Introduction

Chapitre 1

Objectifs généraux

Spécifier, concevoir, réaliser des composants logiciels, des applications informatiques

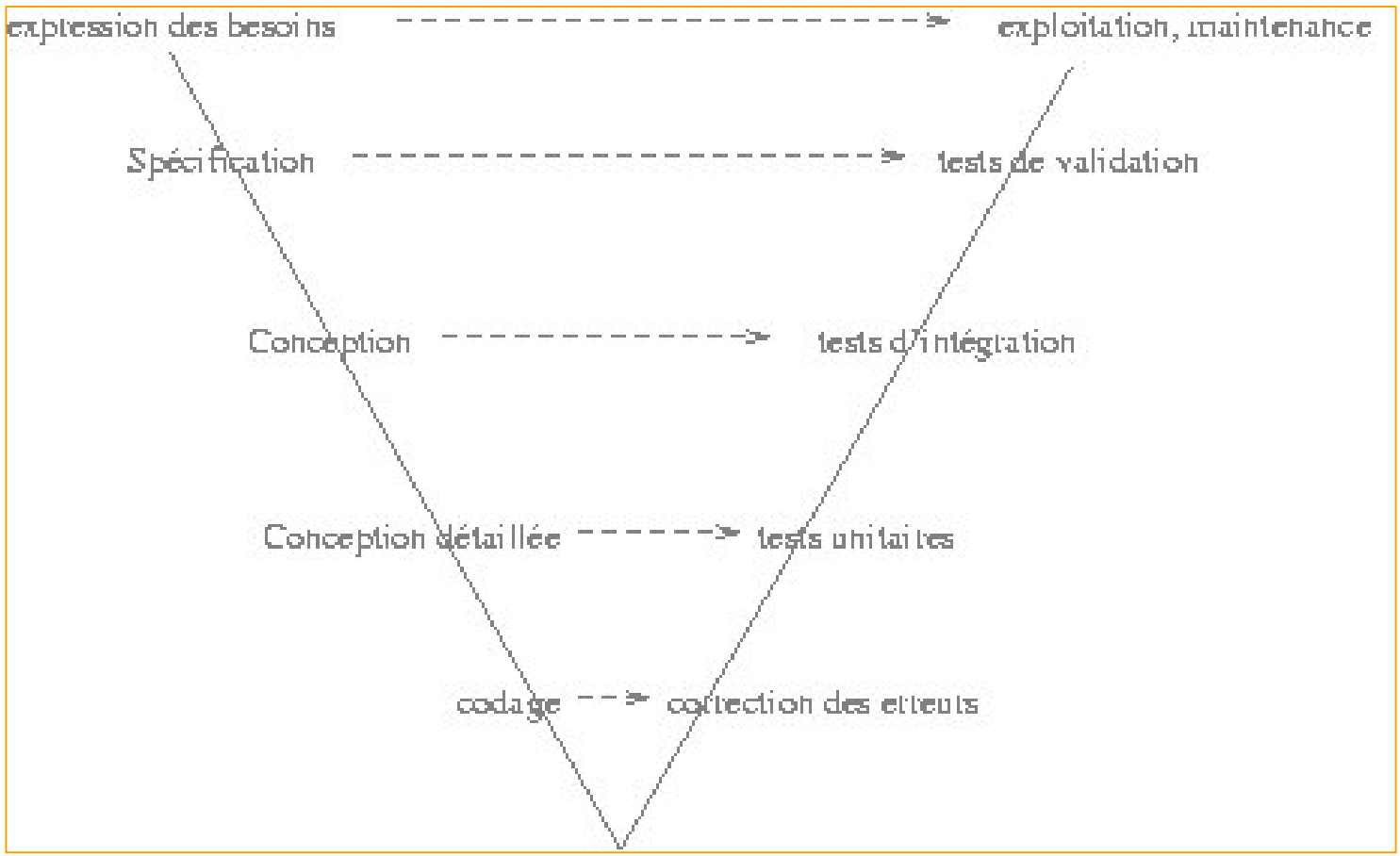
Avec les qualités requises pour un produit industriel :

efficacité
fiabilité
lisibilité
réutilisabilité
extensibilité

Donner les bases durables du métier de concepteur, développeur d'applications informatiques

