

Génie Logiciel Planification

Lamine BOUGUEROUA
Lamine.bougueroua@esigetel.fr

Planification

Expression des besoins

Spécification / Analyse

Conception

Implémentation

Tests

Recette

Maintenance

- Planification d'un projet :
 - définir le problème
 - produire le calendrier du projet
 - Estimation des coûts
 - Maîtrise des coûts
 - *Work Breakdown Structure*
 - Le modèle des processus
 - *Program Evaluation and Review Technique*
 - Diagramme de Gantt
 - confirmer la faisabilité du projet
 - doter le projet en personnel
 - lancer le projet



Work Breakdown Structure

Intérêt :

identifier des tâches et des sous-tâches ;
associer des critères d'achèvement aux tâches.
diagramme des travaux

La structure d'un WBS :

- structure arborescente (sans boucle) ;
 - chaque tâche doit avoir un critère d'achèvement (le plus souvent un livrable) ;
 - l'achèvement de toutes les sous-tâches doit entraîner l'achèvement de la tâche ;
- remarque : le premier niveau de décomposition correspond souvent au modèle de cycle de vie adopté.



Exemple de WBS

- Choisir la recette
- Réunir les ingrédients -> Ingrédients réunis
- Cuisiner
 - Mélanger
 - Ajouter l'eau -> Saladier d'eau
 - Ajouter la levure et la farine -> Mélange
 - ...
 - Cuire -> Pain cuit
- Manger
- Nettoyer

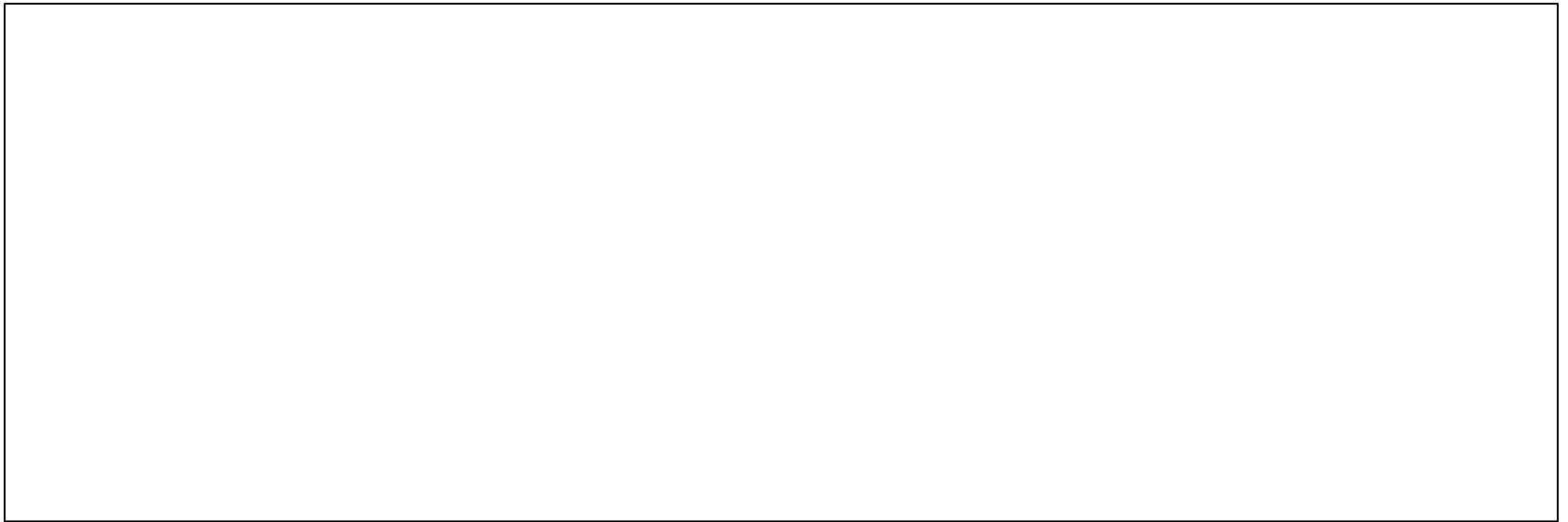


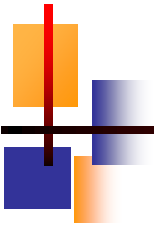
Modèle des processus de conception d'un logiciel

- Un modèle de processus logiciels décrit :
 - les tâches ;
 - les artefacts (fichiers, documents, données...) ;
 - les auteurs ;
 - les décisions (facultatif).
- Règles à observer :
 - deux tâches doivent être séparées par un artefact ;
 - une tâche ne peut être exécutée tant que ses artefacts d'entrée n'existent pas ;
 - il doit y avoir au moins une tâche de début et une de fin ;
 - il doit y avoir un trajet depuis chaque tâche jusqu'à la tâche de fin.



Exemple d'un modèle des processus





Pert Evaluation and Review Technique

Intérêt :

optimiser et planifier l'ordonnancement de tâches ;

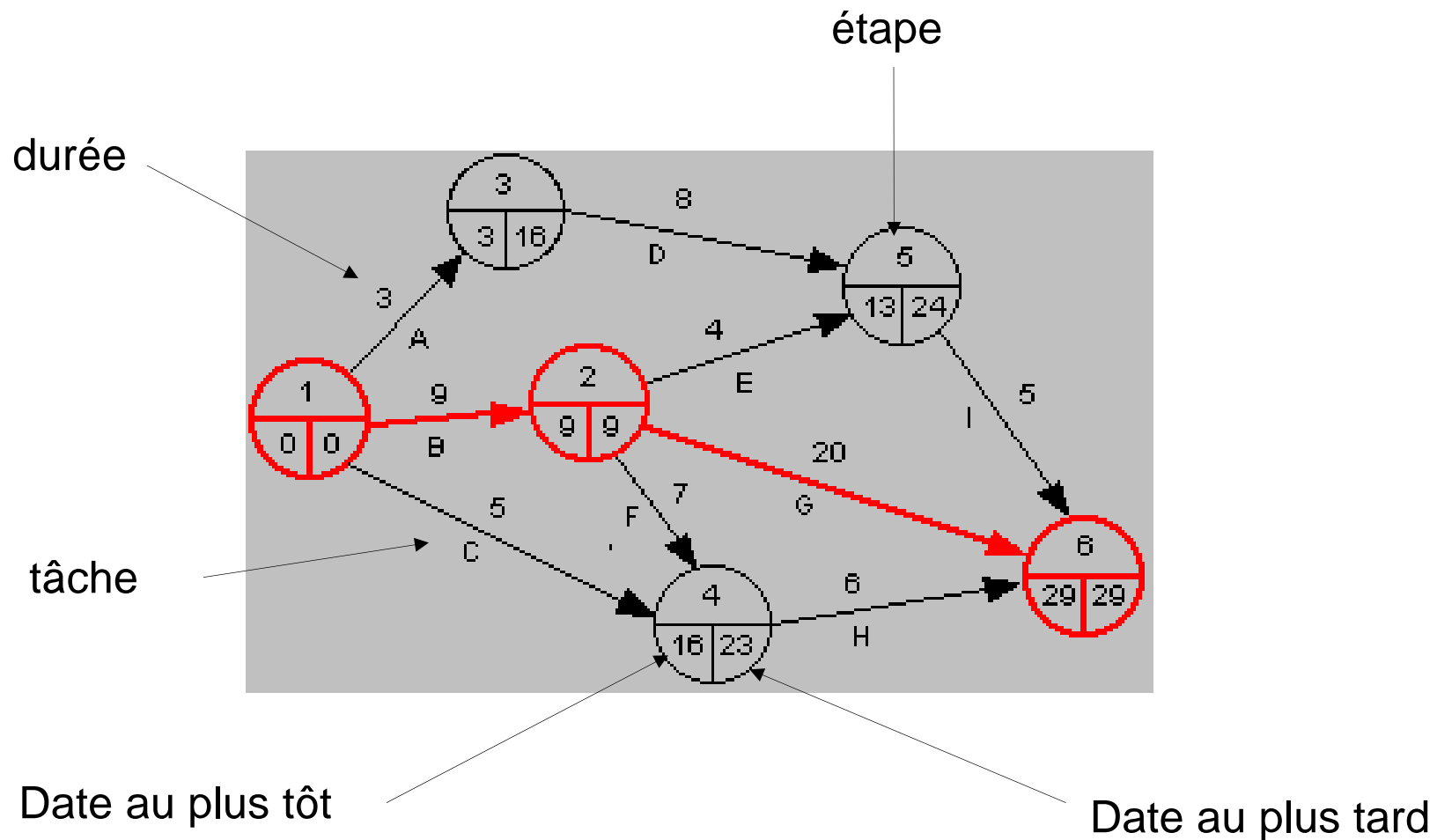
indique pour chaque étape du projet :

dates au plus tôt : quand se terminera le projet ? Quel est le délai nécessaire pour atteindre une étape détermine ?

dates au plus tard : quand doit démarrer le projet pour être achever à temps ? A quelle date chaque étape doit-elle être atteinte pour que le projet ne prenne pas de retard ?

- chemin critique : les tâches qui ne doivent pas prendre de retard sous peine de retarder tout le projet.

Exemple d'un diagramme de Pert



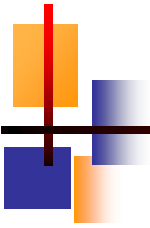


Diagramme de Gantt

Intérêt :

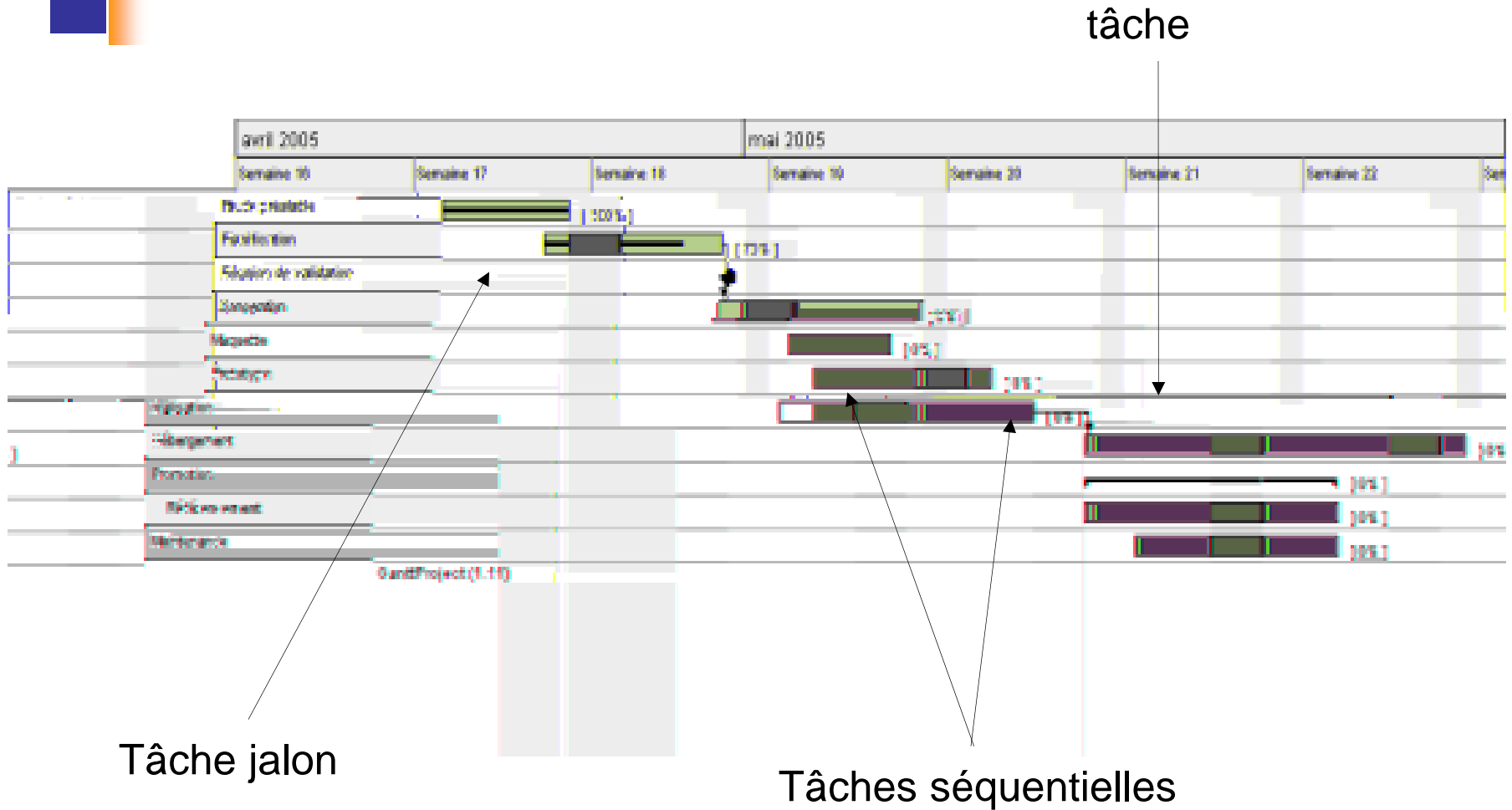
optimiser et planifier l'ordonnancement de tâches ;

représenter graphiquement l'avancement du projet ;

utilisé par la quasi-totalité des chefs de projet dans tous les secteurs :

Accessoirement : bon moyen de communication entre les différents acteurs d'un projet.

