef de projet est attirée quand l'avancement est inférieur au temps passé. Dans le Tableau 5.2, le temps total passé pour la tâche «Codage du module Traitement » T=13 est inférieur à l'avancement R=12.

3.1.3 Bilan individuel mensuel

Le *bilan individuel mensuel* donne pour chaque intervenant une évaluation de sa performance.

La partie gauche du bilan comprend pour un mois *n,* et pour chaque tâche :

1. La charge affectée ;
2. Le reste à faire à la fin du mois précédent *($R(n-1)$) ;*
3. Le temps passé du mois *n: $T(n)$;*
4. Le reste à faire à la fin du mois *n ($R(n)$);*
5. L'avancement du mois n : $A(n) = R(n-1) - R(n)$
6. Le coefficient d'utilisation de la ressource pendant le mois *n:*

$$\frac{T(n)}{Nombre\ de\ jours\ ouvrables\ du\ mois}$$

7. La vitesse du mois: $\frac{A(n)}{T(n)}$

La partie droite donne un récapitulatif depuis le début du projet:

1. Le *temps total passé* : la somme de tous les temps consommés ;
2. Le *coefficient d'utilisation :* calculé à partir de la date de l'arrivée de la ressource sur le projet.
3. La *performance* : mesure le degré d'atteinte des objectifs pour la totalité des tâches en cours ou achevées (n'inclut pas les tâches non encore démarrées):

$$\frac{\textit{Charge affectée} * 100}{\textit{Temps total passé} + \textit{Reste à faire des tâches ouvertes}}$$

Exemple

Ali a commencé à travailler sur le projet à partir du deuxième mois, et il est chargé de réaliser 3 tâches: A, B, C.

Mois 3 (20 jours)	*Tache*	*Charge affectée*	*R(2)*	*T(3)*	*R(3)*	*A(3)*	*Coefficient d'util.*	*Vitesse*	Récapitulatif depuis le début du projet		
									Temps total	Coefficient d'util.	Performance
Ali	Tache A	14	0						15		14/(15+0)= 93%
	Tache B	21	18	14	0	18		18/14=1,29	17		21/(17+0)= 123%
	Tache C	15	15	2	14	1		1/2=0,50	2		15/(2+14)= 94%
	Total	50	33	16	14	19	16/20=0,8	19/16=1,19	34	40/34=0,85	50/(34+14)= 104%

Tableau 5.3. Exemple d'un Bilan individuel mensuel d'un intervenant (Ali)

Le chef du projet que le coefficient d'utilisation de la ressource Ali est relativement satisfaisant (0,85). Sa vitesse durant le mois 3 est satisfaisante (1.19), il a compensé son retard sur la tâche C par une vitesse sur la tâche B. Sa performance globale depuis le début du projet est largement satisfaisante (104%).

3.2 Suivi du projet

Le chef de projet a besoin d'avoir périodiquement une vue de synthèse de l'état du projet, qui permettra au maître d'œuvre, responsable contractuel du projet, de faire le point avec le maître

d'ouvrage. Cette vue de synthèse est représentée dans le *tableau d'avancement du projet*. Ce tableau est alimenté par les récapitulatifs mensuels, regroupe les tâches dans des lots donnant lieu à des livrables, et comprend trois parties :

1. **Rappel des éléments du mois n–1** : pour toutes les tâches/lots: le temps passé ($T(n-1)$) et le reste à faire ($R(n-1)$).
2. **Les éléments du mois *n*** : pour toutes les tâches: Le temps passé ($T(n)$), le reste à faire ($R(n)$) et l'avancement ($A(n)$).

	Mois n-1		Mois n				Récapitulatif depuis le début du projet			
Lots	T	R	T	R	A	Évolution charge restante	Charge initiale	Temps total passé	Évolution globale charge %	% Avancement

Figure 5.3. Structure du tableau d'avancement du projet

Cela permet de calculer la tendance du passé récent entre le mois n-1 et le mois n :

Évolution de la charge restante =T(n) – A(n)

$$= T(n) - (R(n-1) - R(n))$$

$$= (T(n) + R(n)) - R(n-1)$$

Ce paramètre indique si durant le mois la charge restante du projet augmente ou non. Si sa valeur est négative, la charge s'allège, si elle est positive la charge s'alourdit.

3. Les éléments récapitulatifs depuis le début du projet :
   - La charge initiale ;
   - Le *temps total passé* représente le temps de travail affecté au projet depuis le démarrage.
   - Évolution globale de la charge =Temps total passé + R(n) – Charge initiale

Cet indicateur compare la charge du projet avec l'avancement. S'il est positif, on prévoit de dépasser la charge *prévue.*

$$Pourcentage\ d'évolution = \frac{Évolution\ globale\ de\ la\ charge\ *\ 100}{Charge\ initiale}$$

$$Pourcentage\ d'avancement = \frac{Charge\ initiale - R(n)\ *\ 100}{Charge\ initiale}$$

Exemple

	Mois 3		Mois 4				Récapitulatif projet				
Lots	Temps passé	Reste à faire	Temps passé	Reste à faire	Avancement	Evolution de la charge	Charge initial	Temps total passé	Evolution en jours	Evolution en %	Achèvement %
Lot 1	15	0				0	41	53	12	29	100
Lot 2	0	0				0	18,5	17	-1,5	-8	100
Lot 3	14	2,5	3	0	2,5	0,5	37,5	36	-1,5	-4	100
Lot 4	9	13,5	23,5	0	13,5	10	21	31,5	10,5	50	100
Lot 5	0	10	8,5	0	10	-1,5	10	8,5	-1,5	-15	100
Total	38	26	35	0	26	9	128	146	18	14	100

Tableau 5.4. Exemple d'un tableau d'avancement d'un projet

4 La méthode de la valeur acquise (Earned Value Analysis : EVA)

Nous avons présenté un certain nombre d'indicateurs pour mesurer l'avancement d'un projet, cependant, ces indicateurs sont basés sur les évaluations des intervenants (comptes rendus), qui peuvent être subjectives et imprécises, en plus, évaluer l'avancement d'un grand nombre de tâches, qui sont à divers stades d'achèvement est fastidieux, à cet effet, il existe une technique alternative moins subjective, et plus simple à appliquer, qui s'appelle l'analyse de la valeur acquise (Earned Value Analysis : EVA).

L'EVA est une technique quantitative qui permet d'avoir une évaluation objective de la performance des coûts et du calendrier d'un projet. L'EVA est particulièrement utile dans les grands projets de longues durées, et fournit des mesures précises et fiables dès 15% de l'avancement du projet[21].

[21] Roger, S. P., & Bruce, R. M. (2015). *Software engineering: a practitioner's approach*. McGraw-Hill Education.

4.1 Métriques de base

L'EVA utilise trois valeurs primaire pour chaque tâche :

4.1.1 Valeur planifiée :VP (Planned Value : PV)[22].

C'est le coût planifié pour une tâche (ou projet) entre la date de début de la tâche/projet et une date *t*, autrement dit, c'est ce qu'on a planifié de consommer jusqu'à la date ***t***. La VP à la fin planifiée du projet ou d'une tâche correspond au budget du projet, et s'appelle le Budget à l'achèvement (Budget at completion : BAC).

La valeur planifié d'un projet à un moment donné, est la somme des valeurs planifiées pour toutes les tâches du projet qui auraient dû être commencées ou terminées à ce moment dans le calendrier du projet.

4.1.2 Coût réel :CR (Actual cost :AC)[23]

C'est le coût total effectivement dépensé sur une tâche (ou un projet) dépensé jusqu'à la date *t*.

4.1.3 Valeur acquise :VA (Earned value : EV)[24].

Basé sur l'avancement réel du projet et représente ce que l'on aurait dû dépenser pour le travail effectivement réalisé jusqu'à la date *t*.

4.2 Méthodes de mesure

Afin de déterminer objectivement la valeur acquise, le chef du projet peut utiliser diverses méthodes de mesure.

4.2.1 Formule fixe (Fixed Formula)

La méthode de la formule fixe pour déterminer l'avancement s'applique aux lots et aux tâches de courte durée (< 2 mois). Cette méthode applique un pourcentage d'achèvement au début et à la fin d'une tâche, pour mesurer la valeur acquise à partir du budget à l'achèvement (BAC). Généralement, les pourcentages utilisés dans la formule sont 0/100, 50/50 ou 25/75.

- **La technique 0/100.** La valeur acquise d'une tâche restera à 0 même si elle démarre, à la fin de la tâche, 100% du budget de la tâche (BAC) est attribué à sa valeur acquise. La

[22] Egalement appelé: Coût Budgété du Travail Prévu :CBTP (Budgeted Cost of Work Scheduled : BCWS)
[23] Egalement appelé: Coût Réel du Travail Effectué :CRTE (Actual Cost of Work Performed :ACWP)
[24] Egalement appelé: Coût Budgété du Travail Effectué (CBTE) (Budgeted Cost of Work Performed : BCWP)

technique 0/100 est adaptée pour les tâches de courte durée (moins d'un mois) comme l'achat du matériel ou de brèves réunions ou déplacements.

- **La technique 50/50.** La moitié du budget est attribué à la valeur acquise lorsque la tâche démarre, l'autre moitié lorsqu'elle est terminée. La technique 0/50 est adaptée pour les tâches relativement court (moins de 2 mois).
- **La technique 25/75%.** 25% du budget est attribué à la valeur acquise lorsque la tâche démarre, et le 75% restant est attribué lorsqu'elle est terminée. La technique 25/75 accorde plus de poids à la fin du travail qu'à son démarrage, mais elles motivent également l'équipe de projet à signaler les tâches démarrées, ce qui peut améliorer la connaissance des travaux en cours.

4.2.2 Pondération des jalons (Milestone Weighting)

La méthode de pondération des jalons attribue une valeur budgétaire à chaque jalon. Ce n'est qu'à la fin de chaque jalon que le budget est attribué à la valeur acquise. La pondération des jalons est utilisée comme méthode pour les lots de travaux à long terme et devrait idéalement avoir des jalons périodiquement (chaque mois).

4.2.3 Le pourcentage achevé

La méthode du pourcentage achevé s'appuie sur l'évaluation du pourcentage achevé de la tâche, pour déterminer sa valeur acquise à partir de son budget à l'achèvement (BAC). Cette méthode est souvent considérée subjective et à éviter, car l'évaluation du pourcentage achevé est réalisée par les intervenants qui donnent souvent des évaluations subjectives (exagérées). On a tendance à franchir la barre des 90 % assez rapidement, puis y demeurer jusqu'à la fin du lot de travaux ou de l'activité.

Exemple

Considérons les tâches d'un projet et leurs budgets à l'achèvement (BAC), ainsi que les valeurs planifiées et les coût réels à la fin de la période 12 (Tableau 5.5). Le planning du projet est représenté à la Figure 5.4.

Tâche	Budget à l'achèvement (BAC)	Valeur planifiée (VP)	Coût réel(CR)
A	15 000 DA	15 000 DA	8 000 DA
B	60 000 DA	60 000 DA	62 000 DA
C	5000 DA	-	-
D	30 000 DA	-	-
E	40 000 DA	-	-
F	15 000 DA	-	-
G	15 000 DA	-	-
Projet	180 000	75000 DA	70 000 DA

Tableau 5.5. Liste des tâches et des couts associés à la fin de la période 12

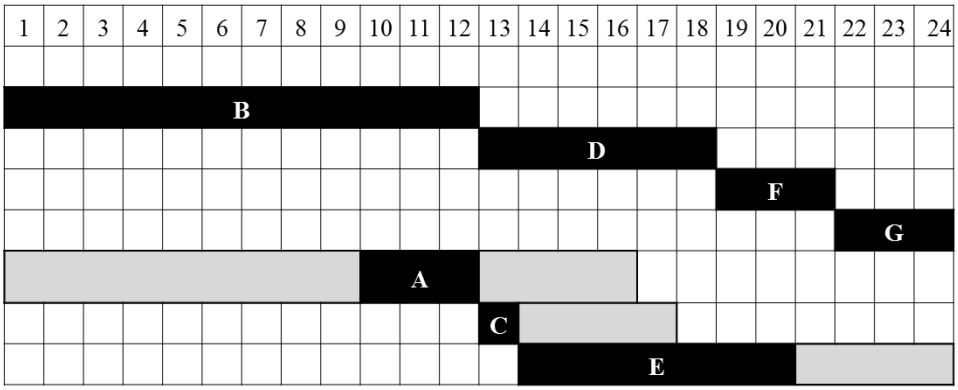


Figure 5.4. Planning du projet

La valeur planifiée du projet à la fin de la période 12 égal à la somme des valeurs planifiées des tâches à cette date, à savoir les tâches A et B. Ainsi :

Calcul de la valeur planifiée
VP= 15000+60000=75000 DA.

Calcul du coût réel
CR=8 000+62 000=70 000 DA

Le projet a consommé effectivement 70000 DA

Calcul de la valeur acquise

On suppose qu'à la fin de la période 12, la tâche B est achevée, la tâche A est démarrée mais pas encore achevé.

- Si on applique la technique 0/100 pour mesurer la valeur acquise :
  VA=15000*0+60000*1=60000 DA.
- Si on applique la technique 50/50 pour mesurer la valeur acquise :
  VA=15000*0.5+60000*1=67500 DA.
- Si on applique la technique 25/75 pour mesurer la valeur acquise:
  VA=15000*0.25+60000*1=63750 DA.

