REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNMENT SUPERIEUR ET LA RECHERC SCIENTIFIQUE

UNIVERSITE IBN -KHALDOUN DE TIARET

FACULTE DES SCIENCES APPLIQUÉES

DEPARTEMENT DE GENIE ELECTRIQUE

MEMOIRE DE FIN D'ÉTUDES

Pour l'obtention du diplôme de Master

Domaine : Sciences et Technologie

Filière : Génie Electrique

Spécialité : Informatique Industrielle

THÈME

ÉTUDE DES SYSTÈMES DE SUIVI DES PROCESSUS DE QUALITÉS

Préparé par : ZEGGOU Tamani

Devant les jurys :

Nom et prénom	Grade	Qualité
Mr : ACED Med Redha	MAA	Président
Mr : BELHADJI Youcef	MAA	Examinateur
Mr : SAHLI Belgacem	MCB	Encadreur

PROMOTION 2015/2016

Remerciement

Je tiens à présenter mes remerciements à Mr. SAHLI Belgacem qui m'a proposé le Thème sur le développement des systèmes de suivi des processus de qualité qui est l'objet de mon travail, et qu'il trouve ici l'expression de mon gratitude pour ses conseils, son Orientation et toute sa disponibilité.

Mes vifs remerciements s'adressent également à tous mes enseignants pour tous les services qu'ils m'ont rendus.

Merci enfin à tous mes collègues d'étude et mes proches d'avoir donné un second souffle à mon travail.

Dédicace

Je dédie ce mémoire

A mes chers parents ma mère et mon père

Pour leur patience, leur amour soutien et leurs encouragements.

A mes frères, A mes amies et mes camarades.

Sans oublier tous les professeurs que ce soit du primaire, du moyen, du secondaire ou de l'enseignement supérieur.

CHAPITRE III : PROCESSUS DE QUALITES

CHAPITRE IV : SYSTEME DE MANAGEMENT DE LA QUALITE

Introduction Générale

Les applications informatiques dites de contrôle-commande ont envahi l'environnement industriel et notre vie quotidienne. Depuis quelques décennies, les besoins de plus en plus accrus en termes de technicité ont conduit à intégrer une très forte automatisation dans tous les produits industriels ou destinés à l'usage « grand public »

Dans ce projet, on va donner le développement lié entre les deux parties « matériel et logiciel », une présentation succincte du matériel est faite. En revanche, nous allons nous intéresser particulièrement à l'aspect informatique ou plus exactement à l'aspect génie logiciel.

Tout d'abord Le suivi est l'évaluation en permanence de l'exécution d'un projet au regard des échéances convenues, ainsi que de l'utilisation par les bénéficiaires du projet, des ressources, de l'infrastructure et des services mis à leur disposition » et L'évaluation c'est la mesure de l'impact, de la performance et de l'efficacité d'un projet par rapport à ses objectifs initiaux.

Le système de suivi est d'abord basé sur les objectifs du projet Plus les objectifs sont clairs et plus le Suivi & Evaluation sera efficace.

L'analyse des objectifs du projet permet de voir si toutes les parties engagées ont la même vision des objectifs et des cibles à atteindre.

Et un processus est un ensemble d'activités corrélées ou interactives qui transforme des éléments d'entrée en éléments de sortie, et ces éléments sont soit des objets matériels (pouvant être perçus comme des flux par la logistique à des fins d'évaluation) soit des informations soit les deux notes :

- Les éléments d'entrée d'un processus sont généralement les éléments de sortie d'autres processus amont.
- Les processus d'un organisme sont généralement planifiés et mis en œuvre dans des conditions maîtrisées afin d'apporter une valeur ajoutée.
- Lorsque la conformité du produit résultant ne peut être immédiatement ou économiquement vérifiée, le processus est souvent qualifié de « procédé spécial ».

En fin ce projet consiste à faire le suivi des processus et donc de développer une application recouvrant toutes les activités de ces processus.

Alors dans les deux premiers chapitres (chapitre 1, chapitre 2) on va voir tous ce qui concerne le système temps réel, Et concernant le 3ème et le 4ème chapitre, on va parler sur le processus de qualités et les systèmes de management de la qualité.

Chapitre : I

SYSTEME TEMPS REEL

I.1 Introduction :

Un **exécutif temps réel** peut être employé à la place d'un système d'exploitation généraliste par un système de contrôle-commande lorsqu'il est soumis à des contraintes de temps, ou bien lorsqu'il doit être embarqué sur un microcontrôleur.

Le comportement d'un système informatique est qualifié de «**temps réel**» lorsqu'il est assujetti à l'évolution d'un procédé qui lui est connecté et qu'il doit piloter ou suivre en réagissant à tous ses changements d'états.

Un système temps réel est défini comme un système dont le comportement dépend :

- **De l'exactitude des traitements effectués**
- **Du temps où les résultats sont produits**

Un **retard** = le fait de rater une échéance = erreur du système.

Alors Un système est dit **Temps Réel** lorsque l'information après acquisition et traitement est **encore pertinente**". Cela veut dire que dans le cas d'une information arrivant de façon périodique, le temps d'acquisition/traitement doit être inférieur à la période de rafraîchissement de cette information. Pour cela, un RTOS doit être **déterministe et préemptif** pour donner la main durant le prochain traitement à la tâche de plus forte priorité prête.

Exemple d'un RTOS (Système d'Exploitation Temps Réel): Le µ C/OS II

La notion de temps réel correspond à la façon dont les tâches sont exécutées dans le temps : le temps d'exécution des tâches étant déterminant pour la commutation des tâches, le noyau temps réel exécute en premier les tâches dont le temps d'exécution est critique.

I.1.1. Limitations des systèmes d'exploitation généralistes

Lorsqu'une application de contrôle-commande est soumise à de fortes contraintes de temps, l'utilisation d'un système d'exploitation généralistes est inadaptée car leurs objectifs suivants :

- Garantir l'indépendance des processus et les protéger les uns vis-à-vis des autres notamment grâce à la MMU (*Memory Management Unit*), ce qui implique une mise en œuvre lourde des communications entre les processus,
- Augmenter la vitesse moyenne de traitement à l'aide d'optimisations locales ; utilisation d'un cache disque permettant un accès asynchrone, mais non prédictible aux mémoires de masse ; utilisation des optimisations des processeurs tels les *pipelines* et la mémoire cache nuisant à la prédictibilité du temps d'exécution (*les optimisations processeur peuvent être utilisées par les noyaux temps réel aussi*),
- Limiter le surcoût processeur dû au système d'exploitation, ce qui conduit souvent à une gestion grossière du temps : l'unité de temps typique est la milliseconde ou la dizaine de millisecondes,

- Favoriser l'équité dans l'ordonnancement au détriment du temps de réponse utilisant des quanta de temps de l'ordre d'une dizaine de millisecondes,
- Faciliter la maintenance du système en offrant de nombreux processus de maintenance en concurrence avec les processus des utilisateurs, ce qui augmente la charge processeur et mémoire du système,
- Rendre transparente l'utilisation de la mémoire, en utilisant la mémoire virtuelle, ce qui nuit à la prédictibilité

De plus pour arriver à ces buts, les systèmes d'exploitation généralistes sont le plus souvent monolithiques, en d'autres termes, le système d'exploitation est muni 'un noyau de base permettant la gestion des processus, leurs ordonnancement et la gestion de la mémoire et de la mémoire virtuelle.

I.1.2 Noyau, exécutif et systèmes d'exploitation temps réel

Les noyaux temps réels assurent la gestion de tâches (ordonnancement, outils de synchronisation et de communication, gestion des interruptions) et de la mémoire. Ils sont appelés micronoyaux quand ils ont une faible empreinte mémoire (taille mémoire nécessaire à leur exécution). Un noyau temps réel est apte à être embarqué sur un microcontrôleur ou un microprocesseur disposant de peu de mémoire.

Un exécutif temps réel est une surcouche du noyau : il offre les services nécessaires à l'utilisation des entrées/sorties, du réseau, des timers, des fichiers … etc. le plus souvent ces services sont modulaires. Et il est possible de configurer spécifiquement un exécutif en fonction de l'architecture utilisée. Ces modules peuvent être configurés de façon monolithique ou sous forme de tâches de service

Les exécutifs temps réel sont moins ouverts, moins flexibles et optimisent moins la vitesse moyenne des traitements que les systèmes d'exploitation généralistes mais ils favorisent la prédictibilité temporelle

Alors un système d'exploitation temps réel est un système d'exploitation complet reposant sur un exécutif temps réel.

I.2. Concepts des exécutifs temps réel

Les langages de programmation utilisés pour développer une application s'appuient sur les services fournis par l'exécutif temps réel afin de gérer des tâches de les faire communiquer, se synchroniser, de gérer le temps, de traiter les interruptions matérielles. Ces services influencent l'état des tâches sur lequel s'appuie l'ordonnanceur afin de gérer l'exécution du système.

I.2.1 Gestion des tâches

Les noyaux temps réel gèrent les tâches suivant le même principe que les systèmes d'exploitation généralistes. Cependant étant donné que les tâches sont gérées finement, certains noyaux distinguent la création et l'initialisation d'une tâche.

Une tâche est initialement gérée, ainsi, elle devient existante mais non initialisées. Elle doit être initialisée ce qui la met en état prête. C'est dans cet état qu'elle requiert un processeur, lorsque l'ordonnanceur le décide, suivant la politique d'ordonnancement choisie cette tâche se voit allouer un processeur afin d'être exécutée.

De l'état exécutée, une tâche préemptée ou bien se bloquée en attendant un message, un événement ou bien l'accès à une ressource. Lorsqu'elle se met en attente pendant un certain temps ou bien jusqu'à une certaine date, On dit elle est endormie

Une tâche peut généralement être supprimée à partir de tout état. Afin de prévenir la perte d'une ressource suite à la suppression d'une tâche la détenant, certains exécutifs proposent des primitives de protection contre la suppression.

I.2.2 Outils de communication et de synchronisation

Ce paragraphe présente les outils de communication et de synchronisation permettant aux tâches d'interagir. Auparavant, nous revenons sur les outils de base que sont les sémaphores et les moniteurs, qui seront utilisés par la suite pour donner une implémentation d'outils de communication et de synchronisation, ainsi que sur les variables conditionnelles en effet, tous les outils ne sont pas nécessairement proposés par les exécutifs, et le concepteur d'une application doit parfois lui-même implémenter, en utilisant les outils de base.

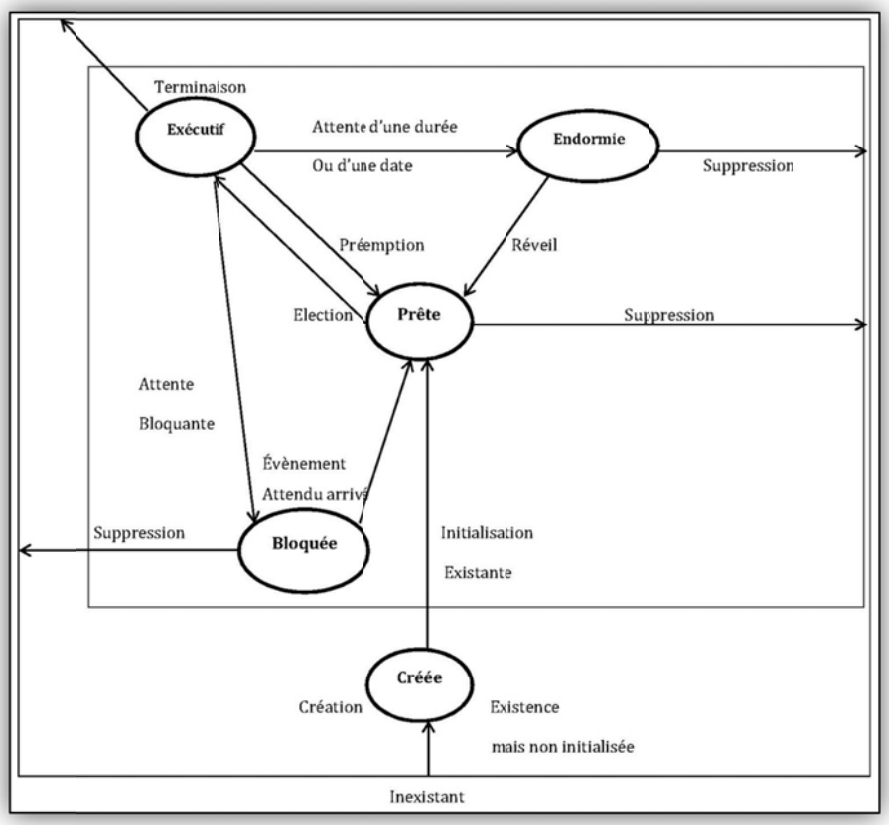


Fig. 1.1. [1] Graphie simplifié des états possibles
des tâches gérées par un noyau temps réel

❖ **Éléments de base**

Sémaphores

Le sémaphore est une variable soit binaire (deux états sont possibles : libre et pris), soit n-aire. Dans les cas des sémaphores n-aire, appelés sémaphore à compte. la valeur 0 indique que le sémaphore est pris et une valeur différente de de zéro indique qu'il y a un certain nombre d'instances libre.

Un sémaphore binaire est caractérisé par une valeur booléenne est une file d'attente des tâches en attente du sémaphore. La file d'attente peut être gérée en fonction de l'exécutif sous-jacent. Soit de façon FIFO, soit sous la forme de plusieurs files FIFO gérées par priorité des tâches désirant accéder au sémaphore. Souvent appelé mutex (mutual exclusion) lorsqu'il est utilisé pour assurer l'exclusion mutuelle, le sémaphore binaire peut être couplé. Suivant les exécutifs, avec un protocole à priorité héritée ou bien un protocole à priorité plafond afin d'éviter le phénomène. Dans le cas du protocole à priorité plafond, il est nécessaire de caractériser le sémaphore par une priorité plafond c'est-à-dire la plus grande priorité parmi les tâches susceptibles de l'utiliser.

Un sémaphore à compte ou sémaphore compteur peut être utilisé pour garantir une exclusion mutuelle lors de l'accès à une ressource multi-instance, ou encore permettre un accès de type

lecteur/écrivain à une ressource, ou bien encore pour effectuer une synchronisation à compte. Dans les deux premiers cas comme le mutex, il peut être couplé à un protocole à gestion de ressource.

- **Moniteurs**

Un moniteur est un outil strictement plus puissant que le sémaphore, il existe deux types de moniteur :

✓ Le moniteur classique, appelé **moniteur de Hoare,** dans lequel la non restante des primitives est couplée avec un mécanisme d'attente (wait) et de réveil (signal) permettant de mettre en attente ou réveiller une tâche accédant au moniteur sous certaines conditions.

✓ Le **moniteur à la Ada** (objet protégé) permet de mettre une garde (barrière logique) à l'entrée de chaque primitive du moniteur, ce qui permet une gestion plus fine et plus transparente des conditions d'accès que dans le cas du moniteur de Hoare.

- **Variables conditionnelles**

Peu d'exécutifs et langages de programmation proposent nativement le moniteur. La primitive wait d'un moniteur Hoare consiste, de façon atomique (non préemptible), à libérer le verrou du moniteur, puis au moment du signal, réveil de la tâches et prise du verrou. Ce fonctionnement n'est pas implémentable par sémaphore et nécessiterait une portion de code non préemptible.

De nombreux exécutifs proposent donc un outil appelé variable conditionnelle effectuant lors d'un wait de façon atomique ; libération du verrou, passage de la tâche à l'état bloquée en attente de la variable conditionnelle, puis au moment du signal, réveil et prise du verrou.

Notons qu'en général, il est possible de prendre en compte les priorités des tâches en attente d'une même variable conditionnelle

Variable conditionnelle	
Verrou du Moniteur	File d'attente De tâches

Tableau 1.1. [1] Caractérisation d'une variable conditionnelle

- **Les variables conditionnelles** sont des outils de façon conjointement utilisées avec des sémaphores d'exclusion mutuelle pour la construction de moniteurs de Hoare.
- **Signaux**

Les signaux sont des événements (sans données) pouvant être envoyés à une ou plusieurs tâches simultanément. Il y a deux types de signaux : les signaux synchrones internes à une tâche ou

un processus et les signaux asynchrones provenant une tâche ou processus, ou bien une source matérielle externe.

Dans les cas des signaux ; le terme synchrone caractérise un signal interne et le terme asynchrone caractérise le signal externe.

- ***Signaux synchrones***

Un signal synchrone est un résultat d'un événement interne à une tâche. Cet événement est traité immédiatement (de façon synchrone à son occurrence) par la tâche. Lorsqu'un événement interne a lieu (par exemple erreur arithmétique comme une division par zéro par exemple ou une violation de segmentation mémoire ou bien le redimensionnement d'une fenêtre graphique gérée par une tâche, etc.), une occurrence du signal asynchrone associé a lieu. A chaque signal est associée une action par défaut. Typiquement, l'action par défaut consiste à terminer la tâche, ou bien le processus père.

- ***Signaux asynchrones***

Les signaux asynchrones sont des signaux provenant d'une source externe à l'adresse d'une tâche ou un processus (signal privé), ou bien à plusieurs tâches et/ou plusieurs processus (signal public). Le principe fondamental repose sur la sensibilisation des tâches à différents signaux il y a deux façons pour une tâche de se sensibiliser à un signal : une tâche peut déclarer une ASR (Asynchronous Service Routine) et la lier à un signal ; la tâche devient alors réceptive au signal, et lorsque le signal a lieu, l'ASR est appelée dans le contexte (en utilisant la même pile d'exécution de la tâche, interrompant le traitement en cours. Il est aussi possible pour une tâche de se mettre explicitement en attente d'un signal : la tâche devient donc bloquée et l'occurrence de l'événement la fait passer dans l'état prête.

- ✓ **Signal fugace**

Si l'occurrence ne peut être prise en compte, elle est perdue.

