Analyse et programmation 1

Techniques algorithmiques

INFO1.08 - Techniques algorithmiques

heig-vd

Thèmes abordés

- Comment concevoir un algorithme
- Quelques techniques algorithmiques usuelles

heig-vd

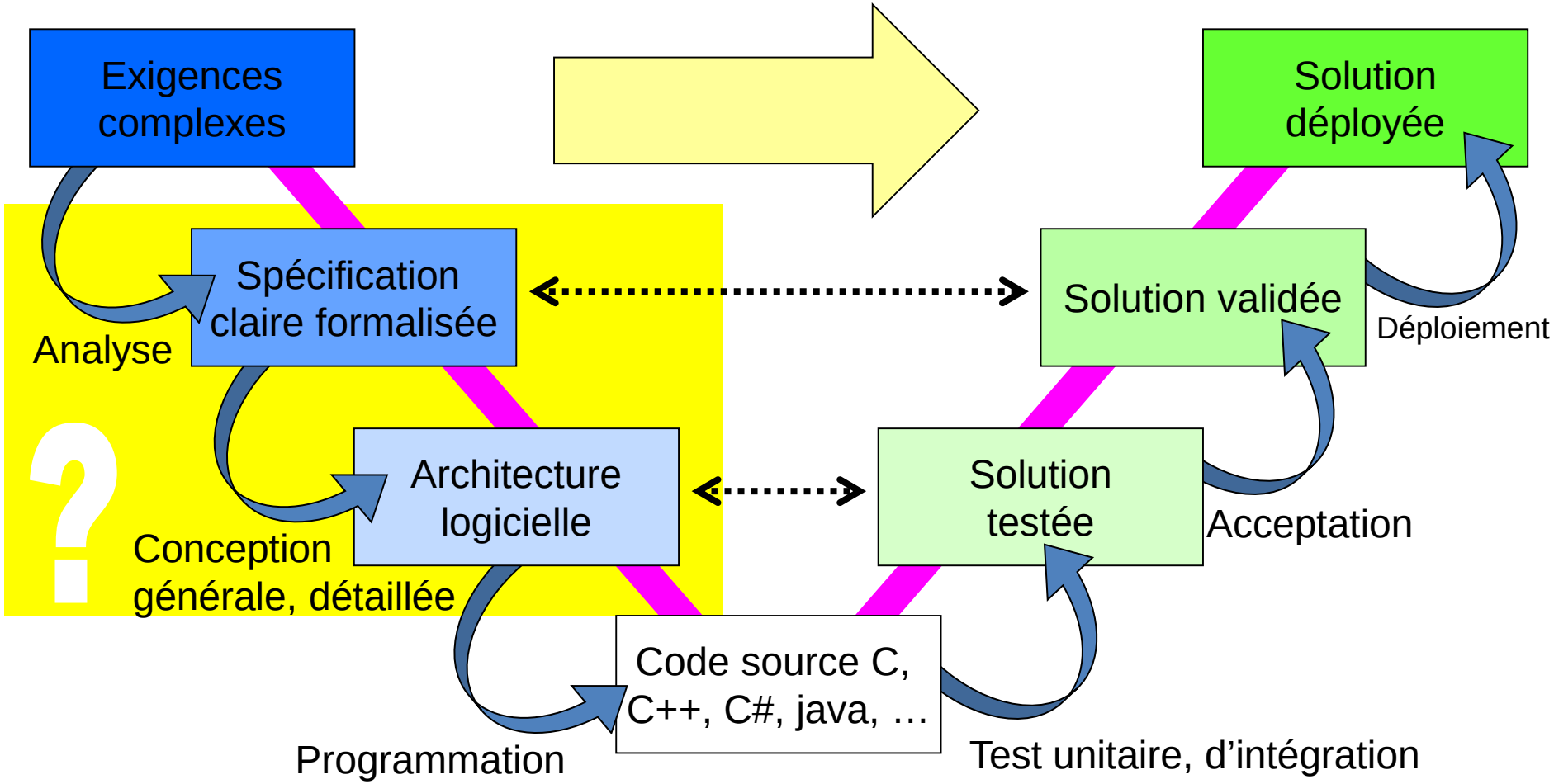
Comment concevoir un algorithme

- Méthodologies
- Quelques techniques de conception
  - Les organigrammes
  - Les structogrammes
  - Le pseudo code
  - UML

heig-vd

Rôle de l'analyse et conception

Dans le processus de développement logiciel



Méthodologies

Nature du problème

- Les acteurs de la réalisation d'un programme
  - L'utilisateur (client): connaît le besoin, de façon informelle.
  - Le programmeur : connaît (plus ou moins) la programmation.
  - Besoin d'un outil de communication entre les deux.
- Les cahiers des charges
  - Sont en général de haut niveau.
  - Exemple : piloter une machine de production.
- Les instructions à disposition
  - Sont de très bas niveau.
  - Exemple : if, for, i++
- Problème du programmeur d'application
  - Comment passer du cahier des charges aux instructions en C.