4.3 Indicateurs complémentaires

D'autres indicateur clés sont déterminées à partir de ces trois valeurs de base. Les plus courants et les plus utiles sont l'écart du coût, l'écart de délai, l'indice de performance du coût et l'indice de performance du délai.

4.3.1 Écart de coût :EC[25]

L'écart de coût (CV) et l'indice de performance des coûts (IPC) sont deux indicateurs qui fournissent une mesure de la performance du projet par rapport aux coûts.

L'écart de de coût (EC) est la différence entre la valeur acquise et le coût réel :

$$\boldsymbol{EC = VA - CR}$$

- Si l'EC est négatif, ça signifie qu'on dépense plus que prévu;
- Si l'EC est positif, cela indique qu'on consomme moins que prévu
- Si l'EC est nul, les dépenses réelles correspondent aux dépenses planifiée.

On peut exprimer l'écart de délai par l'i**ndice de performance du coût :IPC[26].** L'IPC est le rapport entre la valeur acquise et le coût réel :

$$\boldsymbol{IPC = \frac{VA}{CR}}$$

[25] En anglais Cost Variance : CV
[26] En anglais Cost performance index :SPI

L'IPD exprime le taux de couverture des dépenses réelles (CR) par les dépenses prévus (VA) :

- Si l'IPC est inférieur à 1, cela indique que le coût de réalisation est plus élevé que prévu ;
- Si l'IPC est égal à 1, on considère que le coût de réalisation est conforme à la planification;
- Si l'IPC est supérieur à 1, cela signifie que le coût de réalisation est inférieur à celui prévu.

4.3.2 Écart de délai : ED[27]

L'écart de délai (CD) et l'indice de performance est un indicateur qui fournissent une mesure de la performance du projet par rapport aux délais.

L'écart de de coût (ED) est la différence entre la valeur acquise et le coût planifié :

$$\boldsymbol{ED = VA - CP}$$

- Si l'ED est négatif, on avance moins vite que ce que l'on avait planifié.
- Si l'ED est positif, on avance plus que prévu,
- Si l'ED est nul, l'avancement est conforme à la planification.

On peut exprimer l'écart de délai par Indice de performance du délai : IPD[28]. L'IPD est le rapport entre la valeur acquise et la valeur planifiée :

$$\boldsymbol{IPD = \frac{VA}{VP}}$$

L'IPD exprime le taux d'avancement ou du retard du projet par rapport planning :

- Si l'IPD est inférieur à 1, cela indique qu'on est en retard ;
- Si l'IPD est égal à 1, cela indique que l'avancement est conforme à la planification ;
- Si l'IPD est supérieur à 1, cela signifie que l'avancement réel est supérieur à celui prévu.

Exemple

Continuons avec l'exemple précèdent :

$$\boldsymbol{EC = VA - CR = 67500 - 70000 = -2500}$$

$$\mathbf{IPC} = \boldsymbol{\frac{VA}{CR}} = \boldsymbol{\frac{67500}{70000}} = \mathbf{0.96}$$

27 En anglais Schedule variance :SV
28 En anglais Schedule performance index :CPI

Les indicateurs du coût (EC<0 et IPC<1) indiquent que les dépenses sont légèrement supérieures aux prévisions, en fait, les coûts prévus couvrent 96% des dépenses réels.

$$\boldsymbol{ED = VA - VP = 67500 - 75000 = -7500}$$

$$\textbf{IPD } = \frac{\boldsymbol{VA}}{\boldsymbol{VP}} = \frac{\boldsymbol{67500}}{\boldsymbol{75000}} = \boldsymbol{0.9}$$

Les indicateurs du calendrier (ED<0 et IPD<1) indiquent qu'on est en retard par rapport à la planification. L'IPD est de 0,90, nous avons donc réalisé 90% des tâches planifiés.

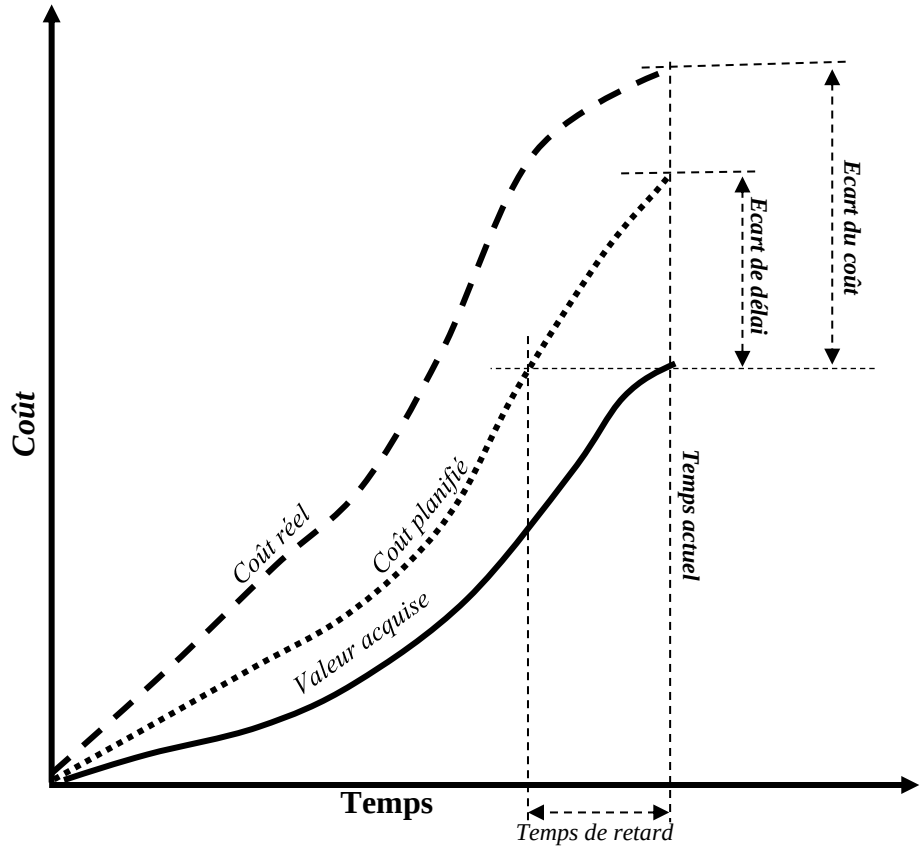


Figure 5.5. Métriques et indicateurs de la méthode de la valeur acquise

Chapitre 6 .

Initiation à Microsoft Project

1 Introduction

Microsoft Project est outil logiciel de gestion de projet, développé par Microsoft . Il est conçu pour aider le chef de projet à accomplir de nombreuses activités, particulièrement :

- Elaboration du planning du projet: MS Project permet la création des tâches d'un projet, la définition de leurs durées, ainsi que les contraintes d'enchainement entre les tâches. Puis, MS Project se charge de calculer les dates de début et de fin des tâches, ainsi la date début et de fin du projet.
- Gestion des ressources : MS Project permet au chef de projet de définir les ressources du projet avec leurs types, coût, etc., de les affecter aux tâches, de détecter et de résoudre les surutilisations des ressources, etc.
- Suivi du projet : Pendant le suivi du projet, Microsoft Project permet au chef du projet de simplifier le processus de mise à jour de la planification initiale pour prendre en compte les données réelles, et de visualiser les estimations de départ et les données réelles et de calculer les écarts entre les deux.

2 Interface de Microsoft Project

Lorsque vous ouvrez un nouveau projet, la fenetre qui s'affiche comporte principalement, la barre d'outils d'accès rapide, l'onglet fichier, le ruban, le volet de la liste des tâches et le volet du diagramme de Gantt(Figure 6.1).

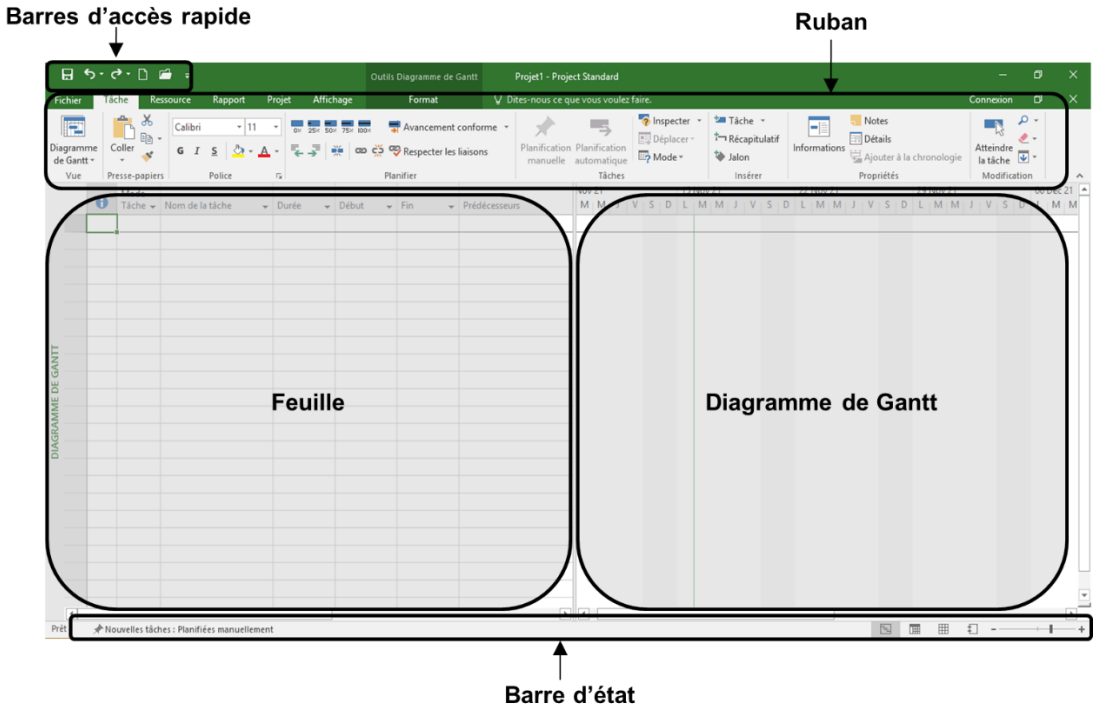


Figure 6.1. Interface de Microsoft Project

2.1 La barre d'accès rapide

La barre d'accès rapide, au-dessus et à gauche du ruban, permet d'accéder rapidement à un certain nombre de fonctions (nouveau, enregistrer, etc.).

2.2 Le ruban

Le ruban permet d'accéder facilement aux outils et commandes les plus couramment utilisés. Le ruban est composé de plusieurs **onglets**. Chaque onglet contient plusieurs **groupes** qui réunissent des **commandes.** Lorsque vous changez d'onglet, les comandes disponibles sur le ruban changent (Figure 6.2).

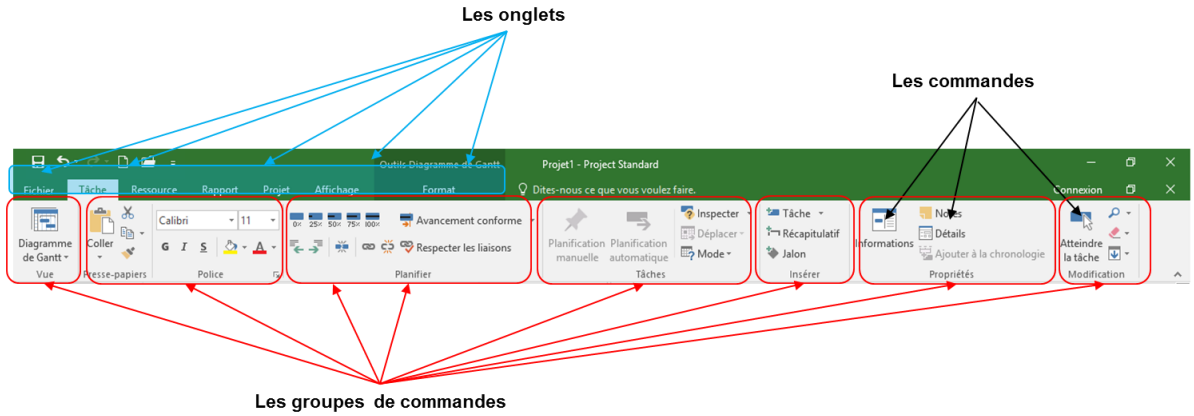


Figure 6.2. Le ruban Microsoft Project

2.3 L'onglet Fichier

L'onglet Fichier donne accès aux commandes sur les fichiers (l'ouverture, l'enregistrement, l'impression, etc.). Enfin, il permet d'accéder aux Options de Project.

2.4 Feuille

Semblable à une feuille de calcul, la feuille sert à saisir et afficher les données du projet selon la vue activée (Liste des tâches, tableau de ressources, etc.).

2.5 Graphique

C'est une représentation graphique de données de la feuille (Diagramme de Gantt des tâches, histogramme des ressources, etc.).

2.6 Barre d'état

La barre d'état, en bas de la fenêtre Project, contient principalement des commandes pour basculer entre les différentes vues et régler le niveau de zoom.

3 Création et paramétrage d'un nouveau projet

Cette section est un guide pour la création d'un nouveau projet, et la définition de sa date de début, ainsi que le paramétrage du calendrier du projet.