Exemple d'un Diagramme de Gantt



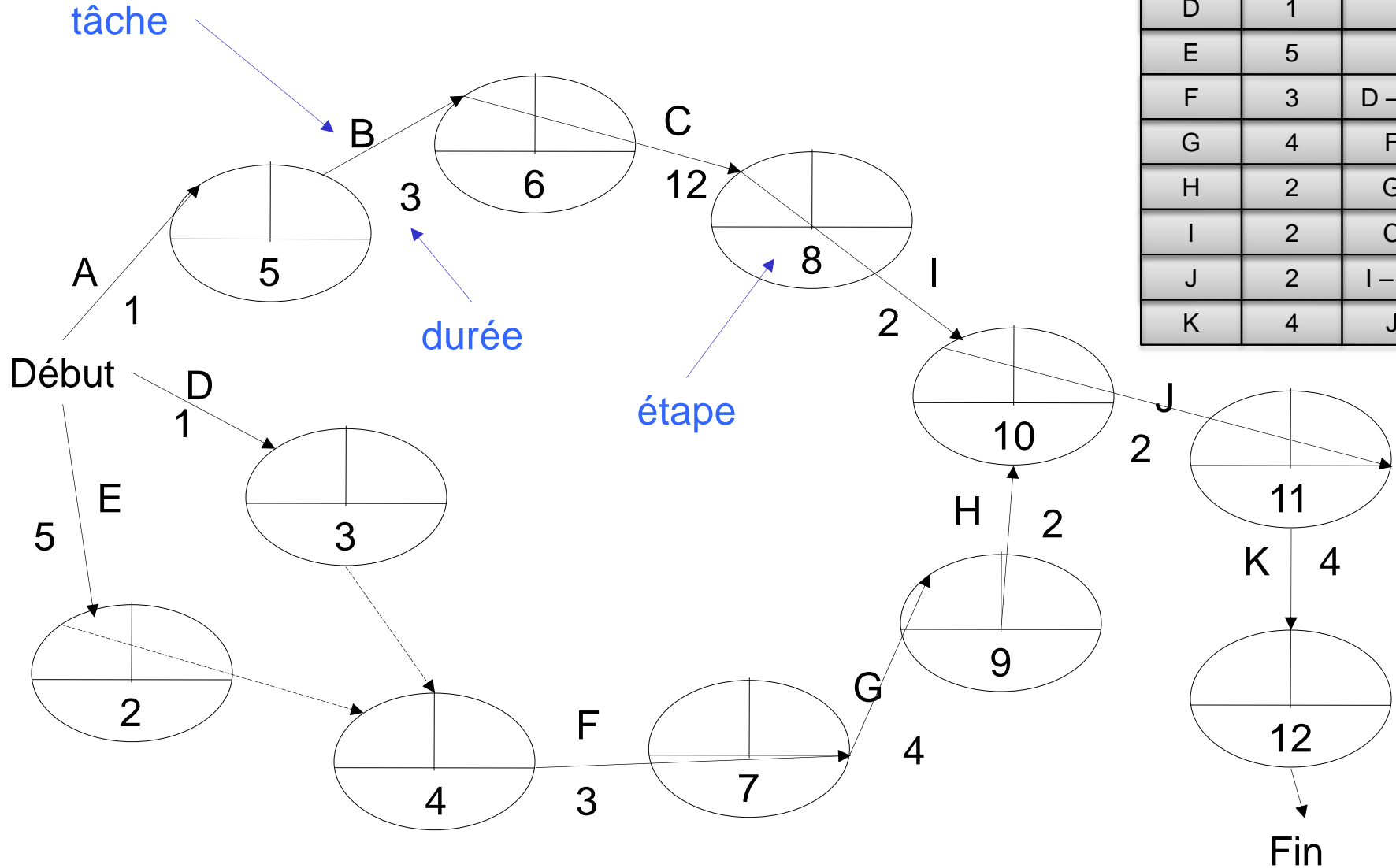


Mise en œuvre de Pert

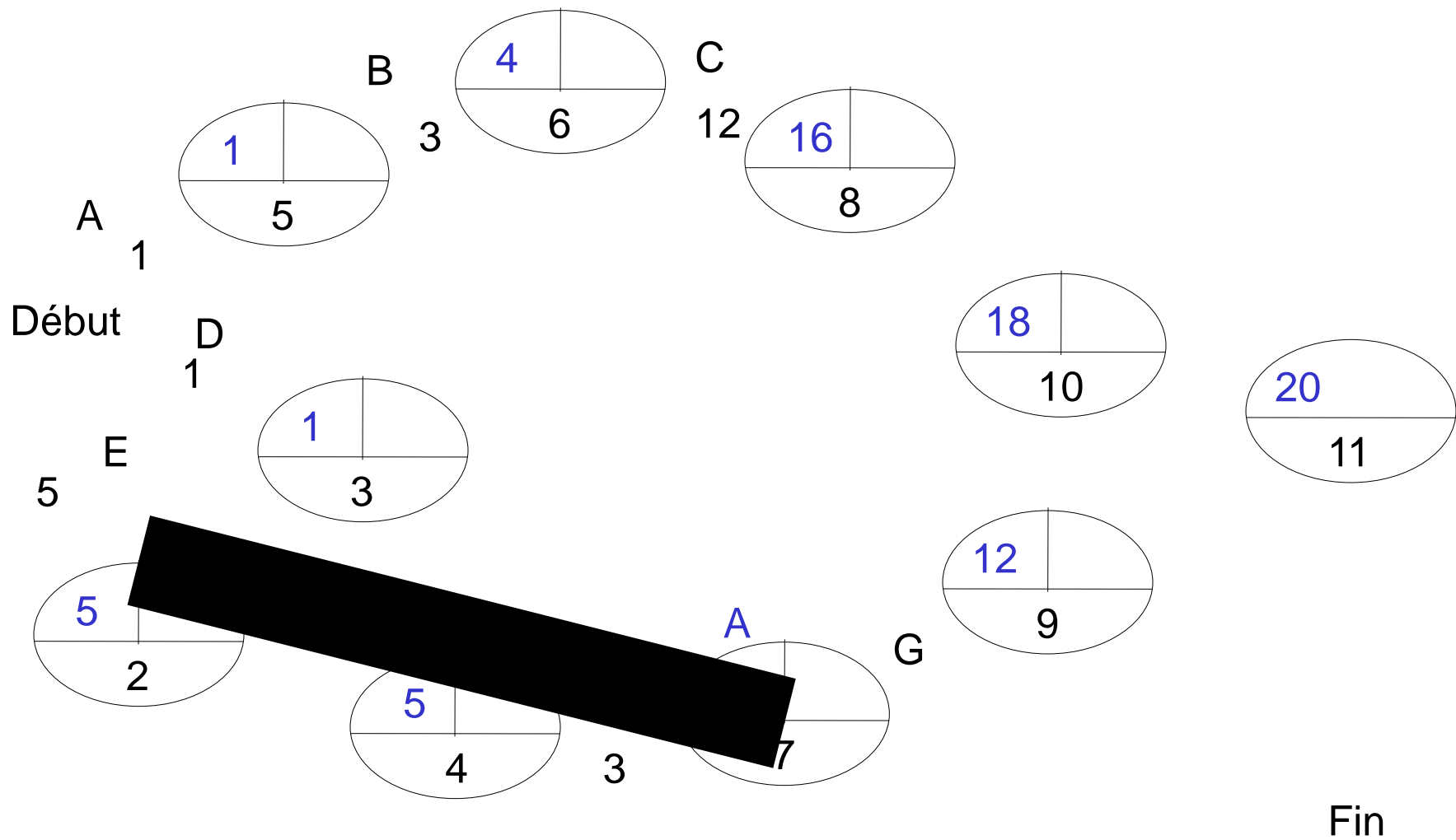
tâche	durée	prédécesseur
A	1	
B	3	A
C	12	B
D	1	
E	5	
F	3	D – E
G	4	F
H	2	G
I	2	C
J	2	I – H
K	4	J

Mise en œuvre de Pert

A	1	
B	3	A
C	12	B
D	1	
E	5	
F	3	D – E
G	4	F
H	2	G
I	2	C
J	2	I – H
K	4	J



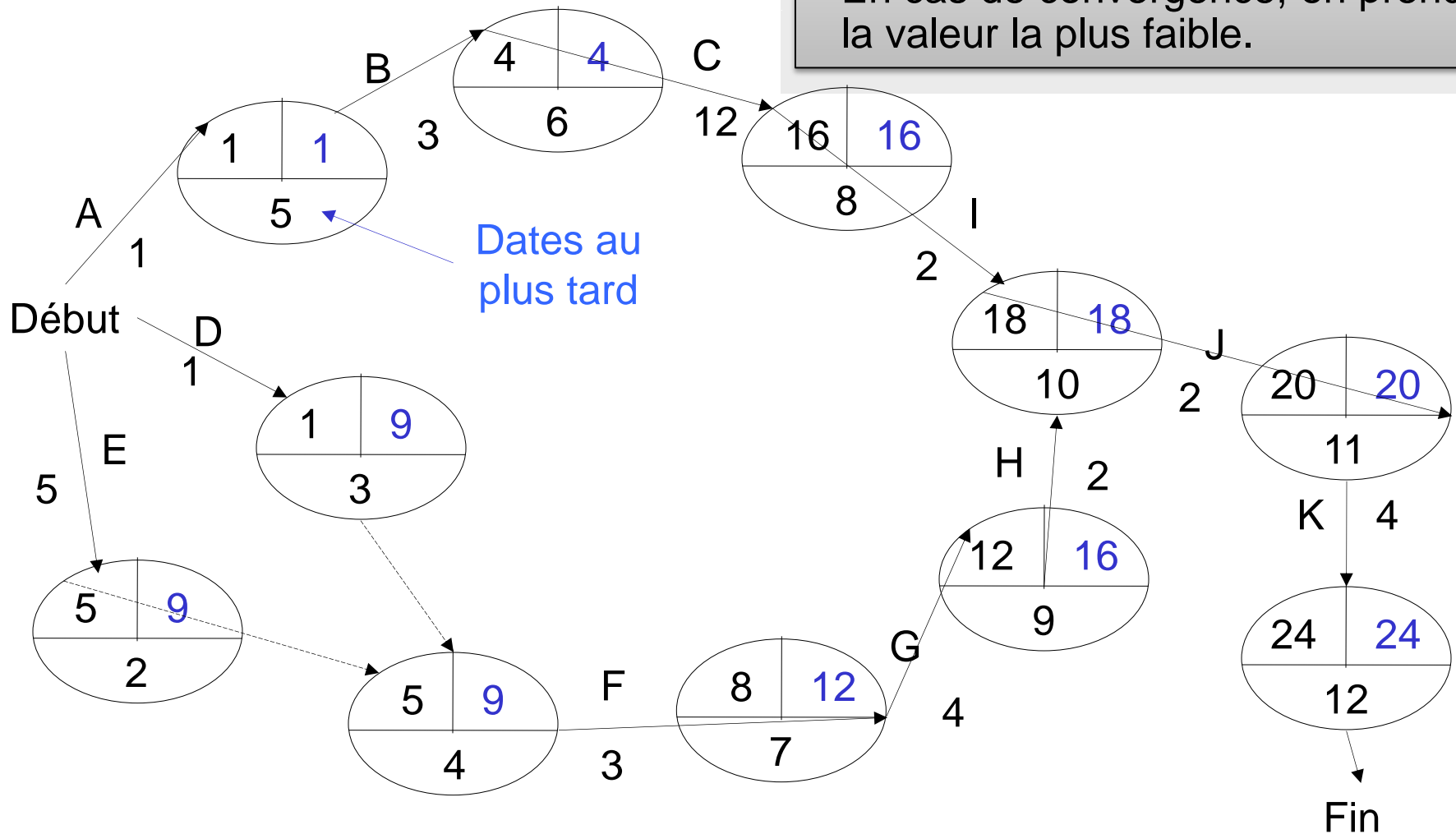
PERT : dates au plus tôt



PERT : dates au plus tard

Calcul de droite à gauche en retranchant la durée de la tâche précédente ;

En cas de convergence, on prend la valeur la plus faible.