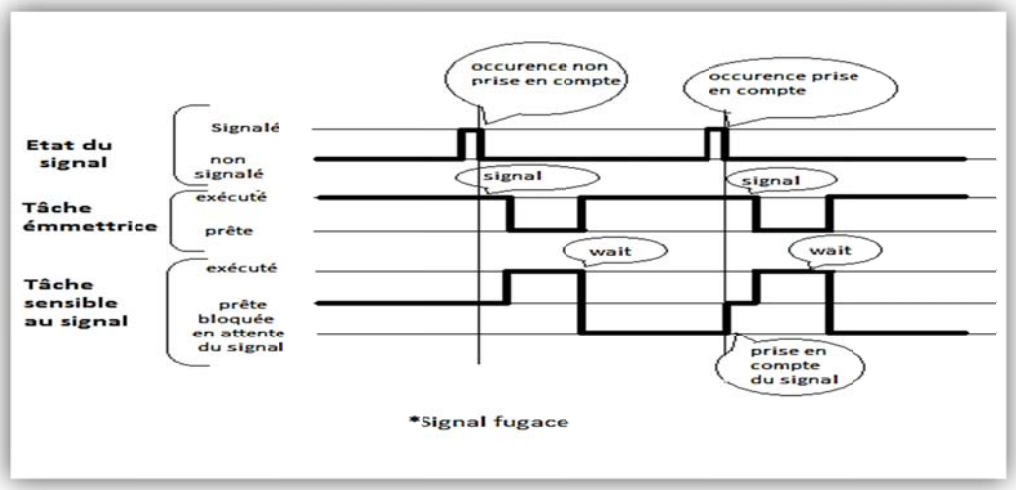


Fig.1.2. [1] Les occurrences non prises en compte immédiatement sont perdues

✓ **Signal mémorisé**

Si elle ne peut être prise en compte, l'occurrence et mémorisée. Si une autre occurrence du même événement a lieu d'ici à sa prise en compte, elle est ignorée. Ce type de mémorisation suppose qu'une tâche (typiquement tâche recevant le signal) réinitialise l'occurrence.

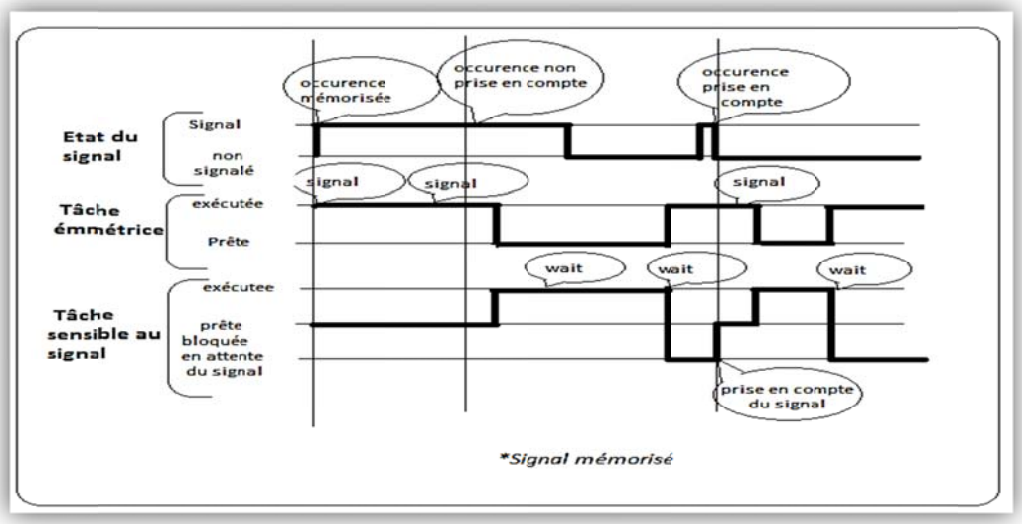


Fig.1.3. [1] Signal mémorisé au plus une occurrence est immédiate

✓ **Signal à compte**

Les occurrences non prises en compte immédiatement sont comptées pour utilisation ultérieure, dans ce cas, la prise en compte du signal suppose une décrémentation du compte.

- **Communication par message**

Il existe différents outils permettant d'assurer des communications par messages :

- Boite aux lettres ;
- Tube ;
- Rendez-vous ;

- Tableau noir ;
- ……

Chacun de ces outils a ses spécificités propres qui sont présentées dans les paragraphes suivants.

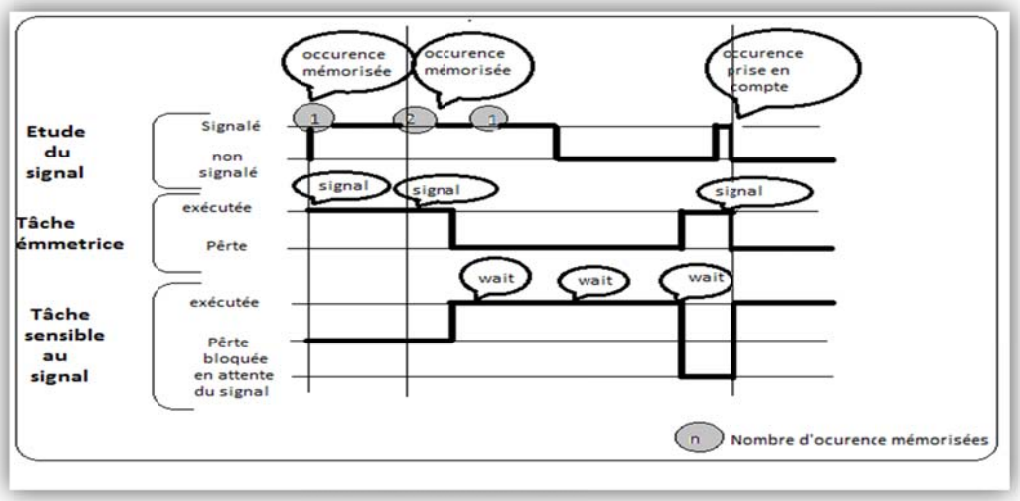


Fig.1.4. [1] Le nombre d'occurrence non prises en compte est mémorisé

La boite aux lettres

La boite aux lettres permet une communication asynchrone puisque l'analogie existant entre communication asynchrone et communication par boite postale permet d'appréhender très simplement le concept.

Une boite aux lettres est constituée d'une zone d'échange tampon (*buffer*) dans laquelle une tâche dite émettrice peut déposer des données. La taille de la zone d'échange est donnée par le nombre de messages maximum multiplie par la taille de chaque message.

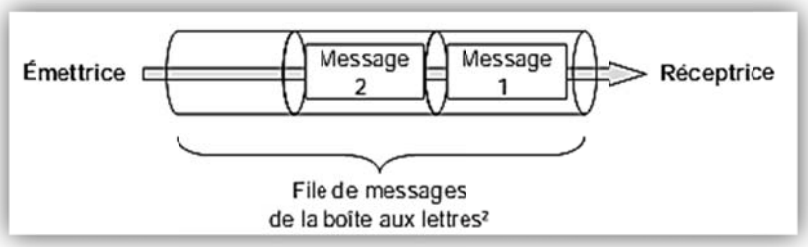


Fig.1.5. [1] Boite aux lettres FIFO

Les données de la boite aux lettres sont gérées en FIFO (i.e. premier déposé/premier retiré) ou bien avec une file FIFO par niveau de priorité. Une tâche dite réceptrice, retire les données dans l'ordre d'arrivée ou de priorité. En fonction du langage support et de l'implémentation, la priorité peut être :

- Orientée message : le message est muni d'une priorité influençant l'ordre dans lequel il va être lu :

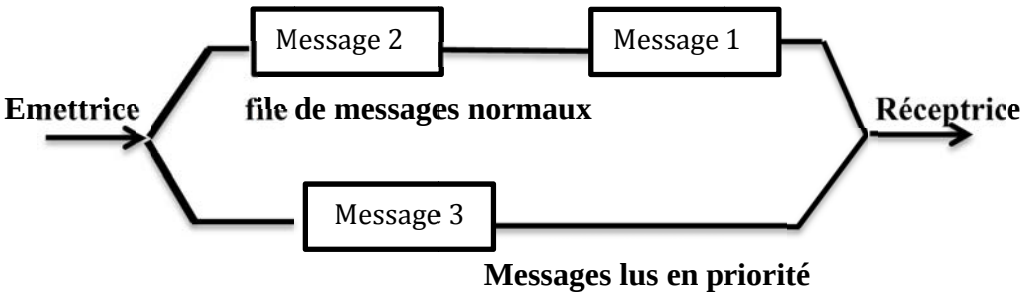


Fig.1.6. [1] Boîte aux lettres à priorités de messages.

- Orientée tâche, la priorité du message est liée à la priorité de la tâche émettrice, ce qui n'a de sens que si plusieurs tâches sont à même d'émettre de messages dans la même boite aux lettes .

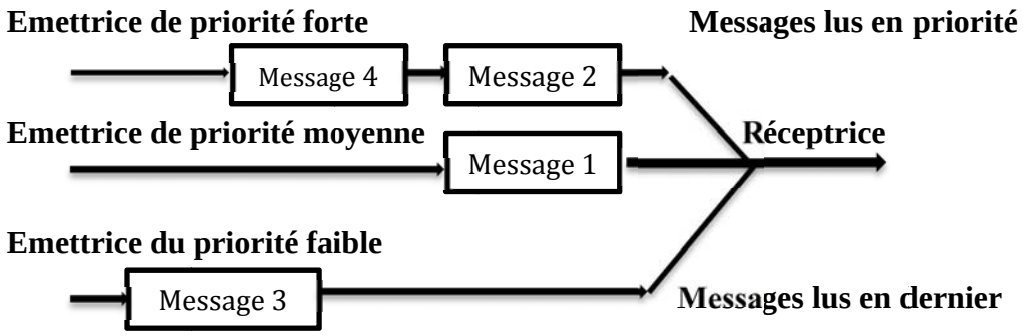


Fig.1.7. [1] Boite aux lettres à priorités de tâches émettrices.

On parle de communication asynchrone car, en règle générale, une tâche émettrice n'as pas besoin d'attendre que la tâche réceptrice soit à l'coute pour lui envoyer des données.

Cependant, les communications par boites aux lettres présentent quelques contraintes :

- Si la boite est vide, une tâche désirant recevoir des données est mise en attente dans l'état bloquée jusqu'à ce que des données aient été déposées dans la boite.
- Une boite aux lettres à une taille bornée, et lorsqu'une tâche doit déposer des données dans une boite pleine, en fonction de l'implémentation choisie, soit le message le plus ancien est écrasé par le nouveau on parle alors de boite aux lettres à écrasement (ou bien de RT FIFO), soit la tâche passe dans l'état bloquée jusqu'à ce que des données aient été retirées de la boite, on parle alors de boite aux lettres sans écrasement. Dans le cas général, le terme boite aux lettres désignera une boite sans écrasement et on précisera « avec écrasement » dans le cas contraire.

Une boite aux lettres est généralement utilisée pour une communication entre une émettrice et une réceptrice ou bien *n* émettrice et une réceptrice. Bien que cela soit possible. Il est assez rare que l'on utilise pour une boite aux lettres pour faire communiquer *n* émettrices et *m* réceptrices.

Les boites aux lettres peuvent être à attente bornée aussi bien pour l'émission que pour la réception : en cas d'attente dépassant un délai spécifié une émission une réception de message est avortée, la primitive renvoyant alors une erreur.

Une boite aux lettres peut donc être représentée par un tampon de messages qui peut être soit une file, soit un ensemble de files classées par priorités, une file d'attente de tâche désirant émettre (cas ou la file est pleine et sans écrasement) et une file d'attente de tâches en attente de message (cas ou file est vide) comme pour les messages, c'est les files d'attente peuvent être gérées en FIFO ou bien gérées sous forme de files de priorités

Boite aux lettres		
File(s) d'attente d'émettrice	Tampon file(s) de message	File d'attente de réception

Boite aux lettres	
Tampon= file(s) de message	File d'attente de réception

Tableaux.1.2. [1] Composition d'une boite aux lettres

Lorsque le langage utilisé ne propose pas le type de boite aux lettres désiré, il est possible d'en implémenter simplement à partir de sémaphores, en utilisant la technique du producteur/consommateur ou bien de manière plus élégante en utilisant des moniteurs.

- **Le tube**

Un tube (ou pipe) permet comme la boite aux lettres une communication unidirectionnel par passage de message. Cependant, La philosophie du tube repose sur le concept de flots d'octets (comme tout périphérique Unix), exactement comme dans un fichier : une tache transmettant des données à travers un tube les envoie comme un flot d'octets dans un fichier. Les données insérées les unes à la suite des autres dans le tube. Bien que pouvant tout à fait être stockées sur disque, sont destinées à être lues en FIFO par une autre tâche. Le tube utilise un concept assez proche du producteur/consommateur. Un producteur place *n* octet dans le tube à chaque fois qu'il veut transmettre un message vers le consommateur, qui prélèvera les messages *n* octets par *n* octets.

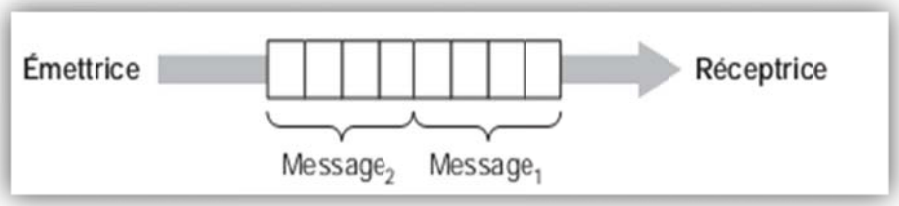


Fig.1.8. [1] Tube de communication

Le tube est bloquant en lecture, mais non bloquant en écriture : dans le cas où le tube est plain, l'écriture ne se bloque pas et renvoi une erreur au producteur. En effet théoriquement, un tube peut être stocké sur disque et donc utiliser un tampon limité seulement par la taille du disque.

Un tube peut donc être vu comme une boite aux lettres avec perte de messages dans le cas où il est plein sa structure est limitée à une file d'octet pouvant être stockée sur disque, et une file d'attente de réceptrice(s).

Tube	
Tampon= file d'octet	File d'attente de réceptrices

Tableau.1.3. [1] Composition d'un tube

Cet outil de communication étant à même d'utiliser des fichiers physiques, il n'offre aucune garantie de délai de communication

- **Le socket**

Le socket est outil de communication bidirectionnelle par message implémenté sur un protocole réseau (typiquement TCP ou UDP). Il permet à deux processeurs ou tâches de communiquer après établissement d'une connexion de type client/serveur

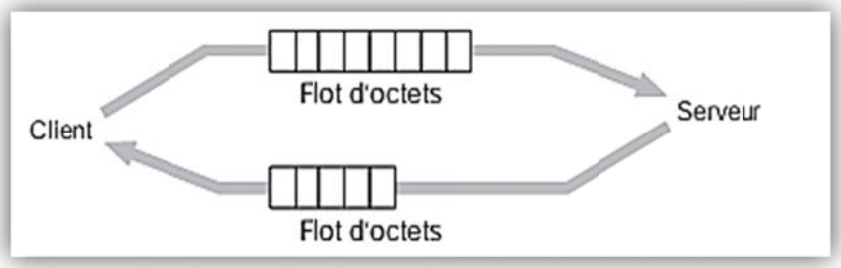


Fig.1.9. [1] Communication par socket

Les messages sont des flots d'octet, comme pour le tube chaque élément communiquant est muni de deux tampons : l'un pour à envoyer (par exemple le protocole TCP, utilisant la technique du fenêtrage TCP, peut être amené à attendre un accusé de réception avant de continuer l'émission de segments TCP supplémentaires, la bande passante étant limitée, des données peuvent tout simplement être en attente d'accès au médium de communication), et l'autre pour les messages

reçus mais non encore lus. Le socket permet une communication entre 2 tâches ou processus (socket TCP ou UDP classique), ou bien entre *n* tâches ou processus (cas de la multidiffusion implémentée sur UDP).

Un socket se caractérise eu niveau de chaque élément communiquant par deux tampons bornés : les données arrivées mais non lues, et les données en attente d'envoi, et une file d'attente par tampon pouvant contenir la tâche ou processus communiquant sur le socket.

Socket			
File d'attente D'émission	Tampon de sortie= file d'octets	Tampon d'entrée= file d'octets	File d'attente de réception

Tableau.1.4. [1] Caractérisation d'un socket au niveau d'un processus ou d'une tâche

- **Le rendez-vous**

Le rendez-vous est un mode communication synchrone bidirectionnelle : on peut le comparer à une communication téléphonique. Ce mode de communication est assez rarement mis en œuvre (il était présent en Ada 83, mais a été rendu obsolète par l'arrivée des moniteurs dans Ada 95).

Une tache, dite acceptante, attend un rendez-vous comme par analogie ou pourrait être en attente devant un téléphone. Une tâche dite appelante peut demander à effectuer un rendez-vous avec une tâche acceptante, de la même façon qu'on pourrait téléphoner à quelqu'un.

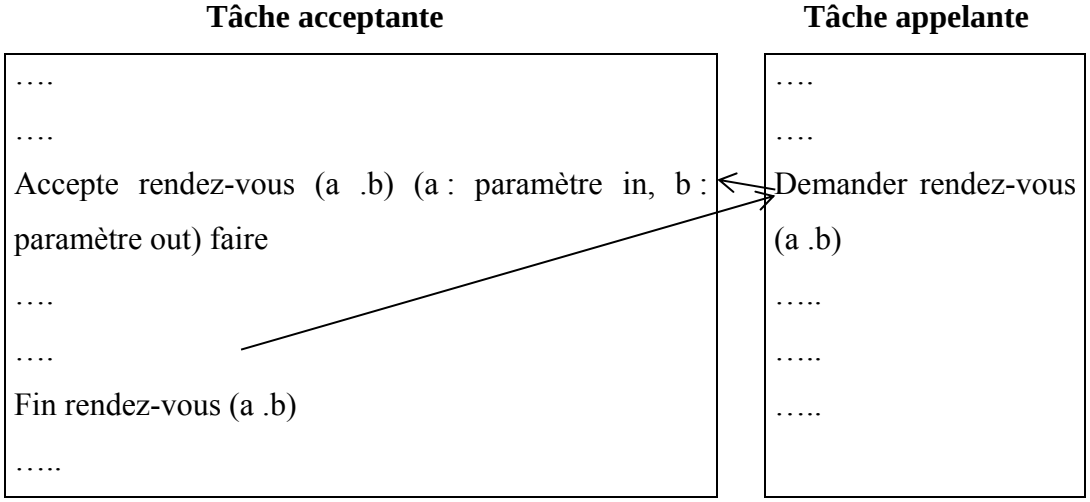


Tableau.1.5. [1] Principe du rendez-vous

Un rendez-vous n'a lieu que lorsque la tâche acceptante est sur une instruction accepte et que la tâche appelante demande l'obtention du rendez-vous : la tâche appelante peut passer des

données à la tâche acceptante, puis est bloquée jusqu'à la fin du rendez-vous exécuté par la tâche acceptante. Des données peuvent alors être passées de la tâche acceptante à la tâche appelante. Les chronogrammes d'exécution possibles d'un rendez-vous.