Méthodologies
Présentation

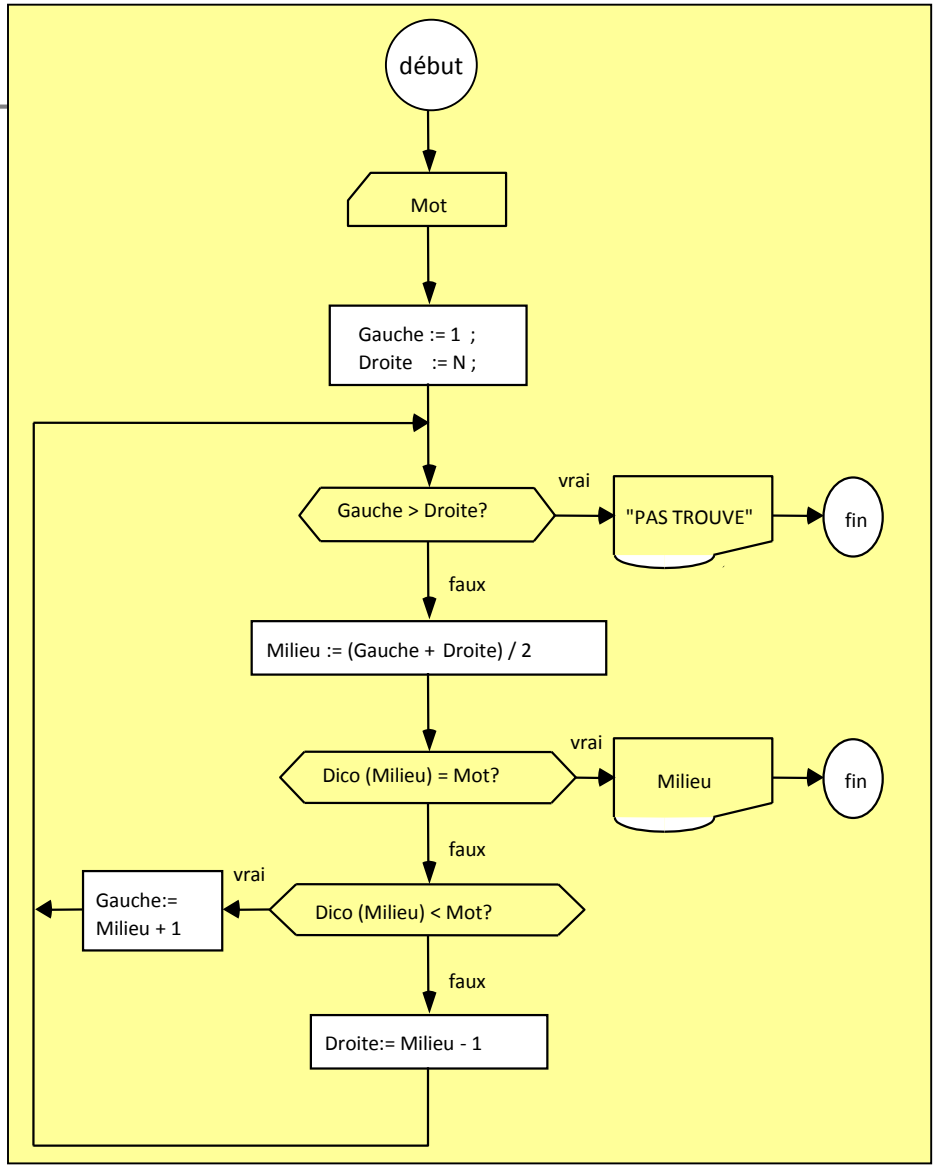
- Solution : utiliser une méthodologie
  - Définir des notations
    - Si possible compréhensibles par le programmeur et l'utilisateur.
  - Proposer une démarche
    - Cheminement pour la résolution des problèmes.
- Constat
  - Il existe des dizaines de méthodologies.
  - Les méthodologies ne savent pas programmer à notre place.
  - La programmation reste un art difficile.
    - Nécessite de la pratique.
  - Une méthodologie
    - Peut vous proposer un cadre formel.
    - Ne peut pas réfléchir à votre place.

Comment trouver un algorithme ?

- Pour analyser un problème
  - Commencer par le résoudre à la main.
    - Papier, crayon.
  - Une fois qu'une démarche est trouvée, analyser étape par étape ce que vous faites.
  - Identifier ensuite les structures de contrôle.
  - Et noter les grandes lignes de l'algorithme.
- Est-ce efficace de perdre du temps à réfléchir ?
  - Une fois que le bon algorithme est trouvé, la programmation devient un simple problème de traduction.
  - Une fois les bases du langage assimilées, la programmation prend peu de temps.