Permettre une adaptation rapide aux différents langages de programmation

Cycle de vie du logiciel



Environnement de programmation

applications informatiques	**génie logiciel**
développement d'une application	cycle de vie
étapes de développement	analyse, conception, codage, tests, maintenance
environnement de programmation	outils, méthodes, langages
méthodes	UML, SADT, SART, Merise
outils	interface utilisateur, gestionnaire de données, compilateur, chargeur, éditeur de liens
langages	Ada, C++, Eiffel, Java

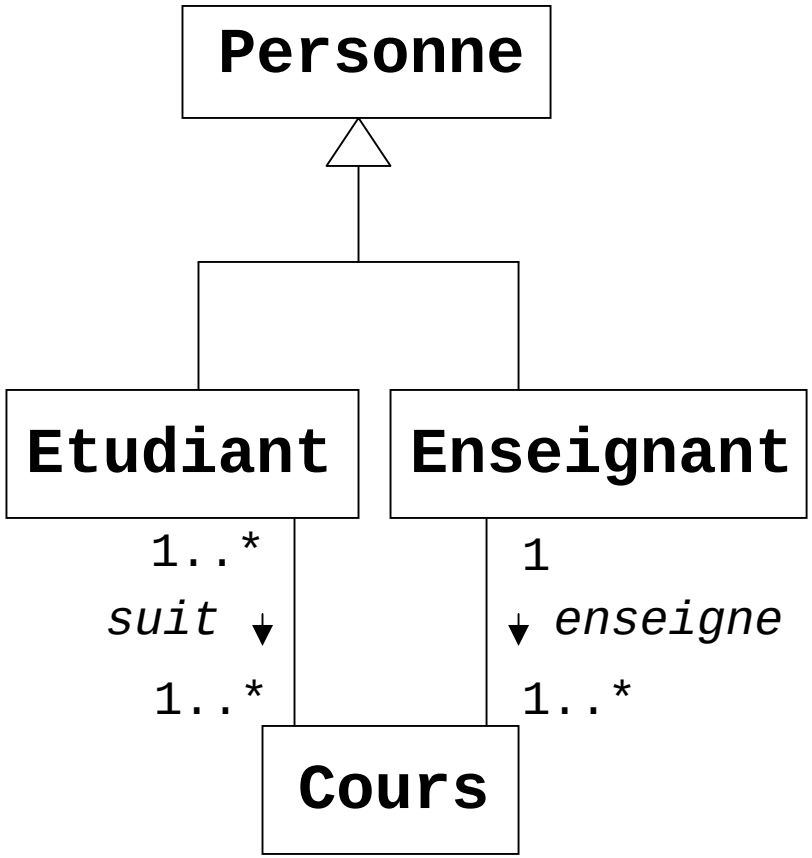
Les étapes du cycle de vie
- Analyse -

Dans cette étape, on s'attache à répondre à trois grandes questions :

- comprendre le problème, identifier les données et les résultats attendus
- dégager les grandes fonctionnalités du système (spécification fonctionnelle
- identifier les ressources nécessaires (matérielles et humaines)

Cette phase est indépendante de tout langage d'implantation.