3.1 Création d'un projet

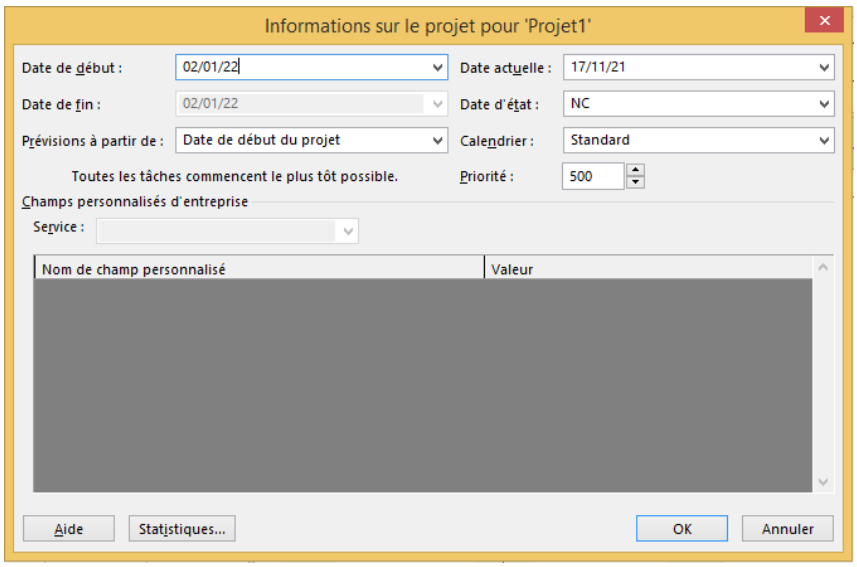
Pour créer un nouveau projet :

1. Onglet Fichier →Nouveau.
2. Dans la liste des templates, cliquez sur "Nouveau Projet" ou sur tout autre template approprié pour concevoir le nouveau projet.

3.2 Définir la date de début de projet

Pour définir la date de début du projet :

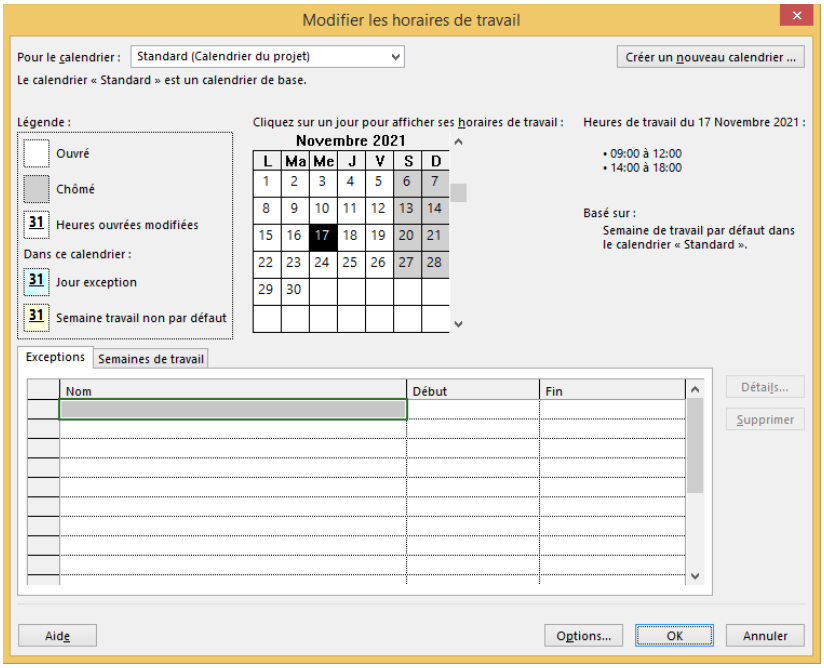
1. sélectionnez la *commande* « **Information sur le projet** » situé dans le *groupe* « **Propriété** » sous **l'onglet** « **Projet** », une boite de dialogue s'affiche et vous permet de spécifier diverses informations sur votre projet.



2. Dans la zone **Prévisions à partir de,** sélectionnez une option pour indiquer le type de **planification** du projet. En sélectionnant une date de début, vous allez travailler en mode **«Le plus tôt possible»** (ASAP :As Soon As Possible). En sélectionnant la date de fin, vous travaillerez en mode **«Le plus tard possible** (ALAP : As Late As Possible).
3. Saisissez la date de début dans le cas du mode ASAP, ou la date de fin dans le cas du mode ALAP.

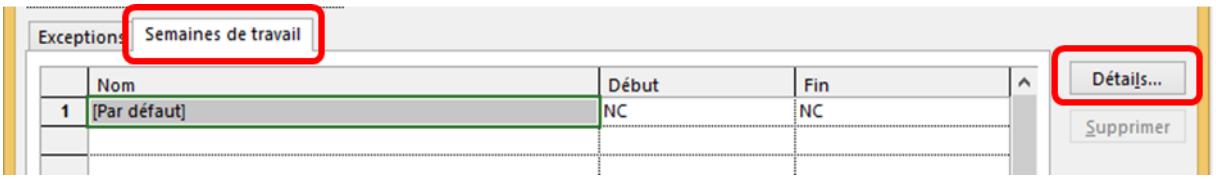
3.3 Paramètres du calendrier

Le calendrier détermine les périodes ouvrées des ressources et des tâches. Il est possible d'introduire des changements au calendrier standard du projet tels que les heures ouvrées, les congés, les jours fériés, etc. Pour paramétrer le calendrier, sélectionnez l'onglet « **Projet** », puis dans le groupe « **Propriétés** », sélectionner la commande « **Modifier le temps de travail** ». Une boite de dialogue s'affiche.



3.4 Semaine de travail

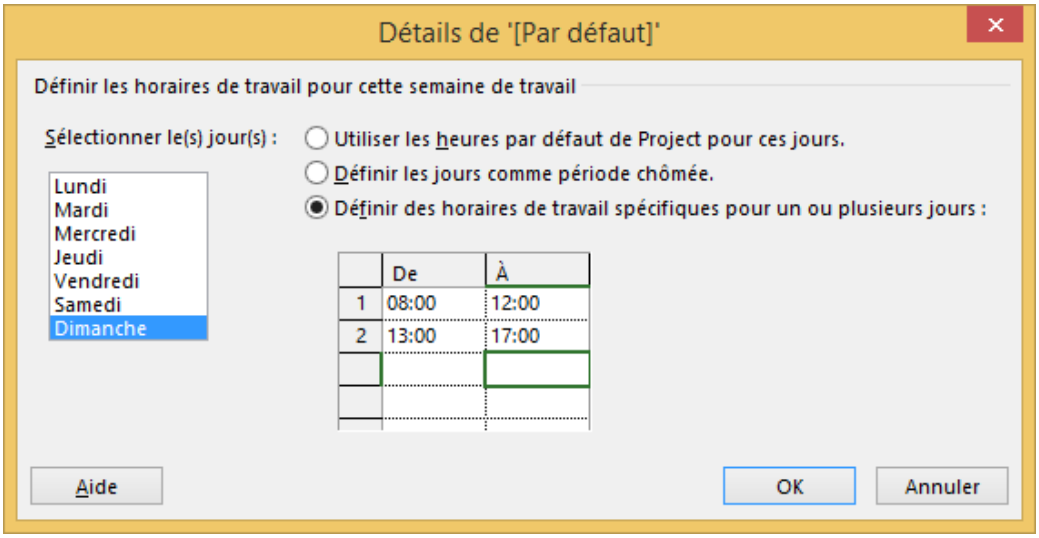
Pour définir les jours ouvrés et chomés de la semaine, aisi que les horaires du travail, sélectionnez l'onglet « **semaine de travail** » dans la boite de dialogue, puis, cliquez sur le bouton "**détails**" .



Dans la boite de dialogue « **Détail de** » qui s'affiche, la semaine, puis :

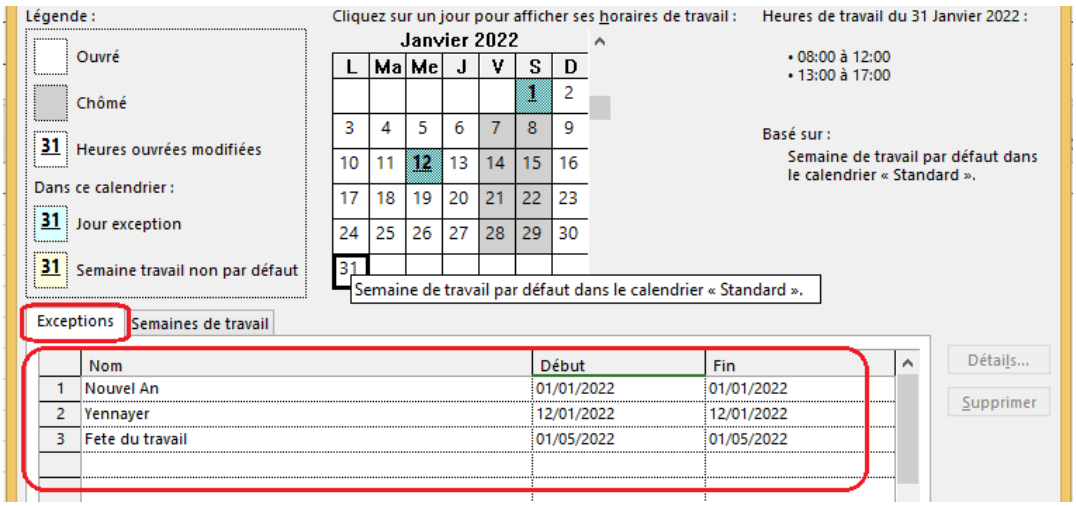
- Définissez les jours chomés en selectionnant le jour à gauche et « Définir le jou comme période chomée» à droite .

- Définissez les horaires de chaque jour en selectionnant le jour à gauche et « Définir les jours comme temps de travail spécifique », puis en introduisant les horaires du travail dans le tableau à droite.



3.5 Les jours fériés et les jours d'exception

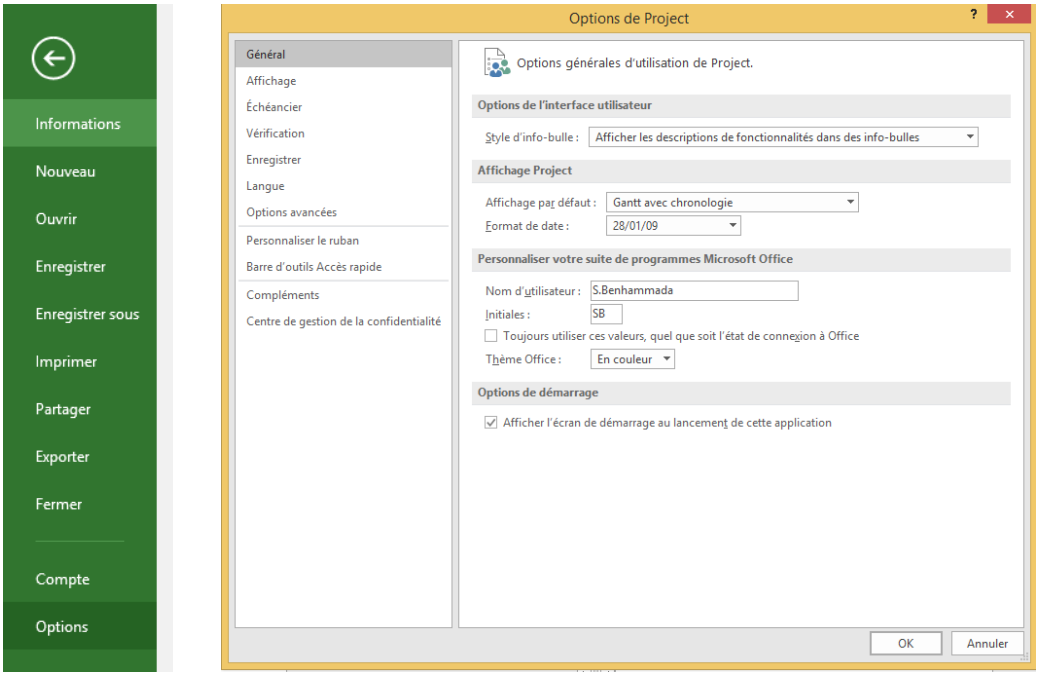
Pour déclarer les jours fériés sélectionnez l'onglet « Exceptions », saisir le congé dans la colonne nom, puis, sélectionnez la date de début et de fin.



1. Pour vérifier la modification du calendrier du projet, remarquez dans le graphique que les jours d'exception sont désormais colorés en gris pour indiquer les périodes chômées, tout comme les week-ends.

