Environnements de développement

Véronique BAUDIN

Pascal DAYRE

Introduction

« La grande diffusion de l'informatique au niveau industriel due à l'essor technologique de ces dernières années a suscité un intérêt sans cesse croissant envers les problèmes posés par la production de logiciels.
Les progrès constants de l'industrie électronique ont des conséquences importantes tant sur le plan économique que technique. En effet, l'accroissement considérable de la puissance du matériel, conjugué à l'abaissement de son coût a permis à l'informatique de diversifier son champ d'application, tout en s'attaquant à la résolution de tâches de plus en plus complexes »

Revue « Génie Logiciel » - numéro 1 - 1984

Notre contexte

En tant que développeur, nous avons certainement tous été confrontés à ce type de situation:

- dans le cadre d'un logiciel qui doit répondre aux besoins de 2 ou 3 chercheurs, ou de 2 ou 3 équipes d'un même laboratoire ou de laboratoire différents

- dans le cadre de projets contractuels du type PCRD, ANR ou autre, dans lesquels nous avons à prendre en compte la présence d'industriels qui n'ont pas forcément les mêmes habitudes de développement

- dans le cadre de l'amélioration ou de la modification de logiciels libres développés par d'autre et sur lesquels on nous a demandé de travailler

Dans tous ces cas de figure, on nous impose d'utiliser ou de choisir un outil ou un ensemble d'outils pour réaliser notre logiciel: mais pourquoi ?

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

I. Contraintes et types d'applications

- Contraintes de réalisation
  - Délais
  - Qualité
  - Ressources humaines et matérielles
- Types d'applications
  - Accessibles totalement par le web
  - Application « standalone »
  - Réalisation de composants à intégrer
  - Réalisation de services
  - .....

Cycle de vie

- Qu'est-ce ?
  - Un processus
    - Phases: création, distribution, disparition
- Pourquoi ?
  - 2 buts
    - Maîtriser des risques, des délais et des coûts
    - Contrôler la qualité / aux exigences

Phases du cycle de vie

- **Définition des objectifs: que doit faire ce logiciel ?**
- **Analyse des besoins et faisabilité: recueil et formalisation des besoins du demandeur et de l'ensemble des contraintes.**
- **Conception générale: élaboration des spécifications de l'architecture globale du logiciel.**
- **Conception détaillée: définition précise de chaque sous-ensemble du logiciel.**
- **Codage : implémentation des fonctionnalités définies lors des phases de conception**
- **Tests unitaires: permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.**
- **Intégration: pour s'assurer de l'interfaçage correct des différents éléments du logiciel.**

Documentation

- **Toutes les informations nécessaires pour utiliser et ajouter des développements au logiciel (manuel d'installation, jeu de tests, manuel(s) utilisateur, spécification, dossier d'architecture, dossier de conception)**

Phases du cycle de vie

- **Recette: vérification de la conformité du logiciel aux spécifications initiales.**
- **Installation et déploiement**
- **Valorisation**
- **Maintenance corrective et évolutive, support.**