Diagramme UML



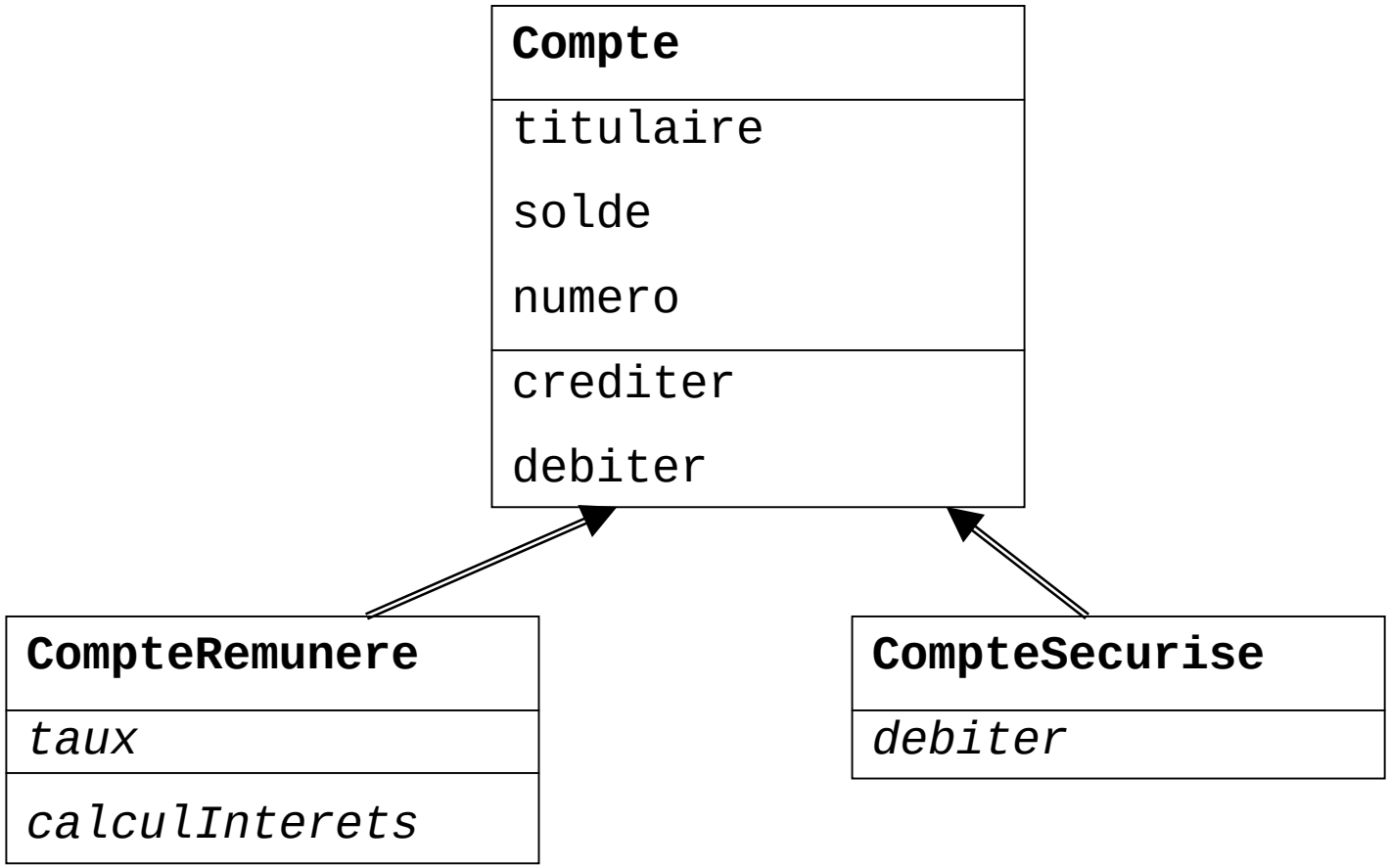
Les étapes du cycle de vie
- Conception -

On décompose le problème en sous-problèmes plus simples :

- Ceci donne naissance à un ensemble d'unités informatiques (composants) qui constituent le logiciel d'application.
- C'est dans cette phase que l'architecture du logiciel est élaborée. Les composants (modules ou unités) et leurs relations (interfaces) sont spécifiés.

Il s'agit d'un processus itératif qui peut conduire à un retour vers l'analyse.

Ada peut être utilisé comme langage de conception.


Compte
titulaire
solde
numero
crediter
debiter
CompteRemunere
taux
calculInterets
CompteSecurise
debiter

Les étapes du cycle de vie

- Codage -

Il s'agit, dans cette étape, d'implanter (coder) la conception, c'est à dire l'ensemble des composants de l'application.