Les organigrammes
Présentation

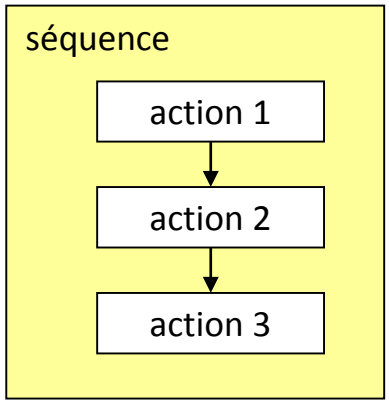
- Représentation graphique d'un algorithme
  - Rectangle : instruction.
  - Losange : décision.
  - Flèches : "saut"

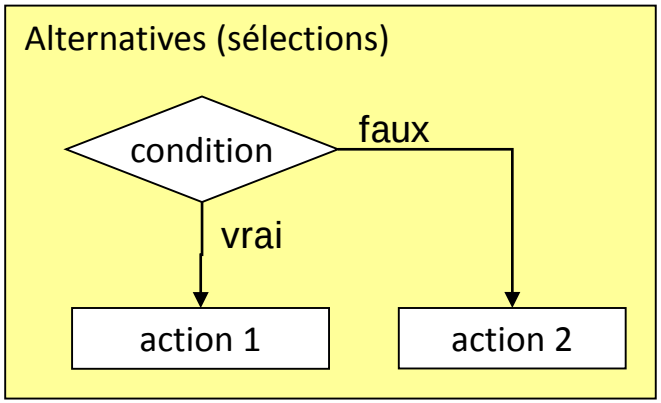


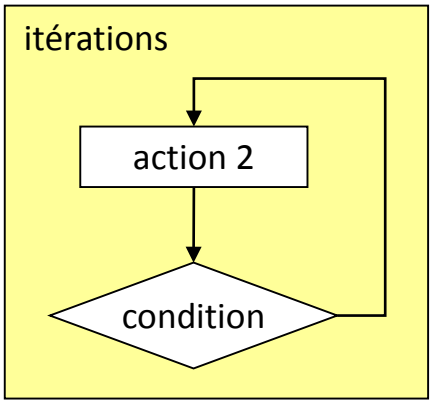
heig-vd

Les organigrammes

Symboles







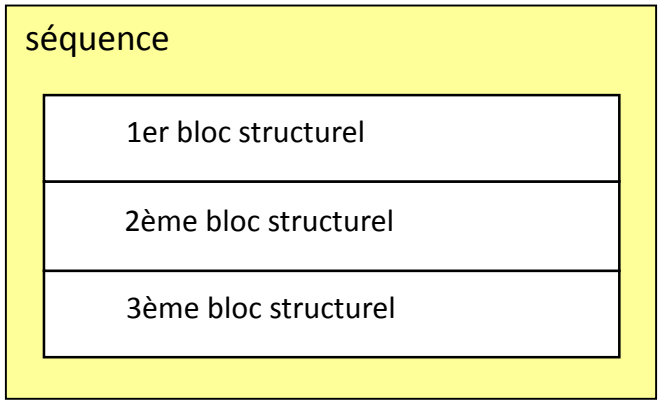
Les organigrammes
Analyse

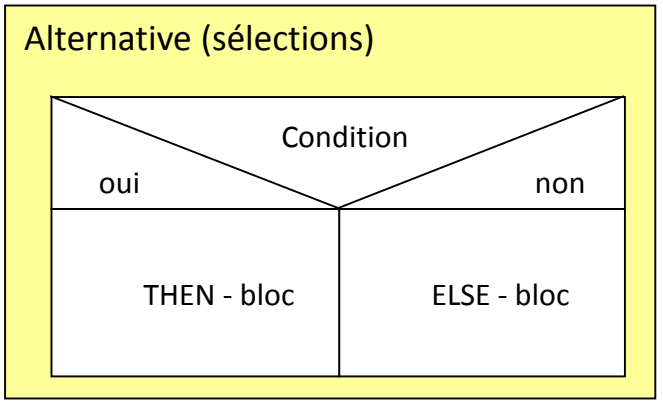
- Permettent de représenter des algorithmes simples
  - Exemple : représenter le fonctionnement d'une instruction.
  - Devient vite difficile à lire sur des problèmes réels.
- Encore utilisés en assembleur
  - Traduit bien les instructions machine
- Traduisent mal la programmation structurée
  - Les branchements traduisent « goto »
  - Pas de symbole représentant for, while, do while
- Pratique industrielle en langage évolué
  - Peu rencontré sur des projets professionnels.
- Donc…

Les structogrammes

Nassi Shneiderman Diagram (NSD) – Présentation (1/2)

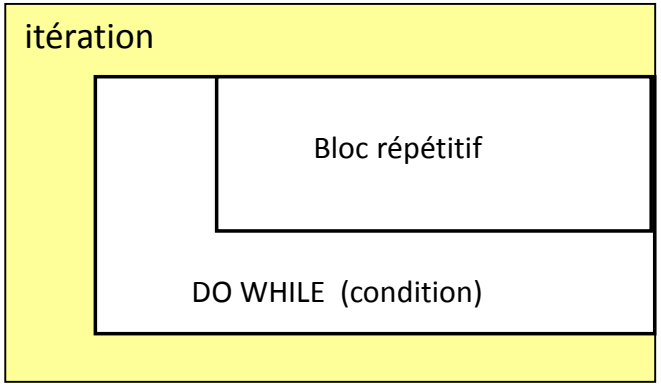
- Représentation graphique des structures de contrôle, inventée en 1972.
  - L'imbrication graphique traduit l'imbrication logique.
  - Différents symboles pour les structures de contrôle habituelles.

