

CENTRALE
L Y O N

Ateliers de Génie Logiciel

Principes de fonctionnement
Exemple : APSE

BTD/GL/AGL 1

CENTRALE
L Y O N

Définition

Atelier de Génie Logiciel : Définition

Système organisé avec ses structures, ses personnes et les tâches réparties entre celles-ci, appréciable par :

- une architecture matérielle choisie,
- des normes, des méthodes , des outils et des langages supports et/ou supportés
- des relations entre atelier et utilisateurs

dans le but de développer et de maintenir du logiciel de qualité

Bertrand DAVID : Génie Logiciel

BTD/GL/AGL 2

CENTRALE
L Y O N

Objectifs

Production industrielle du logiciel :

- standardisation de la production
- maîtrise des coûts et des délais
- contrôle de la qualité du produit

par

- des méthodes
- des normes
- des mesures
- des outils

Bertrand DAVID : Génie Logiciel

BTD/GL/AGL

3

CENTRALE
L Y O N

Buts recherchés

- Aide à l'accomplissement, au contrôle et au suivi des différentes phases du cycle de vie du logiciel
- Augmentation de la cohérence entre les tâches
- Aide à la mise en application des plans d'assurance qualité du logiciel
- Amélioration du confort de travail
- Amélioration de la productivité

Bertrand DAVID : Génie Logiciel

BTD/GL/AGL

4

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Typologie des outils

- **Outils techniques**
 - développement du logiciel
 - développement de la documentation
- **Outils logistiques**
 - tâches de type bureautique
 - contrôle de configuration
 - gestion de versions
- **Outils de contrôle**
 - suivi de projet
 - assurance qualité
 - métrologie

BTD/GL/AGL

5

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Architectures d'ateliers (1)

- **Système fermé :**
 - les outils se connaissent et coopèrent
 - enchaînement des outils
 - approche classique de processus de production

```
graph TD; A[ ] --> B[ ]; B --> C[ ]; C --> D[ ]
```

Le diagramme illustre un processus de production fermé à quatre étapes, représentées par des boîtes rectangulaires alignées verticalement et reliées par des flèches descendantes. Les étapes sont :

- Edition
- Compilation
- Edition de liens
- Exécution

BTD/GL/AGL

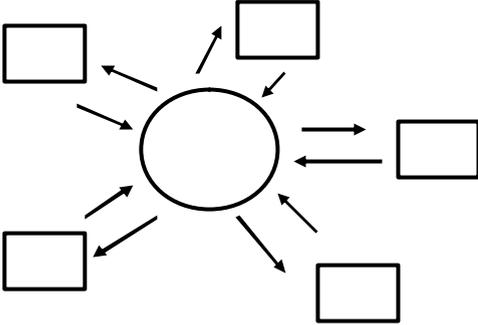
6

CENTRALE
L Y O N

Architectures d'ateliers (2)

Bertrand DAVID : Génie Logiciel

- **Systemes ouverts**
 - Banalisation des tâches de développement
 - Structure de données homogène
 - Enchaînement par des données



BTD/GL/AGL

7

CENTRALE
L Y O N

Classes d'ateliers

Bertrand DAVID : Génie Logiciel

- Environnement classique de programmation
- Atelier boîte à outils
- Atelier orienté méthode
- Atelier orienté langage
- Système support intégré de développement
- Atelier orienté matériel
- Méta-atelier

BTD/GL/AGL

8

CENTRALE
L Y O N

Environnement classique de programmation

Bertrand DAVID : Génie Logiciel

- **Outils**
 - Editeur de texte
 - Compilateurs
 - Editeur de lien
 - Gestionnaire de fichiers
 - Metteur au point

BTD/GL/AGL 9

CENTRALE
L Y O N

Atelier boîte à outils

Bertrand DAVID : Génie Logiciel

- **Exemple Unix**
 - Editeur de texte
 - Compilateurs
 - Editeur de lien
 - Gestionnaire de fichiers
 - Metteur au point
 - Analyseur statique
 - Analyseur dynamique
 - Constructeur d 'analyseurs lexicaux
 - Constructeur d 'analyseurs syntaxiques