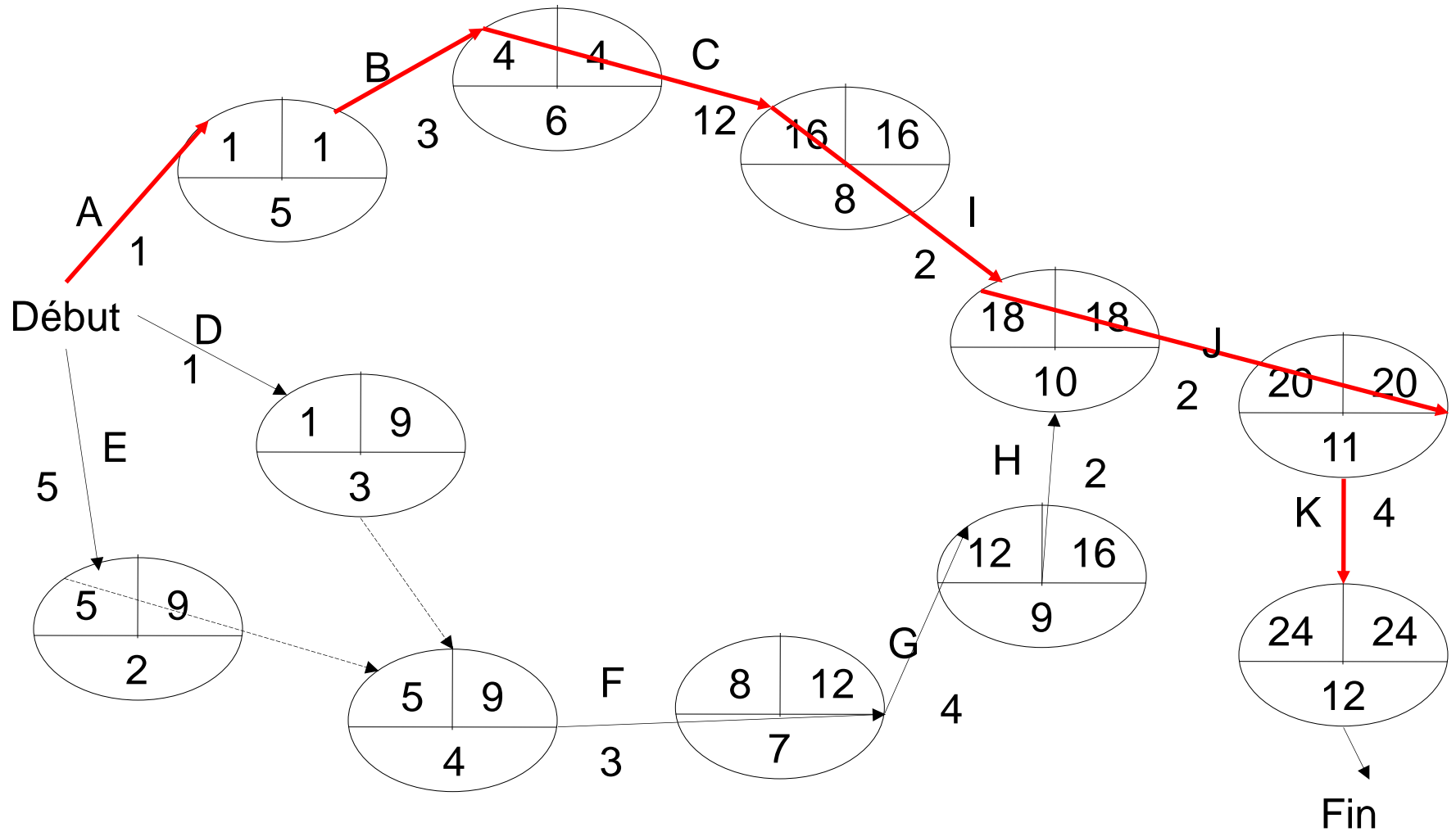


PERT : chemin critique

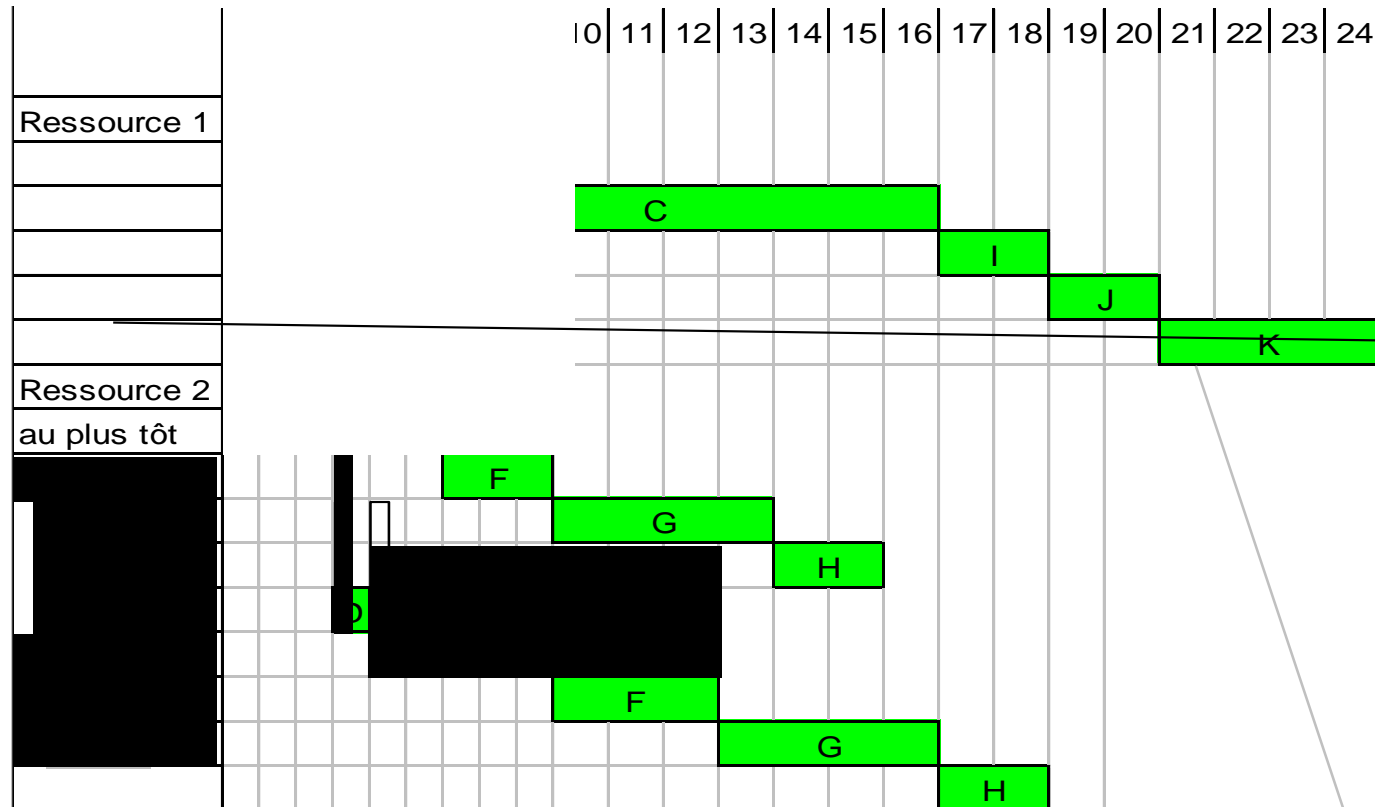
étape	date au plus tôt	date au plus tard	marge
2	5	9	4
3	1	9	8
4	5	9	4
5	1	1	0
6	4	4	0
7	8	12	4
8	16	16	0
9	12	16	4
10	18	18	0
11	20	20	0
12	24	24	0

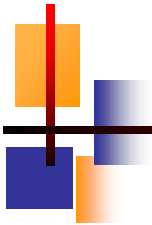
- chemin critique : relie les étapes qui n'ont aucune marge
- Un retard dans une tâche sur le chemin critique entraînera un retard dans le projet entier.

PERT : chemin critique

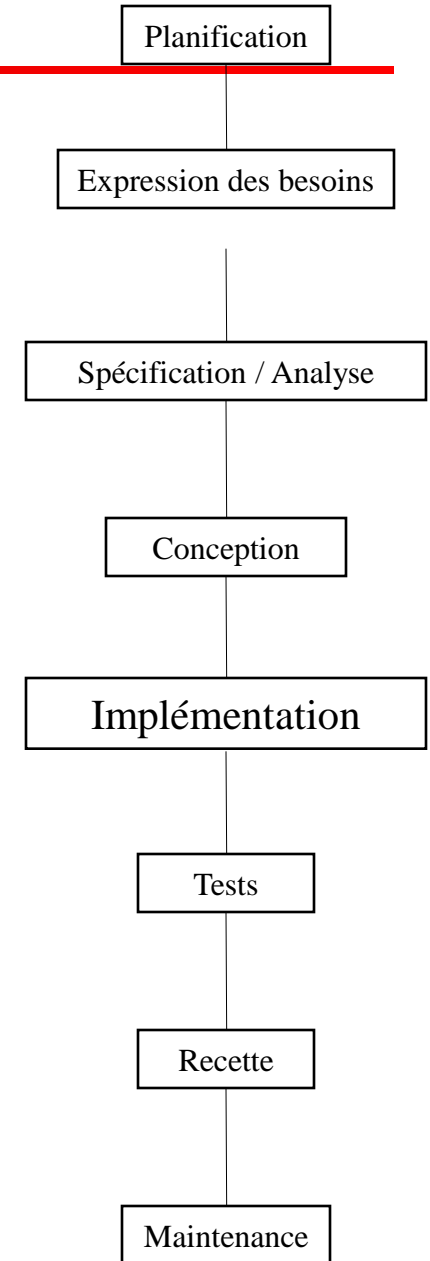
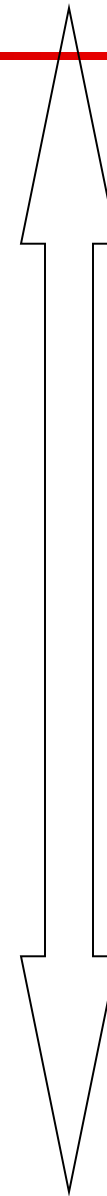


Mise en œuvre de Gantt





Suivi de projet



- Au cours d'un projet, le chef de projet a besoin de mesurer l'avancement du projet pour vérifier la conformité avec la planification établie.
- Il existe de nombreuses mesures parmi lesquelles :
 - Le pourcentage achevé à une date donnée ;
 - La différence entre la quantité de travail estimée et le travail réel ;
 - ...

Le maîtrise des coûts par la technique

Un modèle intégré coûts/délais



INTRODUCTION

Maîtriser les coûts du projet en:

- S'assurant que les dépenses ne dépassent pas les fonds autorisés pour une période donnée;
 - Identifiant les écarts à la planification initiale;
 - Évitant d'inclure des modifications non approuvées;
 - Informant les parties prenantes des modifications approuvées et des coûts engendrés;
 - Surveiller la performance du travail aux dépenses qu'il a entraînées.
-



Système de la valeur acquise – Définition

- Le système de la valeur acquise (VA) fait la comparaison coût/délais.
-

REMARQUE :

Sans un **découpage temporel des coûts** qui correspond aux activités planifiées, le suivi des coûts ne peut pas fournir une information fiable aux fins de contrôle.



Valeur acquise – pré-requis

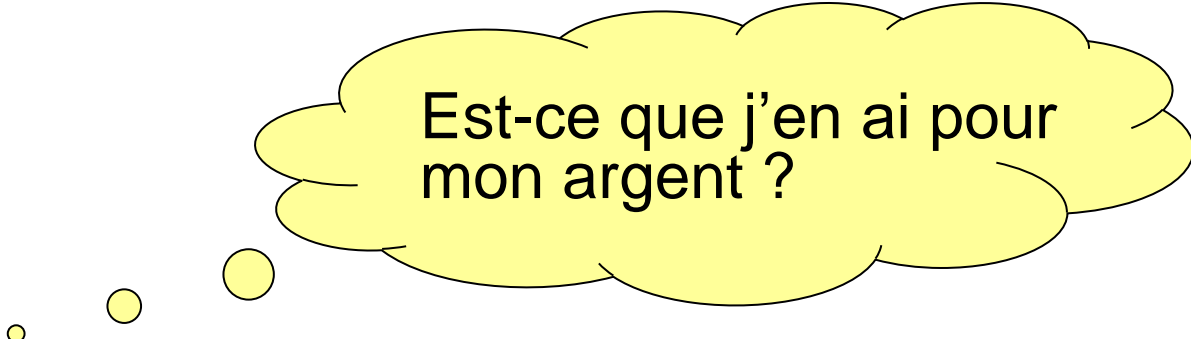
Aperçu du système d'information de projet

Source : Langevin, Y. *et al. Management de projet*, 2007, page 456.



Définition de la valeur acquise

Méthode de mesure de performance qui intègre les mesures de contenu, des coûts et de l'échéancier du projet.



Est-ce que j'en ai pour mon argent ?



Analyse de la valeur acquise – Étapes

1. Définir le travail à l'aide d'une SDP (Structure de découpage du projet)
 2. Développer des calendriers du travail et des ressources
 3. Élaborer un budget réparti dans le temps au moyen des lots de travaux
 4. Recueillir les coûts réels du travail effectué pour réaliser les lots de travaux
 5. Calculer l'écart de prévisions (ED) et l'écart des coûts (EC)
 6. Établir un ratio d'efficacité avec des indices de performance
-



Glossaire des termes

- Valeur planifiée (VP) ou CBTP (coût budgété du travail prévu)
- Valeur acquise (VA) ou CBTE (coût budgété du travail effectué)
- Coûts réels (CR) ou CRTE (coûts réel du travail effectué)
- EC ou **Écart de coûts** (différence entre valeur acquise et coûts réels)
- EP ou **Écart de prévision** ou des délais (différence entre valeur acquise et valeur planifiée)
- IPC ou **Indice de performance des coûts** (ratio de la valeur acquise versus la valeur planifiée)
- IPD ou **Indice de performance des délais** (ratio de la valeur acquise versus les coûts réels)

$$\begin{aligned} EC &= VA - CR \\ EP &= VA - VP \\ IPC &= VA/VP \\ IPD &= VA/CR \end{aligned}$$



Mise en situation : 3 données à calculer

- Programme de gestion de 10 KLOC (quantité de travail)
- La semaine de travail est de 50 heures/semaine
- Le travail est réalisé par 2 personnes
- L'échéancier planifié est de 10 semaines, soit 1 KLOC en moyenne par semaine
- Le coût horaire est de 100 € par heure-personne.

Hypothèse : l'effort est constant tout au long du projet.