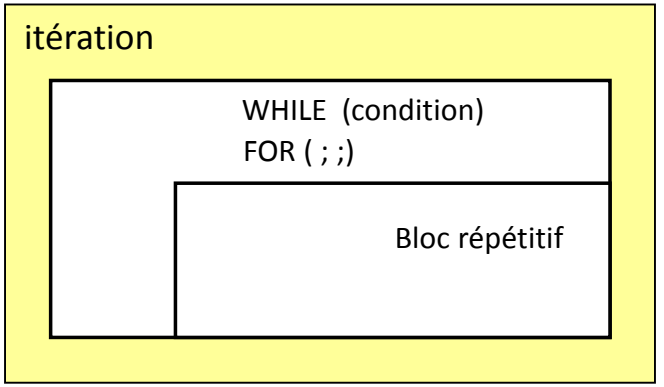


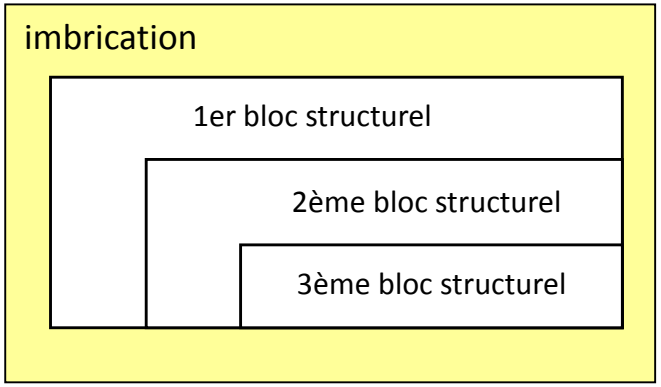


Les structogrammes

Nassi Shneiderman Diagram (NSD) – Présentation (2/2)