On utilise un langage de programmation (Ada, C, Java, C++, Eiffel, Cobol, Fortran, Pascal, ...) pour exprimer le code.

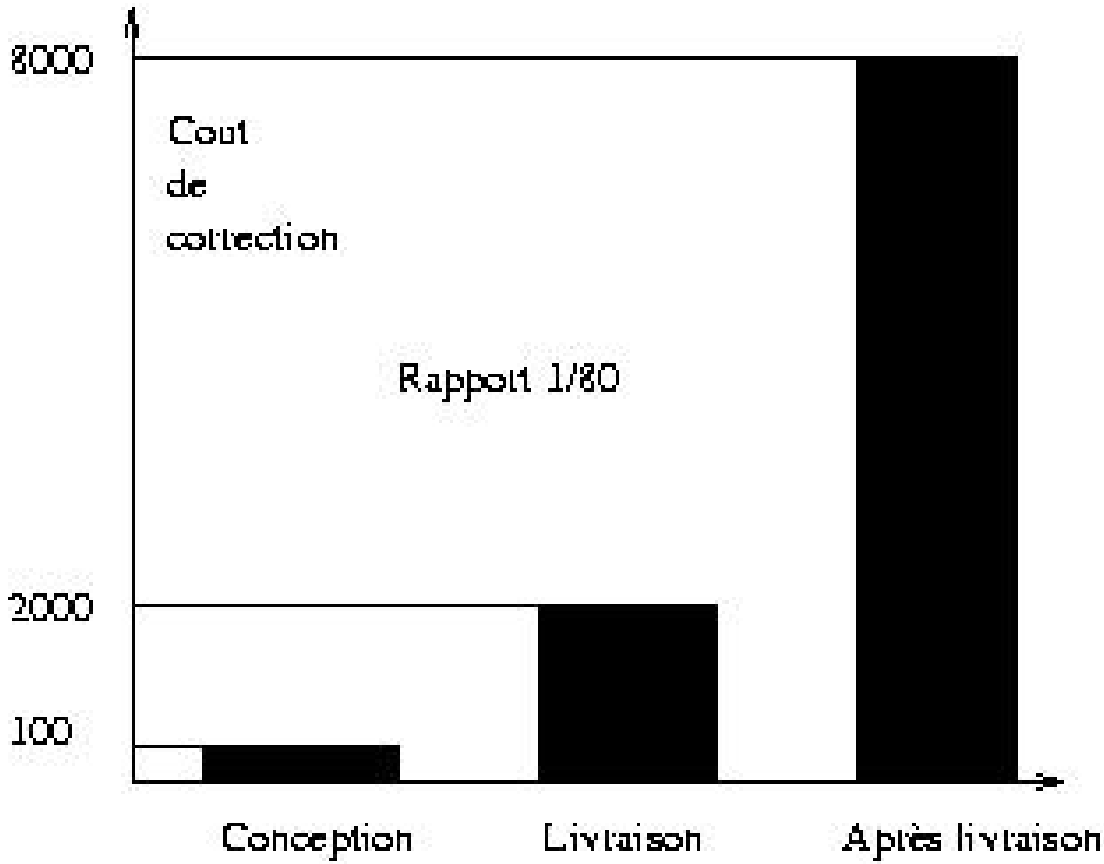
```
package Comptes is
--------- type Compte -------------------------
   type Compte is tagged
      record
         titulaire   : String( 1..10 );
         solde       : Float;
         numero      : String( 1..5 );
      end record;
   procedure debiter(  C: in out Compte;S: in Float );
   procedure crediter( C: in out Compte;S: in Float );
   procedure initialiser
        ( T: in String;S: in Float;N:in String;C: out Compte );
-------- type CompteRemunere -----------------
   type CompteRemunere is new Compte with
      record
         taux       :Float;
      end record;
   function calculinterets( C:CompteRemunere ) return Float;
   procedure initialiser
(T:in String;S:in Float;N:in String;taux:in Float;C:out CompteRemunere );
-------- type CompteSecurise -----------------
   soldeInsuffisant:exception;
   type CompteSecurise is new Compte with null record;
   procedure debiter( C: in out CompteSecurise;S: in Float );
end comptes;
```

Les étapes du cycle de vie
- validation/vérification -

- ✓tests unitaires (niveau composant) :
  - chaque composant fait, individuellement, l'objet de tests
- ✓tests d'intégration (niveau système)

L'évolution du système est contrôlée à travers le processus conception/codage/test.