4 Personnalisation des options du projet

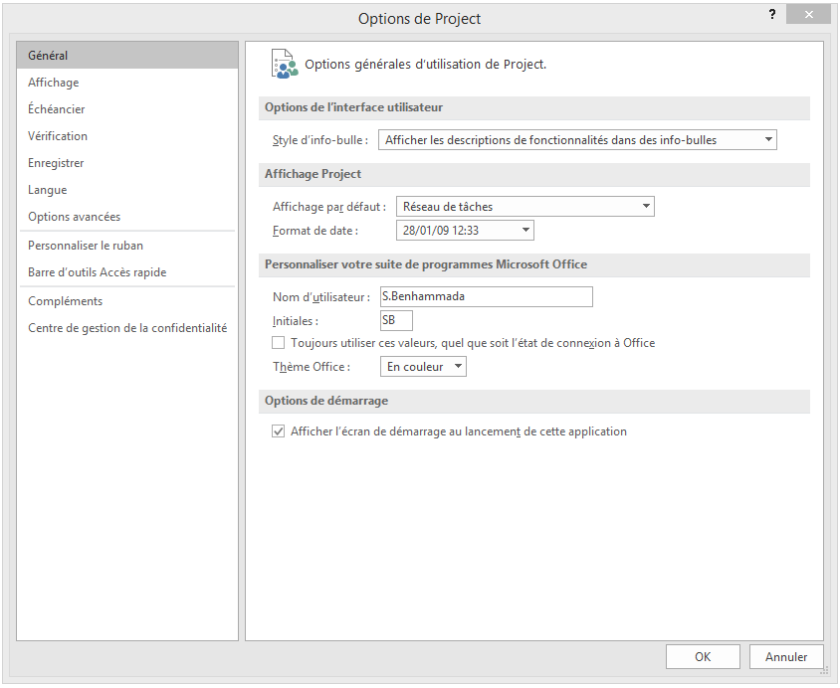
Il est préférable de définir les options du projet au démarrage de celui-ci. Pour personnaliser les options du projet, sélectionnez le menu « **Option »** dans l'onglet « **Fichier »** du ruban, puis, cliquez sur la rubrique dans le volet de gauche pour en afficher le contenu.



4.1 Rubrique « Général »

Dans la rubrique « **Affichage** » vous pouvez définir principalement:

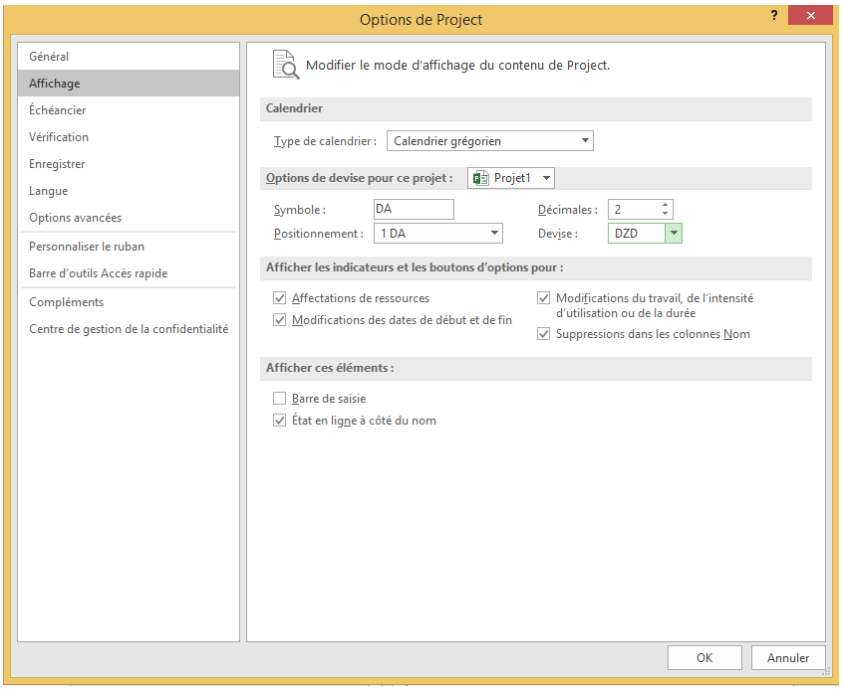
- **L'affichage du projet:** Diagramme de Gantt, Réseau de tâches, etc.
- **Le format de la date :** Il est recommandé de choisir un format de date incluant l'heure, pour éviter de chercher trop longtemps des erreurs dont elle pourrait être la cause.



4.2 Rubrique « Affichage »

Dans la rubrique « **Affichage** » vous pouvez définir principalement:

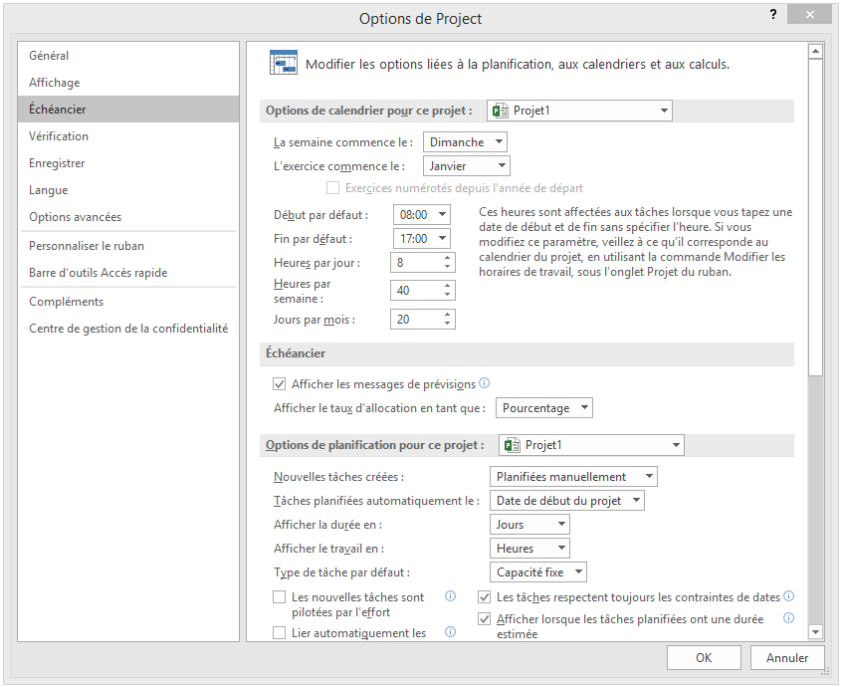
- Le type du calendrier: Grégorien, Hégire, etc.
- La devise pour gérer les coûts du projet : Dinar (DZD), Euro, etc.



4.3 Rubrique « Echéancier »

Dans la rubrique de « **Echéancier** », vous pouvez définir:

- Le 1er jour de la semaine,
- Le 1er mois de l'année,
- Les horaires de travail sur la journée et le nombre d'heures par jour pour les tâches (ces paramètres doivent correspondre au calendrier du projet).
- Le mode de planification de tâches « Nouvelles tâches créées ». Il est recommandé de choisir le mode « Planifiées automatiquement ».
- Le nombre d'heures par semaine, et le nombre de jours par mois.
- L'affichages de la durée des tâches et de la durée de travail des ressources (Mois, Jours, heurs, minutes, etc.) ;
- etc.

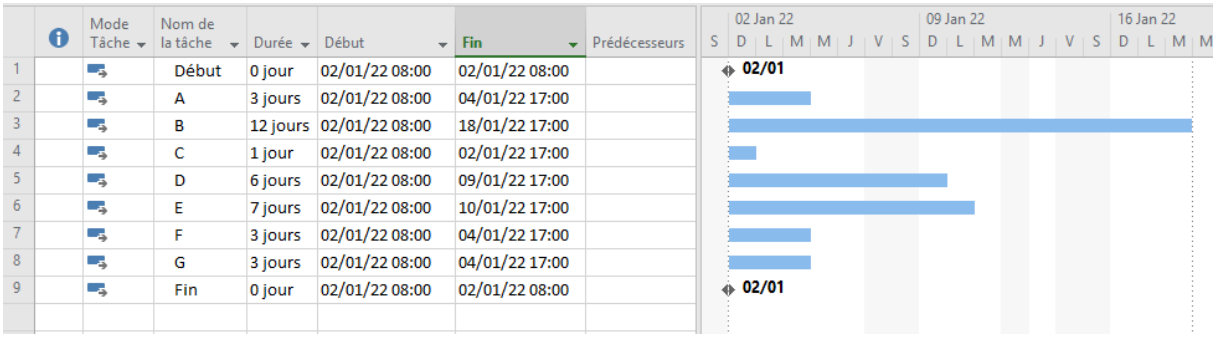


5 Gestion des tâches

5.1 Création, suppression, et insertion des tâches

Pour créer les tâches d'un projet:

1. On saisit la liste des tâches et leurs durées.
2. Les tâches de type jalon (Début et Fin) avec une durée (0).



	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Début	0 jour	02/01/22 08:00	02/01/22 08:00	
2		A	3 jours	02/01/22 08:00	04/01/22 17:00	
3		B	12 jours	02/01/22 08:00	18/01/22 17:00	
4		C	1 jour	02/01/22 08:00	02/01/22 17:00	
5		D	6 jours	02/01/22 08:00	09/01/22 17:00	
6		E	7 jours	02/01/22 08:00	10/01/22 17:00	
7		F	3 jours	02/01/22 08:00	04/01/22 17:00	
8		G	3 jours	02/01/22 08:00	04/01/22 17:00	
9		Fin	0 jour	02/01/22 08:00	02/01/22 08:00	

Pour supprimer une tâche :

- Sélectionnez la ligne de la tâche à supprimer;
- Appuyez sur la touche **Suppr** du clavier, ou clic droit, commande « Supprimer la tâche » du menu contextuel ;

Pour insérer une nouvelle tâche :

- Sélectionnez la ligne de la tâche qui doit suivre celle qui sera insérée ;
- Sélectionnez la commande Insérer une tâche du menu contextuel. Une ligne vide s'ajoute à la liste au-dessus de la ligne préalablement sélectionnée ;
- Saisissez les informations relatives à la nouvelle tâche ;

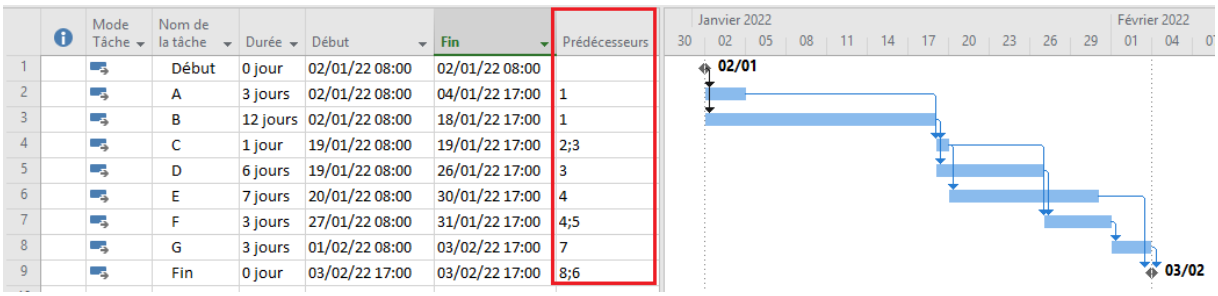
5.2 Contraintes d'ordonnancement entre les tâches

Pour ajouter les contraintes d'enchainement entre les tâches :

- Graphiquement dans le diagramme de Gantt.

ou bien

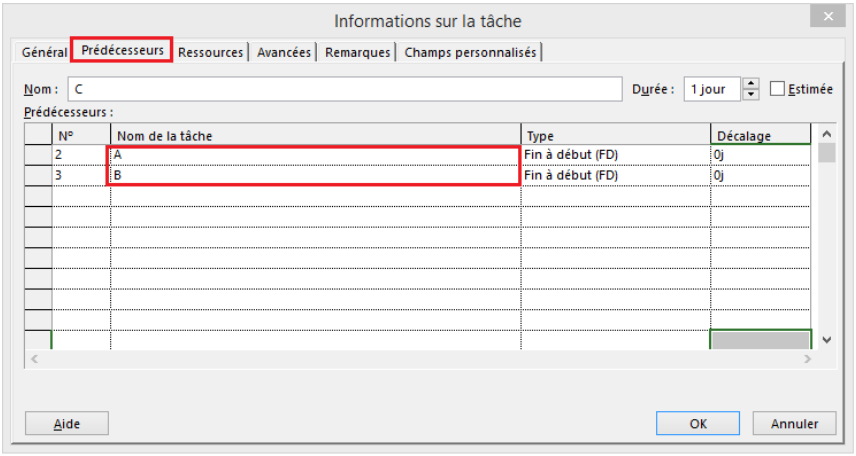
- En indiquant les numéro des tâches antécédentes dans la colonne "prédécesseur", (les numéros doivent être séparés par des points-virgules).



	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Début	0 jour	02/01/22 08:00	02/01/22 08:00	
2		A	3 jours	02/01/22 08:00	04/01/22 17:00	1
3		B	12 jours	02/01/22 08:00	18/01/22 17:00	1
4		C	1 jour	19/01/22 08:00	19/01/22 17:00	2;3
5		D	6 jours	19/01/22 08:00	26/01/22 17:00	3
6		E	7 jours	20/01/22 08:00	30/01/22 17:00	4
7		F	3 jours	27/01/22 08:00	31/01/22 17:00	4;5
8		G	3 jours	01/02/22 08:00	03/02/22 17:00	7
9		Fin	0 jour	03/02/22 17:00	03/02/22 17:00	8;6

ou bien

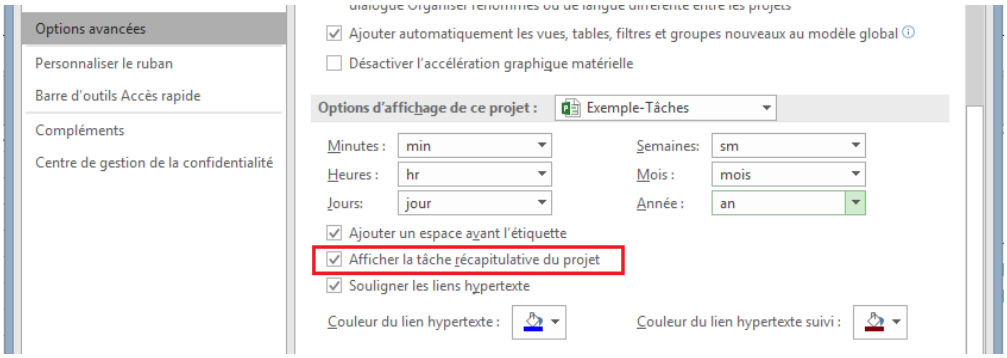
- Double-cliquer sur la tâche pour afficher la boite dialogue « Informations sur la tâche »
- Insérer les contraintes d'enchainement sous l'onglet « Prédécesseurs».