BTD/GL/AGL 10

CENTRALE
L Y O N

Atelier orienté méthode

Bertrand DAVID : Génie Logiciel

- **Les outils facilitent l'application des principes méthodologiques d'une méthode :**
 - gèrent le formalisme et le graphisme
 - effectuent des contrôles de cohérence
- **Exemples :**
 - Merise

BTD/GL/AGL 11

CENTRALE
L Y O N

Atelier orienté langage

Bertrand DAVID : Génie Logiciel

- **Atelier exploite en profondeur la connaissance du langage de programmation sur lequel il s'appuie :**
 - Editeur syntaxique
 - Outils réutilisables proposés et basés sur le même langage

BTD/GL/AGL 12

| | |
|---------------------------------|--|
| CENTRALE L Y O N | Systeme support intégré de développement |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> ● Une représentation interne des objets reposant sur une structure de données commune à plusieurs outils de développement qui en utilisent certains attributs ● La définition et l'utilisation de mécanismes de traduction et de gestion cohérente de tous les objets manipulés ainsi que leurs relations ● La mise en place d'une Base de données d'objets ● La définition d'une architecture banalisant les objets et offrant des postes de travail permettant de visualiser et agir sur plusieurs objets ou états d'objets à la fois |
| BTD/GL/AGL | 13 |

| | |
|---------------------------------|---|
| CENTRALE L Y O N | Objets supportés |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> ● Des objets manipulés sont : <ul style="list-style-type: none"> – des unités de production (modules) – unités de partage (ressources élémentaires) – unités d'évolution (versions) ● Des formes prises sont : <ul style="list-style-type: none"> – texte – programme – graphique – documentation ● Le type et l'abstraction de l'objet dépend de l'étape du cycle de vie et du rôle de l'acteur |
| BTD/GL/AGL | 14 |

CENTRALE
L Y O N

Méta-atelier

Bertrand DAVID : Génie Logiciel

- Définition d'une structure d'accueil, d'une architecture, d'un modèle conceptuel permettant de configurer - paramétrer l'atelier sur le plan :
 - des outils
 - des méthodes
 - des tâches
 - de l'organisation
 - des langages et formalismes cibles