Qualités d'un logiciel



coût de la non qualité

Qualités d'un logiciel : - efficacité -

- ✓ Optimalité de l'utilisation des ressources (temps, espace)
- ✓ Importance de la macro-efficacité qui se reflète dans une bonne structure de la solution au dépens de la micro-efficacité

Qualités d'un logiciel : - fiabilité -

L'industrie du logiciel est l'industrie la moins fiable.

- ✓ Un aspect de la fiabilité concerne la correction. On dit qu'un programme est correct si la solution répond bien au problème posé (spécification). L'application réalise les fonctions attendues par l'utilisateur dans les conditions normales et avec les performances attendues.

- ✓ Un autre aspect de la fiabilité concerne la récupération des pannes. Un programme est considéré comme fiable et robuste s'il propose un mode de fonctionnement dégradé, sans provoquer d'effets de bords.

Qualités d'un logiciel : – lisibilité –

La base d'une bonne lisibilité est le maintien de la relation entre le problème et sa solution.

Au niveau du codage :

- bonne documentation de la solution
- bon style de codage

Au niveau de la conception :

- clarté de l'architecture

Qualités d'un logiciel : - extensibilité -

- ✓ Une application informatique évolue au fil du temps et des besoins de l'entreprise à laquelle elle appartient.
- ✓ Il est important de ne pas devoir réécrire une application lorsque les spécifications subissent des modifications.
- ✓ L'extensibilité est la qualité d'un logiciel peu sensible aux changements de spécification.

Qualités d'un logiciel : – réutilisabilité, modularité –

Les composants logiciels qui forment l'application, conçus de manière à ne pas dépendre du contexte, deviennent réutilisables.

Une application est portable si elle est indépendante du système d'exploitation, du système dE/S et du matériel.

Les critères de modularité sont :

- couplage faible (interconnexion limitée)
- cohésion forte (éléments internes étroitement reliés)

Notre approche

Afin de nous concentrer sur le coeur de l'activité de programmation, nous nous limiterons aux trois phases essentielles :

- spécification
- codage
- tests

Spécification

La spécification est la description du problème à résoudre.

Spécification informelle

- Le problème est décrit en langage naturel.
- La description conserve éventuellement quelques imprécisions, ambiguïtés.
- Elle est souvent incomplète

Spécification formelle

- Il s'agit d'une description complète et rigoureuse du problème, exprimée dans un langage formel (proche des maths).
- Elle permet de faire des preuves de correction et de terminaison d'un calcul.

Codage

Le codage correspond à la traduction de la solution d'un problème dans un langage de programmation (LP).

Par rapport à la phase de spécification, les problèmes posés sont de nature différente. Ils tiennent :

- aux caractéristiques physiques de la machine (E/S)
- à la représentation des données
- à la traduction des actions dans un LP

Tests

En général, un programme ne peut pas être testé pour toutes les données possibles.

Des jeux de tests sont élaborés de manière à visiter tous les chemins que peut prendre l'exécution du programme.

Cela ne fournit pas une preuve de la correction du programme, mais donne un indice de confiance dans son fonctionnement.

Exemple de spécification (1)

Spécification informelle

On veut savoir si un nombre entier n, non négatif, est un carré parfait.
Le résultat produit est :

- vrai si et seulement si n est un carré parfait
- un entier égal à la racine carrée entière de n.