5.3 Tâche récapitulative

Il est souhaitable d'ajouter la tâche récapitulative du projet afin d'afficher la durée, la date de début et la date de fin du projet:

Onglet « Fichier »→ « Options » →Rubrique « Options avancées »→ Options d'affichage de ce projet → Cocher « Afficher la tâche récapitulative de ce projet ».



Ou bien :

Onglet « Format »→ Groupe « Afficher/Masquer » → Cocher « Afficher la tâche récapitulative de ce projet ».

Exemple

Planification d'un projet à l'aide de Microsoft Project

Tâche	Durée (Jours)	Prédécesseurs
A	3	-
B	12	-
C	1	A,B
D	6	B
E	7	C
F	3	C,D
G	3	F

Tableau 6.1. Tâche du projet

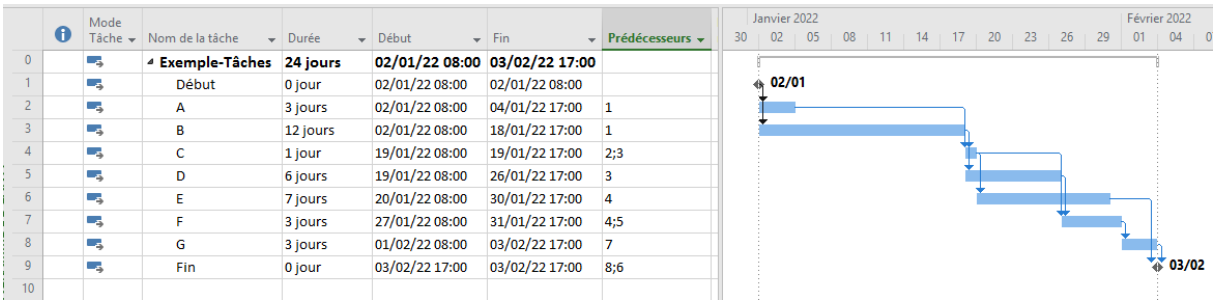
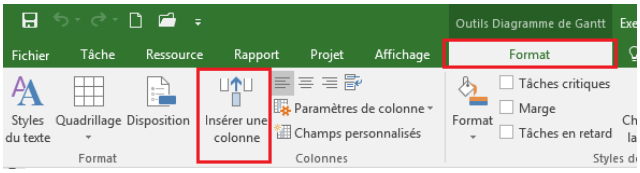


Figure 6.3. Planification du projet dans Ms Project

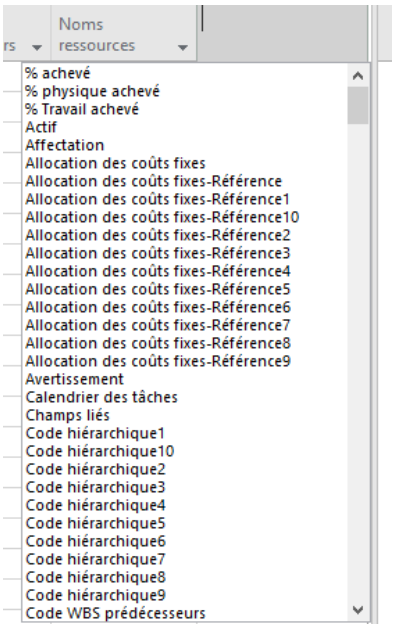
5.4 Insertion de nouvelles colonnes

Il est possible d'insérer de nouvelles colonne dans la liste dans la feuille des tâches

1. Onglet « Format »⟶ «Groupe « Colonnes »⟶ « Insérer une colonne »

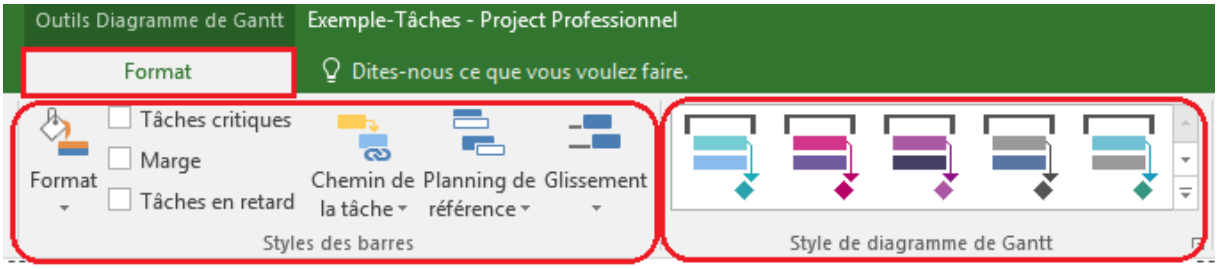


2. Une liste s'affiche, et vous permet d'insérer de nouvelle colonnes, comme par exemple: Le début au plus tard, La fin au plus tard, La marge libre, La marge totale, Le coût, etc.



5.5 Personnalisation du diagramme de Gantt

Il est possible de personnaliser le diagramme de Gantt à l'aide des groupes de commandes « Style des barres » et « Style du diagramme de Gantt » dans l'onglet « Format ».



5.5.1 Affichage du chemin critique

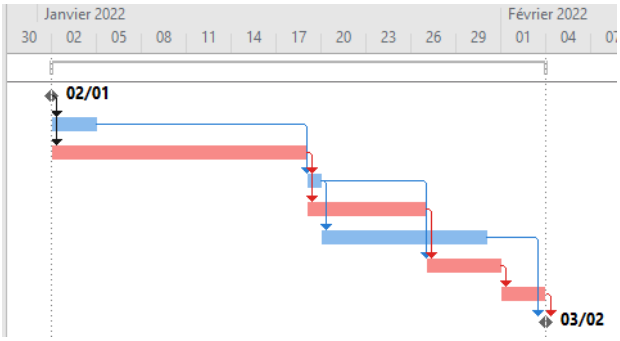
Pour de nombreux projets, le facteur de la durée est essentiel: la date limite du projet doit être respectée. Si un projet est limité par des contraintes de dates, l'examen du chemin des tâches critiques du projet est indispensable.

Pour afficher les tâches du chemin critique :

Onglet « Format » → Groupe « Style de barres » →Cocher Tâches critiques

Exemple:

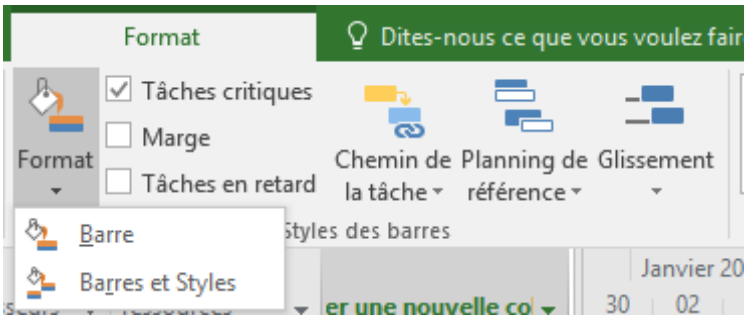
Les tâches du chemin critique apparaissent en rouge sur le diagramme de Gantt.



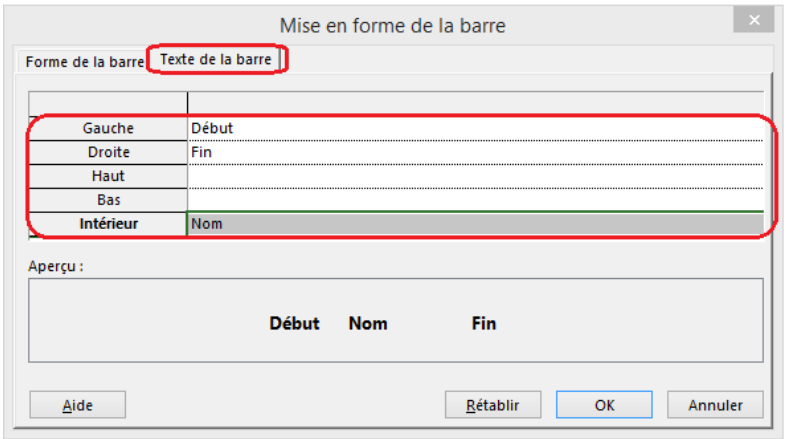
5.5.2 **Affichage d'informations sur le diagramme de Gantt**

Il est possible d'afficher des informations sur les tâche dans le diagramme de Gantt (nom de la tâche, sa durée, la date de début, la date de la fin, etc.). Pour ce faire

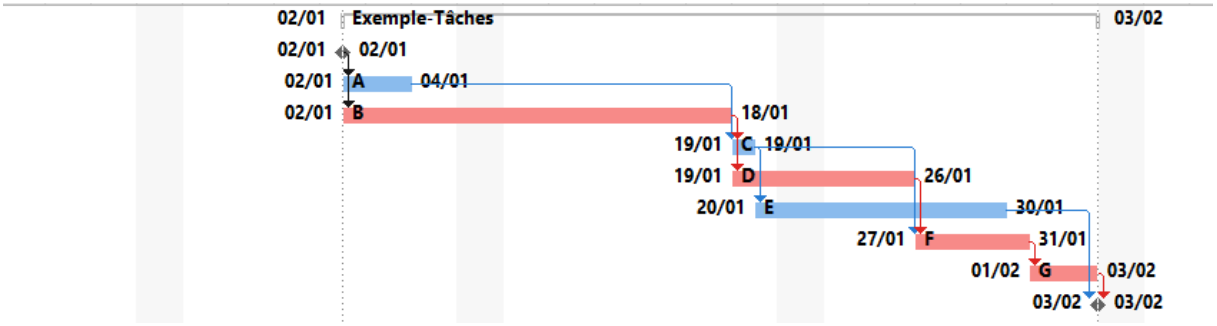
1. Sélectionner les tâches concernées dans la liste des tâches
2. Onglet « Format » → Groupe « style de barres » → « Format » → « Barre »



3. Dans la boite de dialogue « Mise en forme de la barre », cliquer sur l'onglet « Texte de la barre ».
4. Sélectionner les informations que vous souhaitez afficher, puis, cliquer sur « ok ».



Exemple

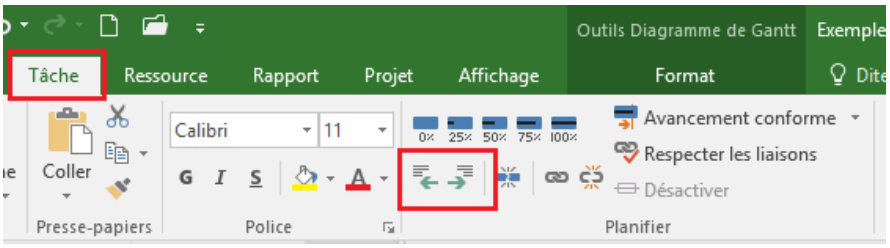


5.6 Hiérarchisation des tâches

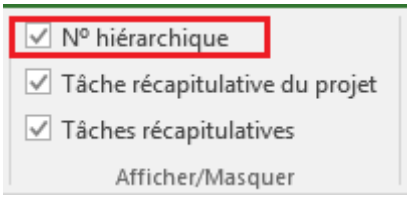
Tout projet se décompose en phases successives. Chaque phase se décompose en tâches (WBS).

Hiérarchiser les tâches consiste à les structurer en tâches subordonnées et tâches récapitulatives (phases). Pour hiérarchiser les tâches:

1. Saisir tous les tâches (récapitulatives et subordonnées), et sélectionner les tâches subordonnées dans la liste des tâches.
2. Onglet « Tâches »⟶ Groupe « Planifier », puis, utiliser les commandes « **Abaisser** » et « **Hausser** » pour définir le niveau hiérarchique de chaque tâche. Lorsque vous abaissez le niveau d'une tâche par rapport à celle située au-dessus, cette dernière devient une tâche subordonnée.



3. Pour afficher les numéros hiérarchiques des tâches: Onglet Format⟶ Groupe Afficher/Masquer⟶ Cocher la case « N° hiérarchique ».



