# PLAN

I. Pourquoi : les besoins, les types d'applications

II. Comment : les technos et pratiques dont on dispose pour mettre en œuvre les applications

III. Avec quels outils
III.1 Introduction aux IDE
III.2 L'utilisation de base des IDE
III.3 L'utilisation avancée des IDE par l'exemple

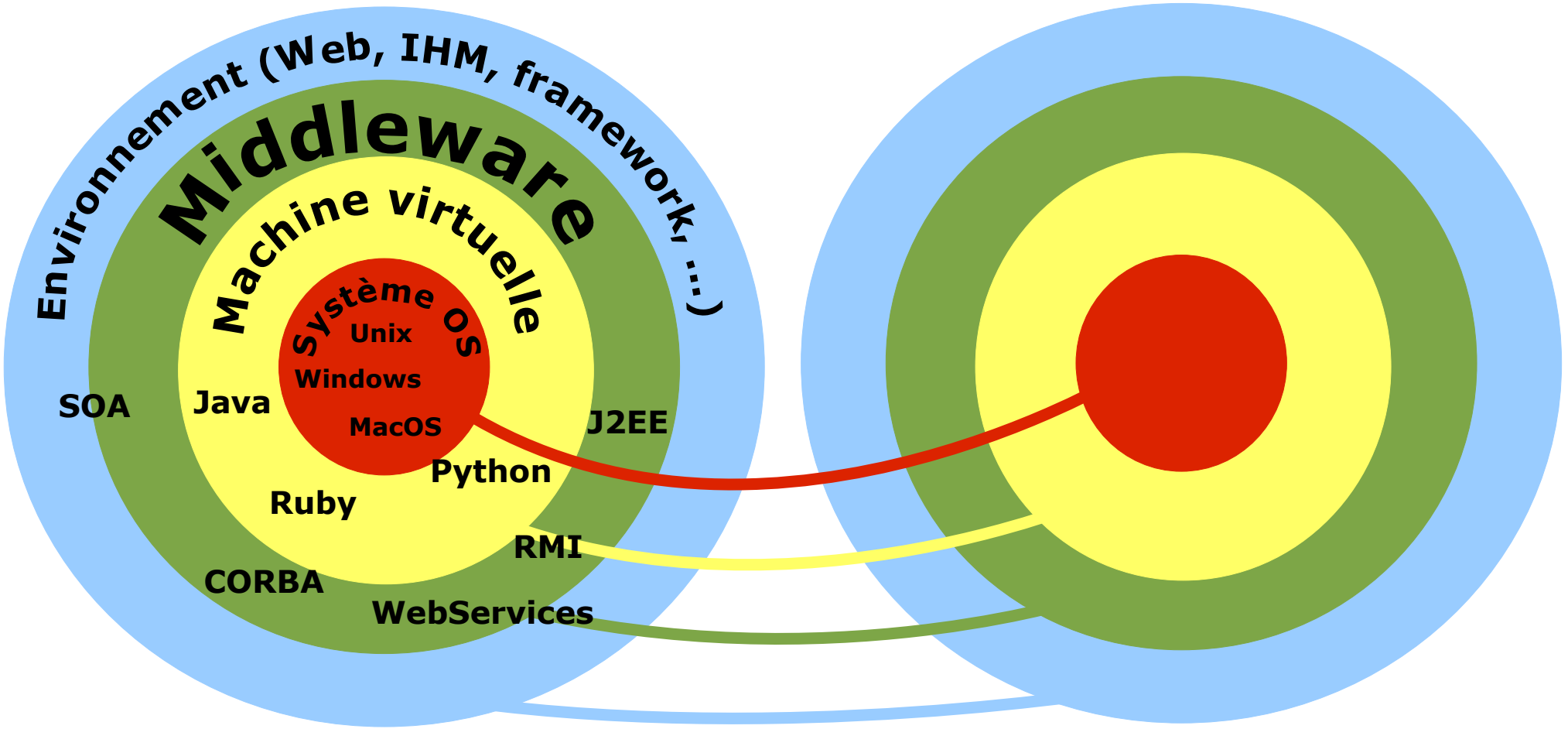
IV Travail coopératif : forges, forum, wiki, visioconférences, blog…

V Conclusion

# II. Technologies et pratiques de mise en oeuvre

- **Organisation :**
  - Tenir compte d'un contexte existant ?
  - Développer seul ou en équipe ?
  - Application ou service ?
- **Contraintes ou choix techniques :**
  - Langage (s) ?
  - Machines cibles ?
  - Type d'architecture: à plat, client/serveur, n-tiers, ....
  - Interface graphique ?
  - Utilisation de bases de données ?
  - Utilisation de bibliothèques, API, composants ... ?