Spécification formelle

$$\{n>=0\} \rightarrow \{(\exists\ m \in [0..n] \wedge \text{vrai} \wedge m*m=n) \vee (\exists m \in N \wedge \text{faux} \wedge m*m<n<(m+1)*(m+1)\}$$

Exemple de spécification (2)

Spécification informelle

Calcul de n^p, p est un entier naturel mais il peut être nul sauf si n est nul

Spécification formelle

$\{p \in \mathbb{N} \wedge [(p \geq 0) \vee (p \neq 0 \wedge n = 0)]\} \rightarrow \{résultat = n^p\}$

Pourquoi Ada
– crise du logiciel –

- complexité croissante
- coût
- qualité

En 1976, une étude met en lumière le coût prohibitif du logiciel dû à la complexité croissante :

- 75 $ par instruction développée
- 4000$ par instruction modifiée (après livraison)

Pourquoi Ada
- une étude (1) -

Un autre étude menée en 1979 par le gouvernement américain et portant sur 487 projets illustre le coût exhorbitant de la maintenance :

- 41,8% est dû à un changement de spécification
- 17,4% est dû à un changement dans le format des données
- 12,4% à des sorties d'urgence
- 9% à des défaillances nécessitant un débogage

Pourquoi Ada
- une étude (2) -

La même étude indique que :

- 47% des applications délivrées ne sont jamais utilisées
- 29% payées mais non terminées
- 19% abandonnées ou réécrites
- 3% utilisées après modifications
- 2% utilisées sans changement

Pourquoi Ada
– une autre étude –

Une enquête effectuée aux USA en 1986 auprès de 55 entreprises révèle que 53% du budget total d'un logiciel est affecté à la maintenance :

- 34% à la maintenance évolutive : modification des spécifications initiales
- 10% à la maintenance adaptative : nouvel environnement, nouveaux utilisateurs ;
- 17% à la maintenance corrective : correction des bogues ;
- 16% à la maintenance perfective : améliorer les performances sans changer les spécifications ;
- 6% à l'assistance aux utilisateurs ;
- 6% au contrôle qualité ;
- 7% l'organisation et au suivi ;
- 4% divers.

Pourquoi Ada

Nécessité d'un langage général de haut niveau :

- pour applications embarquées
- pour applications de gestion
- pour applications scientifiques

Nécessité d'un langage normalisé

- évitant la multiplication des dialectes
- imposant la certification des compilateurs selon une norme internationale

Cahier des charges d'Ada

L'accent est mis sur la diminution du coût des logiciels en tenant compte de tous les aspects du cycle de vie.

Les caractéristiques mises en avant sont :

- privilégier la facilité de maintenance sur la facilité d'écriture (maintenance = ½ du coût global du logiciel)
- appliquer un contrôle de type extrèmement rigoureux pour diagnostiquer les erreurs le plus en amont possible
- permettre une programmation sûre. Le logiciel traite lui-même les situations anormales
- rendre les applications portables pour ne plus lier les logiciels à un constructeur et donc les rendre indépendantes de toute plate-forme
- permettre des implantations efficaces et donner accès à des interfaces de bas niveau (temps réel)

Ada : norme internationale (1)

- ✓ Ada est l'aboutissement d'une longue lignée de langages impératifs et procéduraux (Pascal, algol, C).
- ✓ Il synthétise les meilleurs apports de ces langages et les intègre dans un ensemble cohérent.
- ✓ Il est utilisé dans des domaines variés : CAO, médical, traitement linguistique, gestion, temps réel, ...
- ✓ Ada est une norme, aucun dialecte, sur-ensemble ou sous-ensemble n'est admis afin de garantir la portabilité des applications

Ada : norme internationale (2)

Une révision périodique de la norme est effectuée. Ada 95 est une révision d'Ada 83. Elle est compatible avec l'ancienne norme.