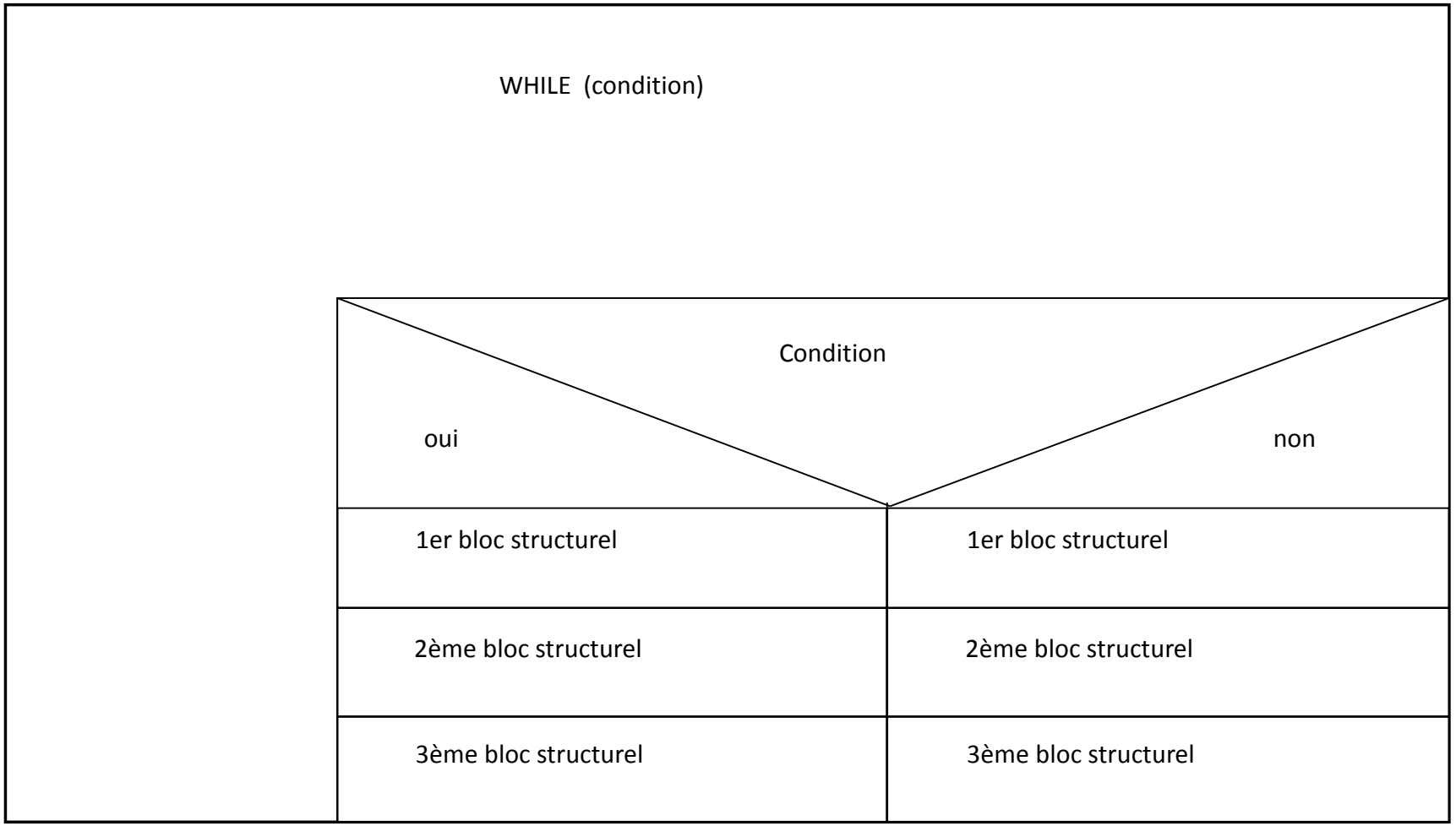




heig-vd

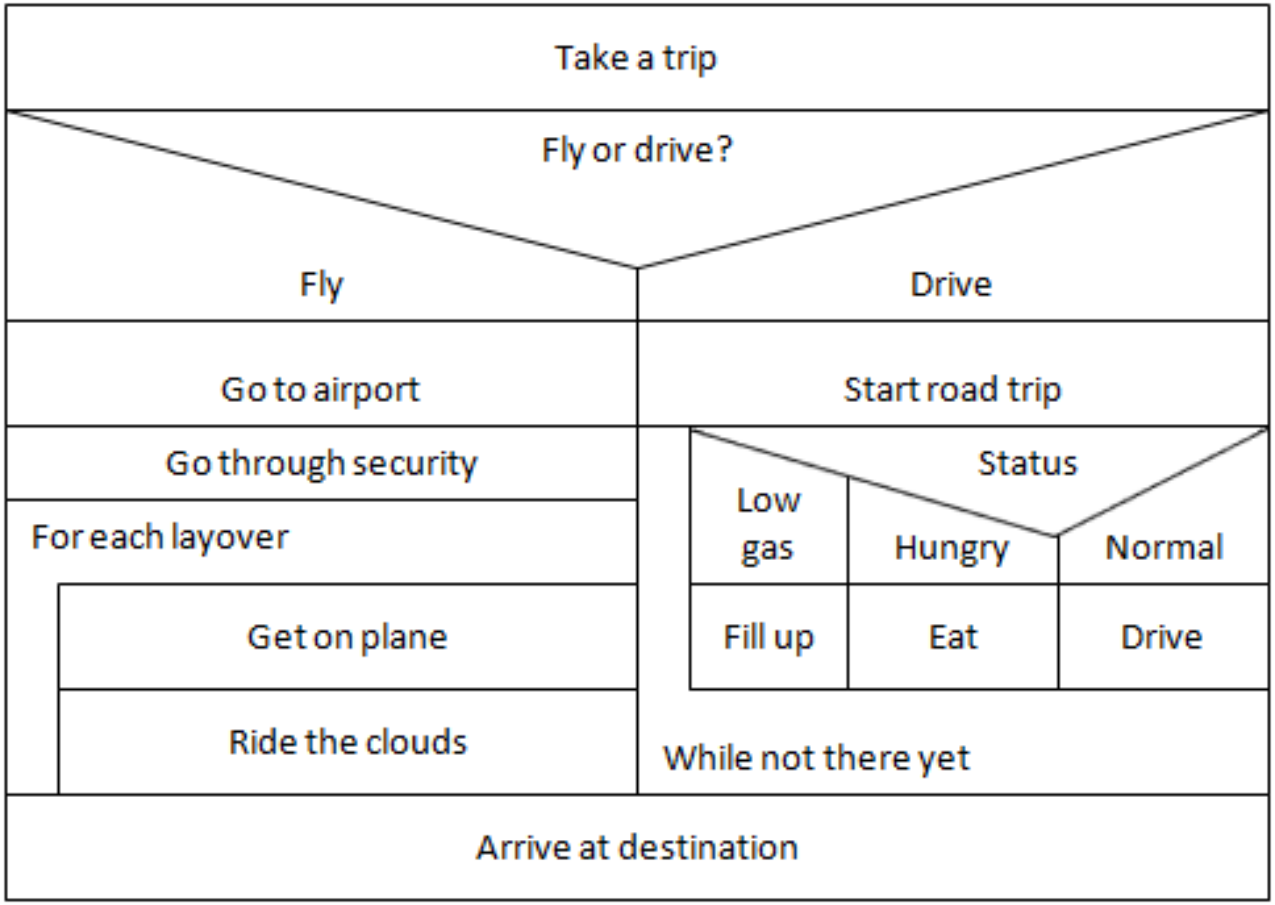
Les structogrammes

Illustration



Les structogrammes

Illustration



Les structogrammes
Analyse

- Permettent de représenter des algorithmes structurés
- Traduisent bien la programmation structurée
  - Montrent la décomposition hiérarchique.
  - Bonne façon de réfléchir.
- Limitations
  - Des niveaux d'imbrication multiples deviennent peu lisibles.
  - Un outil d'édition est indispensable (ou alors une bonne gomme).
- Pratique industrielle en langage évolué
  - Peu rencontré sur des projets professionnels.
- Donc…

Le pseudo code

Présentation

- **Principe du pseudo code**
  - Pseudos instructions traduisant les structures de contrôle.
    - SI . . . ALORS . . . SINON . . .
    - SELECTION SUR I . . . 1 => . . . 2=>
    - FAIRE . . . TANT QUE . . .
    - TANT QUE . . . FAIRE
  - Indépendance par rapport à la syntaxe réelle du langage cible.
  - On laisse de côté l'aspect déclaration des variables.
  - Moyen rapide et simple de structurer ses idées.
- Exemple

```
FAIRE
   Saisir X
   SI X < 0 ALORS
      Afficher un message d'erreur
TANT QUE X < 0
```