# Mise en oeuvre de l'architecture logicielle



# Première étape

- Comprendre ce que désire(nt) l'(es) utilisateur(s)
  - *Description de l'activité*
  - *Vocabulaire*
  - *Description du problème*
  - *Ebauche de solution*

    - UML : pour communiquer

    - Maquettage : pour valider/vérifier la demande

# Le travail du développeur

- Quelque soit la méthode retenue: 5 étapes en 1 ou plusieurs cycles
  - Conception
  - Codage
  - Tests unitaires
  - Intégration
  - Recette

# Etape: conception

- Outils de conception :

du papier/crayon aux  outils sophistiqués pour la mise en oeuvre de différentes approches

| Langages/Techniques de description | Outils de mise en oeuvre |
|---|---|
| UML | BoUML, Papyrus, eUML, ... |
| Réseaux de Pétri | CPN-AMI, CPNTools, TINA, HiQPN-Tool, JPetriNet, ... |
| Méthode formelle B | B4free , ComenC, ... |
| Notation Z | Community Z Tools |

# Etape: Codage

- Sélection des mécanismes/technologies à utiliser
  - en fonction des besoins identifiés et des contraintes d'utilisation définies
    - Quelles machines cibles ?
    - Quel type d'interface ?
    - Quel type d'utilisateur ?
    - Quel type de déploiement ?
    - Nécessité d'intégration dans un environnement existant ?
    - .....
- Choix d'un langage de programmation:
  - à partir des objectifs à atteindre pour l'application à développer, quelques question à se poser, très dépendantes de l'application

# Etape : Codage vs langage

- Faisabilité :
  - Ressources pour satisfaire aux spécifications fonctionnelles de l'application
  - Ressources pour faire déployer et fonctionner les applications écrites sur la/les machine(s) cibles
  - Aspects techniques:
    - connexion avec des bibliothèques graphiques, de calcul, de connexion réseau,.... des SBGD,
    - existence de codes réutilisables
    - utilisable pour des applications locales, client/serveur, n-tiers, web .....
    - présence d'un « garbage collector », .....
- Fiabilité et performances de l'application :
  - Existence d'outils pour
    - test,
    - logging, (traces d'exécution)
    - analyse de code
  - Existence d'outils de
    - profiling (mesures de performances)
    - pooling (gestion de groupe de serveurs, ....)
    - .....

# Etape : Codage vs langage

- Utilisabilité
  - Possibilité de  modulariser en fichiers distincts
  - Portabilité des sources  et/ou des binaires ou bytecodes sur différents OS
  - Documentations facilement accessibles (constructeur/fournisseur, cours EnsSup, .....)
  - Outils d'aide au développement
  - Environnement de développement intégré
- Maintenabilité, extensibilité, flexibilité :
  - Langage orienté objet
  - Existence d'outils  implémentant les principaux design patterns
  - Existence  d' outils pour effectuer de la génération de code  et du « reverse engineering » (UML <=>code source)
- Pérennité:
  - Utilisé  en entreprise et en enseignement supérieur
  - Existence de communauté(s) d'utilisateurs actives
  - Normalisation ISO

# Etape: codage

- Outils de codage :

des outils mono-fonction aux outils intégrés pour le développement des différente types de logiciels

| **Fonctionnalité** | **Outils de mise en oeuvre** |
|---|---|
| langage | C/C++, Fortran, Java, HTML, XML, PHP, Python, Ruby, ..... |
| Editeur simple ou syntaxique | vim, emacs, NotePad++,jedit,gedit, Komposer,geany, .... |
| Débogage | Gdb , debugger NetBeans et Eclipse, pdb (Python debugger), jdb (Java debugger), Jswat (debugger java graphique) |
| Automatisation des tâches de compilation, génération de code, .... | ANT, MAVEN, Make, autotools, .... |
| Versionning | CVS, SVN, Git, ..... |
| Tests unitaires et mesures de performances | Junit, Cobertura, HeapAnalyser (Eclipse), VisualVM(profiler pour java), Jprobe, Gprof, python-profiler, ..... |
| Analyse de code | Checkstyle, findbugs, ..... |
| Création d'interfaces graphiques | Plugin VE (Eclipse) |
| Documentation du code | Javadoc , doxygen |
| Bases de données | |
| Déploiement d'applications | JavaWebStart, plateformes OSGi (Equinox, OSCAR, .....) |

En fonction du type de logiciel à développer, des technologies utilisées, des choix d' outils vont se faire de façon naturelle

# Etape: Tests unitaires

- Objectif: vérifier le « bon fonctionnement » d'une classe, d'une méthode, d'une fonction, d'une procédure, .....
  - Lors de la première mise en oeuvre
    - En utilisant les use cases définissant le fonctionnement attendu de l'application
  - Après des modifications
    - En effectuant des tests de non-régression
    - En re-utilisant les use-cases pour vérifier le fonctnnement des nouvelles fonctionnalités
- Types de tests
  - Manuels: définis à partir des uses cases, et en relation avec les utilisateurs finaux (si possible): tests significatifs pour l'application
  - Automatique:  sur une spécification en logique temporelle des propriétés impérative de l'application, on utilise un Model Checker (fourniture de contre-exemple lorsqu'une propriété n'et pas vérifiée)
- Exemples d'outils
  - junit, Dart,QMTest,JFunc, GUItest

# Etape: Intégration

- Objectif:
  - s'assurer que les briques logicielles développées fonctionnent correctement entre elles.
  - Vérifier que les fonctionnalités de l'applications correspondent aux spécifications
- Types de tests
  - Définis sur la base des use cases et des spécifications initiales
  - Difficilement automatisables
  - Mais il existe des outils d'évaluation de la couverture du code par les tests
    - Cobertura
    - clover

# Etape: Recette

- Point de vue du développeur
  - Eléments logiciels:
    - Logiciel
    - Script d'installation ou procédure de déploiement
    - Système de suivi et de maintenance : trac, bugzilla, ...
  - Documentation
    - Manuel d'installation
    - Mode d'emploi
    - Jeux de tests (au moins de l'étape intégration)