Pour garantir la conformité à la norme, les compilateurs Ada sont validés :

- le processus de test des compilateurs est devenu un standard international (ISO/IEC-18009:1999)
- la validation est réalisée par des labos indépendants

Actuellement, 26 compilateurs sont validés (Aonix, Intermetrics, ...)

dont quelques compilateurs gratuits :

http://www.adahome.com/Resources/Compilers/Free.html

L'histoire d'Ada en deux mots (1)

1968 -> 1973, le DoD constate :

- le coût du logiciel commence à dépasser le coût du matériel des principaux systèmes de défense
- un accroissement de 51% du coût de ses systèmes informatiques
- la diversité et l'inadéquation des langages de programmation utilisés (450)
- le manque d'environnement de développement

1975 -> 1977, établissement du cahier des charges pour un langage de programmation de haut niveau

L'histoire d'Ada en deux mots (2)

1978 -> 1979 Examen des propositions

1980 Publication du manuel de référence

1983 Approbation du manuel de référence par la norme ANSI
(American National Standards Institute)

1987 Normalisation ISO (International Organization for Standardization),
Certification des compilateurs (3500 tests)

Caractéristiques principales du langage

Ada supporte les concepts :

- de modules (paquetages)
- de généricité
- d'exceptions
- de tâches (fils de contrôle parallèles avec communication)

Il possède une riche bibliothèque de modules prédéfinis.

Il s'interface avec d'autres langages comme C, fortran, cobol,...

Le compilateur Ada d'Intermetrics peut générer du J-code (code pour la machine virtuelle java). Il est aussi possible d'utiliser Ada pour développer des applettes Java.

Bibliothèques pour réaliser des interfaces graphiques

Ada possède de riches bibliothèques pour réaliser des interfaces graphiques :

http://libre.act-europe.fr/GtkAda/

http://w3imagis.imag.fr/Membres/Alexandre.Meyer/teaching/info3/index.fr.html

Applications

Ada est le langage le plus populaire pour les applications critiques :

- mise en jeu des vies humaines
- importance des coûts liés aux pannes

Domaines d'applications :

- transports
  - conduite automatique de trains
  - systèmes de contrôle en avionique
- production d'énergie
  - conduite de centrales nucléaires
  - conduite de barrages

Entreprises impliquées

Information complète : http://www.adaic.org/atwork/index.

Industries aéronautiques et spatiales :

Boeing, Nasa (station spatiale), Chandler Evans (système de contrôle de moteur), ESA (observation spatiale, mission saturne conjointement avac la Nasa), Ford Aerospace, Intermetrics, Lockheed, Rockwell Space System (navette spatiale), ...

Contrôle du trafic aérien (CTA) :

Hughes (système de CTA canadien), Loral FSD (système de CTA US), Thomson-CSF (système de CTA français).

Bateaux : Vosper Thornycroft Ltd (contrôle de navigation).

Trains :

European Rail (système d'aiguillage), EuroTunnel, Extension du métro de Londres, GEC Alsthom (systèmes de contrôle des signaux de trains et TGV), TGV France (système d'aiguillage), Westinghouse Australia (système de protection automatique des trains).

Médical : Coulter Corp(Onyx analyse de l'hématologie)

Nucléaire.

Bilan de l'utilisation d'Ada

Les études économiques sur l'utilisation du langage Ada font apparaître que, par rapport aux autres langages, Ada :

- coûte moins cher en développement
- raccourcit la phase d'intégration
- provoque moins d'erreurs résiduelles

Comment se procurer un compilateur Ada

Télécharger le compilateur gnat gratuitement (et pour toute plate forme) à partir de l'URL :

http://www.lisi.ensma.fr/ftp/enseignement/ada/

- Téléchargez les fichiers: adagide-install.exe gnat-3.15p-nt.exe les fichiers .zip
- Exécutez gnat-3.15p-nt.exe puis adagide-install.exe
- Ajout de la documentation : dezippez docs.zip (ou docs-pdf.zip)

Quelques sites utiles

Site de référence (Compilateurs, bibliothèques, cours en ligne, documentation)
http://www.adahome.com/

Son manuel de référence :
http://www.adahome.com/rm95/rm9x-toc.html

Bibliographie

Ingénierie du logiciel avec ADA, Grady Booch, InterEditions

Programmer en ADA, J.P.G.Barnes, Addison Wesley

Tutoriels en ligne : http://www.adahome.com/Tutorials/

Page Web de l'UE

http://deptinfo.cnam.fr/new/rubrique.php3?id_rubrique=226