BTD/GL/AGL 17

CENTRALE
L Y O N

Trilogie : Outils - Méthodes - Langages

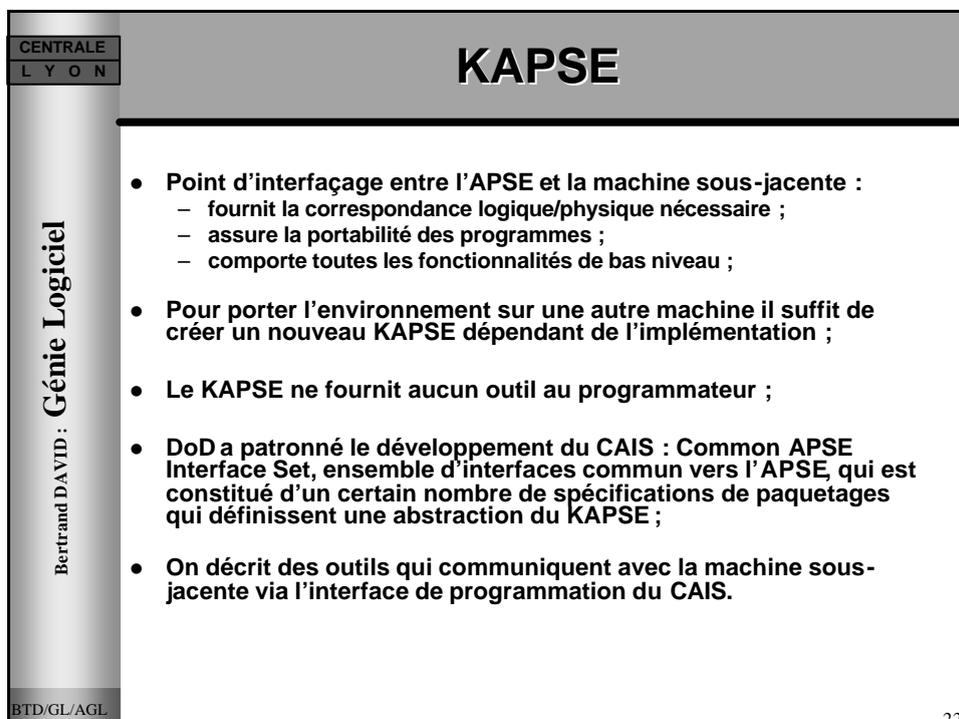
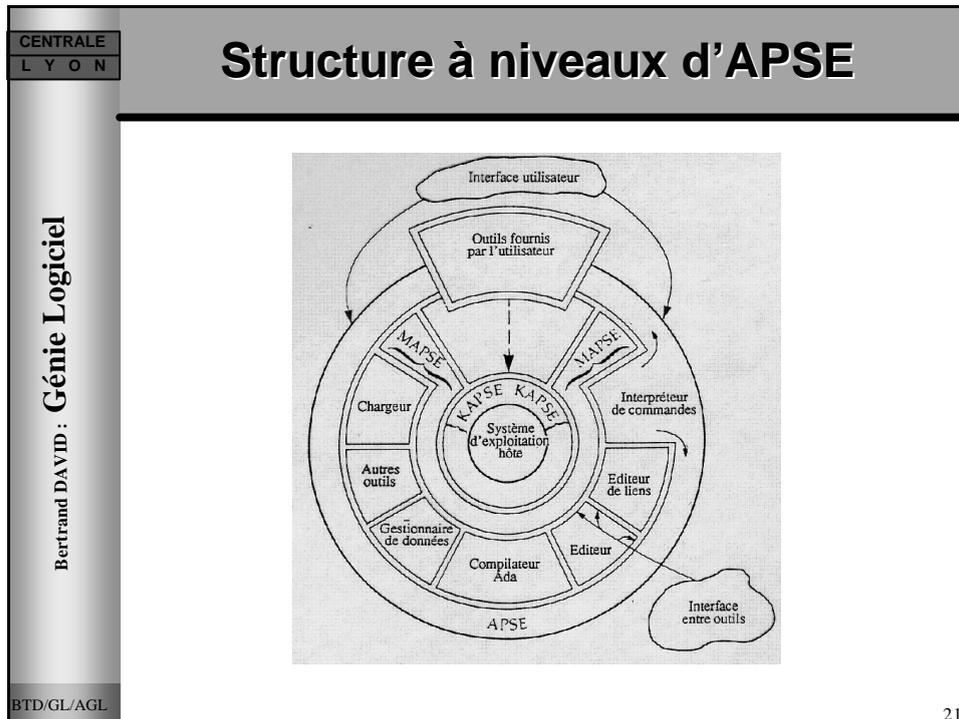
Bertrand DAVID : Génie Logiciel

| Composante principale / secondaire | OUTILS | LANGAGE | METHODE |
|------------------------------------|-------------------------------|---|---|
| OUTILS | Atelier Boite à Outils | Atelier de production pour un langage donné | Atelier Boite à Outils avec un schéma directeur |
| LANGAGE | Atelier dirigé par le langage | Environnement classique de programmation | Schéma directeur pour un langage |
| METHODE | Atelier classique | Atelier avec une cible précise | Environnement de conception |

BTD/GL/AGL 18

| | |
|---------------------------------|--|
| CENTRALE L Y O N | L'environnement support de programmation pour ADA : |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> ● APSE : ADA Programming Support Environment <ul style="list-style-type: none"> – environnement hôte/cible : cible du développement peut être la même machine ou une machine différente – base de programmes : bibliothèques, unités de programme, codes source et objet, documentation,... – avec comme objectif de constituer le point de contrôle central au chef de projet. |
| BTD/GL/AGL | 19 |

| | |
|---------------------------------|---|
| CENTRALE L Y O N | Structure à niveaux pour APSE |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> ● niveau le plus interne : système d'exploitation de la machine hôte avec la base de données, ● niveau suivant : noyau d'environnement support de programmation Ada (Kernel Ada Programming Support Environment : KAPSE), assure la correspondance logique/physique pour le reste de l'APSE ; ● au-dessus du KAPSE se trouve l'environnement support de programmation Ada minimal (Minima Ada Programming Support Environment : MAPSE), qui contient l'ensemble d'outils minimal permettant le développement de programmes ; ● au plus haut niveau se trouve l'APSE lui-même, qui comporte un ensemble d'outils avancés supportant les différentes phases du cycle de vie. |
| BTD/GL/AGL | 20 |