① Les coûts budgétés

- ② Les écarts (aussi appelés variations)
 - ③ Les indices de performance
-



Étape 1: Calcul du VP de référence

- **VP (Valeur planifié) = BAF (budget à la fin)**

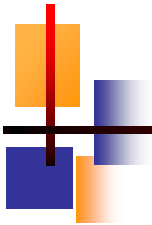
Effort

10 semaines x 50 hrs/sem x 2 personnes = 1000 hrs-pers.

Coût: **VP = BAF initial**

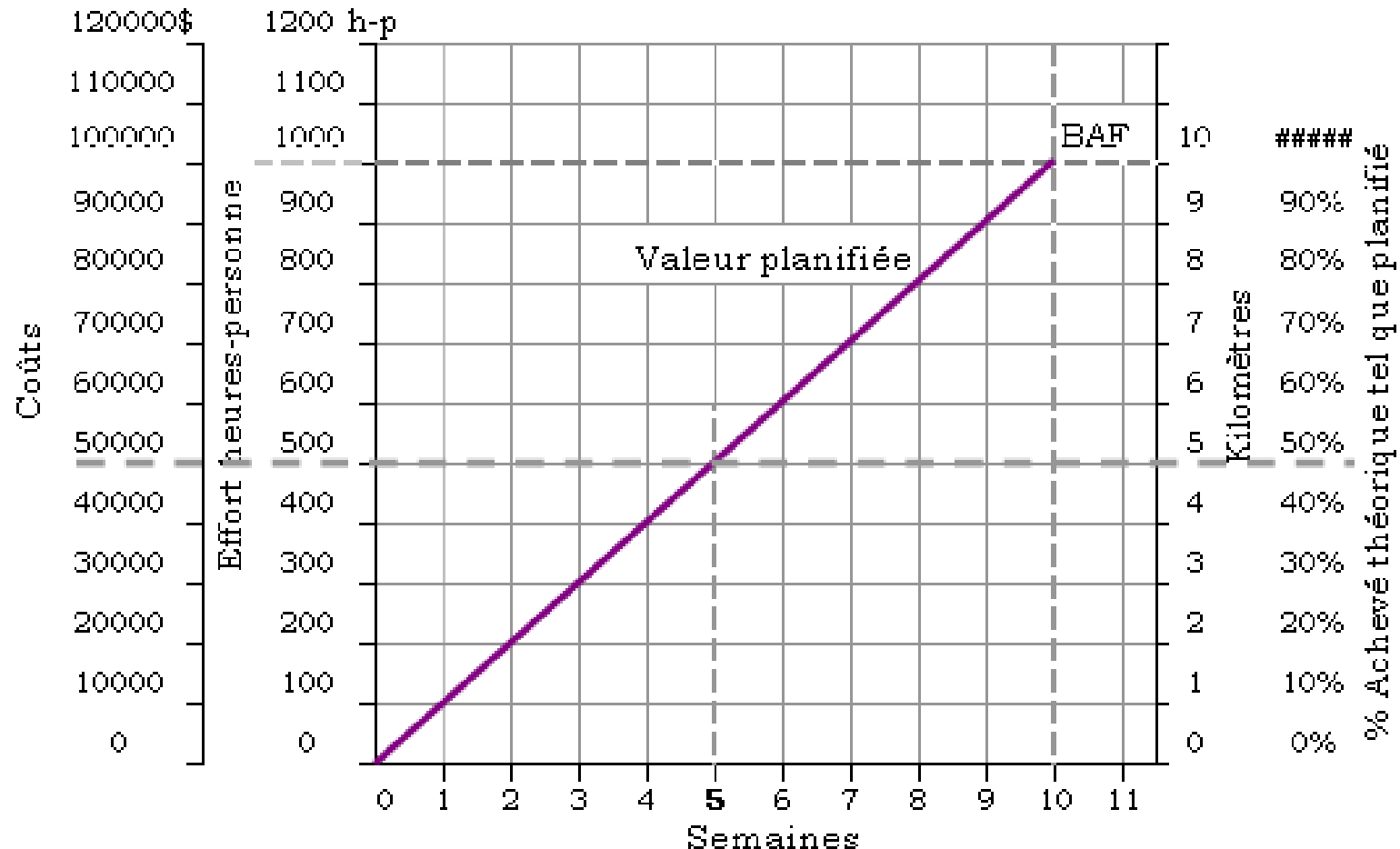
1000 hrs-personne x 100 € /hr-personne = 100 000 €

Le rythme de progression des travaux budgété (estimé)
est de : 10 kLOC / 10 semaines = 1 kLOC/sem.



Représentation graphique « VP »

- Le coût budgété du travail prévu (**CBTP**) ou **V**aleur **P**lanifiée





Situation

Situation :

- L'équipe s'est présentée à la date prévue. Il a fallu 2 semaines pour prendre les arrangements et ententes.
 - Le travail débute avec un retard de 2 semaines.
 - Depuis la reprise, le projet est à 3^{ième} semaine d'exécution au rythme prévu.
-



Étape 2: Calcul de la Valeur acquise à la date d'état

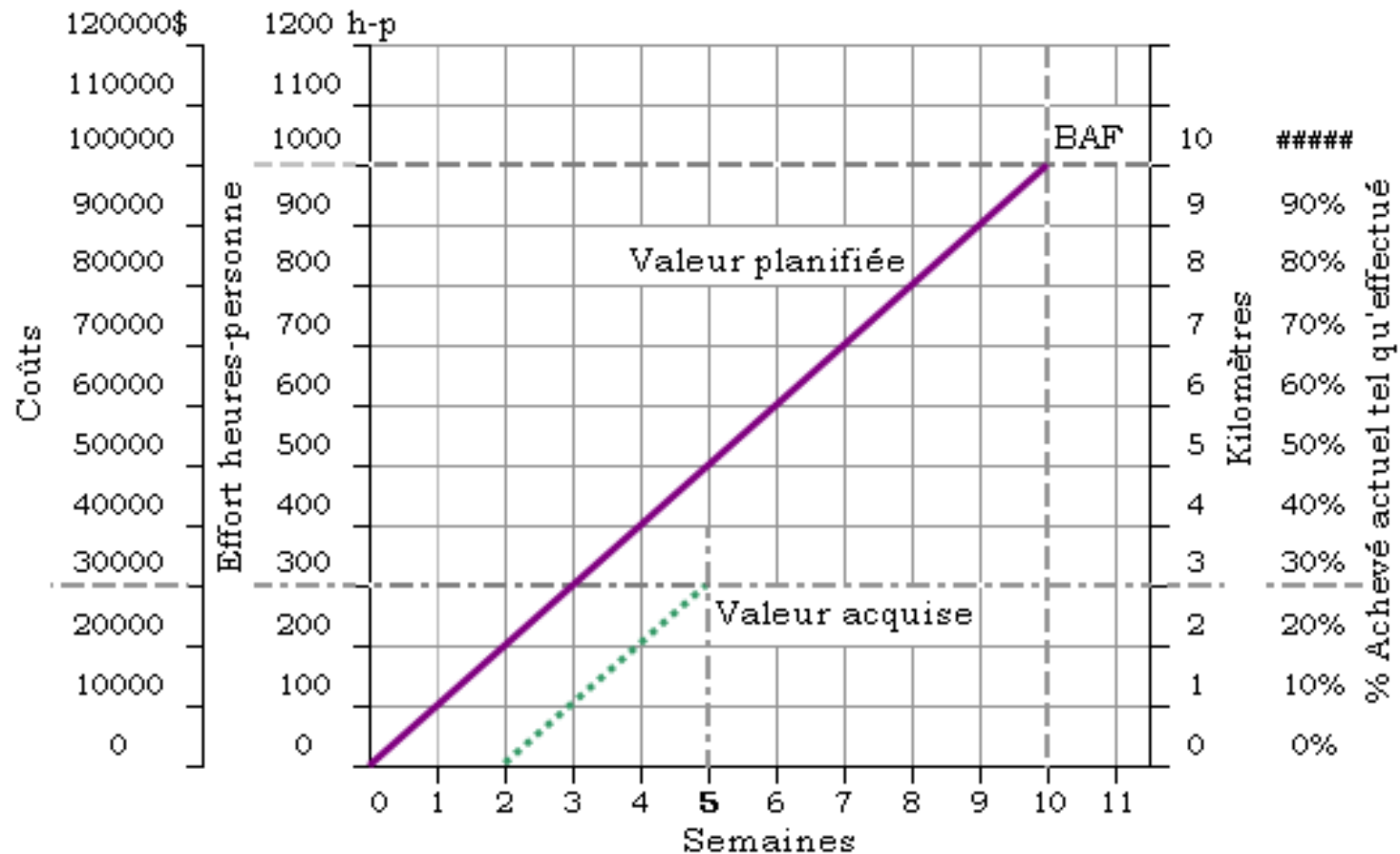
- **VA après 3 semaines**
(**3 kLOC** ou 3 kLOC/ 10 kLOC = **30%**)

VA = BAF(**budget à la fin**) * % Achievé

$$100\,000\text{ €} \times 0,3 = 30\,000\text{ €}$$

Représentation graphique de la « VA »

- Le coût budgété du travail effectué (**CBTE**) ou **V**aleur **A**cquise





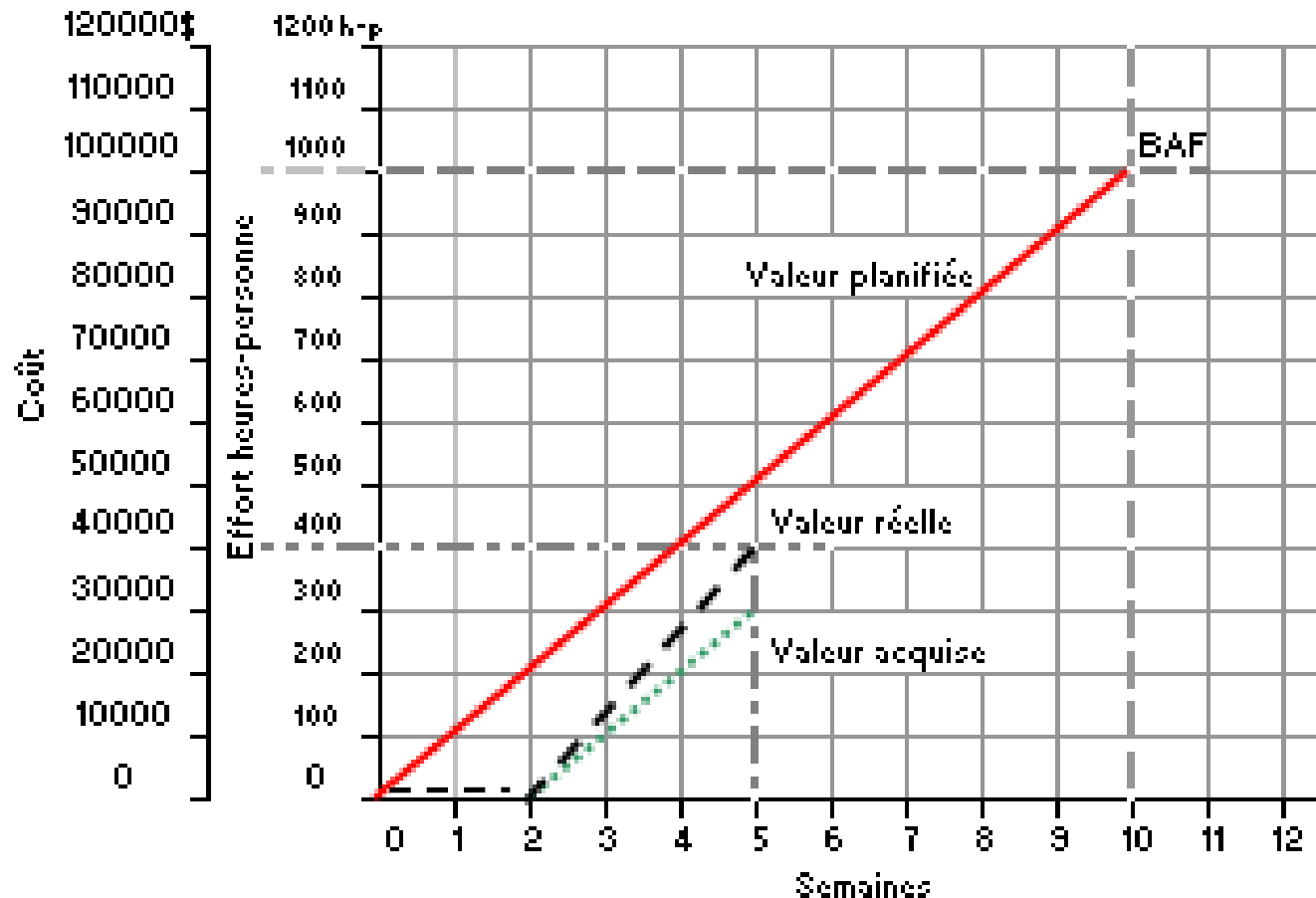
Étape 3: Calcul des coûts réels

- Selon la sommation des relevés aux feuilles de temps des ressources,
l'effort est de **400 h-p** à 5ième semaine.