Exemple

	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
0	**Exemple-Tâches**	**24 jours**	**02/01/22 08:00**	**03/02/22 17:00**	
1	1 Début	0 jour	02/01/22 08:00	02/01/22 08:00	
2	**2 Phase 1**	**13 jours**	**02/01/22 08:00**	**19/01/22 17:00**	
3	2.1 A	3 jours	02/01/22 08:00	04/01/22 17:00	1
4	2.2 B	12 jours	02/01/22 08:00	18/01/22 17:00	1
5	2.3 C	1 jour	19/01/22 08:00	19/01/22 17:00	3;4
6	**3 Phase 2**	**8 jours**	**19/01/22 08:00**	**30/01/22 17:00**	
7	3.1 D	6 jours	19/01/22 08:00	26/01/22 17:00	4
8	3.2 E	7 jours	20/01/22 08:00	30/01/22 17:00	5
9	**4 Phase 3**	**6 jours**	**27/01/22 08:00**	**03/02/22 17:00**	
10	4.1 F	3 jours	27/01/22 08:00	31/01/22 17:00	5;7
11	4.2 G	3 jours	01/02/22 08:00	03/02/22 17:00	10
12	5 Fin	0 jour	03/02/22 17:00	03/02/22 17:00	11;8

6 Gestion des ressources

Une ressource est toute personne, équipement, matériau et frais (de déplacement, hébergement) utilisés pour accomplir les tâches d'un projet.

6.1 Types de ressources

Microsoft Project prend en charge trois types de ressources. Il s'agit de ressources de travail, Ressources de travail, Ressources matérielles, et Coûts.

Ressources de travail

Les ressources de travail comprennent les personnes et l'équipement nécessaires (qui effectuent le travail nécessaire pour accomplir les tâches d'un projet. Les ressources de travail consomment du temps pour accomplir des tâches.

Figure 6.4. Exemple de ressources de travail d'un projet de construction: ouvrier, pelleteuse, la grue, etc.

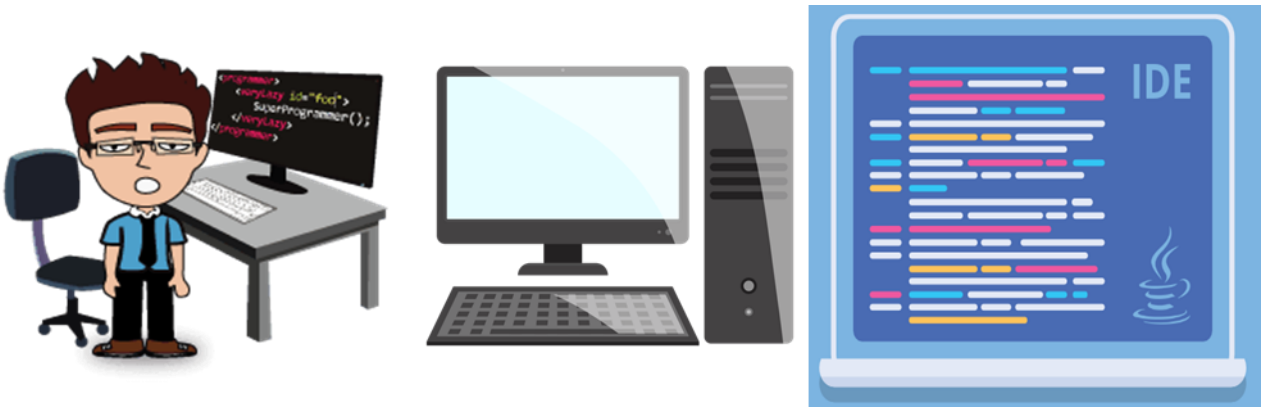


Figure 6.5. Exemple de ressources de travail d'un projet informatique: Programmeur, Hardware, Environnement de développement, etc.

Ressources matérielles

Ce sont des consommables que vous utilisez au fur et à mesure de l'avancement du projet. Par exemple, un projet de construction peut avoir besoin de consommer l'acier ou le béton utilisés tout au long du projet.

Figure 6.6. Exemples de ressources matérielles

Ressources de coût

Les ressources de coût représentent un coût financier associé à une tâche dont vous devez tenir compte. Les exemples incluent des catégories de dépenses telles que les frais de transport, les frais d'hébergement, etc.

6.2 Déclarer les ressources

La déclaration de ressources se fait dans le tableau des ressources:

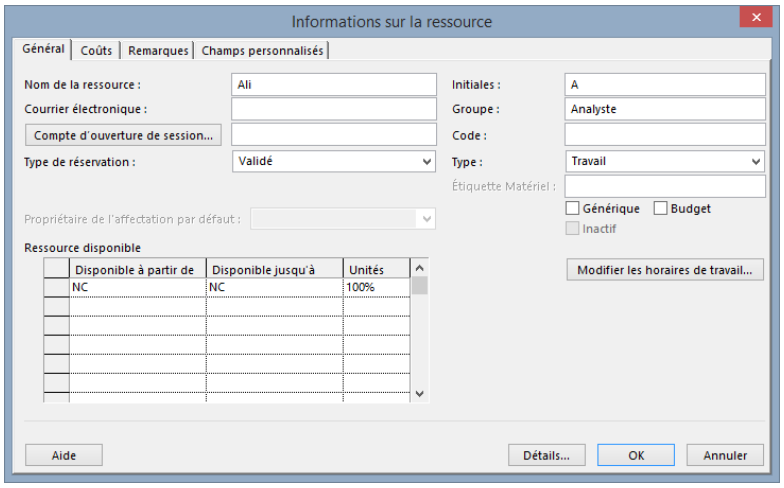
Onglet « Affichage »→ Groupe « Affichage des ressources » → Tableau des ressources

Nom de la ressource	Type	Étiquette Matériau	Initiales	Groupe	Tx. standard	pac max	Tx. hrs. sup.	Coût/Utilisation	Allocation	Calendrie de base
Mohammed	Travail		M	Programmeur	500,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard
Omar	Travail		O	Concepteur	1 000,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard
Ali	Travail		A	Analyste	1 000,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard
Karima	Travail		K	Technicien	300,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard
Salim	Travail		S	Testeur	400,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard
Aicha	Travail		A	Programmeur	500,00 DA/hr	100%	0,00 DA/hr	0,00 DA	Proportion	Standard

Le *Tableau ressource* est composé des champs suivants:

1. **Nom de la ressource:** Champs permettant d'indiquer le nom de la ressource.
2. **Types**: type de la ressource qui peut être *Travail*, *Matériel*, ou *Coût*.
3. **Etiquette:** l'unité de mesure pour une ressource **matérielle**,
4. **Initial**: Abréviation du nom de la ressource. **Exemple**: **Nom**: Mohamed Ali. **Abréviation:** MA
5. **Groupe**: Classement administrative ou autre de la ressource. Par exemple, dans un projet logiciel on peut avoir les groupes : Programmeur, Analyste, Testeur, etc. Dans un projet de construction, on peut avoir les groupes: Plombier, Maçon, etc. Le groupe peut être utilisé pour le tri des ressources pour faciliter la recherche.
6. **Capacité max**: Représente le nombre de ressources (Unités) de même type disponibles. 500% signifie que l'on dispose d'une équipe de 5 personnes. Si la valeur est inférieure à 100% il faut comprendre que la ressource est affectée au projet à temps partiel. Par exemples: Ali travaille à mi-temps sur votre projet. Vous allez donc définir une valeur de 50% dans le champ Capacité max de Ali.
7. **Taux standard**: représente le coût de la ressource par heure lorsqu'elle est utilisée durant ses heures normales de travail.
8. **Taux heures sup:** représente le coût des heures supplémentaires de la ressource par heure.
9. **Coût/utilisation:** représente le coût d'utilisation de la ressource indépendamment de la durée d'utilisation de la ressource. Ce champs est utilisé, si la ressource applique un tarif global plutôt qu'un tarif horaire.
10. **Allocation :** Mode d'attribution des coûts réels des tâches en cours. Il existe trois possibilités de calcul, soit Proportionnel, au Début ou à la Fin de la tâche.
1. **Calendrier de base:** mentionne le calendrier qui doit être utilisé comme calendrier de base pour une ressource (standard, équipe de nuit, 24 heures).

Pour voir et /ou modifier les informations d'une ressource, sélectionnez la dans le tableau des ressources et cliquez sur le bouton Informations : Onglet Ressources→ Groupe Propriétés→ Informations. *Ou bien* double-cliquer sur la ressource.

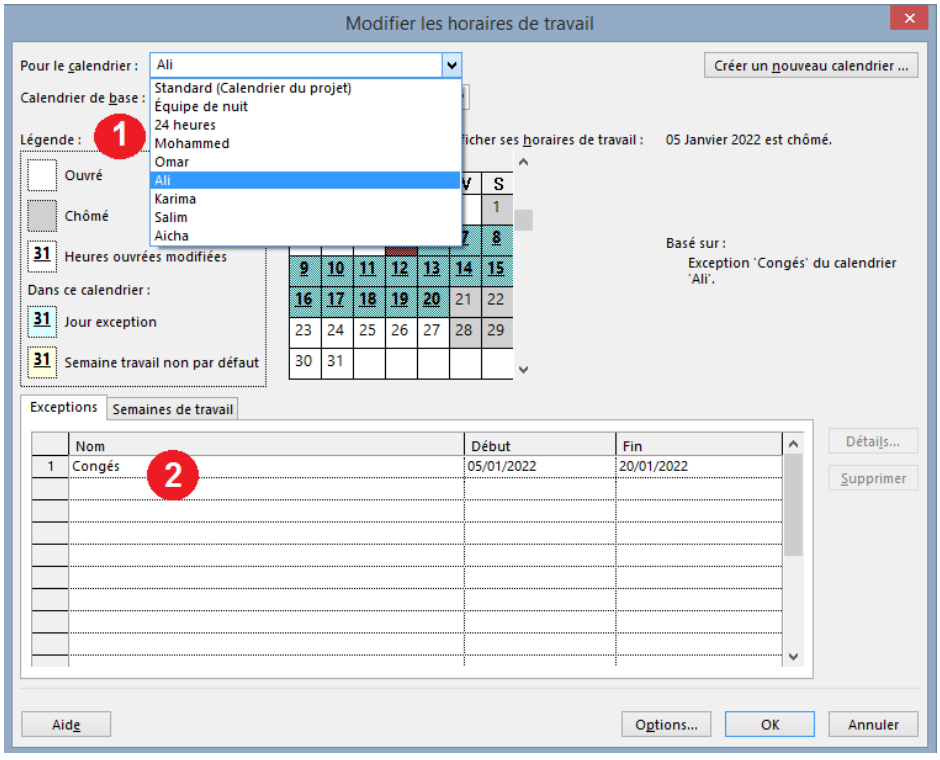


6.3 Personnaliser le calendrier des ressources (congé, absence, etc.)

Par défaut, Microsoft Project affecte le Calendrier standard à toutes les ressources. En plus, il est possible de personnaliser le calendrier d'une ressource pour prendre en compte les périodes de congé, d'absence ou les horaires particuliers de chaque ressource.

Pour personnaliser le calendrier d'une ressource :

2. Onglet « Projet »→ Groupe propriétés »→ Modifier le temps de travail.
3. Sélectionner le nom de la ressource dans la liste déroulante « Pour le calendrier »
4. Saisissez les congés, absences, etc. sous l'onglet Exception

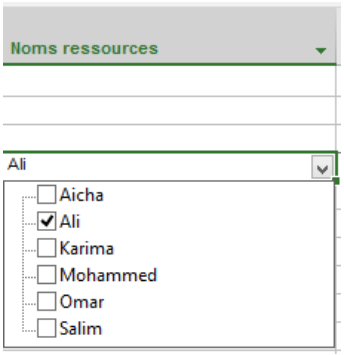


6.4 Affectation des ressources aux tâches

L'affectation des ressources aux tâches, permet de répondre à des questions telles que les suivantes : Qui doit travailler sur quelles tâches et quand ? Avons-nous suffisamment de ressources pour accomplir les travaux requis par le projet ? Existe-t-il des ressources surutilisées?

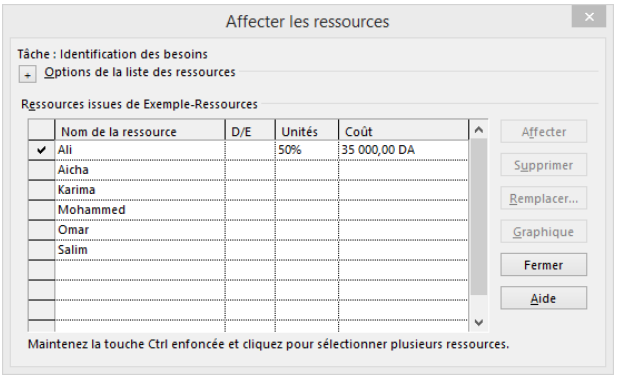
1ière Méthode :

Sélectionner une ressource dans la colonne « **Nom de la ressource** »



2ième Méthode

1. Sélectionner une tâche ;
2. Onglet « Ressources » → Groupe « Affectations » → « **Affecter les Ressources** » ;



3. Dans la boite de dialogue qui s'affiche, sélectionner la ressource à affecter ;
4. Dans le champ **Unités**, précisez le nombre d'unités à affecter de cette ressource, si la valeur est différente de 100%.
5. Cliquer sur le bouton « **Affecter**».
6. A la fin de l'affectation des ressources, cliquer sur le bouton « **Fermer** »

6.5 Détecter les surutilisations

Lorsqu'une ressource est en **surutilisation**, cette dernière apparait en rouge, avec un message (dans le tableau des ressources). Les tâches affectés à la ressource sont indiqué dans la colonne « indicateur ».