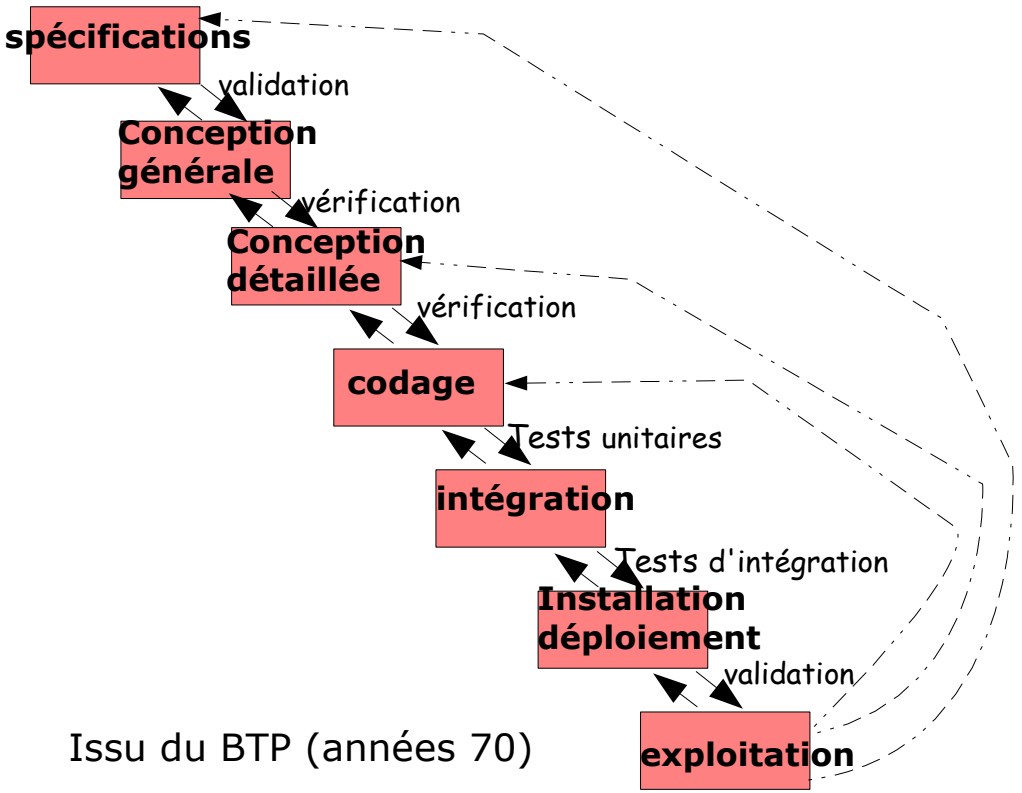
Phases du cycle de vie

- **Migration vers une nouvelle application**
- **Arrêt progressif du service ou de l'application**

Quelques points clés pour le développement de logiciel

- Bien comprendre les demandes des utilisateurs finaux
- Tenir compte des modifications du cahier des charges
- Eviter de découvrir trop tard les défauts sérieux du projet
- Traiter au plus tôt tous les points critiques du projet
- Définir une architecture robuste et adaptée
- Bien maîtriser la complexité du système
- Bien communiquer avec l'utilisateur final
- Favoriser la réutilisation
- Faciliter le travail en équipe