Le pseudo code

Présentation

- Principe du pseudo code (ici en Anglais, pour se rapproche du code)
  - Pseudos instructions traduisant les structures de contrôle.
    - IF . . . THEN. . . ELSE. . .
    - SWITCH CASE I . . . 1 => . . . 2=>
    - DO. . . WHILE. . .
    - WHILE. . . DO

- Exemple

```
DO
   Saisir X
   IF X < 0 THEN
      Afficher un message d'erreur
WHILE X < 0
```

Le pseudo code

Table de symboles

- Variables utilisées dans le pseudo code
  - Dans les cas simples, le type et le rôle d'une variable sont évidents.
  - Dans les autres cas, il faut écrire une table de symboles.
- Table de symboles
  - Elle précède le pseudo code
  - Elle précise
    - Le nom de l'objet : X
    - Sa nature (variable, constante) :
    - Son type de données : réel, entier, …
    - Un descriptif quand le nom n'est pas assez explicite.
  - Correspond aux déclarations de variable
- Exemple

  Somme : réel, accumulateur pour la somme des réels saisis

Le pseudo code

Analyse

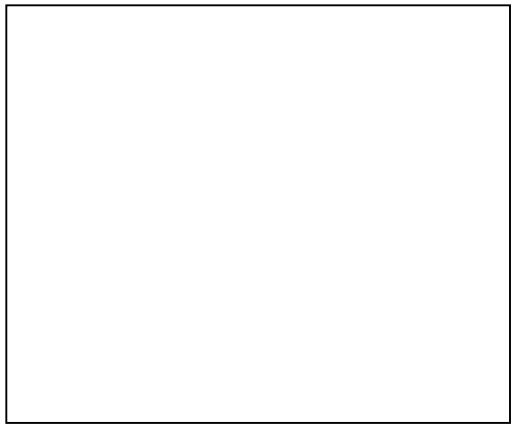
- Avantages
  - Très similaire au code final
  - Donc, pas d'effort de traduction particulier.
  - Très efficace, on ignore les exigences syntaxiques du langage cible.
  - Structuré.
  - Documente l'algorithme pour les évolutions futures.
  - Ne nécessite aucun outil particulier.
- Inconvénients
  - Laisse beaucoup de liberté.
  - Textuel : permet certaines ambiguïtés.

UML - Langage d'analyse et de conception

- **UML**
- Qu'est ce qu'UML ?

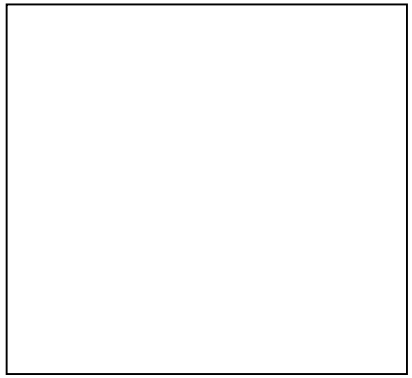
- UML : Unified Modeling Language
  - Langage de **Modélisation** Unifié
  - Utilisé pour l'analyse et la conception des logiciels
  - Forte coloration orientée objet
  - Langage essentiellement graphique
  - Facile à lire et à comprendre
  - UML n'est pas une méthode, mais un langage
  - UML définis 13 types de diagrammes