L'acceptation de rendez-vous, de même que la demande de rendez-vous sont des instructions bloquantes, un rendez-vous peut donc être caractérisé par un pointeur vers la tâche acceptante, sont état (en attente de rendez-vous ou non). Et une file d'attente gérée en FIFO ou bien un ensemble de file d'attente FIFO gérées par niveau de priorité des tâches appelante le rendez-vous peut se ramener à l'utilisation de deux boites aux lettres.

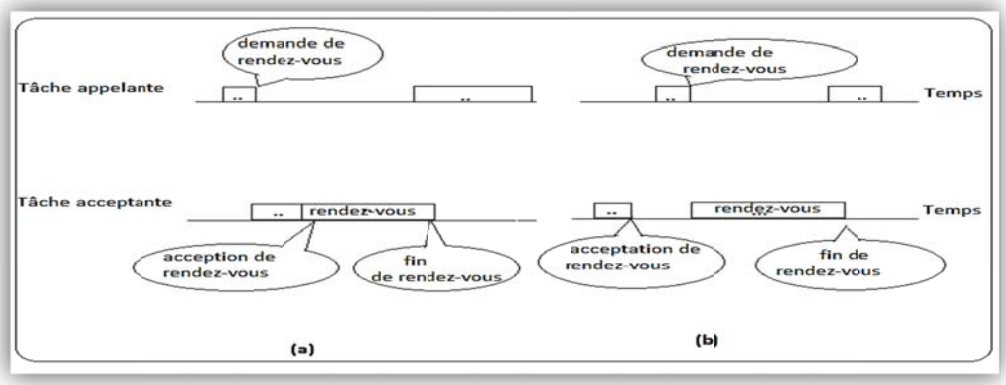


Fig.1.10. [1] Chronogramme d'exécution possible d'un rendez-vous :

(a) La demande a lieu avant l'acceptation, (b) l'acceptation a lieu avant la demande

On peut noter que dans cette implémentation, la boite aux lettres de demande de rendez-vous est nécessairement publique, car toute tache peut demander un rendez-vous, alors que la boite utilisée pour signifier la fin du rendez-vous est spécifique à chaque tache demandant le rendez-vous.

- **Le tableau noir**

Le tableau noir est conceptuellement le plus simple des moyens de communication asynchrone : il utilise une zone de mémoire commune pouvant contenir un message .l'écriture d'un message écrase le message précèdent, et la lecture est non bloquant et non destructive (une valeur déjà lue peut être relue tant qu'elle n'a pas été écrasée par une écriture.

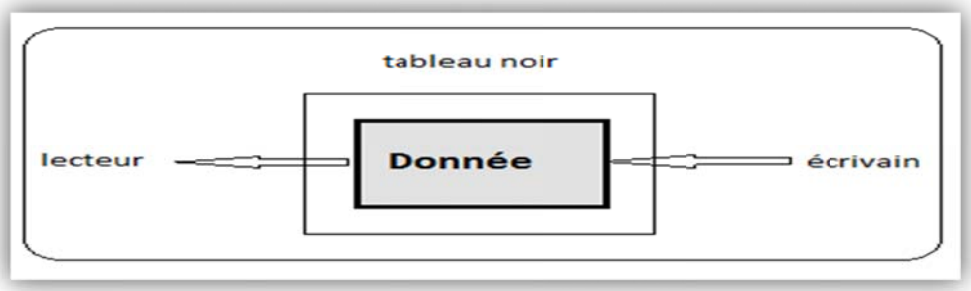


Fig.1.11. [1] Communication par tableau : La lecture est non destructive

Entre processus, il existe des mécanismes particuliers de déclaration de zone de mémoire commune (le segment mémoire des processus est protégé des autres processus). Entre taches, le mécanisme s'apparent à l'utilisation d'une variable classique. Il nécessite cependant un mécanisme de protection garantissant L'exclusion mutuelle : en effet, il peut être dommageable qu'une tache modifiant le message soit interrompue par une tache lisant le message, car ce dernier pourrait alors être incohérent .par contre il n'est pas gênant que plusieurs lectures aient lieu en même temps le mécanisme de protection d'un tableau noir s'apparente au problème de la lecture / écrivaine. L'un des problèmes qui pose souvent est le choix d'une valeur initiale pour ce type d'outille de communication

Tableau noir	
Valeur actuelle	Mécanisme(s) d'exclusion mutuelle

Tableau.1.6. [1] Caractérisation d'un tableau noir

Le mécanisme d'exclusion mutuelle utilisé est typiquement un sémaphore binaire, ou si le langage support le propose en lecture/écriture**.** Dans les deux cas si possible, il convient d'utiliser un protocole de gestion de ressources (protocole à priorité plafond ou à défaut à priorité héritée) afin d'éviter le phénomène d'inversion de priorités.

- **Bilan sur la communication par message**
- **Synchronisation**

La synchronisation consiste à assurer des propriétés de fonctionnement mutuel des tâches, ainsi l'exclusion mutuelle nécessite une synchronisation. De la même façon, si une tâche doit absolument avoir eu lieu avant une ou plusieurs autres, sans pour autant avoir à transmettre de message, la contrainte de précédence est une synchronisation. Enfin, si plusieurs tâches doivent s'attendre en un point donné de leur code, une synchronisation de type rendez-vous a lieu.

- **Exclusion mutuelle :**

Ce type de synchronisation s'agit d'empêcher les sections critiques utilisant la même ressource de se préempter mutuellement. Une variante de l'exclusion mutuelle correspond à distinguer l'accès en lecture à l'accès en écriture : dans ce cas plusieurs lectures peuvent avoir lieu simultanément. Une synchronisation d'exclusion mutuelle est donc définie par une ressource et une file d'attente pouvant être gérée en FIFO ou en files FIFO classées par priorités des tâches.

Exclusion mutuelle	
Ressources	Files d'attente

Tableau.1.7. [1] Caractérisation d'une exclusion mutuelle

Comme dans le cas du tableau noir, pour lequel la valeur (stockée à un emplacement mémoire) est une ressource critique, le mécanisme d'exclusion mutuelle utilisé est typiquement un sémaphore binaire, ou si le langage support le propose un sémaphore en lecture/écriture. Dans les deux cas, si possible, il convient d'utiliser un sémaphore de gestion de ressources (protocole à priorité plafond ou à défaut à priorité héritée) afin d'éviter le phénomène d'inversion de priorités.

- **Synchronisation n/1**

La synchronisation n/1 (n producteurs,1 tâche en attente de déclanchement) traduit une contrainte de précédence, typiquemen²t, une tâche n'étant activée que sur certaines conditions (déclanchement par une autre tâche, ou bien déclanchement suite à une interruption) est en attente de synchronisation et s'exécute sur déclanchement de celle-ci

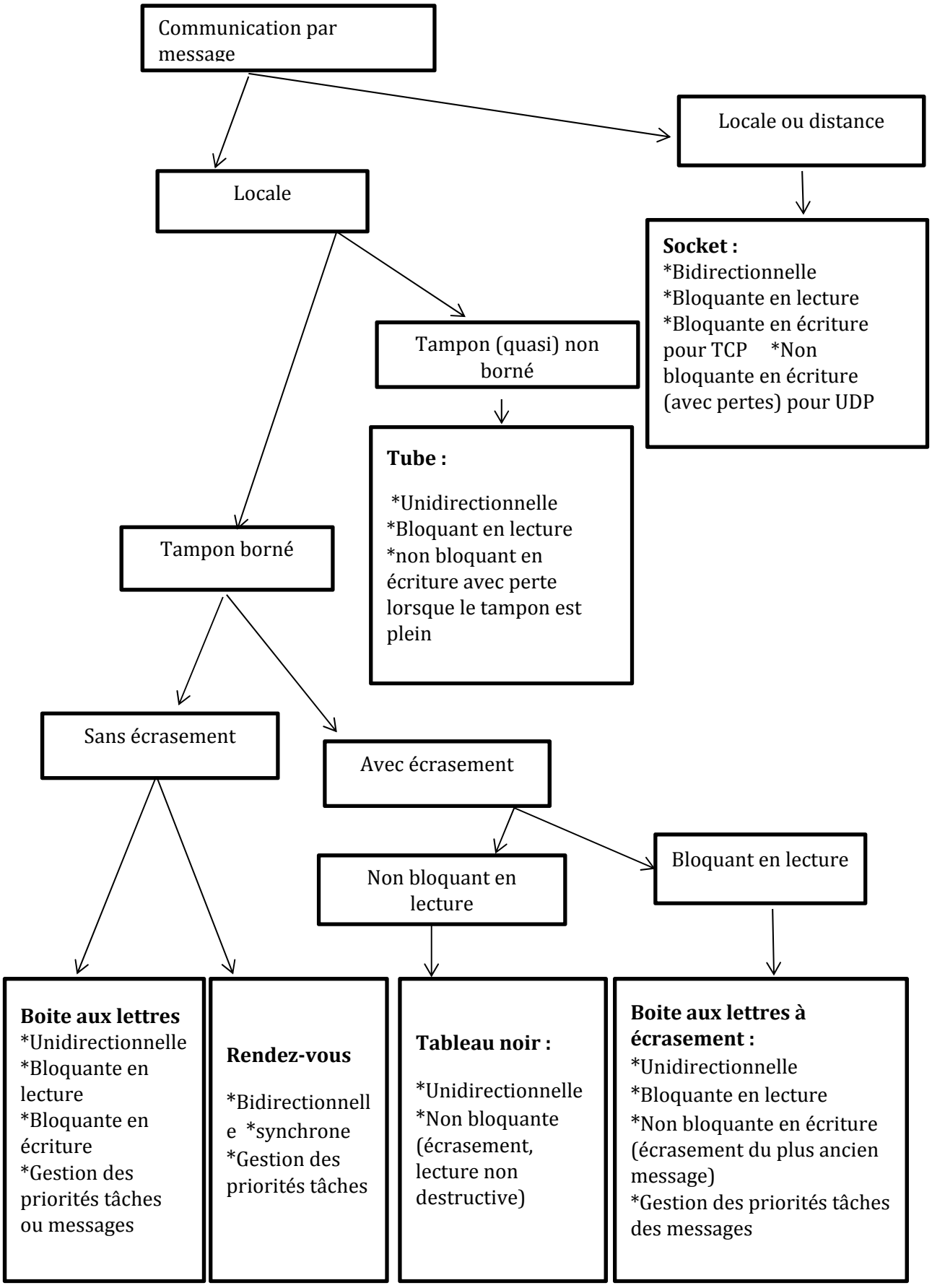


Fig.1.12. [1] Outils de communication par message

Une synchronisation peut être binaire ou à compte : lorsqu'une tâche est en attente sur une synchronisation binaire, si un déclenchement n'a pas été pris en compte et qu'un second déclenchement a lieu il écrase le dernier déclenchement et la tâche en attente ne sera déclenchée qu'une fois. Dans le cas d'une synchronisation tâche en attente de synchronisation sera déclenchée autant de fois que la synchronisation est déclenchée. Ces deux sémantiques sont similaires aux sémantiques associées aux signaux mémorisés et signaux à compte.

Une synchronisation à compte peut être caractérisée par une file d'attente (qui contiendra au plus une tâche en attente) et à un entier contenant le nombre de déclenchement non encore pris en compte.

Une synchronisation binaire peut être caractérisée par un booléen traduisant l'état de la synchronisation (déclenchée ou non), et une file d'attente contenant au plus une tâche.

Synchronisation n/1 à compte	
Nombre de déclenchements non prise en compte	File d'attente

(a)

Synchronisation n/1 binaire	
Un déclenchement en attente ? (booléen)	File d'attente

(b)

Tableaux.1.8. [1] Caractérisation d'une synchronisation n/1 (a) à compte, (b) binaire

- **Synchronisation à diffusion :**

La synchronisation à diffusion est une extension à *n* producteurs, et *m* tâches en attente de déclenchement de la synchronisation *n/1*. Si le nombre de tâches en attente est connu, cela revient à utiliser *m* synchronisations *n/1*. Cependant, dans ce cas un répartiteur de synchronisation doit être utilisé. L'ordre de prise en compte des déclenchements doit être cohérent avec les priorités des tâches déclenchées (déclenchement de la tâche la plus prioritaire d'abord), ce qui implique que soit le répartiteur connaît les priorités des tâches, soit il est plus prioritaire que toutes les tâches en attente de synchronisation

- **Rendez-vous synchronisé :**

Dans les rares cas où l'on souhaite que *n* tâches soient en même temps à un emplacement spécifique de leur exécution, on y définit l'attente d'un rendez-vous synchronisé à *n* tâches. Un rendez-vous synchronisé est caractérisé par un nombre de tâches attendues au rendez-vous, et une file d'attente de tâches arrivées au rendez-vous

I.2.3 Gestion des interruptions

Il y a deux types d'interruption : les interruptions logicielles et les interruptions matérielles.

- **Interruptions logicielles**

Les interruptions logicielles sont déclenchées par une tâche (ou l'exécutif lui-même) lors de son fonctionnement : par exemple débordement lors d'un calcul, erreur de segmentation, défaut de page sur pagination, division par zéro, etc. Dans ce cas, c'est la tâche de traiter elle-même l'interruption. Enfin certaines interruptions logicielles auront pour effet de réinitialiser le système (erreur de parité lors d'un transfert ente la mémoire centrale et le processeur, erreur du bus de données, etc.). Une interruption logicielle peut se traduire sous la forme de signal synchrone.

- **Interruption matérielle**

Les interruptions matérielles sont générées par une source externe au processeur comme par exemple un contrôleur de dispositif d'entrées /sorties (clavier, périphérique de stockage …) ou bien une horloge qui peut être programmée afin de déclencher une interruption au bout d'un certain temps ou à une certaine date. Dans ce cas il faut qu'un traitement associé au noyau air lieu, son rôle principal sera de relayer l'interruption vers la tâche de traitement adéquate.

Physiquement, une interruption matérielle est apportée par une ligne d'interruption le nombre d'interruption possible est limité par le matériel, ainsi sur une architecture de type x86, il y a 16 interruption matérielle, les interruptions matérielle sont souvent appelées **IRQ** (*interrupt ReQuest*)

I.2.4 Gestion du temps

La gestion du temps sur les systèmes d'exploitation généralistes se base sur une horloge de faible résolution (de l'ordre de la milliseconde) ou bien utilise de façon grossière une horloge de haute résolution.

Seules les interruptions correspondant à des instants programmés ont lieu, permettant ainsi une extrême finisse temporelle, sans nuire aux performances du système.

Les exécutifs et systèmes d'exploitation temps réel se basent donc sur les horloges pour proposer l'une des deux méthodes suivantes de gestion de temps :

- **Noyau dirigé par le temps :** seule la notion de tick permet de gérer le temps, les tâches ou processeur ne peuvent être réveillés que sur des ticks, c'est le cas par exemple des exécutifs VxWorks et RTEMS.
- **Noyau dirigé par les événements :** les horloges sont programmables à leur granularité la plus fine, et il est possible de réveiller une tâche ou un processeur d'une façon très fine, c'est le cas par exemple des exécutifs et systèmes d'exploitation de type POSIX et OSEK/VDX.

Les noyaux dirigés par les événements sont strictement plus puissant que les exécutifs dirigés par le temps : il est possible sur les premiers d'obtenir un fonctionnement similaire aux derniers, mais l'inverse n'est pas vrai. Les noyaux dirigés par les événements peuvent atteindre une résolution d'horloge de l'ordre de la microseconde, voire moins, alors que sur un noyau dirigé par le temps il est quasi impossible sans dispositif matériel additionnel d'atteindre une résolution inférieure à quelque centaine de microsecondes (de plus, à ce grain temporel, le surcoût processeur dû au noyau est prohibitif).

I.3. Principales normes temps réel

Ce paragraphe présente deux normes temps réel, POSIX et OSEK, les outils et concepts définis dans ces normes définissent les services rendus par un exécutifs ou un système d'exploitation compatible avec ses normes, appliqués au langages C. Cependant, le langage utilisé par un programmateur tirant parti de ces services peut être un autre langage que le C (typiquement Ada), il existe d'autres normes temps réel, comme ITRON/BTRON, norme principalement développée au Japon.

I.3.1 La norme POSIX

POSIX (Portable Operating System Interface) est un standard initialement normalisé en 1988 par l'IEEE sous la norme de P1003 et par l'ISO/IEC sous le nom ISOC/IEC-9945. Le but de cette norme est de normaliser l'accès aux services offerts à différents niveaux par les systèmes d'exploitation Unix, de sorte à assurer le plus de portabilité possible aux programmes.

I.3.2 La norme OSEK/VDX

La norme OSEK/VDX est née en 1995 dans la fusion d'un consortium de constructeurs d'automobiles allemands (OSEK est l'acronyme *de Offene Systeme und deren Schnittstellen Fur die Elektronik im Kraftfahzeug)* et d'un consortium de constructeurs d'automobiles français (VDX est l'acronyme de Vehicle Distributed eXecutive).

Le but de la norme OSEK/VDX est de définir un exécutif adapté au contrôle embarqué dans les systèmes automobiles, composés de plusieurs unités de contrôle distribuées sur un ou plusieurs réseaux de terrain. La normalisation a pour effet de diminuer les coûts d'intégration de composants entre les constructeurs automobiles et les instrumentiers.