Les modèles de cycle de vie: cascade

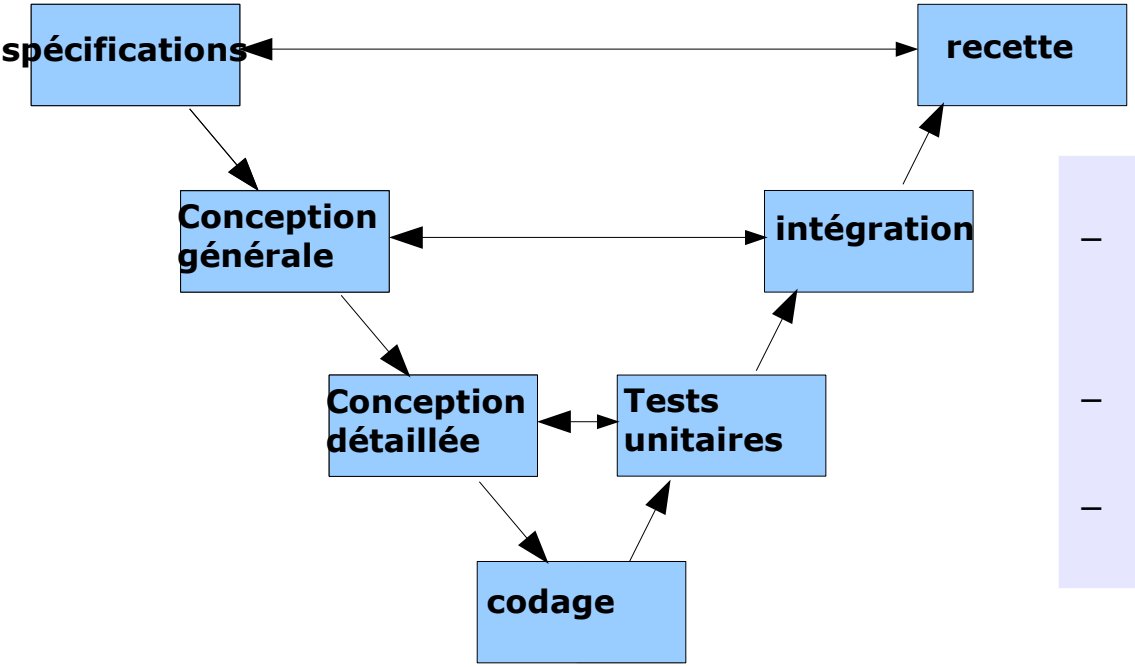


Issu du BTP (années 70)

Critiques

- Connaissances des besoins / exigences au départ
- Conception réalisée à partir des spécifications (pas de retour arrière à ce niveau)
- Codage / implémentation: idem
- Accorde une très grande importance aux documents (cycle déterminé de relecture, commentaires, nouvelle version)
- Démarrage tardif du codage