UML – 6 diagrammes structurels



Class

Package



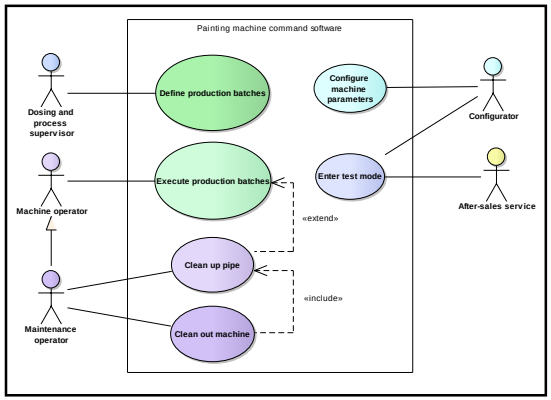
Object

Component

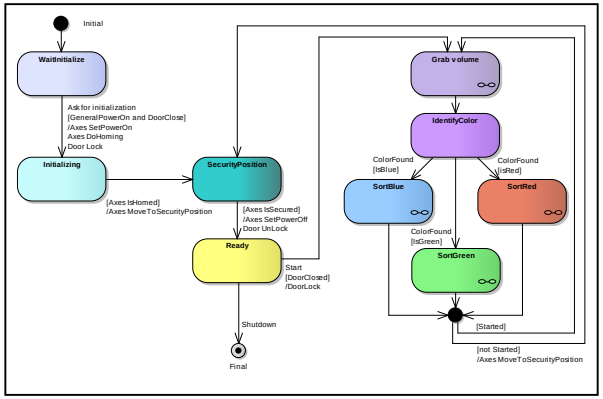
Composite Structure

heig-vd

UML – 7 diagrammes comportementaux



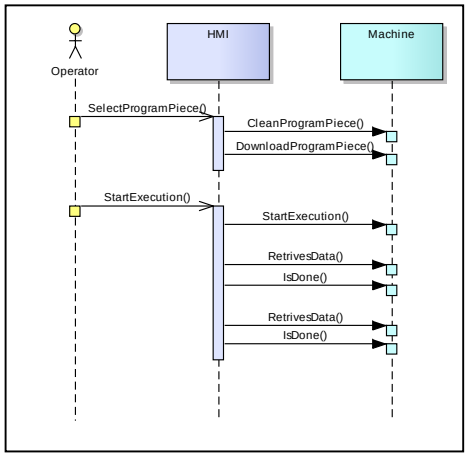
Use case



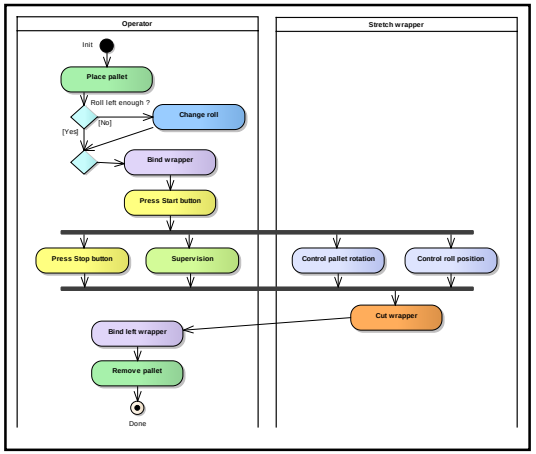
State machine

Timing

Interaction Overview



Sequence



Activity

Communication

Décomposition par raffinage successif

Une démarche hiérarchique naturelle

- Exemple : afficher les tables de multiplication de 1 à n

Pour i = 1 jusqu'à n

    Afficher la table de multiplication de i

Afficher la table de multiplication de i :

    Pour j = 1 jusqu'à n

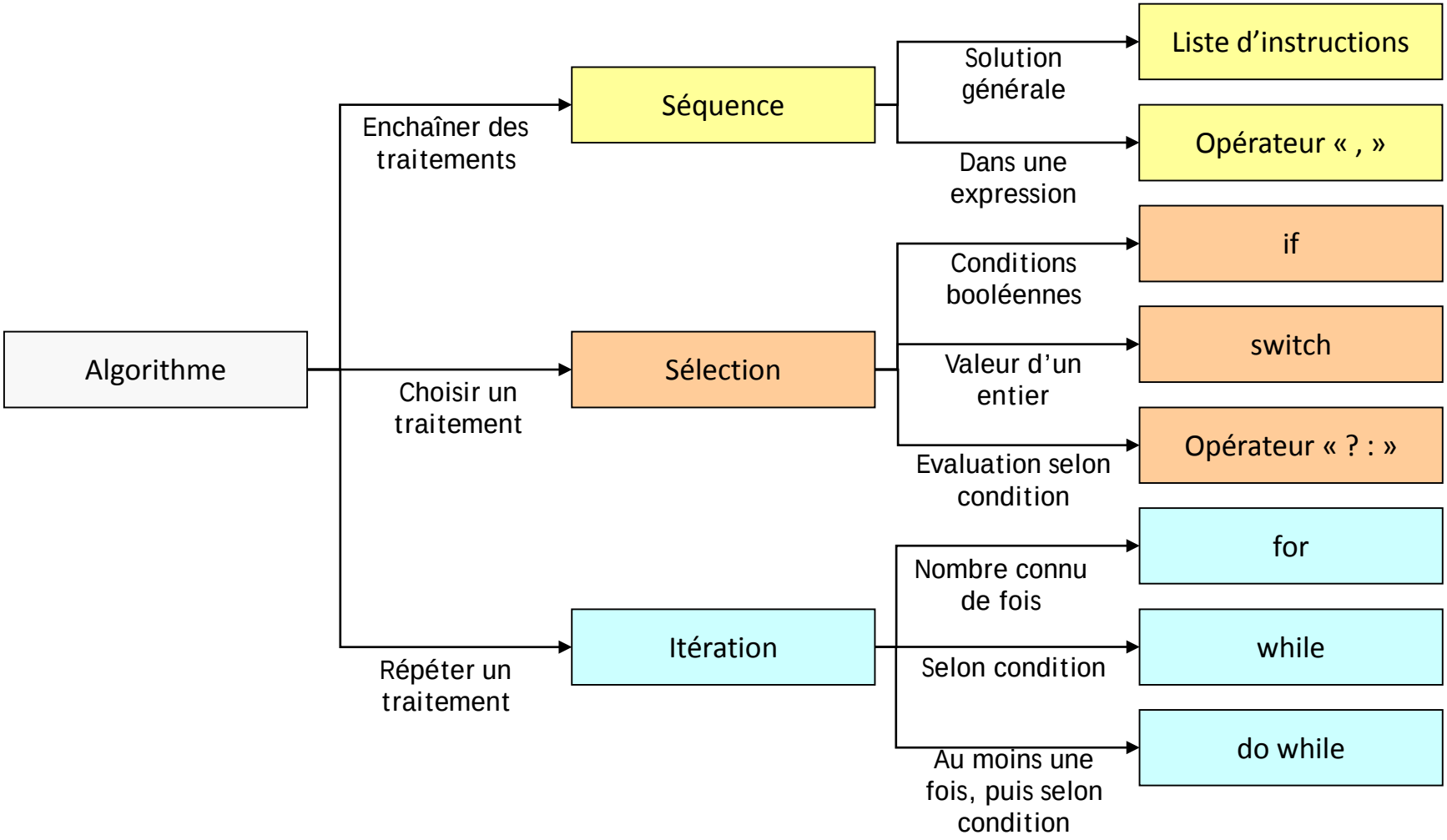
        Afficher j, « x », i, « = » , j * i