- **Exemple**: La ressource « Ali » est sur utilisées (Elle est affectée aux tâches « A » et « B » qui se déroulent au même temps)

Pour faire un diagnostic précis des surutilisations, vous pouvez procéder comme suit:

1. Onglet « Affichage »→ Groupe « Fractionner l'affichage »
2. Activer la case « Détails »
3. Dans la liste déroulante, choisissez « Graphe des ressources »
4. Activer la fenêtre du haut, sélectionner une tâche. Les ressources affectées à la tâche sélectionnée apparaissent dans la fenêtre du bas.

6.6 La surutilisation des ressources

Les ressources sont surutilisées lorsque la capacité **maximum** d'une ressource dans le tableau des ressources est dépassée et/ou lorsque, pour une période donnée, le temps disponible de la ressource est inférieur au temps affecté. La principale tâche, lors des affectations des ressources aux tâches, est d'identifier et de résoudre ces surutilisations de ressources.

6.6.1 Détecter les surutilisations

Lorsqu'une ressource est en surutilisation, cette dernière apparait en rouge, avec un message (dans le tableau des ressources). Les tâches affectées à des ressources surutilisées sont indiquées dans la colonne « indicateur » de la liste de ressources.

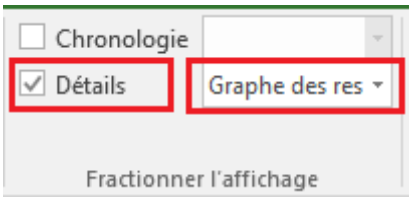
Exemple

La ressource « Ali » est surutilisées (Elle est affectée aux tâches « A » et « B » qui se déroulent au même temps).

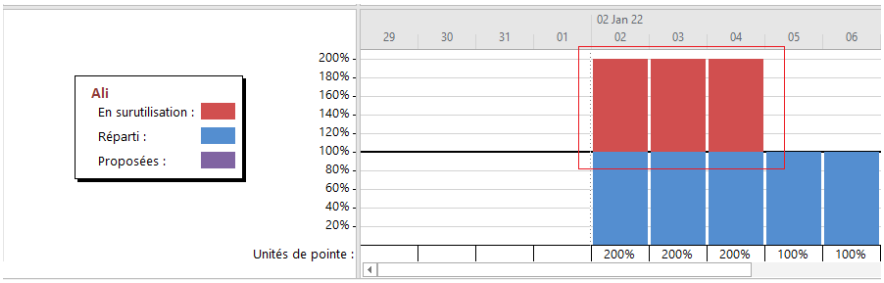
		Nom de la ressource	Type	Étiquette Matériau
1		Mohammed	Travail	
2		Omar	Travail	
3		Ali	Travail	
4		Karima	Travail	
5		Salim	Travail	
6		Aicha	Travail	

		Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressou
0			**Exemple-Ressour**	**24 jours**	**02/01/22 08:00**	**03/02/22 17:00**		
1			Début	0 jour	02/01/22 08:00	02/01/22 08:00		
2			**Phase 1**	**13 jours**	**02/01/22 08:00**	**19/01/22 17:00**		
3			A	3 jours	02/01/22 08:00	04/01/22 17:00	1	Ali
4			B	12 jours	02/01/22 08:00	18/01/22 17:00	1	Ali

1. Pour faire un diagnostic précis des surutilisations, vous pouvez procéder comme suit:
2. Onglet « Affichage »→ Groupe « Fractionner l'affichage »
3. Activer la case « Détails »
4. Dans la liste déroulante, choisissez « Graphe des ressources »



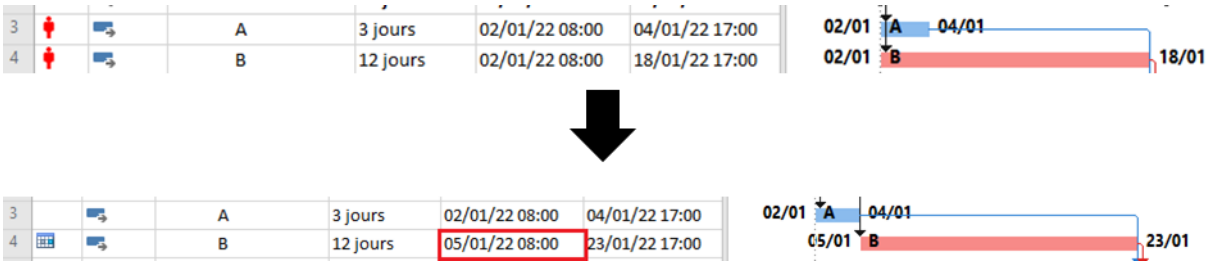
5. Activer la fenêtre du haut, sélectionner une tâche.
6. Le graphe des ressources affectées apparais dans la fenêtre du bas.



6.6.2 **Résoudre les surutilisations**

Les surutilisations des ressources peuvent être résolues de plusieurs manières :

1. **Déplacement des tâches:** Modifier la date de début d'une tâche dans la lise des tâches, ou déplacer la tâche dans le diagramme de Gantt.



2. Substitution d'une ressource surutilisée par une autre.

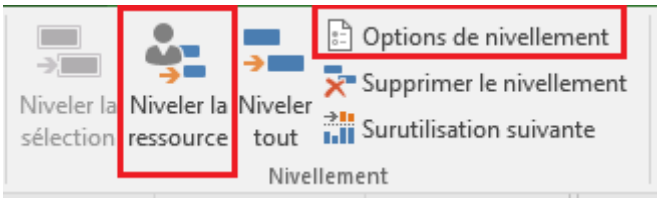
S'il y'a des ressources disponibles, il est possible de remplacer la ressource surutilisée par une autre.

Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
	A	3 jours	02/01/22 08:00	04/01/22 17:00	1	Ali
	B	12 jours	02/01/22 08:00	18/01/22 17:00	1	Aicha

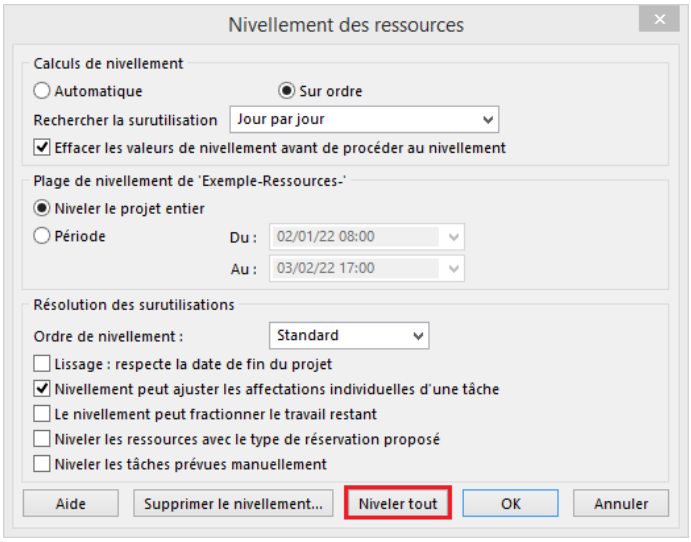
3. Nivellement des ressources

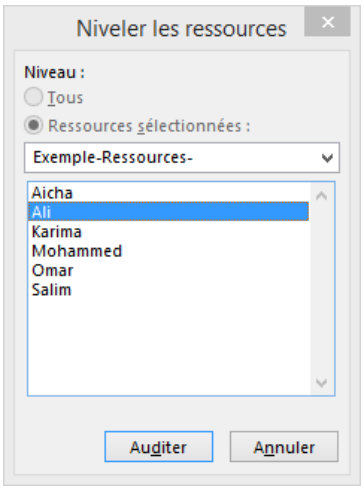
Le nivellement résout automatiquement les surutilisation en retardant ou en fractionnant les tâches en fonction des paramètres de la boite de dialogue *Nivellement de ressources:*

1. Onglet « *Ressources* » → Groupe « *Nivellement*» → « *Options de nivellement*»



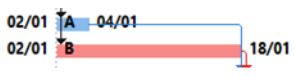
Il est possible de niveler toutes les ressources en cliquant sur le bouton « Niveler tout », ou niveler chaque ressource à part en sélectionnant la commande « Niveler la ressource ».



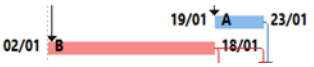


Dans notre exemple, MS Project a résolu facilement la surutilisation en retardant la tâche A.

3		A	3 jours	02/01/22 08:00	04/01/22 17:00	1	Ali
4		B	12 jours	02/01/22 08:00	18/01/22 17:00	1	Ali



3		A	3 jours	19/01/22 08:00	23/01/22 17:00	1	Ali
4		B	12 jours	02/01/22 08:00	18/01/22 17:00	1	Ali



6.7 Gérer les modifications d'affectation

Le planificateur peut modifier les données initiales pour une tâche donnée, il peut changer:

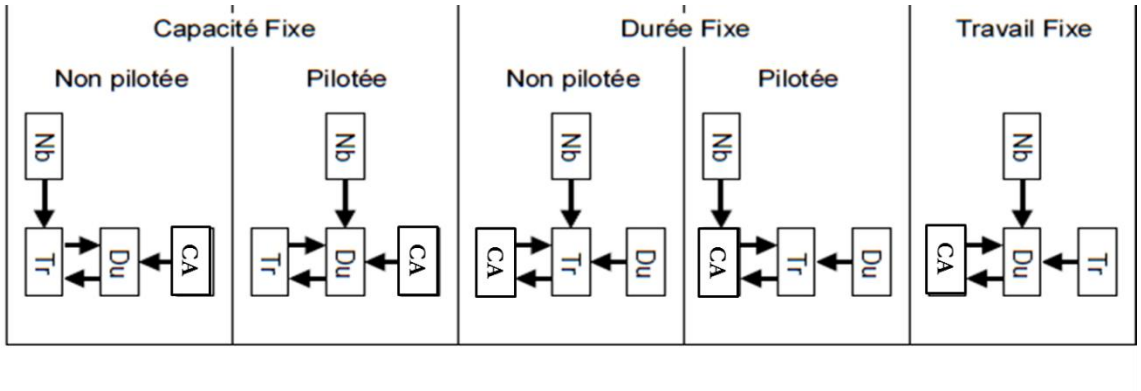
1. Le ***nombre*** de ressources affectées à une tâche,
2. La ***durée*** d'une tâche,
3. Le ***travail*** d'une tâche (l'effort)
4. Les ***unités d'affectation*** (ou la ***capacité***): la disponibilité d'une ressource pour une tâche. (La valeur par défaut est de 100 %).

Le travail, la durée et la capacité (%) sont déterminés par la formule suivante :

$$\boldsymbol{Travail = Durée \times Capacité}$$

La modification de l'un des 4 paramètres influe sur les autres.

MS Project propose 5 types de tâches, choisis par l'utilisateur:

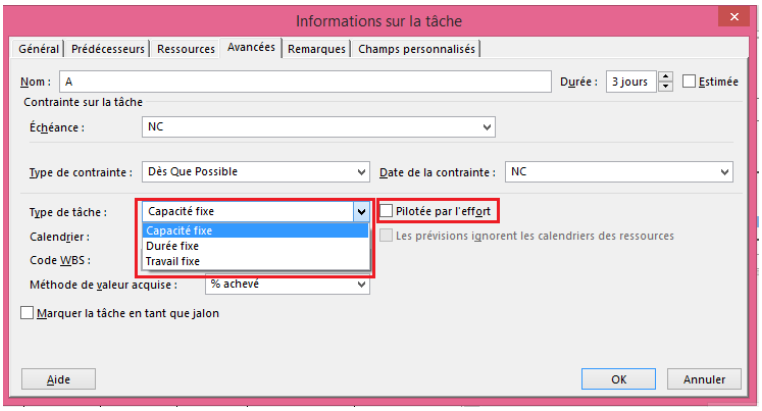


- ***Durée* fixe.** Une tâche à **Durée fixe** conserve la valeur que vous avez saisie dans le champ **Durée.**
- ***Travail fixe.*** Une tâche à **Travail fixe** à une tâche, conserve le travail, si l'affectation des ressources est modifiée, Microsoft Project recalcule la durée.
- ***Capacité fixe.*** Le type de tâche **Capacité fixe** la capacité des ressources affectées à la tâche.
- ***Tâche pilotée par l'effort.*** Pour les types de tâche, **Capacité fixe** et **Durée fixe.** Si l'option **Piloté par l'effort** est activée, Microsoft Project maintient le travail total de la tâche à sa valeur en cours, quel que soit le nombre de ressources affectées à la tâche.

Pour modifier ces paramètres pour une tâche:

1ère méthode:

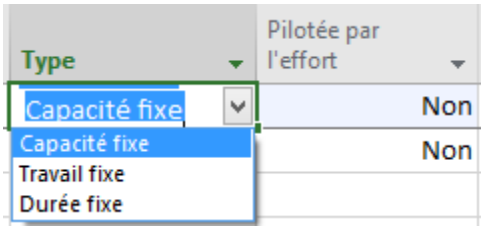
1. Double-clic sur la tâche pour ouvrir la boite de dialogue « Informations sur la tâche »,
2. Activer l'onglet « Avancées »
3. Choisissez le type de la tâche, et cocher/décocher la case « Pilotés par l'effort » .