Bien qu'initialement conçue pour le domaine automobile, cette norme est très bien adaptée à d'autres domaines, comme les applications de contrôle-commande, moins utilisée que POSIX dans le cas général, elle a l'avantage d'être plus facile d'accès, de par sa spécialité.

I.4 Programmation des systèmes multitâches

I.4.1 Programmation C, Ada et LabVIEW

I.4.1. a Présentation générale des trois langages

Nous avons choisi de présenter l'implémentation de systèmes de contrôle-commande à travers trois langages de programmation :

- **Le langage C**, car parmi les représentants des langages impératifs, c'est langage le plus répandu pour l'implémentation de systèmes de contrôle-commande. De plus, il est le langage de référence pour les systèmes d'exploitation, et est à la base de différentes normes comme POSIK et OSEK/VDX ;
- **Le langage Ada,** recommandé dans les systèmes à haut niveau de sûreté
- **Le langage LabVIEW,** langage graphique flots de données est quant à lui très utilisé dans le contrôle de procédés industriels.

➢ **Le langage C**

Le langage C a vu le jour à la fin des années 60. Son rôle initial était de permettre la portabilité de la majeure partie du code d'un système d'exploitation d'une architecture matérielle à une autre. En 1973, il est utilisé avec sucés pour écrire un système d'exploitation, c'est aujourd'hui encore l'un des langages de programmation les plus utilisés.

Le langage C, est un langage de type impératif, il est faiblement typé à compilation séparée.

Le langage C est sensible à la casse (majuscule/minuscule).

Le langage C dispose de bibliothèque exceptionnelle de composant logiciels, et la quasi-totalité des langages de programmation peuvent s'interfacer avec le langage C .

Le langage C, est un langage de programmation normalisé, aujourd'hui, des centaines de compilateurs C et environnements de développement associés existant, aussi bien en logiciel libre (le plus connu étant GNU C Compiler, GCC) qu'en logiciel commercial.

➢ **Le langage Ada**

L'idée de langage Ada naît au milieu des années 70 sous l'impulsion du département de la défense américain.

Il est nativement multitâche: en langage Ada, une tâche est un type que l'on peut instancier. Ada est insensible à la casse (majuscules/minuscules) .

Sa première version a été normalisée en 1983 et la seconde version, Ada 95, permet la programmation objet et surtout toute forme de communication asynchrone et synchronisation grâce à l'introduction de moniteurs.

Ada est un langage normalisé, plusieurs compilateurs Ada existent sur la plupart des plateformes, le plus connu des compilateurs libres est GNAT.

- **Le langage LabVIEW**

Le langage labVIEW est un des rares représentants des langages flots de données. C'est un langage graphique, typé, modulaire, et en tant que langage flots de données, il est naturellement parallèle.

Il est relativement récent, puisqu'il a fait son apparition sur Macintosh au milieu des années 80, Initialement à la programmation d'instruments virtuels utilisant des cartes d'acquisition de la société National instruments.

LabVIEW est représenté par une interface graphique nommée face avant et une description du fonctionnement interne sous forme de flots de données nommées diagramme Depuis, le langage LabVIEW a évolué et s »est élargi au fil des versions successives, jusqu'à devenir un langage complet de programmation, plaçant LabVIEW parmi les langages les plus agréables à utiliser pour les applications de contrôle-commande. [1]

Chapitre : II

ETUDE AVENCEE DES SYSTEMES TEMPS REEL

II. Introduction

II.1.1 Présentation générale de l'ordonnancement

Les méthodes d'ordonnancement sont indépendantes des caractéristiques temporelles intrinsèques des tâches. En effet, l'affectation des priorités à un ensemble de tâches se fait de façon non formelle.

Les algorithmes basés sur les priorités peuvent être à priorité fixes ou variables. Dans le cas où les priorités sont variables, l'ordonnancement met à jour les priorités à chaque réveil des tâches ou aux instants des appels de primitives, ou à chaque top d'horloge. Il s'appuie ensuite sur le répartiteur (dispatcher) qui choisit la tâche prête la plus prioritaire et lui octroie le processeur

Les algorithmes à priorités sont appelés algorithmes d'ordonnancement en ligne car il se base sur l'état instantané du système pour prendre une décision. Les ordonnancements basés sur les séquences préétablies s'appellent des algorithmes hors-lignes, dans ce cas la séquence d'ordonnancement est construite à partir d'une vision complète du système, puis exécutée par un séquenceur.

II.1.2 Algorithme d'ordonnancement

Un algorithme d'ordonnancement étant défini comme un algorithme capable de donner une description (séquence) du travail à effectuer par le ou les processeurs une séquence est dire valide si les échéances des tâches sont respectées.

Un algorithme est dit fiable pour une configuration de tâches s'il produit une séquence valide sur une durée infinie quelles que soient les valeurs des premières dates et de déclenchement des différentes tâches. Une configuration est dite ordonnançable s'il existe au moins un algorithme fiable.

Dans un contexte de tâches et algorithme d'ordonnancement (affectation de priorités), nous allons qualifier l'algorithme d'ordonnancement étudié selon deux aspects :

- **Optimalité :** si la configuration des tâches est ordonnançable dans cette catégorie d'algorithmes, alors elle le sera avec l'algorithme étudié
- **Ordonnançabilité :** la capacité à pouvoir prévoir l'ordonnancement de la configuration de tâches en se basant sur des conditions nécessaires et/ou suffisante ou des simulations de l'exécution.

II.1.3 Temps discret

Nous allons utiliser un temps-discret, c'est-à-dire que le temps est considéré comme une valeur entière qui évolue par incrément de 1. De même, les paramètres des tâches vont être affichés sous forme de valeur entière sans unité temporelle précisée. Dabs les applications industrielles, ces grandeurs sont exprimées en fraction de milliseconde, voire en milliseconde la précision dépond de l'horloge du processeur. Pour faire la traduction de l'application réelle (grandeurs réelles) vers l'application formalisée (grandeurs entières), nous pouvons soit procéder à un arrondi au plus proche entier, soit considérer un quantum équivalent au temps unité de base, par exemple 25µs correspond à une unité de temps dans l'analyse formelle de l'application. Ce temps est aussi fonction de la granularité temporelle du noyau temps réel. De plus les temps affichés dans une application sont en général considérés comme des temps relatifs au lancement de l'application.

II.2 Modélisations des tâches :

Dans un système temps réel, les contraintes de temps découlent de la dynamique du procédé contrôlé. Il existe différents types de contraintes de temps, qui seront affinés dans le chapitre portant sur l'ordonnancement :

- **Contraintes de bout en bout :** le procédé doit effectuer une ou des réactions sur le procédé (typiquement via des actionneurs) en un temps contraint. Typiquement, ces contraintes influent sur les contraintes de tâches faisant partie d'une chaîne de l'acquisition à la commande.
- **Contraintes de non réentrance :** dans un système réel, les tâches effectuent un traitement cyclique (soit périodique, soit répété à chaque événement attendu). Chaque itération de cette « boucle » s'appelle une instance de tâches. Une tâche ne pouvant pas être réentrante, il faut typiquement s'assurer que chaque instance d'une tâche puisse se terminer avant le prochain événement déclencheur de la tâche, ou avant sa prochaine période.
- **Contraintes de régularité** (ou **gique**) lorsqu'une tâche fait de l'échantillonnage, elle est périodique et doit être la plus régulière possible. on retrouve le même type de contraintes lorsque la tâche doit délivrer un signal continuellement modifié.

Tous ces types de contraintes sur le système se traduisent en termes de contraintes temporelles individuelles sur les tâches. Chaque tâche est donc caractérisée par des contraintes temporelles, la difficulté réside alors dans l'étude du comportement temporel du système et du respect des contraintes définies sur chacune des tâches. L'algorithme d'ordonnancement choisi va satisfaire certaines contraintes de temps et être un compromis pour d'autres caractéristiques temporelles analysées *a posteriori.*

II 2.2.1 Modélisation formelle des tâches indépendantes

Afin de pouvoir analyser de manière rigoureuse l'ordonnançabilité d'une configuration de tâches, l'optimalité d'un algorithme d'ordonnancement ou la séquence d'exécution d'une application multitâche, il est nécessaire d'avoir un modèle mathématique des tâches. ce modèle doit permettre de prendre en compte toutes les caractéristiques opérationnelles et temporelles d'une tâche d'une application quelconque. Pour cela nous allons considérer successivement les différents cas :

- Tâches périodiques
- Tâches apériodiques
- Tâches avec contraintes de précédence
- Tâches avec partage de ressource critiques

La combinaison de ces différents modèles de tâches permet d'analyser une application multitâche réelle industrielle quelconque

Le premier paramètre commun à l'ensemble des modèles de tâches est la Durée **Ci** d'une tâche **Ɔi.** la durée de la tâche est directement liée au code de la tâche. On évalue généralement la durée d'exécution pire cas **Cmax** et une durée minimale **Cmin .**

L'évaluation de cette durée de la tâche peut être effectuée de deux manières différentes. Soit le code est analysé instruction par instruction, et l'ensemble des durées de ces instructions est additionné (ensemble des durées d'exécution du processeur référencées pour un processeur et une fréquence quartz données). Mais dans ces deux cas, une évaluation tés précise de cette durée d'exécution est très difficile pour essentiellement trois raisons :

– Une analyse exhaustive de tous les chemins d'exécution du code est parfois impossible (jeux de tests complexes, nombre de combinaisons très élevé)

– Les capacités d'exécution des processeurs pour améliorer leur efficacité (mémoire cache à plusieurs niveaux, pipline multiple, etc.) vont conduire à une incertitude sur la durée d'exécution qui dépend fortement du contexte précédent du processeur

– L'occurrence d'interruptions pendant l'exécution d'une tâche qui oblige le processeur à une prise en compte minimale.

Pour cette étude, nous considérerons une valeur de la durée d'exécution pire cas d'une tâche sur l'ordonnancement d'une configuration.

❖ **Modélisation des tâches périodiques :**

Comme nous l'avons vu dans les exemples précédents, ces tâches correspondent par exemple à des tâches de scrutation de capteurs pour effectuer des mesures régulières de grandeurs physiques, la modélisation des tâches périodiques va reposer sur trois paramètres temporels :

* **r0 :** date de réveil de la tâche, c'est-à-dire la première date à laquelle la tâche demande le processeur ;

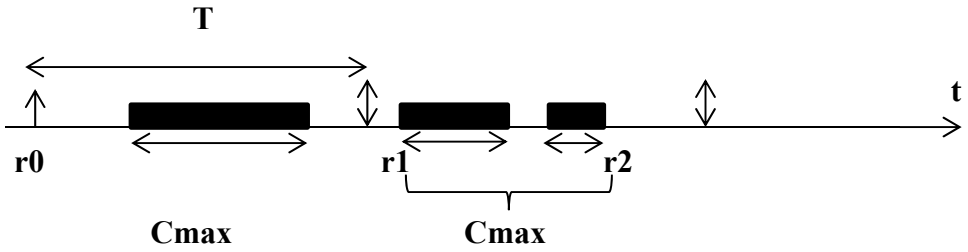
* **C = Cmax :** durée d'exécution maximale définie avec les restrictions exposées précédemment

* **T :** période d'exécution, c'est-à-dire la fréquence de renouvellement de la demande d'exécution de la tâche.

La date de réveil **rk** de la **k**$^{\text{ième}}$ instance ou occurrence d'une tâche est donc définie par :

$$rk = r0 + kT \quad \text{(II.1)}$$

La figure ci-dessous présente l'exécution de deux occurrences d'une tâche périodiques à échéance sur requête dans un diagramme de Gantt. Nous pouvons noter que la tâche s'exécute de façon complète dans la première occurrence, c'est-à-dire que, lorsqu'elle obtient le processeur, elle garde pendant toute sa durée d'exécution **C = Cmax.** En revanche, lors de la deuxième occurrence, la tâche est préemptée une fois lors de son exécution et son exécution s'effectue alors en deux fois.



FigII.1 [1].1 Représentation de l'exécution d'une tâche périodique à échéance sur requête.

❖ **Modélisation des tâches apériodiques :**

En ce qui concerne les tâches apériodiques, le seul paramètre connu est la durée d'exécution **C** de la tâche. La date de réveil ou demande processeur est aléatoire car elle dépond du contexte d'évolution du procédé et ne peut donc pas être connue *a priori.*

Nous pouvons donc représenter son exécution dans le diagramme de Gantt de la figure ci-dessous, les deux dates de réveil **r** et **r'** ont été choisies aléatoirement. De la même manière que pour les tâches périodiques les tâches apériodiques peuvent être préemptées au cours de leurs exécutions et donc d'exécution en plusieurs fois lors d'une instance.

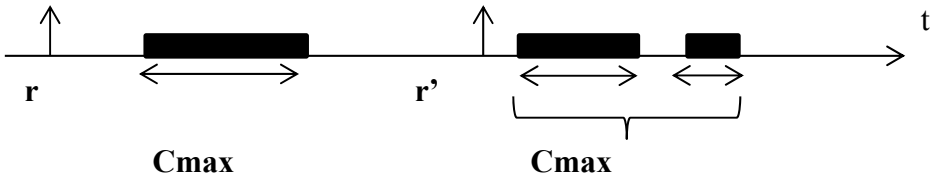


Fig.II.2 [1] Représentation de l'exécution d'une tâche apériodique.

Comme nous nous sommes placés dans un environnement d'exécution stricte, la tâche apériodique, appelée apériodique stricte ou sporadique, doit posséder un délai critique **D.** nous appelons ces tâches apériodiques strictes des tâches sporadiques. L'exécution de ce type de tâches sporadiques est représentée sur la figure ci-dessous. Ce délai critique conduit à des dates d'échéance stricte pour chaque instance d'exécution, soit:

$$\boldsymbol{d = r + D \quad ou\, d' = r' + D} \qquad \textbf{(II.2)}$$

Afin de limiter au maximum les indéterminismes d'exécution et de pouvoir faire une analyse de l'ordonnançabilité de configuration de tâches incluant des tâches apériodiques, nous supposons que les tâches apériodiques strictes possèdent un délai minimum **Δmin** entre deux occurrences ou instances successives, soient **r** et **r'** deux dates de réveil successives d'une tâche sporadiques, nous avons alors la relation suivante : $\mathbf{r' - r \leq \Delta}$

Il est impératif d'avoir une tâche terminée avant une nouvelle demande d'exécution, cela conduit aux inégalités suivantes : $\mathbf{0 \leq C \leq D \leq \Delta min}$

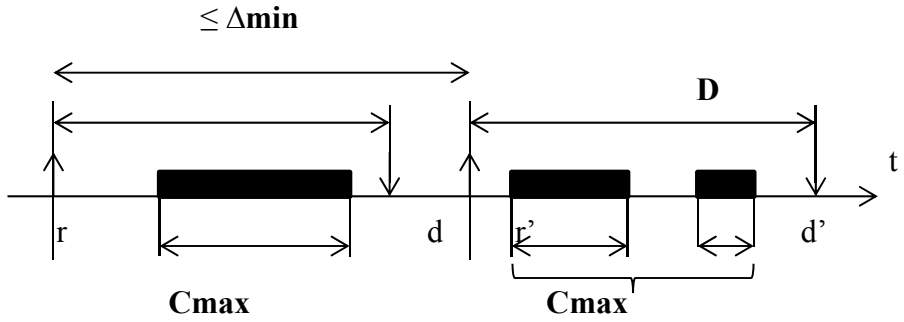


Fig.II.3 [1] Représentation de l'exécution d'une tâche apériodique avec une échéance stricte ou tâche sporadique.

II.2.2 Autres paramètres temporels des tâches

Nous allons nous intéresser à des paramètres qui sont utiles pour caractériser une ou plusieurs tâches dans une séquence d'exécution. Nous distinguons les paramètres statiques qui ne varient pas en fonction du temps d'avancement dans l'exécution de la tâche et des paramètres dynamiques qui dépendent de l'instant où ils sont calculés. Certains de ces paramètres peuvent servir de base pour un algorithme d'ordonnancement.