Méthodologie
Quelle approche choisir

- Lorsque le problème est compliqué
  - Préférer l'approche basée sur le pseudo code.
  - Utiliser d'autres formalismes
    - Lorsqu'ils contribuent vraiment à la compréhension.
    - Ou comme outil de communication.
- Lorsque le problème est assez simple
  - L'algorithme peut être conçu directement au niveau du code.
  - Pratique industrielle : c'était le cas courant.
- Un algorithme doit être clair avant d'être programmé
  - La programmation directe d'un algorithme devient possible avec la pratique.

Méthodologie

Comment choisir la structure de contrôle adaptée



heig-vd

Quelques techniques algorithmiques usuelles

- L'échange de variables
- Le comptage
- L'accumulation
- La recherche d'extremum
- Les répétitions imbriquées

Techniques algorithmiques usuelles
L'échange de variables

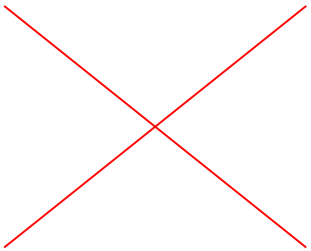
- But : échanger les valeurs des variables.

- Contexte
  - 2 variables contiennent des valeurs différentes.
  - On veut échanger leurs valeurs.

- But
  - Utile pour de nombreux algorithmes.
  - Exemples
    - Ordonner 2 nombres.
    - Rétablir les bornes d'un intervalle.

Techniques algorithmiques usuelles
L'échange de variables

- Implémentation naïve
-



i et j contiennent finalement la même valeur. La valeur initiale de i est perdue !!!

Techniques algorithmiques usuelles
L'échange de variables

- Implémentation fonctionnelle

1. On conserve la valeur initiale de i dans tmp.
2. On copie la valeur de j dans i.
3. On copie la valeur initiale de i dans j.

Techniques algorithmiques usuelles
Le comptage

- But : compter un nombre d'éléments par programmation.

- Exemples
  - Protocole de communication :
    - le nombre de caractères d'un message.
    - le nombre de messages reçus.
  - Caisse enregistreuse : nombre d'articles.

- Le comptage peut être
  - Systématique : prend en compte tous les objets.
  - Sélectif : teste une condition sur l'objet avant de le comptabiliser.

Techniques algorithmiques usuelles

Le comptage - illustration

Techniques algorithmiques usuelles
L'accumulation

- But : faire la somme d'éléments par programme.
- Exemples
  - Caisse de parking :
    - somme des pièces de monnaie introduites par un client.
    - somme des pièces de monnaie dans la caisse.
  - Machine d'assemblage
    - Quantité totale de colle déjà utilisée
  - Calcul de moyenne
    - En combinaison avec un comptage
- L'accumulation peut être
  - Systématique : prend en compte tous les objets.
  - Sélective : teste une condition sur l'objet avant de le comptabiliser.

Techniques algorithmiques usuelles

L'accumulation - illustration

Techniques algorithmiques usuelles
Recherche d'extremum

- But : rechercher le plus petit ou le plus grand élément d'une collection par programme.
- Exemples
  - Station météo, thermomètre électronique de congélateur :
    - Mémoriser les températures minimales et maximales.
  - Système d'information de l'école
    - Trouver l'étudiant le plus méritant

Techniques algorithmiques usuelles

Recherche d'extremum - illustration

Techniques algorithmiques usuelles
Répétitions imbriquées

- But :
  - Répéter plusieurs fois un traitement, lui même constitué d'une répétition.
- Exemples
  - Machine de soudage :
    - Produire 100 pièces
    - Pour chaque pièce, effectuer 15 points de soudure.
  - Afficher les tables de multiplication
    - 10 tables de multiplication
    - Pour chaque table, 10 multiplications.

Techniques algorithmiques usuelles

Répétitions imbriquées - illustration

Techniques algorithmiques usuelles

Répétitions imbriquées – piège classique

- Utilisation de la même variable pour 2 boucles imbriquées

- Ne pas confondre avec deux boucles consécutives

heig-vd

Qu'avons-nous appris ?

- Comment formaliser un algorithme
  - Graphiquement : organigramme, structogramme.
  - Textuellement : pseudo code
- Technique algorithmiques courantes
  - Comptage
  - Accumulation
  - Recherche d'extremum
  - Répétitions imbriquées

heig-vd

Vos questions


Exercices
Techniques
algorithmiques