Les modèles de cycle de vie: en V

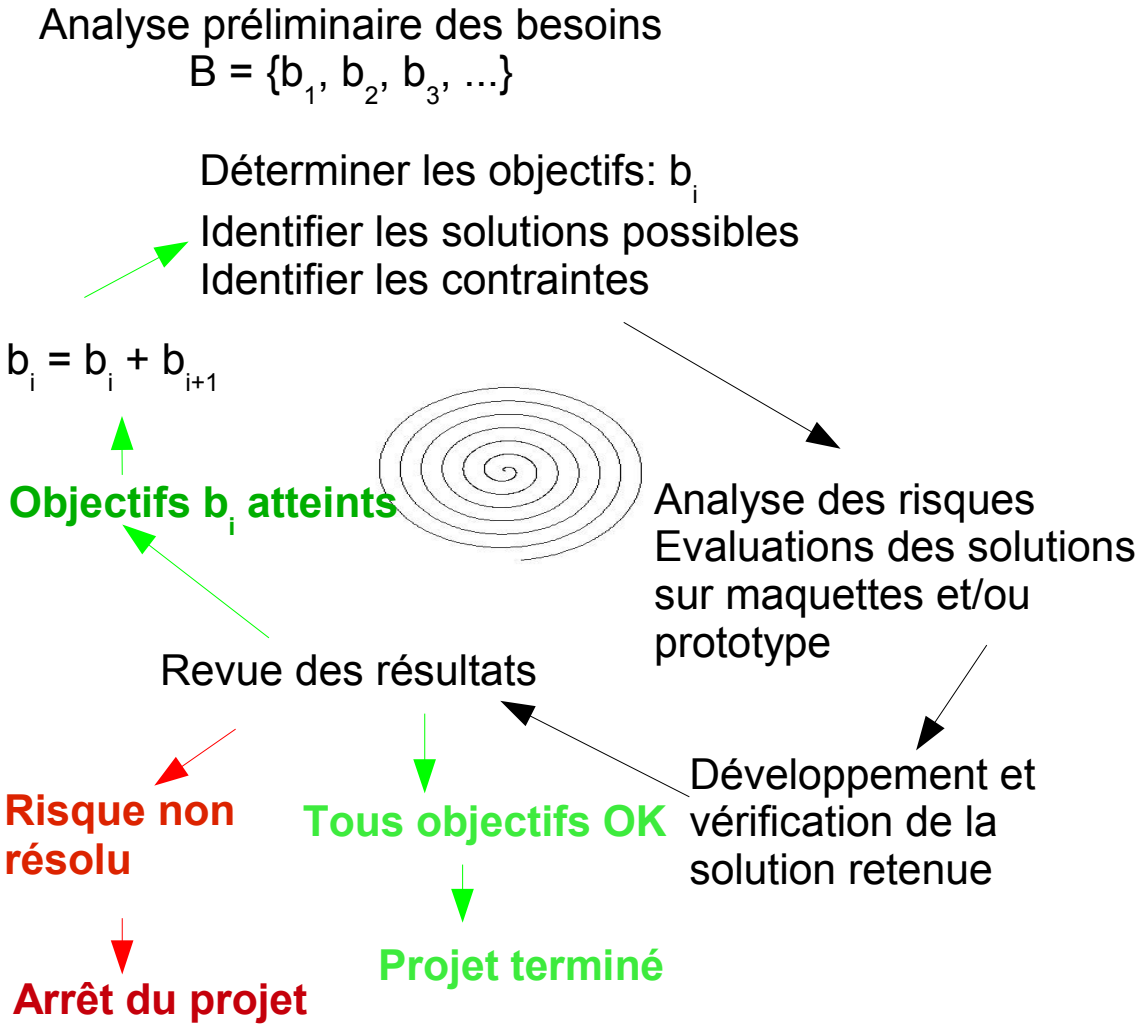


AFCIQ (années 80)

Critiques

- Prise en compte difficile des évolutions du cahier des charges pendant la construction du système
- Pas de validation/vérification à la fin de chaque étape
- Peu ou pas de possibilité de maquettage et/ou de prototypage

Les modèles de cycle de vie: en spirale



Barry W. Boehm 1988

Avantages

- Analyse préliminaire affinée au cours des premiers cycles.
- Maquettes exploratoires pour guider la phase de conception du cycle suivant.
- Le dernier cycle se termine par un processus de développement classique.

Inconvénients

- Modèle moins expérimenté que les deux précédents.
- Mise en oeuvre : grandes compétences et particulièrement adapté aux projets innovants ('importance accordée à l'analyse des risques)

Modèles par incrément

Dans les modèles par incrément un seul ensemble de composants est développé à la fois : des incréments viennent s'intégrer à un noyau de logiciel développé au préalable. Chaque incrément est développé selon l'un des modèles précédents.

Avantages

* chaque développement est moins complexe ;
* les intégrations sont progressives ;
* possibilité de livraisons et de mises en service après chaque incrément ;
* meilleur lissage du temps et de l'effort de développement à cause de la possibilité de recouvrement des différentes phases.

Risques

* mettre en cause le noyau ou les incréments précédents ;
* ne pas pouvoir intégrer de nouveaux incréments.

Suggestion

Les noyaux, les incréments ainsi que leurs interactions doivent donc être faites globalement, au début du projet. Les incréments doivent être aussi indépendants que possible sur le plan fonctionnel mais aussi sur le plan du calendrier du développement.