II.2.2.a Paramètres statiques des tâches

Nous pouvons définir deux paramètres complémentaires du temps d'exécution par rapport à la période ou à l'échéance de la tâche :

- La **laxité L :** C'est-à-dire le temps restant entre la fin d'exécution de la tâche et son échéance. Lors d'une exécution de la tâche au plus tôt, exécution immédiate et sans préemption après la date de réveil, la laxité maximale L max représente tout le temps processeur restant et s'exprime par :

$$L \leq D - C \quad \text{et} \quad Lmax = D - C$$

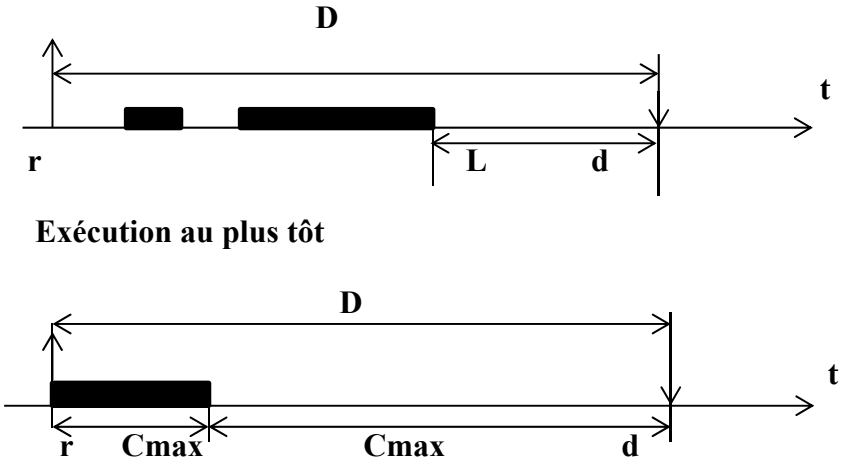


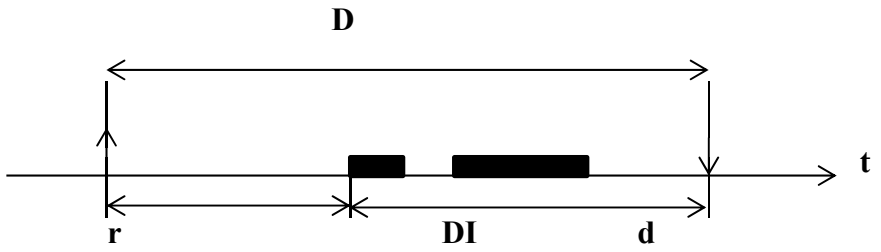
Fig.II.3 [1] Illustration du paramètre « laxité » dans le cas général et dans Le cas d'une exécution au plus tôt

- Le **délai de latence Dl :** C'est-à-dire le temps avant le début d'exécution de la tâche, lors d'une exécution de la tâche au plus tard. Exécution retardée au maximum et sans préemption, le délai de latence maximum **Dlmax** est identique à la laxité maximale et s'exprime par :

$$Dl \leq D - C \text{ et } Dlmax = D - C = Lmax \quad \textbf{(II.3)}$$

Le complément de ces deux paramètres statiques, il est habituel de définir les instants d début d'exécution et de fin d'exécution de la K ième instance d'une tâche, soit la figure ci-dessous :

- Début d'exécution de la K^{ième} instance de la tâche **ζi : Si,k ;**
- Fin d'exécution de la K^{ième} instance de la tâche **ζi : ei,k**



Exécution au plus tard

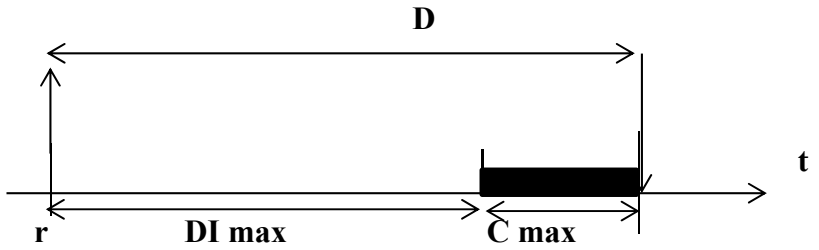


Fig.II.4 [1] Illustration du paramètre « délai de latence » dans le cas général et dans le cas d'une exécution au plus tard

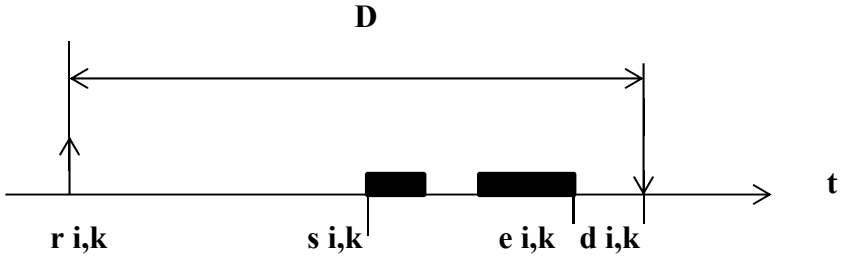


Fig.II.5 [1] Illustration des paramètres « début et fin d'exécution » d'une tâche ᶜi.

- **Temps de réponse :** de la K^{ième} instance de la tâche **ᶜi** définie par la figure si dessous.

$$Tri,k = ei,k - ri,k \qquad \textbf{(II.4)}$$

-Temps de réponse maximum de la **ᶜi :**

$$TRi = max\, k\,\{TRi,k\} \qquad (II.5)$$

-Temps de réponse minimum de la tâche **ᶜi :**

$$TRimin = mink\,\{TRi,k\} \qquad (II.6)$$

-Temps de réponse moyen de la tâche **ᶜi :**

$$TRi, moy = \Sigma k \{TR\ i, k\}/(K+1) \ldots\ldots\ldots \textbf{(II.7)}$$

- **Gigue** (régularité d'exécution) entre deux instances consécutives de la tâche **ζi.** Définie par la relation :

$$gi, k = [(S\ i, k+1 - Si, k) - Ti] / Ti \ldots \textbf{(II.8)}$$

*Gigue maximale de la tâche **ζi :**

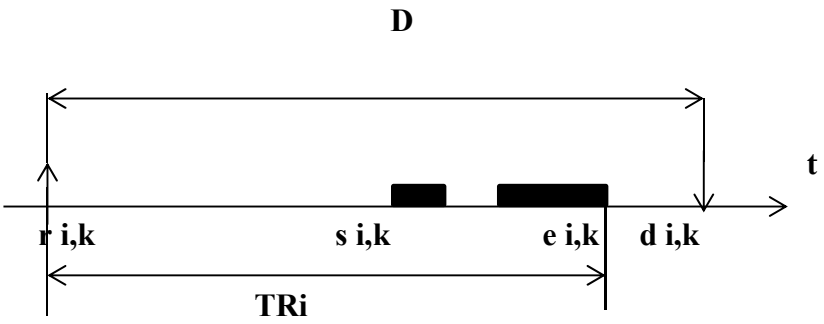
$$Gi = max\ k \{gi, k\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(II. 9)}$$

*Gigue minimale de la tâche **ζi :**

$$Gi, min = min\ k \{gi, k\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(II. 10)}$$

*Gigue moyenne de la tâche **ζi :**

$$Gi, moy = \Sigma k \{g\ i, k\}/(k+1) \ldots\ldots\ldots\ldots \textbf{(II.11)}$$



FigII.6 [1] Temps de réponse de la K^{ième} exécution d'une tâche ζi

II.2.2. b Paramètres dynamiques des tâches

Au cours de l'exécution de l'instance d'une tâche, certains paramètres, qui sont fonction de l'instant **t,** peuvent être utiles à l'ordonnanceur. Etant donnée une tâche avec une durée d'exécution **C,** nous pouvons définir le temps d'exécution restant **C(t)** fonction du temps processeur déjà alloué à la tâche **C exécutée** au cours de cette instance. Ainsi, nous pouvons identifier les trois paramètres suivants :

- **Le temps d'exécution restant C(t)** :

$$C(t) = C - C\ exécutée \qquad \textbf{(II. 12)}$$

- **Le délai critique dynamique D(t)**, c'est-à-dire le temps restant avant la prochaine échéance, soit : **D (t) = d – t**
- **la laxité dynamique L (t),** c'est-à-dire le temps avant le début d'exécution de la tâche, soit :

$$L(t) = D(t) - C(t) = d - t - C(t) \qquad \textbf{(II. 13)}$$

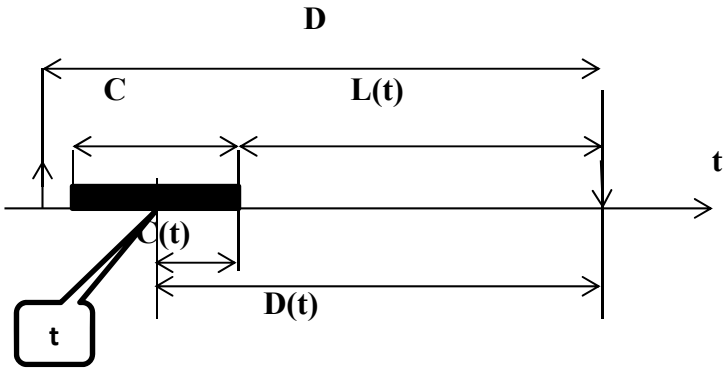


Fig.II.7 [1] Paramètres dynamiques de l'instant d'une tâche en cours d'exécution

II.2.3 Modélisation des tâches dépendantes

Comme nous l'avons déjà vu, la dépendance des tâches entre elles peut se traduire de deux manières : une relation de précédence au niveau de l'ordre d'exécution de deux ou plusieurs tâches ou le partage de ressources critiques. Il est donc nécessaire de pouvoir modéliser ces deux dépendances entre tâches.

II.2.3. a Modélisation de la précédence ente tâches

De façon générale, cette précédence entre tâches est mise en œuvre à l'aide de primitives de synchronisations ou communication intégrées dans le code des deux tâches. Prenons un exemple simple et considérons que deux tâches **ƈ1** et **ƈ2** ne sont pas atomiques. C'est-à-dire que ces primitives sont situées à l'intérieur du code de chacune des tâches.

II.2.3. b Modélisation du partage de ressources critiques entre tâches

Une tâche **ƈi,** de durée totale **Ci,** qui utilise une ressource critique **R** possède dans son code une zone protégée, section critique**,** pendant laquelle elle accède à cette ressource. Cette section critique **SC**i est protégée par les primitives permettant de gérer l'exclusion mutuelle comme un sémaphore. Par conséquence, en termes de temps, l'exécution de cette tâche peut être décrite par 3 valeurs :

***C i,α :** temps avant la section critique ;

***C i,β :** durée de la section critique (ressource utilisée) ;

***C i,γ :** temps après la section critique ;

Ces trois valeurs doivent satisfaire l'égalité suivante :

$$Ci = Ci,\alpha + Ci,\beta + Ci\gamma \quad (\mathbf{II.14})$$

II.3. Ordonnancement des tâches indépendantes périodiques

II.3.1. Algorithmes d'ordonnancement à priorités fixes

II.3.1. a Algorithmes d'ordonnancement « rate monotonic »

Dans un contexte de tâches indépendantes, périodiques, à échéance sur requête et à départ simultané, nous allons effectuer une affectation des priorités aux tâches selon la période : plus la période de la tâche est petite, plus la priorité de la tâche est grande. La tâche conserve cette priorité pendant toute son exécution. Cette règle d'affectation des priorités est appelée l'algorithme d'ordonnancement « Rate Monotonic ou RM »

II.3.1. b Algorithme d'ordonnancement « Deadline Monotonic »

S'il existe au moins une tâche qui n'est pas à échéance sur requête dans la configuration des tâches, alors nous allons utiliser un algorithme d'affectation des priorités basé sur les délais critiques des tâches au lieu des périodes. Cet algorithme d'ordonnancement, appelé « Deadline Monotonic ou DM » (ou Inverse Deadline ou ID), affecte la priorité la plus grande de la tâche dont le délai critique est le plus petit.

II.3.2 Algorithmes d'ordonnancement à sororités variables

Nous avons jusqu'à présent que la priorité affectée à une tâche restait constante pendant toute la durée de l'application. Nous allons nous intéresser à une autre catégorie d'algorithme d'ordonnancement pour laquelle la priorité des tâches caractéristique temporelle dynamique de la tâche.

II.3.2.a Algorithme d'ordonnancement « Earliest Deadline First »

Dans le cas de l'algorithme « Earliest Deadline First ou EDF », la priorité des tâches est variable au cours de leur exécution et fonction de la prochaine échéance. Pour une instance **K** d'une tâche **i** la priorité est liée à la prochaine échéance **d i,k** de cette tâche. A un instant **t**, la priorité peut être calculée à partir du délai critique dynamique **Di (t)** qui s'exprime sous la forme :

$$D(t) = di,k - t = ri,k + Di - t \qquad \textbf{(II.15)}$$

II.3.2.b Algorithme d'ordonnancement « Minimum Laxity »

Dans le cas d'algorithme « Mimum Laxity ou ML » (ou « Lest Laxity « LL). La priorité des tâches est variable au cours de leur exécution et fonction de la laxité dynamique. Pour une instance **K** d'une tâche **i,** la priorité est liée à la laxité dynamique **Li,k (t)** de cette tâche. A un instant **t**, la priorité peut être calculée à partir de :

$$Li(t) = di,k - Ci(t) - t = ri,k + Di - Ci(t) - t$$

$$Ou \quad Li(t) = di,k - Ci - t = ri,k + Di - Ci - t \quad (II.16)$$

II.4 Ordonnancement des tâches indépendantes apériodiques

L'ordonnancement des tâches apériodiques est traité dans le cas des tâches périodiques à contraintes strictes et de tâches apériodiques à contraintes relatives ou à contraintes strictes (tâches sporadiques). Nous supposons que l'ordonnancement des tâches périodiques se fait suivant les algorithmes RM, DM ou EDF pour répondre à la demande d'événements déclenchant des tâches apériodiques, nous pouvons considérer trois méthodes :

- **Traitement en arrière blanc :** les tâches apériodiques sont traitées pendant les temps d'oisiveté du processeur.
- **Traitement par utilisation d'un serveur périodique des tâches apériodiques en environnement à priorité fixe** pour les tâches périodiques (RM) : en plus de toutes les tâches apériodiques, on insère une tâche appelée **serveur**, qui est destinée à traiter les tâches apériodiques et qui possède différentes caractéristiques selon le modèle du serveur :
  - Serveur à scrutation
  - Serveur ajournable
  - Serveur à échange de priorité
  - Serveur sporadique
  - Serveur à vol de temps creux
  - Serveur à échange de priorité étendue
- **Traitement par utilisation d'un serveur périodique des tâches apériodiques en environnement à priorité variable** pour les tâches périodiques (EDF) : en plus de toutes les tâches périodiques on insère une tâche appelée **serveur,** qui est destinée à traiter les tâches apériodiques et qui possède différentes caractéristiques :
  - Serveur dynamique à échange de priorité
  - Serveur dynamique sporadique
  - Serveur à largeur de bande maximale
  - Serveur Earliest Deadline Last (EDL)
  - Serveur à echange de priorité amélioré

II.5 Ordonnancement des tâches périodiques dépendantes

II.5.1Ordonnancement des tâches avec contraintes de précédence

II.5.1.a Définition générale de la précédence

Des tâches peuvent être liées par des contraintes de précédence lorsqu'elles ont les relations de synchronisation (sémaphores, évènements) ou de **communication** (boite aux lettres).

On appelle une contrainte de précédence ente la tâche **τi** et la tâche **τj** le cas où **τi** précède **τj** si **τj** doit attendre la fin d'exécution de **τi** pour commencer sa propre exécution. Nous supposons dans cette section que les tâches ont une forme atomique, c'est-à-dire attente de synchronisation ou de communication en début de tâche et envoi d'un évènement synchronisation ou de communication en fin de tâche. L'expression des contraintes de précédence (ordre partiel sur l'ensemble des tâches) peut se faire par exemple sous la forme d'un graphe comme celui de la figure ci-dessous. Ainsi les six tâches de la configuration (**τ1, τ2, τ3, τ4, τ5, τ6**) sont liées par des relations de précédence décrites par de graphe

Nous pouvons remarquer que nous avons jusqu'à présent noter seulement les précédences dites simples. Pour être complet, il est nécessaire de distinguer les deux cas de précédence :

- **Contraintes de précédence simples :** une contrainte de précédence entre la tâches **τi** et la tâche **τj** ou **τi** précède **τj** si **τj** doit attendre la fin d'exécution de **τi** pour commencer sa propre exécution. Dans ce cas, nous faisons l'hypothèse suivante :
  « Deux tâches périodiques liées par une contrainte de précédence simple sont de même période »

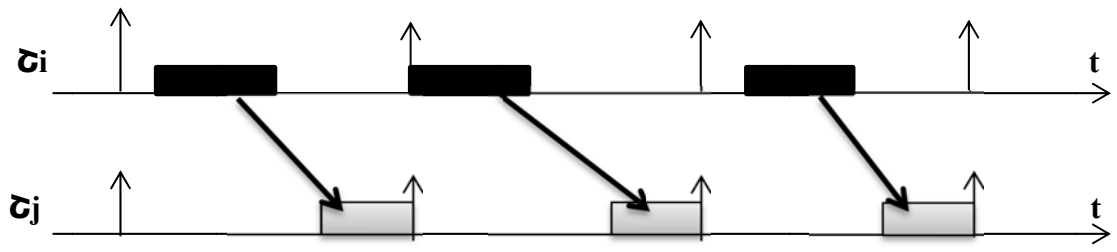


Fig.II.8 [1] Visualisation de la contrainte de précédence entre deux tâches.

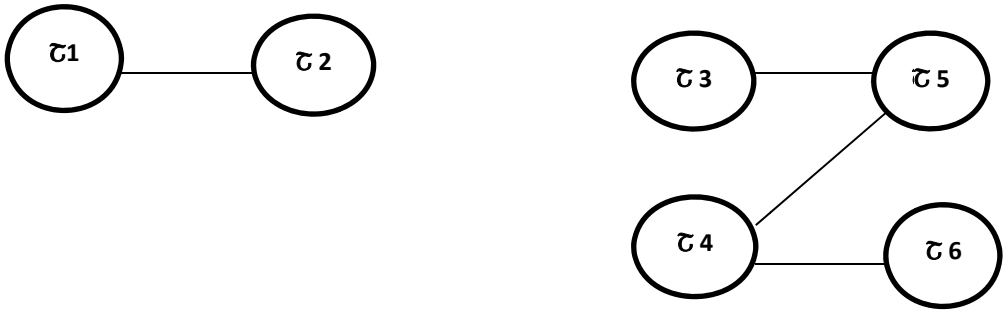


Fig.II.9 [1] Graphe de présentation des contraintes de précédence ente les tâches.

- **Contraintes de précédence généralisée :** le nombre des exécutions de deux tâches liées par une telle relation n'est pas nécessairement le même. C'est le cas lorsque **τi** peut s'exécuter **n** fois avent l'exécution de **τj** ou lorsque **τi** s'exécute une seule fois avant **n** exécutions de la tâche **τj.**

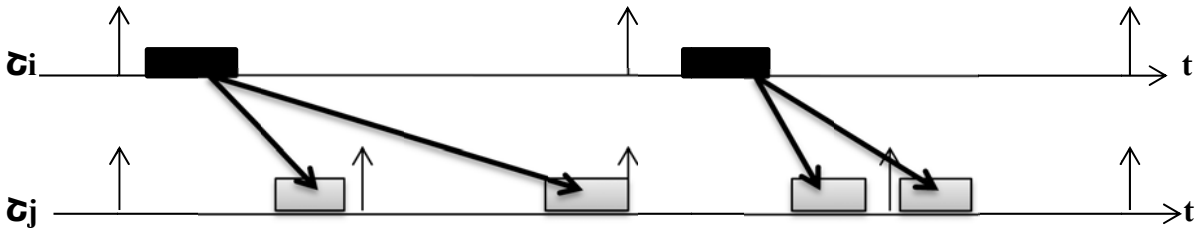


Fig.II.10 [1] Visualisation de la contrainte de précédence généralisée entre deux tâches.