Donc le coût réel est $400 \text{ h/p} * 100 = 40\,000 \text{ €}$

Représentation graphique « CR » »

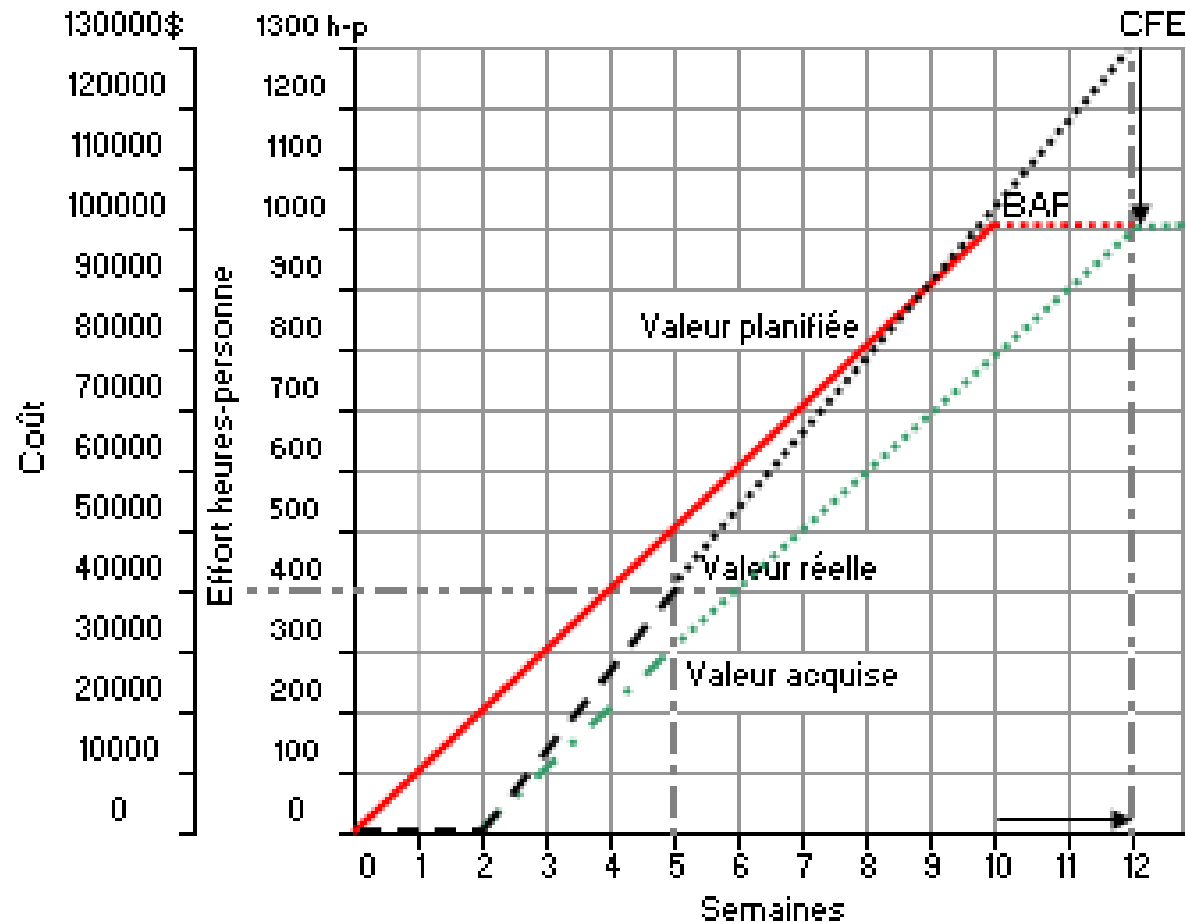
- Le coût réel du travail exécuté (**CRTE**) ou **C**oût **R**éel



- Si l'on fait l'hypothèse que les trois premières semaines sont représentative de ce qui est à venir.
 - Quelles conclusions pouvons nous tirer ?
Le coût final estimé (CFE) est ?
La durée finale est ?
-

Représentation graphique du CFE

- Le coût final estimé (**CFE**)





Plan de la séance

- ① Les coûts budgétés
- ② Les écarts ou variations
- ③ Les indices de performance

Écart de coûts et Écart de délais
EC et ED



Comparaison entre les réels et le planifié

Deux méthodes d'analyse des écarts à partir des 3 données de base (VP, VA, CR)

1. **L'Écart des délais (ED), aussi appelé Écart des prévisions (EP):** *Est-ce que l'affectation est en accord avec les prévisions de l'échéancier?*
 2. **L'écart des coûts (EC):** *Est-ce que le travail effectué a coûté plus ou moins que prévu dans le budget à ce moment du cycle de vie du projet?*
-



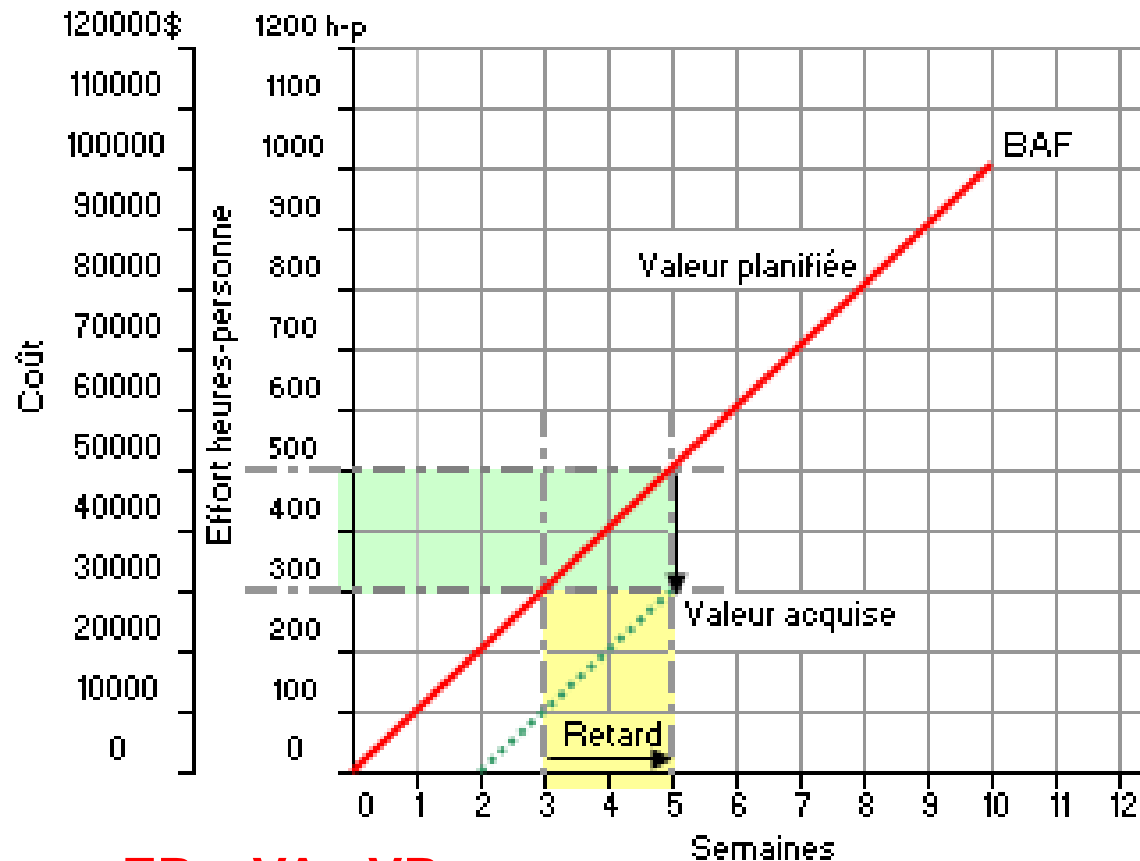
Écart des prévisions (EP)

L'Écart des prévisions (EP): *Est-ce que les travail réalisé est en accord avec les prévisions de l'échéancier?*

L'écart des prévisions

- Valeur Acquise (VA) vs. Valeur Planifiée (VP)

- En temps
- En coûts
- En effort



$$EP = VA - VP$$

Écart de coûts: EC

L'écart des coûts (EC): *Est-ce que le travail effectué a coûté plus ou moins que prévu dans le budget à ce moment du cycle de vie du projet?*

$$EC = VA - CR$$

Si :



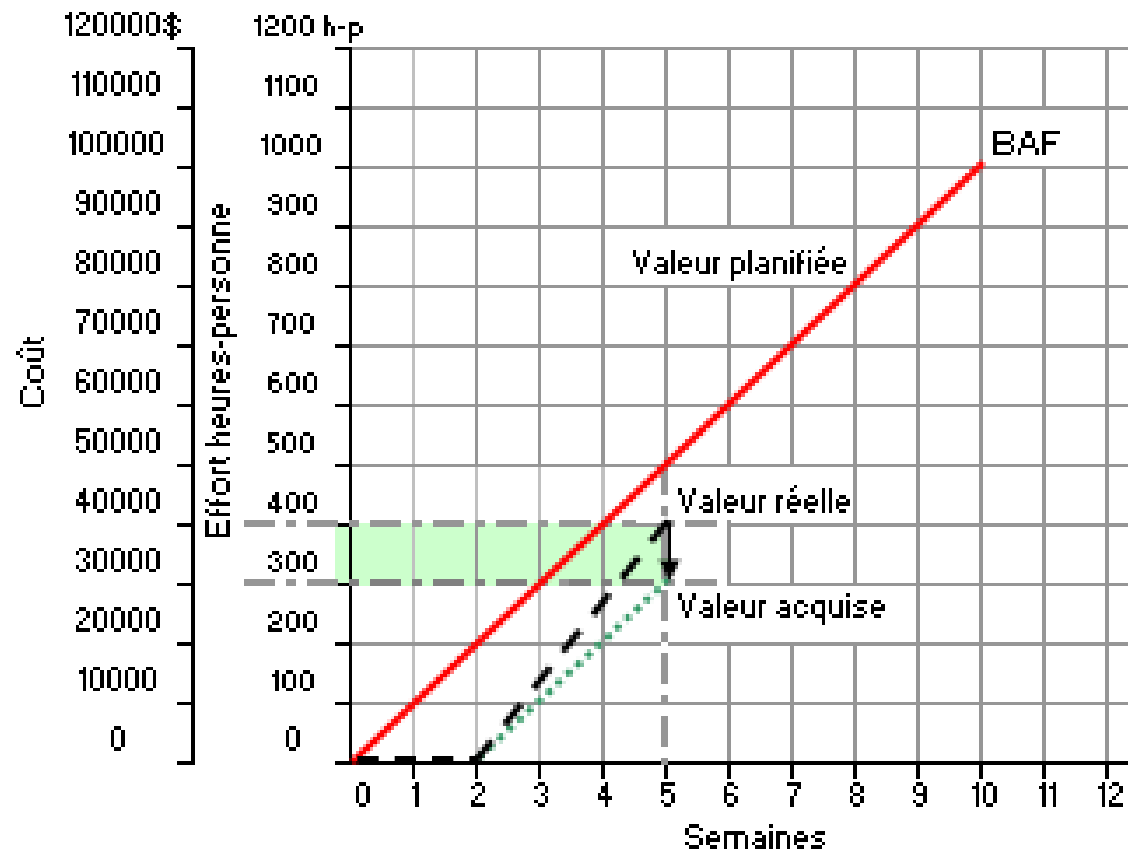
EC est + le coût de la tâche se situe en **dessous** du montant budgété et vos coûts réels sont inférieurs aux coûts prévus.



EC est - le coût de l'affectation est en **dépassement** du budget et vos coûts réels sont supérieurs aux coûts prévus.

L'écart de coûts « EC »

- Coût Réel (CR) vs. Valeur Acquisée (VA)



$$EC = VA - CR$$



Plan de la séance

- ① Les coûts budgétés
- ② Les écarts ou variations
- ③ Les indices de performance

Deux Indices de performance

IPC Indice de performance de coûts
et

IPD Indice de performance de délais

L'indice de performance des coûts (IPC)

- **IPC** ou **indice de performance de coût**, montre le rapport entre les coûts budgétés (ou planifiés) du travail effectué et les coûts réels du travail effectué, jusqu'à la date d'état du projet.

$$\text{IPC} = \text{VA} / \text{CR}$$

Si :

IPC est plus grand que 1 (>1) montre un indice favorable du travail réalisé à date pour le projet



L'indice le plus fiable et le plus utilisé

L'indice de performance des délais (IPD)

- L'indice de performance des délais (aussi appelé Indice de performance des prévisions) est utilisé pour la prévision de la date d'achèvement du projet.

$$\text{IPD} = \text{VA} / \text{VP}$$

Si :

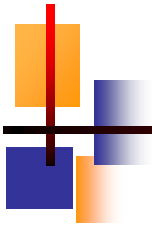


IPD est inférieur à 1 (<1) montre que l'échéancier du projet est compromis.