Processus Unifié

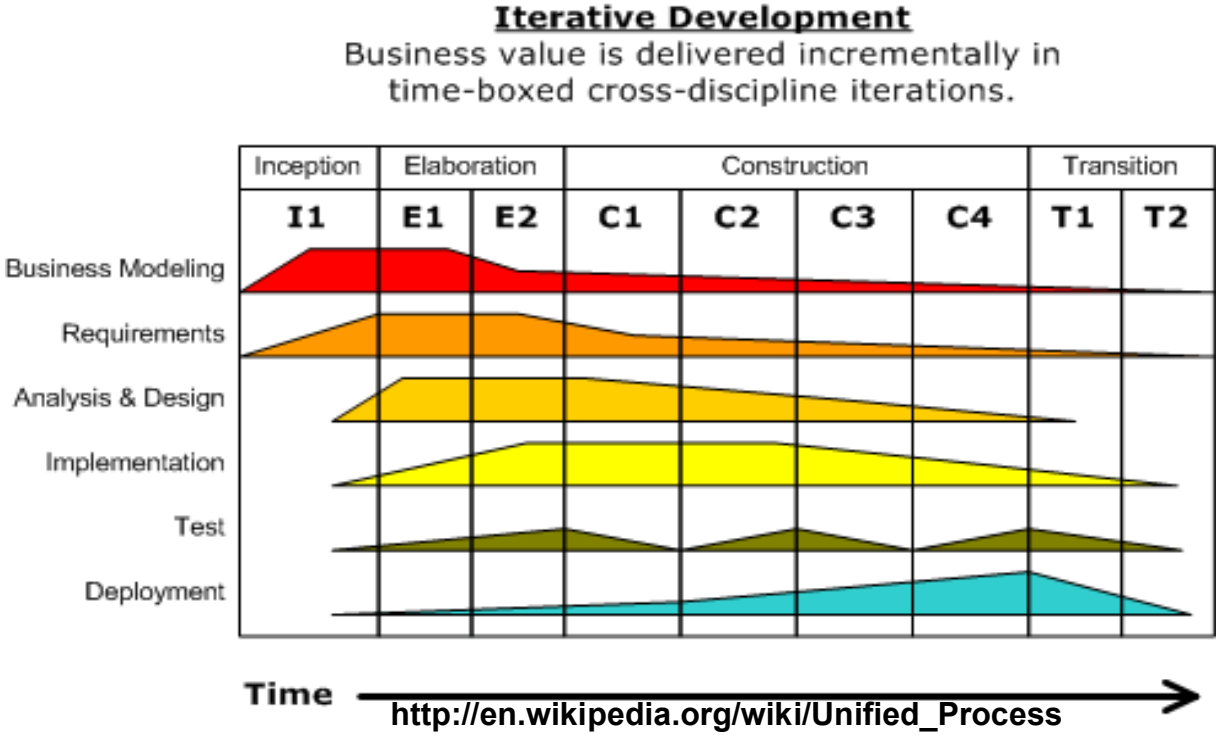
- Apparu dans les années 98: processus générique de développement de logiciel
- Propose un ensemble de bonnes pratiques largement répandues dans les processus de développement
- Permet de définir et d'affecter des tâches et des responsabilités dans une organisation de développement
- Assurer la production de logiciels de bonne qualité, répondant aux demandes des utilisateurs finaux en respectant les contraintes de coût et de délais
- Caractéristiques:
  - processus itératif
  - centré sur l'architecture
  - piloté par des cas d'utilisation
  - orienté vers la diminution des risques

Les 7 bonnes pratiques préconisées dans le Processus Unifié

- Développement itératif
- Développement à base de composants centré sur l'architecture
- Pilotage par les risques
- Gestion des exigences
- Maîtrise des modifications
- Evaluation continue de la qualité
- Modélisation visuelle

Principes généraux de fonctionnement de UP

- Répétition d'un cycle tant que les besoins exprimés par l'utilisateur ne sont pas remplis ou abandonnés