II.6 Analyse d'ordonnançabilité en environnement monoprocesseur

II.6.1 Modélisation et ordonnancement des applications temps réel

Nous sommes partis d'un modèle théorique simple : la tâche périodique est indépendante. Sur la base de ce modèle théorique, nous avons étudié des algorithmes d'ordonnancement optimaux dans leur domaine, permettant d'élaborer des conditions d'ordonnançabilité.

Il est donc extrêmement intéressant de pouvoir transformer toutes les applications temps réel en un ensemble de tâches périodiques indépendantes.

Or nous trouvons dans les applications temps réel, des tâches apériodiques, des tâches dépendantes et des tâches partageant des ressources critiques. La méthodologie d'analyse des applications temps réel consiste donc à transformer l'application initiale pour la rendre compatible avec l'environnement théorique de validation. Ainsi nous avons les transformations suivantes :

- Tâches périodiques indépendantes.
- Tâches apériodiques indépendantes.
- Tâches dépendantes.
- Tâches partageant de ressources.

L'application ainsi transformée est équivalente en termes d'ordonnançabilité à l'application initiale.

II.6.2 Méthode de validation RMA

Cette prise en compte de durée de blocage a permis de mettre en œuvre une méthodologie de validation temporelle d'application temps réel dont l'algorithme d'ordonnancement est fondé sur Rate Monotonic RM (affectation de priorité selon la période) et un protocole de gestion de ressources (héritage de priorité ou héritage de priorité avec priorité plafonnée). La méthodologie est appelée (Rate Monotonic Analyse « RMA »)

II.7 Ordonnancement en environnement multiprocesseur

II.7.1Introduction générale

II.7.1.a Définition

Le système informatique de contrôle est souvent constitué de plusieurs équipements informatique interconnectés par un réseau et ce pour les raisons suivantes :

- Le procédé est par nature constitué d'équipement multiple.
- La sûreté de fonctionnement.
- Les contraintes de temps.
- Les systèmes répartis à contrôle centralisé ou systèmes multiprocesseur ou encore systèmes fortement couplés.
- Les systèmes répartis ou distribués (à contrôle décentralisé) ou systèmes faiblement couplés.

II.7.1.b Placement des tâches dans les systèmes distribués

Le problème de placement ou d'allocation des tâches peut être résolu à partir des paramètres statiques de l'application soit :

- **Caractéristiques statiques des tâches :** durée d'exécution, place mémoire, coûts de communications, interactions avec le procédé, importance,…..
- **Caractéristiques statiques de l'architecture opérationnelle :** Nombre de nœuds, architecture interne des nœuds (mono ou multiprocesseur), vitesse de traitement des processeurs, caractéristiques du réseau de communication (délai de propagation, délai de transmission)

La solution du des tâches sur les différents sites est déterminée à partir de la prise en compte de nombreux critères comme :

- Minimiser le nombre de processeurs
- Equilibrer la charge des processeurs
- Minimiser la communication ente les nœuds
- Prendre en compte les contraintes de résidence
- Minimiser le temps de réponse d'une tâche ou d'un ensemble de tâches ou d'un site
- Respecter le degré de redondance pour chaque tâche

Chapitre : III

PROCESSUS DE QUALITE

III.1. Introduction

Un processus métier est une séquence ordonnée et chronologique des taches destinées à produire un résultat à valeur ajoutée pour le client ainsi que les employés de l'organisation. Cette notion a été toujours présente quel que soit la taille de la structure organisationnelle de l'entreprise.

L'objectif de la gestion des processus métiers est de rendre l'entreprise efficace, flexible et compétitive tout en produisant des biens et des services de qualité à moindre coût. Ainsi l'intégration d'un système de gestion de la qualité a comme objectif de proposer un moyen d'amélioration efficace et continue des résultats produits en conformité aux attentes des clients. La réussite de sa mise en œuvre dépend essentiellement de la flexibilité des processus opérationnel de l'organisation.

III.2 La revue de processus, un outil de pilotage des systèmes de management

Qu'est-ce qu'une revue de processus ? Est-elle obligatoire ? Comment dit-elle réalisée ? Voilà autant de questions que se posent les entreprises en démarche d'amélioration qui ont mis en œuvre l'approche processus de leur organisation.

En effet, elle permet à chaque pilote de processus de s'impliquer (et d'impliquer ses équipes) dans la démarche d'amélioration et de faire état des éventuelles difficultés rencontrées dans la mise en œuvre et/ou le suivi des dispositions établies. La tenue de ces revues de processus permet aux pilotes de servir de relais, de correspondants du responsable du système de management (représentant de la direction) auprès des équipes.

La Revue de processus n'a pas pour objectif de traiter les problèmes spécifiques ou récents, mais constitue un moment privilégié où les acteurs du processus font le point sur le fonctionnement et l'efficacité du processus et, d'une manière plus générale sur l'aptitude du processus à s'améliorer. En d'autres termes la revue de processus n'est pas une réunion qualité !

L'ordre du d'une revue de processus peut prendre en compte les éléments suivants :

- le bilan de la performance des processus (indicateurs de performance)
- le bilan des non-conformités, dysfonctionnements et réclamations rencontrées depuis la dernière revue
- le suivi des actions d'amélioration (correctives et préventives) décidées précédemment
- l'identification des besoins en ressources
- l'ouverture d'actions complémentaires

Dans l'idéal, la « bonne » fréquence de ces revues est mensuelle, en accord avec la périodicité de suivi des indicateurs de performance mais disons que la tenue trimestrielle d'une revue de processus est déjà pas mal… Au-delà, les revues ne seraient pas pertinentes puisque le regard porté sur les performances se ferait à postériori sur des périodes trop reculées. [2]

III.2.1. Le processus conception et développement

III.2.1.1 Définition

La « conception et le développement » est une somme d'activités créatrices qui, partant des besoins exprimés et des connaissances existantes, aboutissent à la définition d'un produit satisfaisant ces besoins et industriellement réalisables.

III.2.1.2 La phase de conception

Elle est donc l'activité qui consiste à développer les meilleures solutions à partir d'un besoin donné à partir de deux phases :

- **La phase d'étude de faisabilité :** Permet de définir des concepts jugés faisables pour les besoins donnés
- **La phase d'avant-projet :** Choix du meilleur concept

III.2.1.3 La phase de développement

Quant à elle, s'appuie sur la phase d'avant-projets et permet de :

- Définir précisément les solutions,
- Préparer la réalisation et l'utilisation du produit. [3]

III.3. Qualités des systèmes informatiques

Les systèmes informatiques disposent d'une gestion de la qualité particulière. La qualité des systèmes informatiques intègre au projet de développement une approche permettant de contrôler autant que possible le produit final. Elle concerne :

- la qualité des processus de réalisation ;
- la qualité des processus d'ingénierie des systèmes, notamment mis en œuvre par le génie logiciel, que l'on nomme également sûreté de fonctionnement des systèmes informatiques (une application particulière de la sûreté de fonctionnement).

La sécurité informatique dépend grandement de la qualité des systèmes. La qualité des systèmes informatiques implique un grand nombre de concepts. L'approche processus doit être complétée de l'approche d'amélioration continue.

III.3.1 Qualité des processus de réalisation

Les enjeux d'un projet informatique peuvent être parfois très importants. Leur gestion doit prendre en compte :

- les besoins fonctionnels explicites ou implicites
- Les coûts
- Les délais
- Les risques

- Les caractéristiques qualités qui font souvent partie des besoins implicites, et ce, tout au long du cycle de développement.

  La démarche de qualité des processus mise en œuvre apporte au projet une plus-value. Trois types de processus peuvent être considérés
- Les processus de pilotage (management, organisation)
- Les processus opérationnels ou de base (pour le client)
- Les processus de support (logistique)

L'ensemble de ces processus constituent la référentielle qualité du projet ou même de l'entreprise. La norme ISO 15504, appelée pendant son élaboration « SPICE », propose une cartographie de processus pour le référentiel. La norme CMMI propose des processus semblables, la différence entre ces deux normes est que la CMMI propose un ordre d'application des processus, alors que l'ISO laisse ce choix aux professionnels qui peuvent même adapter le référentiel. On retrouve par exemple les processus suivants

- Gestion de projet
- Gestion des risques : elle permet de :
  - Définir de façon pertinente des objectifs de coûts, délais, performances ;
  - Organiser la réactivité
  - Atteindre plus sûrement les objectifs du projet.
- Gestion de configuration : elle assure :
  - la traçabilité
  - la visibilité
  - la disponibilité
  - la lisibilité
  - la protection des différents éléments de configuration
- Gestion des modifications : Les modifications sont toujours déstabilisantes. Elles augmentent le coût de développement et peuvent conduire à une régression de la qualité de la documentation et des programmes. Elles ont un impact sur :
  - La maintenabilité
  - La fiabilité
  - La correction

II.3.1.1Outils

III.3.1.1. a La métrologie

La qualité des systèmes informatiques utilise la métrologie comme moyen de contrôle. Elle permet de :

- +Comparer les projets entre eux
- Mesurer les progrès, les régressions
- Fixer des objectifs précis de qualité
- Posséder des détecteurs de non-qualité potentielle
- Permettre une visibilité sur le projet pour le client

Elle permet de travailler sur trois niveaux de mesure et indicateur:

1. Produit ou service qui concerne plus la partie sûreté de fonctionnement

2. Processus :

- Paramètres du processus
- Indices de stabilité
- Taux de dysfonctionnement

3 Clients :

- Taux de service
- Indice d'insatisfaction
- Taux de réponse (favorables/défavorables)

❖ **Qualité des processus d'ingénierie des systèmes**

Toutes les méthodes de sûreté de fonctionnement participent, lors du développement du produit, à la qualité finale, ayant pour objectif la satisfaction du client. [4]

➢ **Qualité des données**

La qualité des données, en informatique se réfère à la conformité des données aux usages prévus, dans les modes opératoires, les processus, les prises de décision, et la planification (J.M. Juran). De même, les données sont jugées de grande qualité si elles représentent correctement le mode de fabrication auquel elles se réfèrent. Ces deux points de vue peuvent souvent entrer en contradiction, y compris lorsqu'un même ensemble de données est utilisé avec un objectif commun.

III.3.1.2 Historique

La plupart des technologies sur les données informatiques sont nées du désir d'envoyer des informations par courrier. Avant l'émergence de serveurs bon marché, les ordinateurs centraux étaient utilisés pour mettre à jour les données (noms, adresses, et autres attributs) afin que les courriers électroniques arrivent correctement à leur destination.

Les mainframes utilisaient des règles métiers pour corriger les défauts dans les données (fautes sur les champs nom et date, défauts de structuration), ainsi que pour suivre les clients qui avaient changé d'adresse, disparu, fusionné, ou expérimenté d'autres événements.

Aux États-Unis, les agences de gouvernement commencèrent à mettre à disposition des données postales à quelques sociétés de service pour gérer les entreprises selon le registre de changement d'adresse national (NCOA). Cette technique a fait économiser à de grandes entreprises de grandes sommes d'argent (millions de dollars) en comparaison de la gestion manuelle des données client. Les grandes entreprises ont réduit leurs frais postaux, les factures et courriers atteignant leurs destinataires plus précisément. Vendue à l'origine comme un service, la qualité des données s'est intégrée au sein des organisations grâce à la disponibilité de technologies serveurs abordables.

Bien que la plupart des entreprises pensent au nom et à l'adresse quand elles se préoccupent de qualité des données, on reconnaît aujourd'hui que la qualité des données est la façon d'améliorer tous les types de données, comme les données sur la chaîne logistique, les données des progiciels de gestion intégrée, les données transactionnelles, etc. Par exemple, mettre en conformité les données de la chaîne d'approvisionnement à un certain standard a une valeur pour une organisation en :

1. évitant de sur stocker des stocks similaires mais légèrement différents
2. améliorant la compréhension d'achats en négociant des remises en quantité
3. évitant les coûts logistiques en stockant et envoyant des pièces détachées à travers une grande organisation.

Alors que les données sur les noms et adresses ont un standard clair avec les définitions des autorités postales, les autres types de données ont peu de standards reconnus. Il y a une tendance de fond aujourd'hui dans l'industrie pour standardiser certaines données qui ne sont pas des adresses. Le groupe GS1 fait partie des groupes qui sont fers de lance dans ce mouvement.

III.4.Importance de la qualité des données

La qualité des données est très importante pour réaliser l'interopérabilité de systèmes complexes. En particulier, elle intervient dans les exigences de traçabilité, qui se manifestent dans plusieurs secteurs économiques :

- Santé et pharmacie,
- Agroalimentaire et grande distribution,
- Chimie,
- Automobile...

Dans le même ordre d'idée, la qualité des données intervient aussi dans l'analyse du cycle de vie des produits.

La qualité des données revêt une grande importance également dans le Direct Marketing ou plus globalement le Customer Relationship Management (CRM) où les données client représentent une source de valeur importante pour les entreprises. Sous un angle plus directement lié à la sécurité des données, elle intervient dans la gestion des documents d'archive, pour l'imputabilité. La qualité des informations est l'un des 11 facteurs du modèle d'intelligence économique (AFDIE). D'après le rapport du CIGREF sur le capital immatériel, la fiabilité et l'audibilité des données conditionnent l'évaluation du capital immatériel des entreprises, et le calcul du retour sur investissement des projets d'ingénierie des connaissances. Il est donc nécessaire de disposer de référentiels de données normalisés pour évaluer la qualité des données. On peut dire aussi que l'objectif poursuivi par la gestion de contenu de parvenir à une convergence des systèmes de gestion de contenu participe d'une démarche qualité dans le domaine des données.

III.5 Qualité des informations dans le modèle d'intelligence économique

Le modèle d'intelligence économique de l'AFDIE identifie six critères de qualité de l'information :

1. **Coût et valeur de l'information :**

La mise en œuvre des normes IAS/IFRS comporte la comptabilisation du capital immatériel en immobilisations incorporelles ; elle pousse l'entreprise à mieux évaluer la valeur de l'information dans sa stratégie.

2. **Connaissances capitalisées et validées**,

Le moyen de capitaliser et de valider les connaissances se fait par des projets d'ingénierie des connaissances.

3. **Mémoire vivante et accessible**,

La mémoire peut être gérée efficacement par une démarche de gestion de contenu, qui visera à unifier les différents systèmes de gestion de contenu présents dans l'entreprise.

4. **Informations et connaissances mieux partagées et protégées**

Il s'agit de mettre en place des communautés de pratique, avec des critères permettant de protéger le patrimoine informationnel.

5. **Informations écrites et orales indispensables avant la prise de décision**,
6. **Information prospective et historique.**

Il s'agit de disposer d'un référentiel historisé et permettant de faire des analyses prospectives selon des axes d'analyse.

III.6 Qualité des données et système décisionnel

Le système décisionnel a pour objet d'aider les décideurs à effectuer des choix pertinents à partir de données historiques. Ceci implique généralement de prendre en charge de grandes quantités de données disparates afin de les calibrer en information suffisamment précise et sûre pour qu'en confiance des actions puissent être définies et lancées. Ce défi classique pour les architectes de systèmes décisionnels est amplifié ces derniers temps par l'augmentation des exigences en matière de variété, de profondeur et de fraîcheur des données historiques à gérer. Dans ce contexte, la vitesse avec laquelle des données peuvent être « nettoyées », « transformées » et intégrées dans un entrepôt de données devient essentielle pour la compétitivité des entreprises.

La qualité d'une donnée dépend d'abord du contexte dans lequel elle a été initialement saisie, mais d'un point de vue décisionnel son intérêt dépend de l'usage qu'un utilisateur peut en faire. La qualité d'une donnée ne doit donc pas être appréciée dans l'absolu mais de façon relative à son intérêt métier. Les besoins métiers évoluant, l'appréciation de la qualité d'une donnée ne peut donc jamais être fixée de façon définitive. Les programmes d'amélioration de la qualité des données doivent être ciblés en fonction des intérêts métier, sinon il y a un fort risque de se créer une charge de travail digne de Sisyphe.

III.7 Normalisation

Il existe très peu de normes relatives spécifiquement à la qualité des données. On peut citer toutefois :

- la norme ISO 19115 relative aux informations géographiques.
- la Norme ISO 8000 relative à la qualité des données de référence (Master data)

Des besoins existent aussi sur les informations contenues dans les ressources informatiques que les informaticiens appellent "non structurées". Des travaux sont en cours pour parvenir à des normalisations des données de référence qui sont utilisées dans ce type de ressources, notamment les ressources Web qui se généralisent et ont besoin de s'interfacer avec d'autres types de ressources informatiques. [5]

- **Qualité logicielle**

En informatique et en particulier en génie logiciel, la qualité logicielle est une appréciation globale d'un logiciel, basée sur de nombreux indicateurs.

La complétude des fonctionnalités, la précision des résultats, la fiabilité, la tolérance de pannes, la facilité et la flexibilité de son utilisation, la simplicité, l'extensibilité, la compatibilité et la portabilité, la facilité de correction et de transformation, la performance, la cohérence et l'intégrité des informations qu'il contient sont tous des facteurs de qualité.

Contrairement à un matériel, un logiciel est un produit qui n'a pas une fiabilité prédictible, de plus il ne s'use pas dans le temps. Donc une anomalie survient ou ne survient pas dans l'exécution du logiciel, l'anomalie est présente de manière latente et peut ne jamais survenir. La qualité d'un logiciel dépend entièrement de sa construction et des processus utilisés pour son développement, c'est par conséquent un sujet central en génie logiciel. Une appréciation globale de la qualité tient autant compte des facteurs extérieurs, directement observables par l'utilisateur, que des facteurs intérieurs, observables par les ingénieurs lors des revues de code ou des travaux de maintenance.

Les problèmes de qualité des logiciels, connus depuis les années 1960, sont par ailleurs à l'origine du génie logiciel, la science de la création de logiciels, y compris toutes les difficultés qui y sont liées - respects des coûts, des délais, du cahier des charges et du niveau de qualité

Il existe plusieurs référentiels de certification du système de management de la qualité en entreprise, en matière d'ingénierie du logiciel comme TickIT.