Exercice

- Au mois 9 d'un projet, les infos suivantes sont disponibles:
 - Coûts réels = 2 000 €
 - Valeur acquise = 2 100 €
 - Valeur planifiée = 2 400 €
 - Calculez EP et EC ?
 - Au jour 51, un projet a :
 - une valeur acquise de 600 €
 - coûts réels = 650 €
 - valeur planifiée = 560 €
 - Calculez EP, EC, IPC et IPD du projet au jour 51 ?
 - Interprétez les résultats.
-



Structure des systèmes d'information

Glossaire des termes



Structure des systèmes d'information

Analyse des écarts – Diagramme coûts/délais

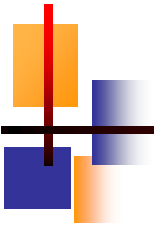
Analyse de la valeur acquise

Tâche	Travail Estimé	Travail Réel	Date Achiev. Estimée	Date Achiev. Effective
1	5	10	25/01/2012	01/02/2012
2	25	20	15/02/2012	15/02/2012
2	120	80	15/05/2012	01/07/2012
4	40	50	15/04/2012	15/04/2012
5	60	50	01/07/2012	
6	80	70	01/09/2012	

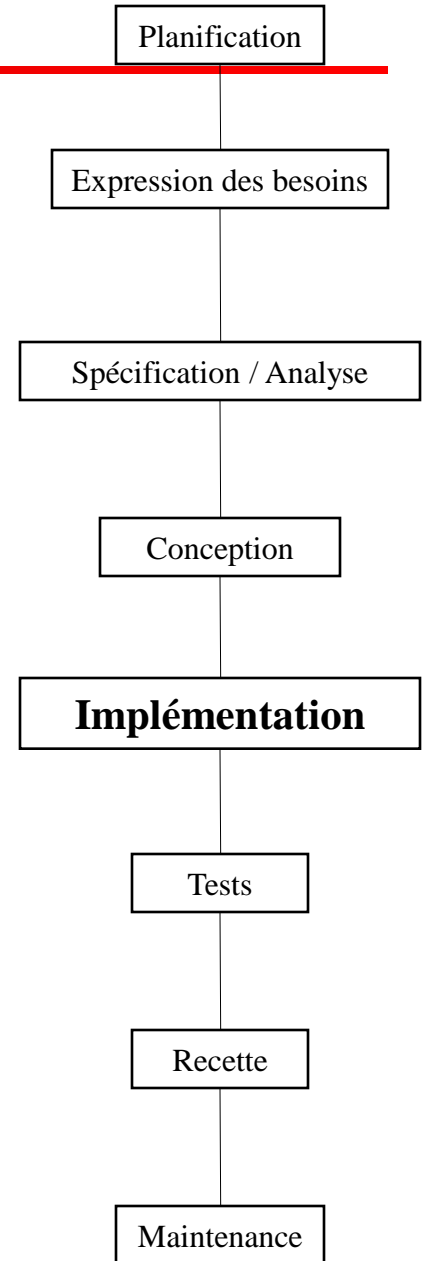
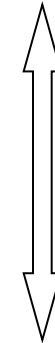
DATE d'aujourd'hui

02/07/2012

CBA	Cout Budgété Achievé	330	Estimation du travail pour le projet entier
VA	Cout Budgété Travail Effectué	190	somme travail estimé tâches achevées
VP	Cout Budgété Travail prévu	250	somme travail estimé tâches devant être achevées
VA %	Valeur Acquis	0,58	CBTE /CBA = % d'ach.
IPD	Indic. Perform. Tempo.	0,76	CBTE/CBTP
EP	Variance Echaéancier	-60	CBTE-CBTP
CR	Cout Réel Travail Effec.	160	Somme travail réel
IPC	Indic. Écart Cout	1,19	CBTE/CRTE
EC	Variance Cout	30	CBTE-CRTE



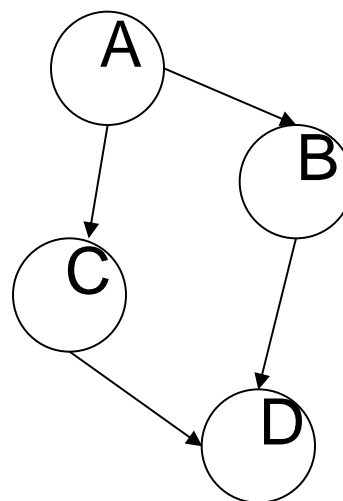
Métriques



Métriques de Mc Cabe (1/4)

- Mesure la complexité structurelle du code ;
- Métrique basée sur un graphe de contrôle.

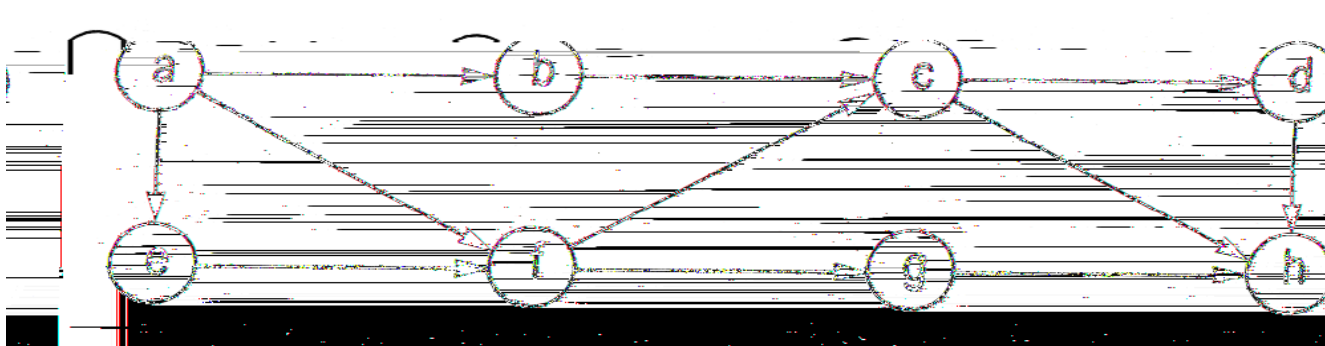
```
// A
...
if( ... ) {
    // B
    ...
} else {
    // C
    ...
}
// D
...
```



Métriques de Mc Cabe (2/4)

$$C = a - n + 2P$$

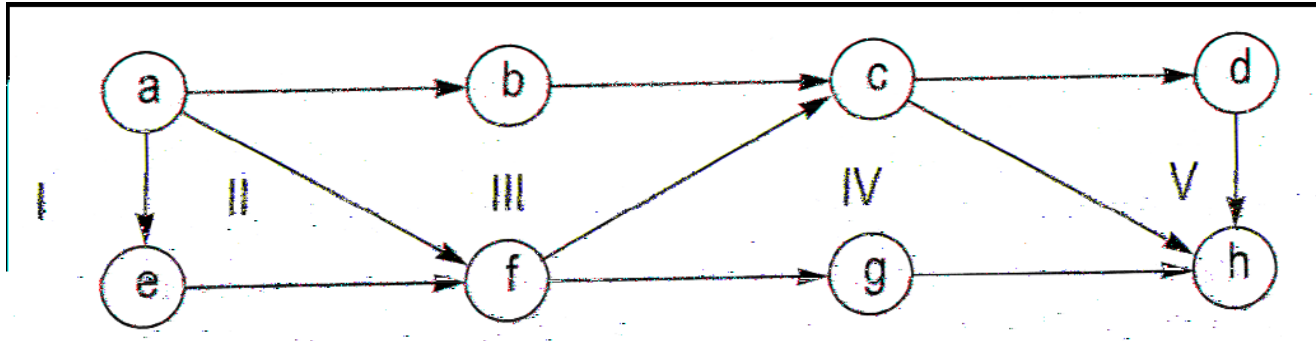
- a = nombre d'arcs du graphe de contrôle ;
- n = nombre de noeuds du graphe de contrôle ;
- p = nombre de composantes connexes.



- a = 11
- n = 8
- p = 1
- $C = 11 - 8 + 2 = 5$

Métriques de Mc Cabe (3/4)

C = nombre de faces (en comptant la face externe) = 5

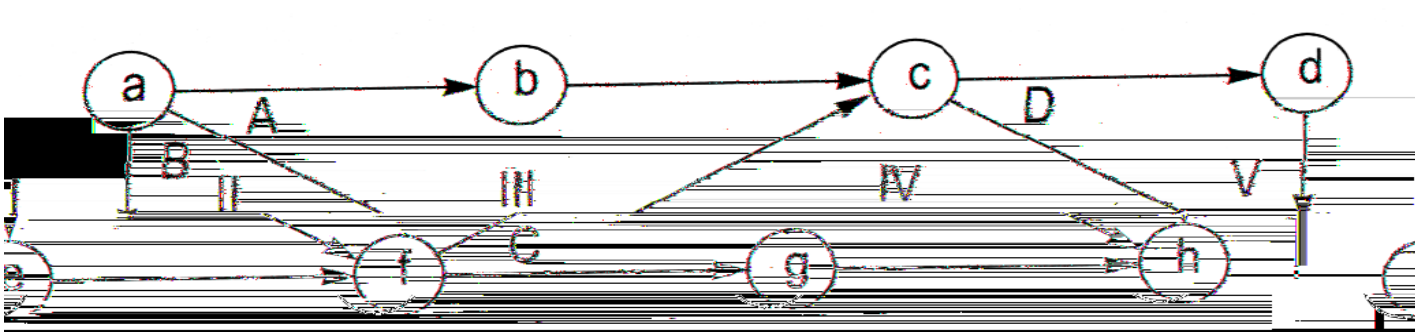


Métriques de Mc Cabe (4/4)

Nombre de décisions du code :

C =

- une instruction if compte pour 1 ;
- une boucle compte pour 1 ;
- une instruction case compte 1 décision de moins que le nombre d'alternatives.



- 4 décisions
- $C = 4 + 1 = 5$

Mesure la complexité d'un programme :

- utilise le code source ;
- basée sur les opérandes et les opérateurs utilisés dans le code ;
- utilisée durant le développement pour suivre les tendances de complexité ;
- permet de calculer :
 - la taille de l'implémentation d'un algorithme ;
 - le niveau de difficulté (propension à l'erreur) ;
 - l'effort à l'implémentation ou pour comprendre un programme ;
 - le temps pour implémenter ou pour comprendre un programme ;
 - le nombre de bugs fournis.



Métriques de Halstead (2/5)

Les opérandes :

- les mots réservés qui identifient le type (char, int, ..., void) ;
- les nombres, les caractères et les constantes ;
- toutes les marques qui ne sont pas des mots réservés.

Les opérateurs :

- Les mots réservés (public, private, if, while, do, new, try, catch, return, ...) ;
- mots réservés qui qualifient la classe de stockage (static, abstract, ...) ;
- les nombres, les caractères et les constantes ;
- Les opérateurs (=, +, /, ==, ...).



Métriques de Halstead (3/5)

Le volume d'un programme (V) indique la taille de l'implémentation d'un algorithme :

$$V = N * \log_2(n)$$

- où $N = N1 + N2$ est la somme du nombre total d'opérandes et d'opérateurs ;
- $n = n1 + n2$ est la somme du nombre d'opérandes et d'opérateurs **uniques**.

$20 < V$ pour une fonction < 1000

$100 < V$ pour un fichier < 8000

N peut être estimé avec $N_{est} = n1 * \log_2(n1) + n2 * \log_2(n2)$

si N_{est} est supérieur à 30% à N il vaut mieux renoncer aux autres mesures de Halstead

Un exemple de métriques de Halstead (4/5)

Les opérateurs (n1=8) ; = while > + - print ()

Les opérandes (n2=6) float x y z 0 1

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

Les opérateurs
(N2=15) :

• ;	6
• =	3
• while	1
• >	1
• +	1
• -	1
• print	1
• ()	1

Les opérandes (N1=15) :

• float	3
• x	3
• y	2
• z	4
• 0	2
• 1	1

$$V = N * \log_2(n) = (N1+N2) * \log_2(n1 + n2) = 30 * \log_2(14)$$

$$Nest = 8 + \log_2(8) + 6 * \log_2(6)$$

Volume potentiel (V^*) est la taille minimale d'une solution au problème :

$$V^* = (2 + n2^*) * \log_2(2 + n2^*)$$

- où $n2^*$ est l'ensemble de valeurs minimales pour n'importe quelle implémentation = valeurs qui ne sont pas initialisées à l'intérieur de l'algorithme (valeurs lues par le programme, passées en paramètres, variables globales).

Niveau d'implémentation L (dans quelle mesure l'implantation actuelle est proche de l'implantation optimale) :

$$L = V^* / V$$

Method Suite for Object Oriented Design :

Weighted Method per Class = WMC = $1/n \sum c_i * M_i$

- où M_i représente la i ème méthode dont la complexité est mesurée par c_i .

Depth of Inheritance Tree = DIT = distance maximale entre un noeud et la racine d'un arbre d'héritage.

Number Of Children = NOC = nombre de sous-classes dépendant directement d'une classe donnée.

Coupling Between Object = CBO = nombre de classes couplées : deux classes sont couplées si les méthodes déclarées dans l'une utilisent des méthodes ou instancie des variables définies dans l'autre classe.

Method Suite for Object Oriented Design :

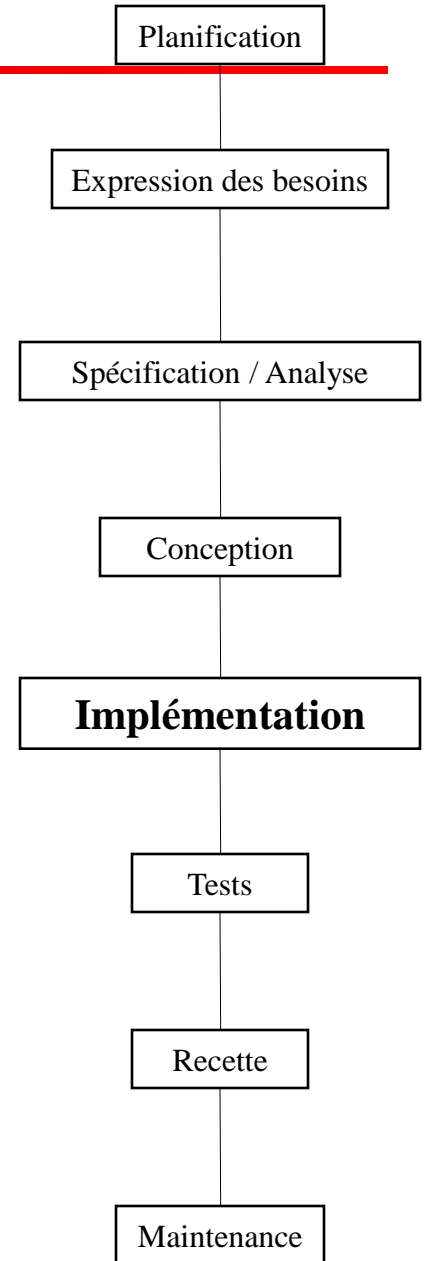
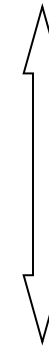
Response For Classe = RFC = ensemble des méthodes qui peuvent être exécutées en réponse à un message reçu par une instance de la classe.