| | |
|---------------------------------|---|
| CENTRALE L Y O N | <h1>MAPSE</h1> |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> • Un ensemble minimal d'outils logiciels, contient tous les outils de base pour le développement du programme : <ul style="list-style-type: none"> - Editeur de texte - Formateur - Compilateur - Editeur de liens - Analyseur statique d'utilisation de variables - Analyseur statique de flot de contrôle - Outils d'analyse dynamique - Routines d'interface avec le terminal - Administrateur de fichiers - Gestionnaire de configuration • STONEMAN suggère seulement des outils, l'implémentation peut donc en ajouter ou enlever d'autres : <ul style="list-style-type: none"> - le modèle de MAPSE suit la démarche édition – compilation – édition de liens – exécution. - compilation peut être par lot ou incrémentale ; - des outils de mise au point : metteurs au point symboliques, outils d'analyse statique ou dynamique peuvent faire partie du MAPSE ; - des outils de chef de projet comme la gestion de configuration, outils de suivi de projet, d'analyse et de suivi du cahier des charges et outils d'évaluation des coûts peuvent y être intégrés. |
| BTD/GL/AGL | 23 |

| | |
|---------------------------------|---|
| CENTRALE L Y O N | <h1>MAPSE (2)</h1> |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> • Des outils de documentation pour administrer les problèmes de la création, du formatage et de la maintenance des documents devraient s'y trouver. • STONEMAN demande que cet ensemble d'outils soit coordonné pour permettre au programmeur de passer d'un outil à l'autre. • Pour faciliter la transportabilité du logiciel entre les outils, de nombreuses implémentations utilisent un langage intermédiaire pour représenter le code source. • Un langage intermédiaire fréquemment utilisé est la notation intermédiaire descriptive attribuée pour Ada (Descriptive Intermediate Attributed Notation for Ada : DIANA). |
| BTD/GL/AGL | 24 |

| | |
|---------------------------------|---|
| CENTRALE L Y O N | <h2>DIANA</h2> |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> • DIANA est généralement produit par le frontal du compilateur et contient toute l'information syntaxique et sémantique essentielle d'une unité de programme. Avec une représentation DIANA, il est possible de changer la cible du logiciel ou d'utiliser des outils qui permettent de l'examiner au niveau source ou au niveau objet. • A la base, DIANA est une abstraction arborescente. Sa structure physique est un arbre syntaxique d'une unité de programme. Cet arbre est attribué par des informations sémantiques. • DIANA est une représentation intermédiaire riche, elle contient toute l'information syntaxique et sémantique que le compilateur est capable de produire. Elle contient, de plus, l'information suffisante pour recréer l'unité de programme à partir de sa représentation DIANA. • Il est donc possible que le compilateur crée une représentation DIANA d'une unité, puis qu'un formateur reprenne cet arbre, ce qui épargne un gros effort à l'outil de formatage. |
| BTD/GL/AGL | 25 |

| | |
|---------------------------------|--|
| CENTRALE L Y O N | <h2>APSE</h2> |
| Bertrand DAVID : Génie Logiciel | <ul style="list-style-type: none"> • La vue la plus large de l'environnement, l'APSE complet, contient des outils qui transcendent les outils de base du MAPSE. • STONEMAN ne précise pas spécifiquement les types d'outils, mais il demande que l'APSE contienne des outils pour : <ul style="list-style-type: none"> - Créer des objets de la base de données - Modifier, Analyser, Transformer - Présenter, Exécuter, Maintenir • Les exigences étant minimales, on peut avoir ici un fort potentiel d'innovation : <ul style="list-style-type: none"> - Editeurs dirigés par la syntaxe - Metteurs au point temps réel - Générateurs automatiques de programmes • On peut distinguer deux classes d'outils : <ul style="list-style-type: none"> - Les outils génériques : outils utilisés pour toutes les tâches de programmation sans prendre en compte une démarche particulière. - Les outils dépendant d'une méthodologie: outils définissant ou supportant une démarche particulière de programmation ou de gestion. |
| BTD/GL/AGL | 26 |