III.8 Indicateurs de qualité logicielle

La norme ISO 9126 définit six groupes d'indicateurs de qualité des logiciels :

- La capacité fonctionnelle. c'est-à-dire la capacité qu'ont les fonctionnalités d'un logiciel à répondre aux exigences et besoins explicites ou implicites des usagers. En font partie la précision, l'interopérabilité, la conformité aux normes et la sécurité
- La facilité d'utilisation
- La fiabilité, c'est-à-dire la capacité d'un logiciel de rendre des résultats corrects quelles que soient les conditions d'exploitation. En font partie la tolérance aux pannes - la capacité d'un logiciel de fonctionner même en étant handicapé par la panne d'un composant (logiciel ou matériel)
- La performance
- La maintenabilité, qui mesure l'effort nécessaire à corriger ou transformer le logiciel. En font partie l'extensibilité, c'est-à-dire le peu d'effort nécessaire pour y ajouter de nouvelles fonctions
- La portabilité, c'est-à-dire l'aptitude d'un logiciel de fonctionner dans un environnement matériel ou logiciel différent de son environnement initial. En font partie la facilité d'installation et de configuration dans le nouvel environnement

Les différents indicateurs sont parfois conflictuels, ou au contraire complémentaires : une augmentation de la capacité fonctionnelle peut avoir un impact négatif sur la performance, la maintenabilité et la fiabilité. Tandis qu'une augmentation de la fiabilité, la maintenabilité ou de la disponibilité ont un impact positif sur l'utilisabilité. En outre, une augmentation de la maintenabilité peut avoir un impact négatif sur la performance.

III.9 La crise du logiciel

Un phénomène de baisse des prix du matériel informatique et d'augmentation des prix du logiciel, accompagné d'une baisse de la qualité des logiciels a été identifié à la fin des années 1960 et nommé la « crise du logiciel ». Cette crise s'apparente aujourd'hui à une maladie chronique de l'industrie du logiciel, dont les symptômes sont les suivants :

- Les délais de livraison des logiciels sont rarement tenus, le dépassement de délai et de coût moyen est compris entre 50 et 70 %
- La qualité du logiciel correspond rarement aux attentes des acheteurs, le logiciel ne correspond pas aux besoins, il consomme plus de moyens informatiques que prévu, et tombe en panne
- Les modifications effectuées après la livraison d'un logiciel coûtent cher, et sont à l'origine de nouveaux défauts. Les adaptations sont bien souvent une nécessité du fait de l'évolution des produits et des attentes des utilisateurs
- Il est rarement possible de réutiliser un logiciel existant pour en faire un nouveau produit de remplacement ; l'amortissement du coût de développement initial est ainsi rendu impossible.

Auparavant minoritaire, le coût du logiciel en 1965 représentait 50 % du coût total d'un système informatique. En 1985 la part du logiciel est de 80 % et les coûts dus à la correction des défauts dans les logiciels (maintenance) représentent jusqu'à trois quarts du coût total d'acquisition, un excédent dû uniquement à la mauvaise qualité du logiciel lors de sa livraison.

Selon une étude réalisée en 1994 par le Standish Group, 53 % des logiciels créés sont une réussite mitigée : le logiciel est opérationnel, cependant le délai de livraison n'a pas été respecté, les budgets n'ont pas été tenus, et certaines fonctionnalités ne sont pas disponibles. Le dépassement des coûts est en moyenne de 90 %, et celui des délais de 120 %, et la qualité moyenne est estimée à 60 %

III.10 Raisons du manque de qualité des logiciels

Un logiciel étant un produit immatériel, les seules représentations observables du logiciel sont le code source, l'interface utilisateur et la documentation (spécification, cahiers de tests, manuels utilisateur, etc.). La quantité de code source (nombre de lignes) est rarement connue à l'avance, ce qui entraîne souvent une sous-estimation de la complexité du logiciel.

Pour chaque module d'un logiciel il existe de nombreuses conditions d'utilisation. La combinaison des différentes conditions d'utilisation des différents modules d'un logiciel amène une explosion combinatoire, et lors de sa construction un logiciel n'est jamais contrôlé dans la totalité des conditions d'utilisation qu'il rencontrera durant son exploitation; ceci pour des raisons pratiques (coût et durée des travaux).

Une autre raison est qu'il n'y a pas de lien entre un défaut mineur et majeur, et une modification mineure ou majeure. Et l'effort de détérioration d'un logiciel n'est pas proportionnel à l'effort de construction. Un défaut mineur peut entrainer un incident majeur, et nécessiter une correction mineure. Dans l'incident du vol 501 d'Ariane 5, une correction mineure aurait suffi à éviter la destruction de la fusée. De même une modification mineure d'un logiciel peut le mettre hors d'usage; un phénomène largement exploité par les virus informatiques.

III.11 Amélioration de la qualité

En génie logiciel, la factorisation des données et du code constituent le moyen universel d'obtention de la qualité. La factorisation des données aboutit au modèle objet (avec usage de l'héritage) dont le correspondant systématique relationnel est idéal lorsqu'il est normalisé (formes normales de Cod). En effet, lorsque les structures de données sont ainsi normalisées, elles deviennent non redondantes, donc minimales en taille, et n'engendrant aucun problème d'incohérence dès lors que l'intégrité découlant de leur type et l'intégrité référentielle sont assurées, contraintes auxquelles il faut ajouter des "règles métiers" consistant en contraintes logiques faisant intervenir plusieurs champs/attributs. Ce travail au niveau des données permet en soi la réduction du code de traitement exploitant ces données. La performance n'en souffre pas si les requêtes sont bien organisées.

Concernant le code, la factorisation permet de n'écrire qu'une fois des instructions similaires, par un usage raisonné des variables intermédiaires et locales, des boucles, des fonctions et des procédures. Cela permet la réduction maximale de la taille du code source (sans perte normalement de lisibilité), et aboutit à ce que les modifications soient le plus locales possibles (donc plus rapides, et plus fiables en termes de non régression). Un gain complémentaire de réduction du code source est apporté par le polymorphisme et la liaison dynamique (qui éliminent les « procédures aiguillage ») en programmation objet (celles-ci sont générées par le compilateur au lieu de devoir être écrites explicitement par le programmeur). Ces factorisations font émerger les bonnes abstractions, la bonne structuration, et permettent le meilleur contrôle possible de l'intégrité des données et de la bonne exécution des traitements, dont les fonctions, appelées en plusieurs points du code source peuvent devenir de fait des services réutilisés, autant que les données qui, étant partagées, sont réutilisées sans duplication. Ce travail permet à une application d'être "modulaire". Ces modules sont les services. Leur interface étant claire et sans effet de bord, le traitement qu'ils réalisent devient caché (boîte noire) pour ses modules clients. Le couplage entre un module et ses modules appelants devient le plus faible possible, du fait que seules les valeurs de paramètres variant d'un appel à l'autre sont passées en argument, les variables invariantes entre appels devant idéalement constituer des attributs d'une classe porteuse, la fonction/module en devenant une méthode (dans

une programmation objet aboutie). En ce sens, on peut parler de couplage faible, et l'on peut substituer une implémentation à une autre. Il est faux en général que l'on obtienne ainsi la minimisation de l'impact de la défaillance d'un module sur les autres modules et sur le fonctionnement de l'application.

L'abstraction vise à diminuer la complexité globale du logiciel en diminuant le nombre de modules et en assurant l'essentiel. Elle peut également apporter une uniformité du logiciel qui augmente son utilisabilité en facilitant son apprentissage et son utilisation.

La dissimulation vise à séparer complètement les détails techniques du logiciel de ses fonctionnalités selon le principe de la boîte noire, en vue d'améliorer sa maintenabilité, sa portabilité et son interopérabilité.

La structuration des instructions et des données rend clairement visibles dans le code source les grandes lignes de l'organisation des instructions et des informations manipulées, ce qui améliore sa maintenabilité et facilite la détection des bugs

De nombreux langages de programmation soutiennent, voire imposent l'écriture de code source selon les principes de structuration, de modularité et de dissimulation. C'est le cas des langages de programmation structurée et de programmation orientée objet. [6]

IV.1. Introduction

Le processus dit un ensemble de moyens et d'activités liés qui transforment des éléments entrants en éléments sortants. Ces moyens peuvent inclure le personnel**,** les installations, les équipements, les techniques et les méthodes**.**

- Le processus répond aux questions : QUOI FAIRE ? POUR QUELLE VALEUR AJOUTÉE?
- La procédure répond aux questions : COMMENT FAIRE ? QUAND ? QUI ?
- Le mode opératoire répond de plus aux questions : OÙ ? SELON QUEL PROCÉDÉ ?
- Le mode pilotage répond de plus à la question : COMBIEN ?
- L'utilisation d'une approche processus peut conduire à élaborer une cartographie des processus qui permet de représenter l'entreprise à travers les liens entre les différents processus.

IV.2 À propos de processus

IV.2.1 Le marché

C'est Le lieu de rencontre (physique ou virtuel) où les organisations offrent des biens ou services, qui seront ou non le sujet d'acquisitions par des tierces personnes morales ou physiques.

L'organisme existe pour produire un bien ou un service. Si ce bien ou service couvre une attente du marché, il pourra être échangé contre des ressources permettant de poursuivre l'activité de l'organisme. Si l'organisme ne produit pas de bien ou service couvrant une attente ou les fournit sans contrepartie, il devra trouver d'autres moyens de se procurer les ressources nécessaires à son fonctionnement.

Le marché est le pourquoi de l'organisme social, sa raison d'être. L'organisme, lui, répond au quoi du processus, de la demande.

	Personne morale ou physique	**Marché**	**Organisme**
Action	**Dispose de ressources**	**Échange ou donne**	**Produit un bien ou service**

Tableau.IV.1. [7] les activités de l'organisme

IV.2.2 L'organisation :

Les prospects entrent en relation avec l'organisme pour sa capacité d'organisation et ses compétences à offrir le bien ou service attendu. L'organisation permet de déterminer les où, quand, et qui des processus.

Les éléments de l'organisation		
		
Ordonner	**Planifier**	**Documenter**

Tableau.IV.2. [7] les éléments de l'organisation

IV.2.3 Le savoir- faire

L'organisme exploite ses méthodes et son savoir-faire afin de réaliser les produits et services. Le savoir-faire répond bien évidemment au comment du processus.

IV.2.4 La mise en œuvre

L'organisation sociale met en œuvre ses outils au travers de ses méthodes et de son savoir-faire. La mise en œuvre détermine avec quoi on réalise le processus.

Les éléments de savoir	
	
La compétence	**La connaissance**

Les éléments de la mise en œuvre

Outils

Tableaux.IV.3-4. [7] qui présentent les éléments de savoir et de mise en œuvre

IV.2.5 Résumé de la démarche

Pour conduire ses activités et réaliser les produits et services, l'organisme met en œuvre des processus intégrant de façon cohérente :

- L'organisation et les compétences
- Le savoir-faire et les méthodes
- Les outils

Le processus est donc un système d'activités qui utilise des ressources pour transformer des éléments d'entrée en éléments de sortie.

IV.3 Les processus en question

IV.3.1 C'est quoi le management par processus ?

Le management par processus consiste pour l'organisation à :

- Identifier les processus et les activités qui les composent ;
- Les décrire
- Préciser les acteurs
- Désigner leur propriétaire (pilote)
- Définir les dispositifs de pilotage
- Améliorer en permanence les processus et leurs activités

Le management par processus distingue :

- ✓ **L'efficacité** ou l'atteinte des résultats : état binaire, réussite ou échec
- ✓ **L'efficience** ou la performance du triplet

- Fonctionnement
- Productivité
- Rendement

IV.3.2 Comment intégrer l'approche processus dans une organisation pyramidale ?

Les organismes sont majoritairement gérés sous une forme de direction pyramidale, comme présenté ci-dessous. L'approche « classique » fait circuler les objectifs de haut en bas. C'est-à-dire avec une approche complètement verticale cloisonnant chaque activité en briques hermétiques, sans que l'une ait connaissance des activités des autres et de leurs relations.

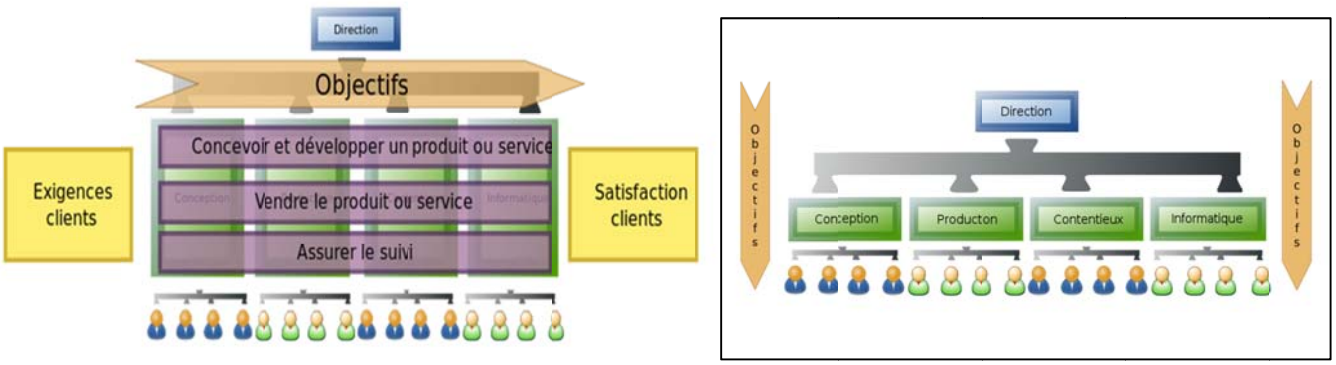


Fig.IV.1. [7] organisation pyramidale

L'approche processus nécessite de faire véhiculer les objectifs de manière transversale, et non verticale. Chacun doit être conscient de son rôle et de son utilité vis-à-vis de l'objectif global.

IV.3.3 Quels avantages offre une démarche de management par processus ?

En mettant en œuvre une telle démarche, l'organisation :

- Intègre les besoins de ses clients
- Optimise et diminue ses coûts de fonctionnement
- Améliore sa productivité interne
- Pilote de bout en bout selon une véritable stratégie
- Formalise ses procédures, ses modes opératoires
- Fait de l'amélioration continue un principe de fonctionnement
- Entre dans une démarche améliorant le professionnalisme
- Limite les problèmes liés aux interfaces
- Améliore sa réactivité dans le traitement des anomalies
- Permet à tous de se situer au sein de l'organisation et de mieux appréhender les finalités de ses activités

IV.3.4 Quels inconvénients apporte une démarche de management par processus ?

- Difficile à mettre en place.
- Dur à comprendre sans préparation.

IV.3.4. a Les différents types de processus

- **Processus de management** qui Déterminent la politique et le déploiement des objectifs dans l'organisme.
- **Processus de réalisation** qui Contribuent directement à la réalisation du produit ou du service.
- **Processus support** qui Contribuent au bon déroulement de la réalisation en leur apportant les ressources nécessaires.

IV.3.5 Processus, procédure, mode opératoire et enregistrement

Terme	Description
Processus	Démarche transformant des éléments entrants en éléments sortants
Procédure	Document qui décrit de façon formalisée les tâches à accomplir pour mettre en œuvre le processus: c'est le mode d'emploi opérationnel
Mode opératoire	Document qui décrit au niveau le plus fin les différentes opérations qui permettent de réaliser la procédure.
Enregistrement	Preuve tangible de l'exécution d'une tâche, activité, opération.

☞ **Tableau.IV.9. [8]** Table de terminologie ordonnée en granularité croissante.

Et chaque élément du tableau contient un ou plus d'éléments de la ligne suivante : un processus se compose d'au moins une procédure, etc.

IV.3.6 Indicateurs de résultat et de fonctionnement

- **Indicateur de résultat :**

C'est un Indicateur permettant de mesurer l'écart entre le résultat attendu et le résultat obtenu.

- **Indicateur de fonctionnement :**

C'est un Indicateur permettant de contrôler le bon déroulement du processus.

IV.3.7 Représentation schématique du processus

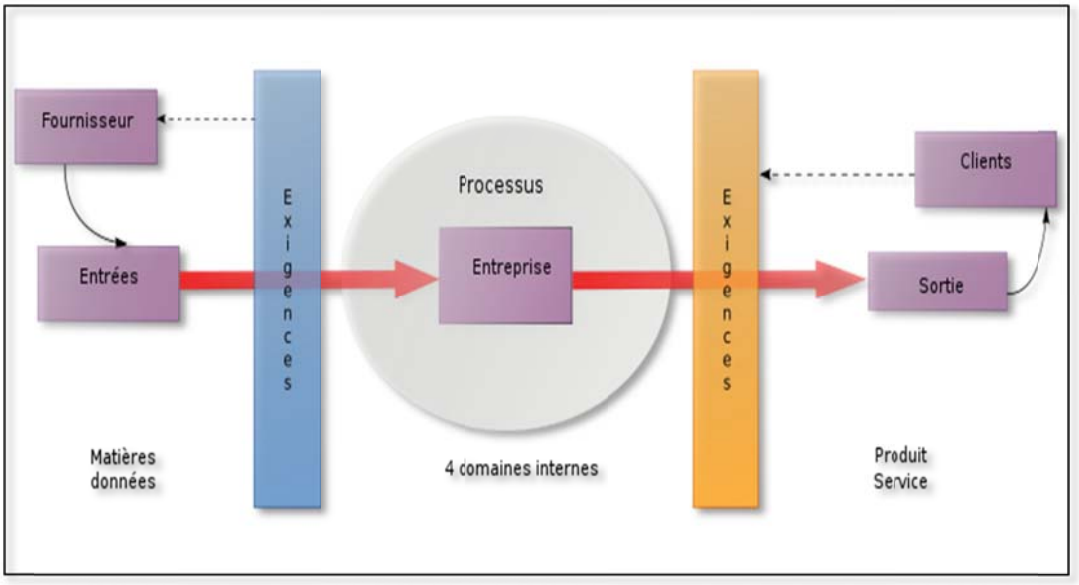


Fig.IV.2 [7] représentation schématique du processus

IV.4 Détermination des processus

IV.4.1 Établir la cartographie des processus

- Les domaines, activités, taches.
- Les flux d'information, leur contenu, leur qualité.
- Les supports associent (informatique ou non).
- L'organisation, les humains.
- Le métier.
- Identifier et maitriser les interfaces**.**

IV.4.2 Définir la cible

Objectif :

- Analyse des domaines, activités et tâches du périmètre de l'étude ;
- Recensement au passage des problèmes et propositions d'amélioration ;
- Tenir compte des particularités de votre entité.