Lack of COhesion in Methods = LCOM = $\max(|P| - |Q|, 0)$

- où P est l'ensemble des paires de (li, lj) ayant une intersection vide ;
- Q est l'ensemble des paires de (li, lj) ayant une intersection non vide
- Li est l'ensemble des variables d'instance utilisées par la méthodes i.



La gestion des versions



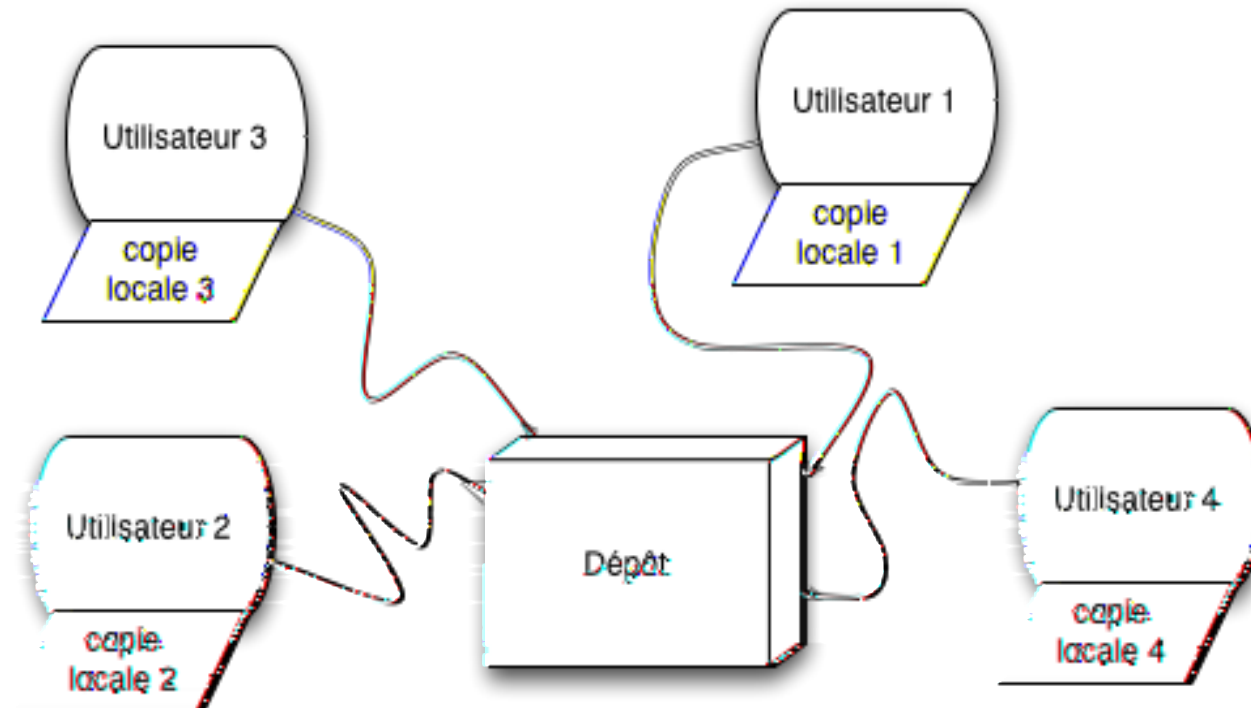


Les logiciels de gestion de versions

- Il existe plusieurs logiciels de gestion de version : CVS, SVN (SubVersion), GIT...
- Subversion a été conçu pour remplacer CVS (reprise des concepts mais changement dans l'implantation).
- Subversion gère les fichiers et les répertoires, ainsi que les changements qu'on y apporte :
 - On peut revenir à d'anciennes versions
 - Examiner la façon dont vos données ont évolué.
- Subversion fonctionne sur un grand nombre de systèmes d'exploitation
- Son interface première est en ligne de commande. Afficher résultats de l'exécution et résultats attendus ;

Subversion en réseau

- Subversion peut fonctionner en réseau :



svn mkdir

- Le dossier trunk contient la « ligne principale » du développement.
- Le dossier Branches contiendra les copies alternatives (ou branches).
- Le dossier tags pour les versions étiquetées.

```
$ svn list file:///var/svn/depot  
/trunk  
/branches  
/tags
```

- Création du dépôt :

```
$ svn import mon-arborescence file:///var/svn/nouveau-depot/un/projet \  
-m "Import initial"
```

```
Ajout mon-arborescence/truc.c
```

```
Ajout mon-arborescence/machin.c
```

```
Ajout mon-arborescence/sous-repertoire
```

```
Ajout mon-arborescence/sous-repertoire/bidule.h
```

-m = message d'explication à joindre à la commande.

- Parcours du dépôt :

```
$svn list file:///var/svn/nouveau-depot/un/projet  
truc.c  
machin.c  
sous-repertoire/
```



Cycle de travail typique

1. Mettre à jour votre copie de travail.
 - **svn update**
2. Faire des changements.
 - **svn add**
 - **svn delete**
 - **svn copy**
 - **svn move**
3. Examiner les changements effectués.
 - **svn status**
 - **svn diff**
4. Éventuellement annuler des changements.
 - **svn revert**
5. Résoudre les conflits (fusionner les modifications).
 - **svn update**
 - **svn resolve**
6. Propager les changements.
 - **svn commit**

3. Examiner les changements effectués.

- **svn status**

? gribouillage.c # le fichier n'est pas suivi en versions

A bazar/pognon/machin.h # le fichier sera Ajouté

C bazar/pognon/tas.c # le fichier entre en Conflit avec une mise à jour

D bazar/poisson.c # le fichier sera supprimé (Deletion en anglais)

M truc.c # le contenu de truc.c a subi des Modifications



Revert

4. Éventuellement annuler des changements.
 - **svn revert**

```
$ svn status truc
? truc
$ svn add truc
A truc
$ svn revert truc
'truc' réinitialisé
$ svn status truc
? truc
```



Résoudre les conflits

5. Résoudre les conflits (fusionner les modifications).

- **svn update**
- **svn resolve**

```
$ svn update
```

```
U INSTALL
```

```
G LISEZMOI
```

```
Conflit découvert dans 'machin.c'.
```

```
Sélectionner : (p) report, (df) diff complet, (e) édite, (h) aide pour plus d'options
```

```
:
```

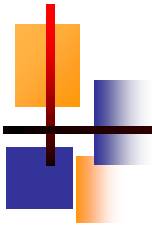
```
$ svn resolve --accept working sandwich.txt
```



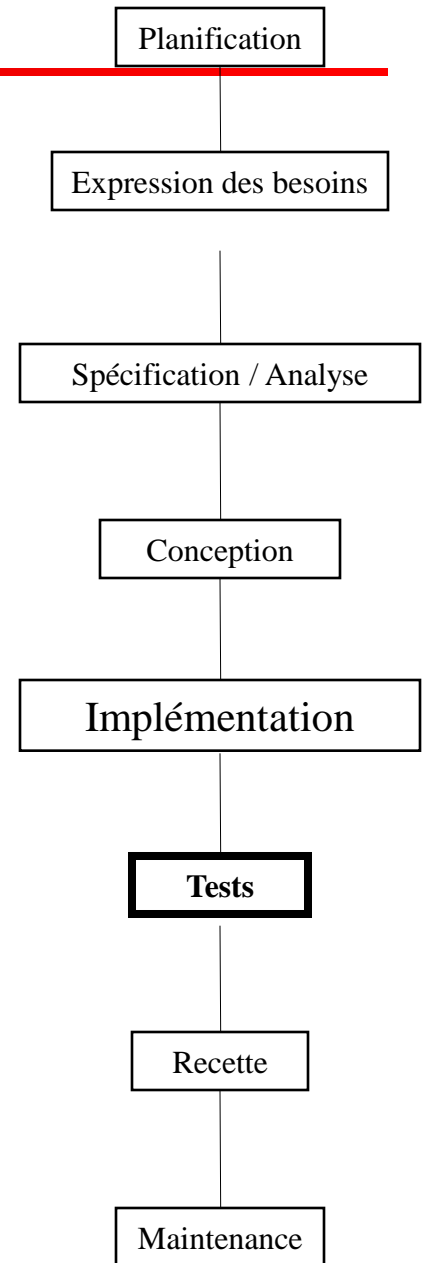
Propager les changements

6. Propager les changements.
 - **svn commit**

```
$ svn commit -m "J'ai corrigé le nombre de tranches de fromage."  
Envoi sandwich.txt  
Transmission des données .  
Révision 3 propagée.
```



Test logiciel





Définition des tests logiciel

Tester un logiciel : Exécuter le logiciel avec un ensemble de données réelles

- Il faut confronter résultats de l'exécution et résultats attendus ;
- Impossibilité de faire des tests exhaustifs :
 - Ex : 2 entiers sur 32 bits en entrée : 2^{64} possibilités. Si 1ms par test, plus de 10^8 années nécessaires pour tout tester
- Choix des cas de tests :
 - Il faut couvrir au mieux l'espace d'entrées avec un nombre réduit d'exemples ;
 - Les zones sujettes à erreurs nécessitent une attention particulière.



Les tests fonctionnels

Identification, à partir des spécifications, des **sous-domaines** à tester :

- produire des cas de test pour chaque type de sortie du programme ;
- produire des cas de test déclenchant les différents messages d'erreur.

Objectif : disposer d'un ensemble de cas de tests pour tester le programme complètement lors de son implémentation

Test “boîte noire” parce qu'on ne préjuge pas de l'implémentation :

- identification possible des cas de test pertinents avant l'implémentation ;
- utile pour guider l'implémentation.



Exemple de tests fonctionnels

Problème : créer un ensemble de cas de test pour un programme qui :

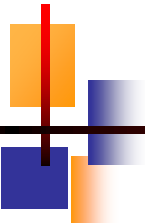
- prend trois nombres a , b et c ;
- les interprète comme les longueurs des côtés d'un triangle ;
- retourne le type de triangle.

Exemple de tests fonctionnels

<u>Sous-domaines</u>	<u>données de test</u>
Scalènes :	$\{ 3, 4, 5 \}$ $\{ 5, 4, 3 \}$ $\{ 4, 5, 3 \}$
Isocèles :	$\{ 5, 5, 8 \}$ $\{ 8, 8, 5 \}$ $\{ 5, 8, 5 \}$ $\{ 8, 5, 8 \}$ $\{ 8, 5, 5 \}$ $\{ 5, 8, 8 \}$
Équilatéraux :	$\{ 5, 5, 5 \}$
Pas un triangle :	$\{ 6, 4, 2 \}$ $\{ 4, 6, 2 \}$ $\{ 4, 2, 6 \}$
Entrées incorrectes :	$\{ -1, 4, 2 \}$ $\{ 4, -6, -2 \}$ $\{ 0, 0, 0 \}$


Les **matrices de test** permettent de :

- formaliser l'identification des sous-domaines à partir des conditions des spécifications ;
- indiquer systématiquement si les combinaisons de conditions sont vraies ou fausses ;
- toutes les **combinaisons possibles** de V et de F seront examinées.



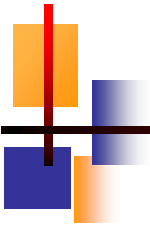
A partir des spécifications, on identifie des conditions d'exécution du programme qui permettront de produire les sorties voulues :

1. $a = b$ ou $a = c$ ou $b = c$
2. $a = b$ et $b = c$
3. $a < b + c$ et $b < a + c$ et $c < a + b$
4. $a > 0$ et $b > 0$ et $c > 0$



Conditions	Cas 1	Cas 2	Cas 3	Cas 4	Cas 5	Cas 6	Cas 7	Cas 8
1.	V	V	V	V	V	F	F	F
2.	V	V	F	F	F	F	F	F
3.	F	V	F	F	V	F	F	V
4.	F	V	F	V	V	F	V	V
Entrées Sortie	{0,0,0} Erreur	{3,3,3} Equi.	{0,4,0} Erreur	{3,8,3} Pas tria.	{5,8,5} Iso.	{0,5,6} Erreur	{3,8,4} Pas tria.	{3,4,5} Scal.

NB : il est impossible d'avoir (3)=V et (4)=F, ou encore (2)=V et (1)=F



Les tests structurels

Tests “boîte blanche” : détermination des cas de test en fonction du code.

Critère de couverture : Règle pour sélectionner les tests et déterminer quand les arrêter :

- au pire, on peut sélectionner des cas de test au hasard jusqu'à ce que le critère choisi soit satisfait.

Oracle : permet de déterminer la sortie attendue associée aux cas sélectionnés :

- Difficile à mettre en place si on veut automatiser complètement le processus de test.