2ème méthode:

Dans la liste des tâches, insérer les colonnes:

1. **Type**: "*Capacité fixe*", "*Durée fixe* " ou "*Travail fixe* "
2. **Pilotée par l'effort:** *Oui*, *Non*



Exemple:

- L'ajout d'une ressource sur une tâche qui consisterait à déplacer un tas de briques réduit la durée de la tâche mais laisse inchangée la valeur du travail. Le modele « Travail fixe » est donc adapté pour cette tâche.
- L'ajout d'une ressource sur une tâche qui consistant à assister à une rénion ne modifie pas la durée de la tâche et la capacité des assistants, mais en augmente le travail. Le modele « Durée fixe, non piloté par l'effort » est adapté pour cette tâche.

Affectation d'une ressource pour chaque tâche :

Mode Tâche	Nom de la tâche	Durée	Travail	Début	Fin	Noms ressources	Type	Pilotée par l'effort
	Déplacer un tas de briques	1 jour	8 hr	02/01/22 08:00	02/01/22 17:00	R1	Travail fixe	Oui
	Rénion	1 jour	8 hr	02/01/22 08:00	02/01/22 17:00	R2	Durée fixe	Non

Affectation de 2 ressources pour chaque tâche :

Mode Tâche	Nom de la tâche	Durée	Travail	Début	Fin	Noms ressources	Type	Pilotée par l'effort
	Déplacer un tas de briques	0,5 jour	8 hr	02/01/22 08:00	02/01/22 12:00	R1;R3	Travail fixe	Oui
	Rénion	1 jour	16 hr	02/01/22 08:00	02/01/22 17:00	R2;R4	Durée fixe	Non

7 Suivi avec MS Project

Le suivi du projet consiste à comparer la planification initiale avec les données réelles constatées lors de la réalisation (dates réelles de début et de fin des tâches, travail réel, dépenses constatées, etc.). Le suivi permet au chef de projet de déterminer si le projet se réalise de façon acceptable.

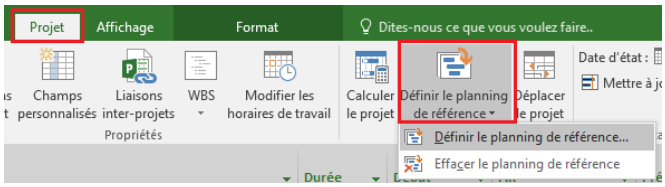
Microsoft Project permet d'enregistrer deux types de planifications : la planification initiale et les planifications temporaires. La première est obligatoire et doit être enregistrée une fois au cours de la gestion du projet; les planifications temporaires sont optionnelles. Elles permettent aux utilisateurs chevronnés d'enregistrer des plans intérimaires et d'établir plusieurs scénarios de planification.

7.1 Enregistrer la planification initiale

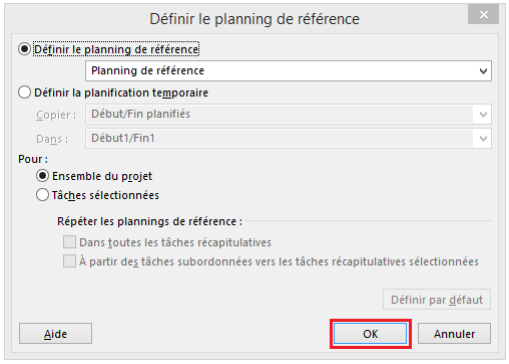
Pour piloter le projet il faut conserver les données de la planification initiale, elle contient les prévisions concernant les durées, les tâches, les ressources, les affectations et les coûts planifiés. Lors de l'enregistrement de la planification initiale, Microsoft Project copie ces informations dans les champs planifiés : Durée planifiée, Travail planifié, Coût planifié, Date de début et de fin planifiée, etc. C'est sur la base des informations contenues dans les champs «planifiés» que l'analyse comparative entre les prévisions et l'avancement réel du projet sera effectuée.

Pour enregistrer une planification initiale :

1. Onglet « Projet »→ Groupe « planifier » → « Définir le planning de référence »



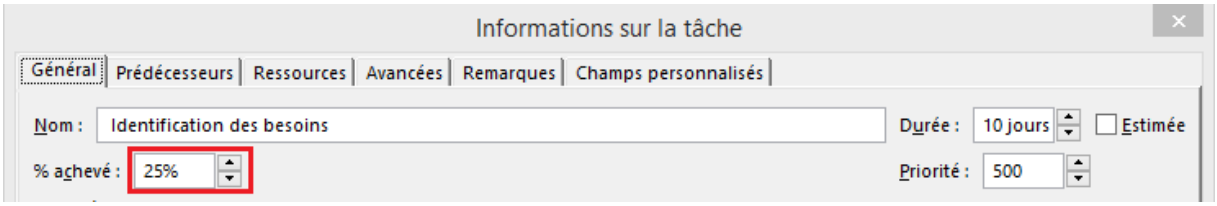
2. Cliquer sur « OK » dans la boite de dialogue qui s'affiche.



7.2 Mise à jour de l'avancement des tâches

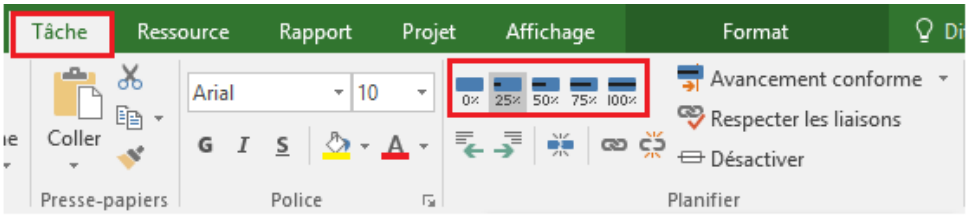
Pour vérifier l'avancement des tâches et le bon déroulement du projet, l'état de chaque tâche doit être périodiquement évalué et mis à jour en fonction des données réelles. Pour indiquer l'avancement réel d'une tâche en saisissant le pourcentage achevé :

1. Double cliquer sur le nom de la tâche pour afficher « Information sur la tâche ».
2. Saisissez le pourcentage achevé, puis, cliquer sur « OK ».



Ou bien:

1. Sélectionner la tâche ;
2. Cliquer sur le pourcentage d'achèvement approprié dans l'onglet « Tâche »→ groupe « Planifier »



Pour indiquer les dates de début et de fin réel des tâches, insérer les colonnes *Début réel* et *Fin réelle* dans le tableau des tâches.

Pour comparer les dates planifiées et les dates réelles, insérer les colonnes « *Planning de référence-Début estimé* » et « *Planning de référence-Fin estimé* ».

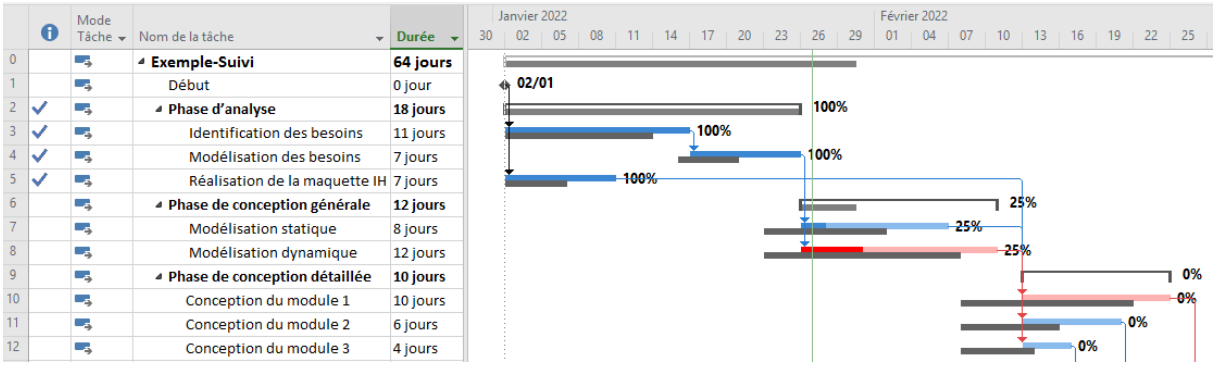
Nom de la tâche	Début réel	Fin réelle	Planning de référence - Début estimé	Planning de référence - Fin estimée
Exemple-Suivi	**02/01/22 08:00**	**NC**	**02/01/22 08:00**	**27/03/22 17:00**
1 Début	NC	NC	02/01/22 08:00	02/01/22 08:00
2 Phase d'analyse	**02/01/22 08:00**	**25/01/22 17:00**	**02/01/22 08:00**	**20/01/22 17:00**
2.1 Identification des besoins	02/01/22 08:00	16/01/22 17:00	02/01/22 08:00	13/01/22 17:00
2.2 Modélisation des besoins	17/01/22 08:00	25/01/22 17:00	16/01/22 08:00	20/01/22 17:00
2.3 Réalisation de la maquette IHM	02/01/22 08:00	10/01/22 17:00	02/01/22 08:00	06/01/22 17:00
3 Phase de conception générale	**23/01/22 08:00**	**NC**	**23/01/22 08:00**	**07/02/22 17:00**
3.1 Modélisation statique	23/01/22 08:00	NC	23/01/22 08:00	01/02/22 17:00
3.2 Modélisation dynamique	23/01/22 08:00	NC	23/01/22 08:00	07/02/22 17:00
4 Phase de conception détaillée	**NC**	**NC**	**08/02/22 08:00**	**21/02/22 17:00**
4.1 Conception du module 1	NC	NC	08/02/22 08:00	21/02/22 17:00
4.2 Conception du module 2	NC	NC	08/02/22 08:00	15/02/22 17:00
4.3 Conception du module 3	NC	NC	08/02/22 08:00	13/02/22 17:00

7.3 Suivre l'avancement avec le « Suivi Gantt »

L'affichage **Gantt Suivi** affiche simultanément la planification initiale et les données réelles.
Pour afficher le « Suivi Gantt » :
Onglet « Affichage » → Groupe « Affichage des tâches→ Diagramme de Gantt→ Suivi Gantt.

Les barres des tâches apparaissent dans des couleurs différentes :

- Les tâches apparaissent sous la forme de fines barres de couleur, rouges pour les tâches critiques et bleu pour les non critiques.
- Le planning de référence apparaît dans l'affichage « Suivi Gantt » sous la forme de barres grises.
- Le pourcentage d'achèvement de chaque tâche est affiché.
- La ligne verticale représente la date d'aujourd'hui.



Bibliographie

1. McConnell, S., *Software estimation: demystifying the black art*. Microsoft press. 2006.
2. Sommerville, I., *Software engineering*. Harlow, England: Pearson Education Limited. 2016.
3. Boehm, B.W., *Software engineering economics*. IEEE transactions on Software Engineering, 1984(1): p. 4-21.
4. Chemuturi, Murali, and Thomas M. Cagley. *Mastering software project management: Best practices, tools and techniques*. J. Ross Publishing, 2010.
5. Unhelkar, B. *Software Engineering with UML*. Boca Raton: Auerbach Publications/CRC Press. 2018.
6. Roger, S. P., & Bruce, R. M. *Software engineering: a practitioner's approach*. McGraw-Hill Education. 2015.
7. Stellman, Andrew, and Jennifer Greene. *Applied software project management*. " O'Reilly Media, Inc.", 2005.
8. Wysocki, Robert K. *Effective project management: traditional, agile, extreme*. John Wiley & Sons, 2011.
9. Morley, Chantal. *Management d'un projet système d'Information-8e éd.: Principes, techniques, mise en oeuvre et outils*. Dunod, 2016.
10. Murray, Anna P. *The Complete Software Project Manager: Mastering Technology from Planning to Launch and Beyond*. John Wiley & Sons, 2016.
11. Abran, Alain. *Software project estimation: The fundamentals for providing high quality information to decision makers*. John Wiley & Sons, 2015.
12. Garmus, David, Janet Russac, and Royce Edwards. *Certified Function Point Specialist Examination Guide*. CRC Press, 2010.
13. Hill, Peter R. *Practical software project estimation: a toolkit for estimating software development effort & duration*. McGraw-Hill Education, 2011.
14. Chemuturi, Murali. *Software estimation best practices, tools & techniques: A complete guide for software project estimators*. J. Ross Publishing, 2009.
15. IEEE Std 610.12-1990. *IEEE Standard Glossary of Software Engineering Terminology*. 1990.
16. Strohmeier, Alfred. *Cycle de vie du logiciel*. Ecole Polytechnique Fédérale de Lausanne 28 (2000).
17. de Barcelos Tronto, I.F., J.D.S. da Silva, and N. Sant'Anna. *An investigation of artificial neural networks based prediction systems in software project management*. Journal of Systems and Software, 2008. 81(3): p. 356-367.
18. Laqrichi, Safae. *Approche pour la construction de modèles d'estimation réaliste de l'effort/coût de projet dans un environnement incertain: application au domaine du développement logiciel*. Diss. Ecole nationale des Mines d'Albi-Carmaux, 2015.
19. Pow-Sang, J. A., Gasco, L., & Nakasone, A.. *Evaluating the applicability of a function point logic file identification technique through controlled experiments. International Journal of Software Engineering and Its Applications*, (2010), 4(3), 29-42.
20. Pow-Sang, J. A., Villanueva, D., Flores, L., & Rusu, C. (2013, October). *A Conversion Model and a Tool to Identify Function Point Logic Files using UML Analysis Class Diagrams*. In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on* (pp. 126-134). IEEE.
21. Leonard, Barry, ed. *GAO Cost estimating and assessment guide: Best practices for developing and managing capital program costs*. Diane Publishing, 2009.