Tâches :

- Réaliser les ateliers par domaines (activités, tâches, MGA) ;
- tenir compte des règles de fonctionnement propres aux différents services.

Modalités :

- Atelier 1/2 journée, - 13h30/17h

Groupes de travail par grand domaine « métier » étudié.

Livrable :

- Activités, tâches, MGA, matrices forces/faiblesses, propositions d'amélioration ;
- Première cartographie.

IV.4.3 Lexique

IV.4.4 Identification des tâches

Un exemple de liste pourrait être :

- Vérifier le paiement du contrat
- Relancer une offre
- Contrôler les anomalies de livraison
- Rechercher la référence d'un produit

Déclarer une feuille de deuil :

- Saisir un ARC
- Prendre un RDV client
- Envoyer au client les conditions régie heure, régie jour
- Etablir un descriptif technique
- Orienter les appels clients vers le service concerné
- Expédier le matériel
- Créer un contrat
- Faire émarger le client

Quelques conseils utiles pour identifier les tâches :

- Utiliser des penses bête à l'endroit, c'est-à-dire avec la bande collante en haut
- Une tâche par pense bête
- Une tâche = verbe + complément(s)
- Eviter les verbes imprécis : gérer, administrer, participer à, travailler, organiser, etc.
- 8 mots maximum par pense bête

IV.4.5 Activités, modèle global d'activité

Cette étape consiste à :

- Regrouper les taches
- Nommer les activités

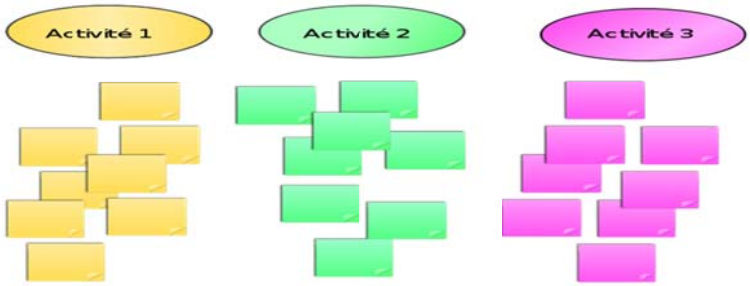


Fig.IV.3 [7] Modèle globale d'activités

IV. 5.1 Établir les Modèles Globaux d'activités (MGA)

Pour cela il convient de :

- Décrire les MGA
- S'assurer que 1 MGA = 1 processus
- D'une première cartographie (répartition par type)
- Identifier les liens inter- processus

IV.5 Matrices forces, faiblesses, améliorations

Ces matrices permettent d'évaluer la perspective qu'ont les collaborateurs des forces et faiblesses de l'organisme. Elles permettent donc non seulement de déceler les faiblesses et les forces, mais aussi de visualiser les éventuelles disparités de point de vue de chaque collaborateur sur le fonctionnement de l'organisme. Les éléments à placer sur la matrice, qui peut évaluer l'importance ou l'amplitude sont :

- Le métier
- Les ressources humaines
- L'organisation
- Le système.

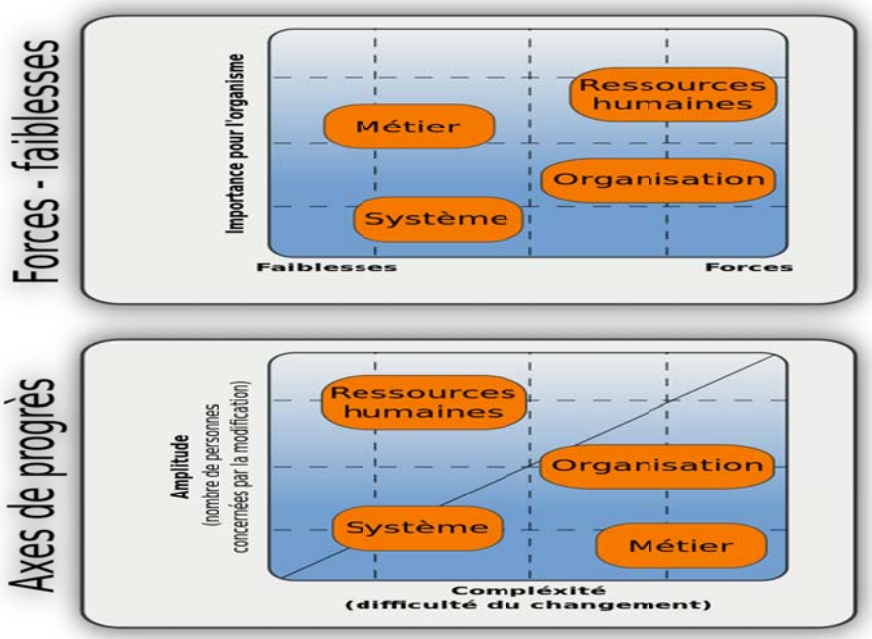


Fig.IV.4 [7] perspectives des forces et des faiblesses de l'organisme

IV.6 Les pilotes de processus

IV.6.1 Les rôles des pilotes de processus

Les pilotes de processus améliorent l'efficacité globale du processus dont ils sont responsables :

- En déterminant les indicateurs les plus pertinents, au regard des objectifs fixés pour le processus
- En s'assurant de la disponibilité effective des ressources nécessaires
- En vérifiant la conformité des données d'entrée et de sortie du processus
- En identifiant les risques et en repérant les points critiques au sein du processus qu'ils pilotent

Les pilotes de processus vérifient l'attente de l'objective qualité sur la base des résultats des indicateurs associés et des valeurs cibles.

La collecte des données peut-être effectué par une tierce personne qui confirme la possibilité de calculer l'indicateur.

Les pilotes de processus déterminent les actions nécessaires pour traiter les dysfonctionnements et surveillent la réalisation des actions.

Ils participent pleinement à la dynamique d'amélioration continue en encourageant l'ouverture de fiches d'entretien et d'amélioration par tous les acteurs du processus.

Ils sensibilisent chacun des acteurs à son rôle au sein du processus.

Ils font évoluer les dossiers de processus, en particulier en intégrant l'évolution des exigences.

Ils communiquent sur les actions menées et sur les résultats obtenus auprès de l'ensemble des agents impliqués.

Les pilotes sont aussi concernés par tous les processus qui interagissent avec le processus dont ils ont la charge. Ils sont identifiés nominativement dans la cartographie des processus.

IV.6.2 Les premiers travaux

Les pilotes de processus peuvent être chargés de rédiger :

- les fiches descriptives de processus
- les procédures correspondantes
- les modes opératoires le cas échéant

Ces actions de rédaction définissent ainsi :

- les enregistrements (traçabilité)
- les indicateurs (mesure)

IV.7 Le CMM

IV.7.1 Objet du modèle

Le Capability Maturity Model (CMM), soit modèle de maturité des capacités, doit permettre de visualiser simplement l'état courant de l'organisation et faire prendre conscience au personnel des dimensions de la qualité.

IV.7.2 Les 5 niveaux du modèle

Le modèle se décline en 5 niveaux, qui sont gravis un à un. Les niveaux et les étapes qui mènent à la suivante sont :

- le niveau initial, franchi avec des processus structurés
- le niveau répétable, franchi avec des processus standards cohérents
- le niveau défini, franchi avec des processus prédictibles
- le niveau géré, franchi avec des processus en amélioration continue
- le niveau optimisé

Le niveau initial :

À ce niveau :

- quelques processus sont définis, le succès dépend essentiellement des efforts individuels
- Les plannings, les budgets, la qualité du produit sont généralement non respectés

Illustration : gestion de projet rarement formalisé (sous forme de plan de développement) et opérationnelle, pas d'estimation de coût.

Le niveau répétable :

À ce niveau :

- Il y a un suivi de coût, une planification pour chaque projet, un définition des exigences
- Un management de projet est mis en place, il est fondé sur la réussite de projets antérieurs

Illustration : existence de mécanisme de suivi de projet, mais difficulté de faire face à des changements importants de personnel ou de technologie.

Le niveau défini

À ce niveau :

- Les processus de réalisation sont institutionnalisés au niveau de l'organisme, c'est-à-dire documentés et appliqués au niveau du projet.

Illustration : bonne visibilité de l'encadrement sur les projets, car les processus, activités et livrables sont conformes aux plans définis au niveau de l'organisme.

Le niveau géré

À ce niveau :

- Des métriques ou indicateurs sont mis en place pour contrôler le bon déroulement des projets et le respect des objectives qualités.

Illustration : Identification des écarts par rapport aux plans établis à l'aide de mesures, et de leur cause (d'origine structurelle ou accidentelle).

Le niveau optimisé

À ce niveau :

- l'organisme à mis en place une action d'amélioration continue pour optimiser les processus de réalisation dans le but de prévenir des défauts.

Illustration : maîtrise optimisée des coûts, des délais et de la qualité. [7]

IV.8 La notion de qualité

IV.8.1 Définition

La qualité est un processus cyclique, qui suit les étapes présentées dans le schéma ci-dessous :

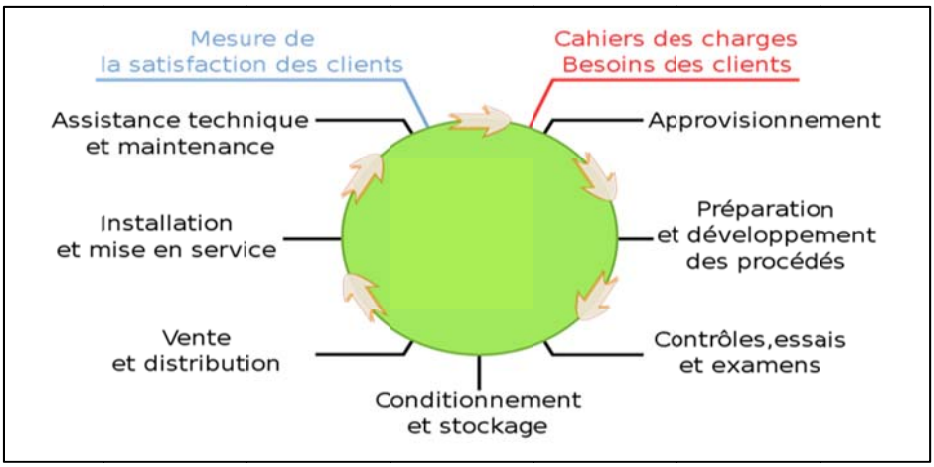


Fig.IV.5. [8] étapes de qualité

IV.8.2 Démarche qualité

Avant de mettre en place une démarche qualité, une entreprise doit avoir une politique qualité qui détermine les objectifs à atteindre en terme de production et de management.

Une fois cette politique qualité approuvée par la totalité des salariés, la démarche qualité est intégrée. La démarche qualité est l'ensemble des actions menées par une entreprise pour :

- Améliorer la qualité et la gestion de la qualité
- Proposer de meilleurs produits, services ou prestations aux clients
- Faire évoluer les salariés

IV.8.3 Mise en place de la démarche qualité

L'intégration de la démarche qualité concerne tous les salariés :

➢ Les cadres et responsables ont le devoir de réorganiser leur service afin de mettre en place les directives dictées par la démarche qualité. Ils doivent tout faire pour atteindre les objectifs et satisfaire la clientèle.

➢ Les employés, de leur côté, doivent appliquer les directives afin que les objectifs de la démarche qualité soient atteints à court, moyen et long terme.

Lors de la réflexion sur la mise en place d'une démarche qualité, l'entreprise invite tous ses salariés à proposer leurs idées et à participer à la création de la démarche qualité. Cette implication est la clé de la réussite.

IV.8.4 La réussite de la démarche qualité

Pour que la mise en place d'une démarche qualité soit bénéfique à l'entreprise, il faut **:**

- Que la démarche qualité et ses directives soient claires et comprises de tous,
- Que le personnel soit formé aux nouvelles tâches qui leurs seront demandées et que leurs conditions de travail soient analysées et si nécessaire améliorées,
- Qu'un représentant qualité soit nommé dans les services principaux :
  - service commercial,
  - service technique,
  - service production,
- Pour les grandes entreprises, qu'un service qualité soit créé,
- Que la non-qualité soit étudiée, que la qualité continue soit assurée,
- Que l'avis du consommateur soit pris en compte.

IV.8.5 Système qualité

IV.8.5.1 Définition du système qualité

Le **système qualité** d'une entreprise regroupe tout les documents concernant ce qui est mise en place en terme de gestion de la qualité.

Le **système qualité** contient les documents relatifs à l'organisation, les actions, les procédures et les moyens mis en œuvre pour que la qualité soit atteinte.

IV.8.5.2 Les documents du système qualité

Le système qualité inclut également tous les documents écrits comme :

- A démarche qualité
- Les certifications, les normes
- Les réglementations
- Les référentielles qualités
- La manuelle qualité

IV.8.6 Les indicateurs de la qualité

Les indicateurs permettent de mesurer la qualité. On distingue deux types d'indicateurs relatifs à quatre idées de la qualité, comme présenté dans le schéma ci-dessous :

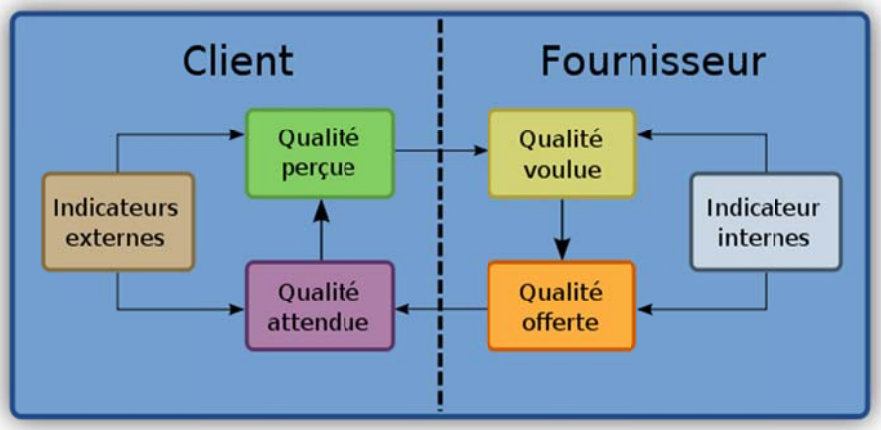


Fig.IV.6 [8] les indicateurs de la qualité

IV.8.7 Critiques et risques liés à la gestion de la qualité

Comme toute activité humaine, la gestion de la qualité n'est pas exempte de défauts et peut mener à des régressions si elle est pratiquée maladroitement, Les griefs les plus courants qui lui sont faits sont :

- **Perfectionnisme :** Un système parfait ... sur le papier
- **Bureaucratie :** Une cathédrale documentaire
- **Taylorisme :** Des documents décrivant les moindres détails
- **Nombrilisme :** Oubli du client
- **Surprotection :** Une assurance tous risques
- **Monopole :** Supporté par les seuls représentants de la qualité
- **Bachotage :** un seul objectif, la certification. [8]

Conclusion générale

Ce rapport de fin mémoire est basé sur le développement des systèmes de suivi et d'évaluation des processus de qualités

Comme le processus est à temps réel, ce travail présente les concepts et mécanismes concernant le système temps réel.

Nous avons montré comment s'effectue le management par processus et aussi comment intégrer l'approche processus dans une organisation. Comme nous avons présenté les systèmes informatiques qui disposent d'une gestion particulière de la qualité. La qualité des systèmes informatiques intègre au projet de développement une approche permettant de contrôler autant que possible le produit final. Elle concerne :

- la qualité des processus de réalisation ;
- la qualité des processus d'ingénierie des systèmes, notamment mis en œuvre par le génie logiciel.

Nous espérons enfin, que dans l'avenir, ce travail sera poursuivi par le développement d'une application en temps réel d'un système de suivi des processus qualité que le manque de temps ne nous a pas malheureusement permit d'achever.

Références et Bibliographie

[1] Francis Cottet Emmanuel Grolleau / systèmes temps réel de contrôle-commande conception et implémentation/ l'usine nouvelle.

[2] Jérémy CICERO/ Qualiblog/ le blog du manager QSE

[3] https://fr.wikipedia.org/wiki/Qualit%C3%A9_des_syst%C3%A8mes_informatiques

[4] https://fr.wikipedia.org/wiki/Qualit%C3%A9_des_donn%C3%A9es

[5] https://fr.wikipedia.org/wiki/Qualit%C3%A9_logicielle

[6] https://fr.wikiversity.org/wiki/Syst%C3%A8me_de_management_de_la_qualit%C3%A9/L%27approche_processus

[7] https://fr.wikiversity.org/wiki/Syst%C3%A8me_de_management_de_la_qualit%C3%A9/La_notion_de_qualit%C3%A9

TABLE DES FIGURES ET DES TABLEAUX

Liste des figures

Liste des tableaux

ملخص:

هذه المذكرة تشرح المفاهيم القائمة على تطوير أنظمة رصد و تقييم عمليات النوعية التي هي في الوقت الحقيقي يسعى هذا العمل الى توفير جميع المعلومات اللازمة على أنظمة الوقت الحقيقي.

باستخدام هذه المفاهيم رأينا كيفية ادارة العملية، وايضا كيفية إدماج مقاربة عملية في المنظمة .

في النهاية، نحن نأمل أن نقدم هذا العمل فكرة عامة عن انظمة الوقت الحقيق، وتطوير نظم لرصد عملية نوعية.

Résumé :

Ce mémoire de fin d'études traite des concepts basés sur le développement des systèmes de suivi et d'évaluation des processus de qualités qui sont en temps réel. Ce travail tente de fournir toutes les informations nécessaires concernant les systèmes temps réel.

Grace à ces concepts, on a vu comment faire le management par processus et aussi comment intégrer l'approche processus dans une organisation.

En fin, on souhaite que ce travail présente une idée générale concernant les systèmes temps réel et le développement des systèmes de suivi des processus de qualités.