Converture de chaque instruction (C0)

Selon ce critère, **chaque instruction doit être exécutée avec des données de test.**

Sélectionner des cas de test jusqu'à ce qu'un outil de couverture indique que toutes les instructions du code ont été exécutées

Exemple de converture de chaque instruction (C0) (1/3)

Noeud	Ligne de code	{3,4,5}	{3,5,3}	{1,0,1}	{4,4,4}
A	read a,b,c	x	x	x	x
B	type = "scalène"	x	x	x	x
C	if(a==b b==c a==c)	x	x	x	x
D	type = "isocèle"		x	x	x

Après le quatrième test, toutes les instructions sont exécutées.

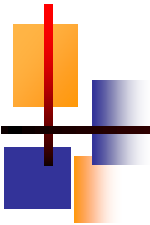
Il est rare que le jeu minimal soit bon d'un point de vue fonctionnel.

Exemple de converture de chaque instruction (C0) (2/3)

Noeud	Ligne de code	{3,4,5}	{3,5,3}	{1,0,1}	{4,4,4}
E	if(a==b && b==c)	x	x	x	x
F	type = "équilatéral"				x
G	if(a>=b+c b>=a+c c>=a+b)	x	x	x	x
H	type = "pas un triangle"			x	

Exemple de converture de chaque instruction (C0) (3/3)

Noëud	Ligne de code	{3,4,5}	{3,5,3}	{1,0,1}	{4,4,4}
I	if(a<=0 b<=0 c<=0)	x	x	x	x
J	type = "données erronées"			x	
K	print type	x	x	x	x



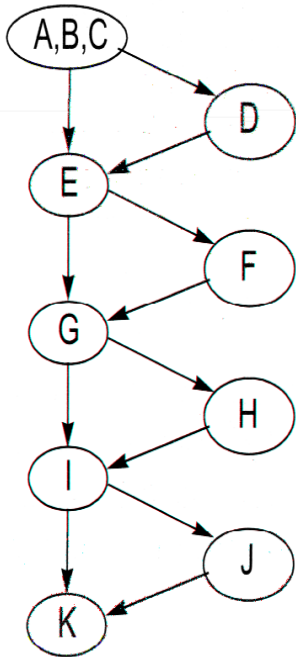
Test de toutes les branches (C1)

Plus complet que C0.

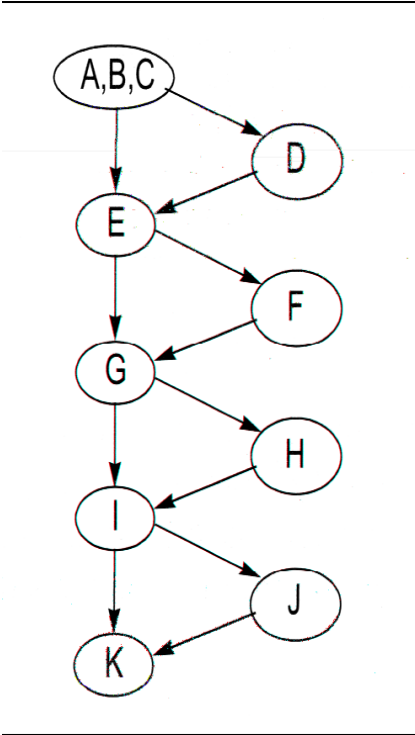
Selon ce critère, **il faut emprunter les deux directions possibles au niveau de chaque décision.**

Nécessite la création d'un graphe de contrôle et de couvrir chaque arc du graphe.

Exemple de test de toutes les branches (C1) (1/3)

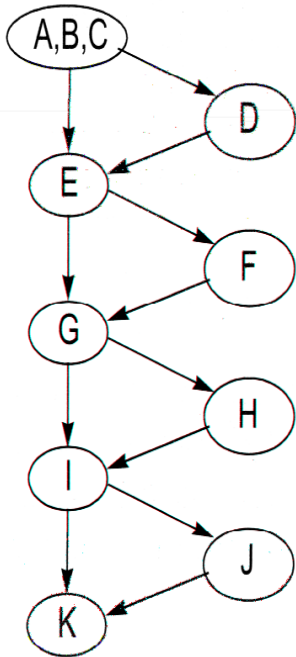


Arcs	{3,4,5}	{3,5,3}	{1,0,1}	{4,4,4}
ABC-D		x	x	x
ABC-E	x			
D-E		x	x	x
E-F				x
E-G	x	x	x	

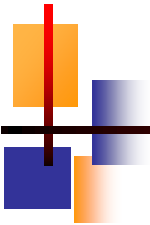


Arcs	$\{3,4,5\}$	$\{3,5,3\}$	$\{1,0,1\}$	$\{4,4,4\}$
F-G				

Exemple de test de toutes les branches (C1) (3/3)



Arcs	{3,4,5}	{3,5,3}	{1,0,1}	{4,4,4}
I-K	x	x		x
J-K			x	



Test de tous les chemins

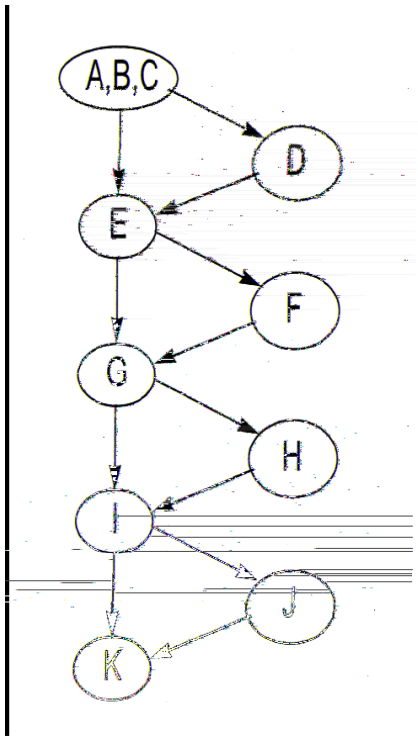
Encore plus complet.

Selon ce critère, **il faut emprunter tous les chemins possibles dans le graphe.**

Chemin : suite unique de noeuds du programme exécutés par un jeu spécifique de données de test.

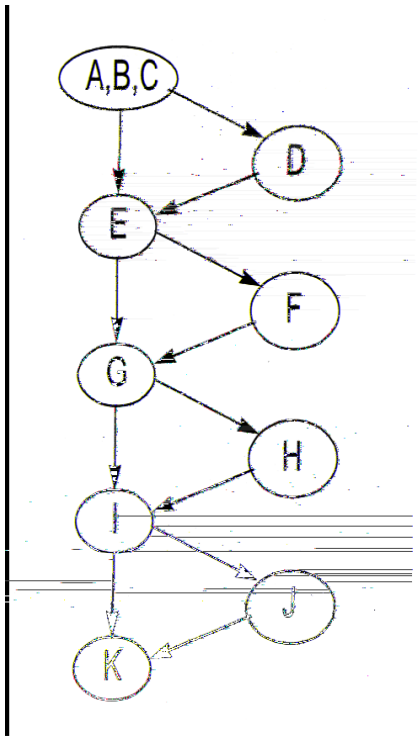
Peu adapté au code contenant des boucles, le nombre de chemins possible étant souvent infini.

Exemple de test de tous les chemins (1/2)

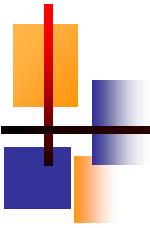


Chemin	V / F	Données	Sortie
ABCEGIK	FFFF	{3,4,5}	scalène
ABCEGHIK	FFVF	{3,4,8}	pas un triangle
ABCEGHIJK	FFVV	{0,5,6}	données erronées
ABCDEGIK	VFFF	{5,8,5}	isocèle

Exemple de test de tous les chemins (2/2)



Chemin	V / F	Données	Sortie
ABCDEGHIK	VFVF	{3,8,3}	pas un triangle
ABCDEGHIJK	VFVF	{0,4,0}	données erronées
ABCDEFGIK	VFVV	{3,3,3}	équilatéral
ABCDEFGHIJK	VVVV	{0,0,0}	données erronées



Couverture de conditions multiples

Un critère de test de conditions multiples impose que :





Exemple de couverture de conditions multiples

Pour `if (a==b||b==c||a==c)`

Combinaison	Données de test	Branche
V??	{3,3,4}	ABC-D
FV?	{4,3,3}	ABC-D
FFV	{3,4,3}	ABC-D
FFF	{3,4,5}	ABC-E

Test structurel

S'appuie sur la **circulation des données** à l'intérieur du programme : elles circulent

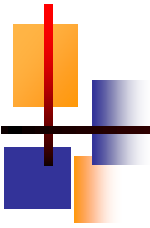
depuis l'endroit où elles sont définies jusqu'à celui où elles sont utilisées

- **def** : définition d'une donnée, correspond à l'attribution d'une valeur à une variable ;
- **c-use** : utilisation pour calcul, la variable apparaît à droite d'un opérateur d'affectation ;
- **p-use** : utilisation comme prédicat dans une condition.

A noter :

- l'utilisation p-use est attribuée aux deux branches conditionnelles ;
- un chemin sans définition, note **def-free**, va de la définition d'une variable à une

utilisation sans passer par d'autres définitions.



Critères de tests de flot de données

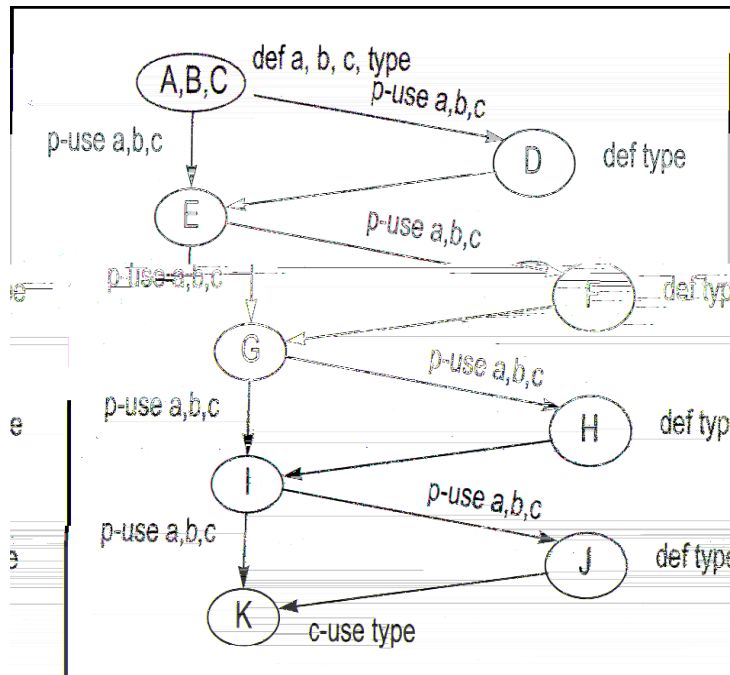
dcu : chaque définition doit être reliée à un c-use en aval par un chemin sans définition.

dpu : chaque p-use doit être reliée à une définition en amont par un chemin sans définition.

du : combinaison des deux précédents.

all-du-paths : le plus exigeant, qui veut que tous les chemins possibles entre une définition et une utilisation doivent être sans définition.

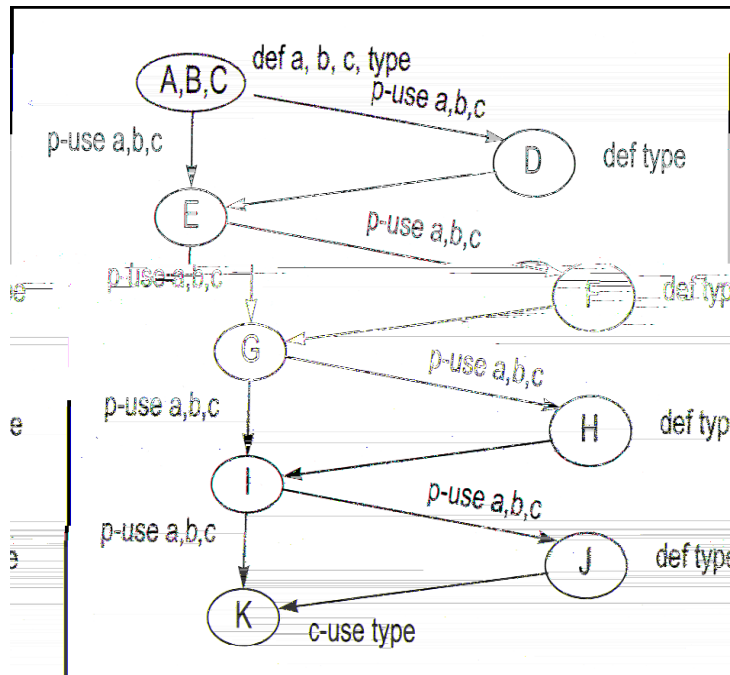
Exemple de tests de flot de données



dcu : le seul c-use porte sur type et se trouve au noeud K

- depuis ABC jusqu'a K : chemin ABCEGIK
- depuis D jusqu'a K : chemin DEGIK
- depuis F jusqu'a K : chemin FGIK
- depuis H jusqu'a K : chemin HIK
- depuis J jusqu'a K : chemin JK

Exemple de tests de flot de données



dpu : les p-use portent sur les variables a, b et c qui ne sont définies que dans le noeud ABC

- depuis ABC jusqu'a l'arc ABC-D
- depuis ABC jusqu'a l'arc ABC-E
- depuis ABC jusqu'a l'arc E-F
- depuis ABC jusqu'a l'arc E-G
- depuis ABC jusqu'a l'arc G-H
- depuis ABC jusqu'a l'arc G-I
- depuis ABC jusqu'a l'arc I-J

du : tous les tests de dcu et de dpu combinés

all-du-paths : mêmes tests que pour du