http://en.wikipedia.org/wiki/Unified_Process

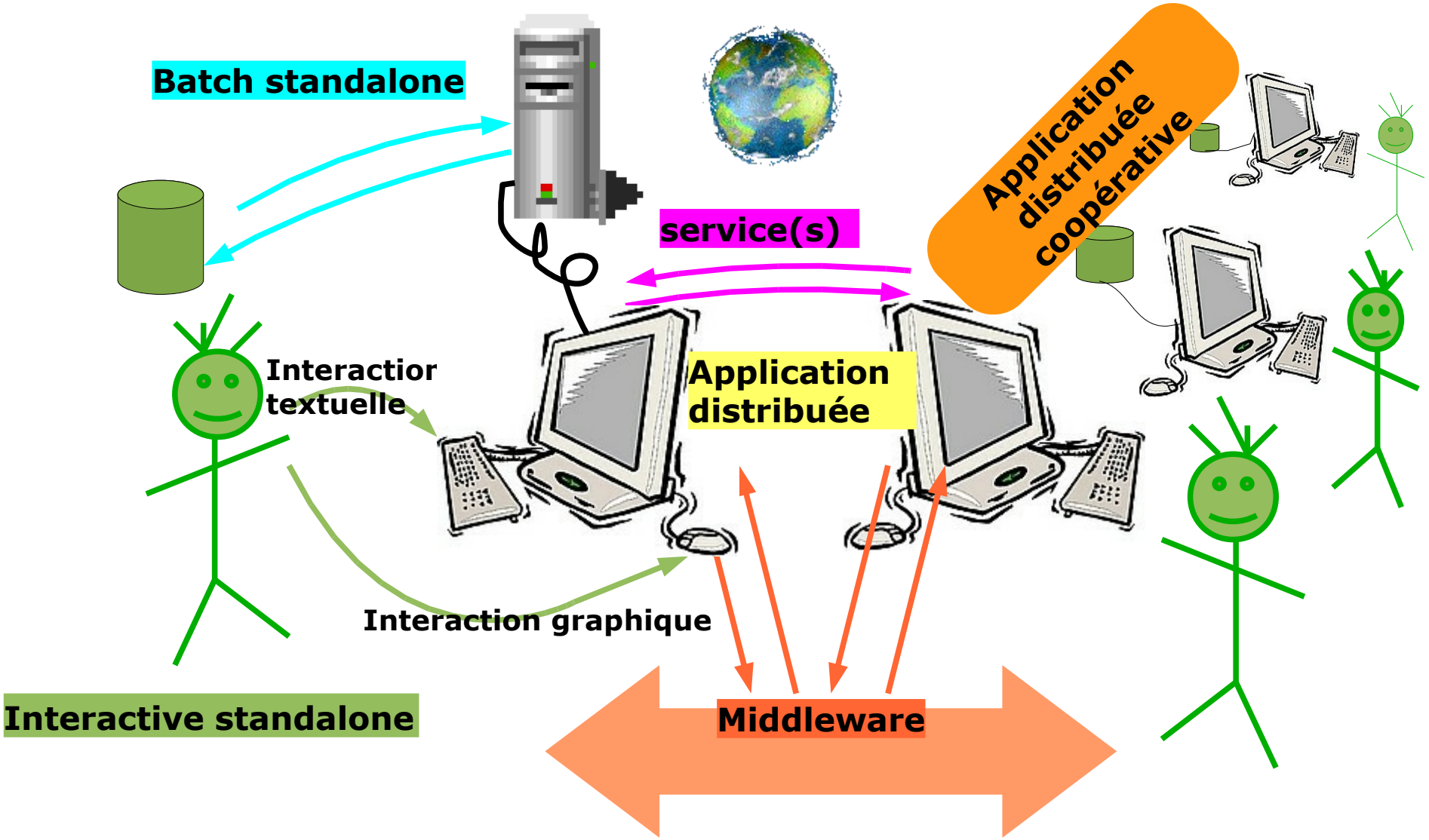
Inception: définition de la portée du projet
Elaboration: planification du projet, spécification des fonctionnalités, architecture de base
Construction: réalisation du produit
Transition: transfert du produit vers les utilisateurs

http://www.irisa.fr/prive/jezequel/enseignement/analyse.pdf

UP: implémentations et outils

- RUP (Rational Unified Process)
  - IBM Rational Methode Composer
    - http://www-01.ibm.com/software/awdtools/rmc/features/index.html
  - OpenUP Eclipse Process
    - http://www.journaldunet.com/developpeur/outils/tutoriel-pratique/08/0130-eclipse-epfc-1/1.shtml
- 2TUP Two tracks unified process ou cycle en Y
  - Version remaniée de UP par Valtech
    - UML2 en action: de l'analyse des besoins à la conception - P Roques, F Vallée Ed. Eyrolles 03-2007
- EUP Enterprise Unified Process
  - Version complétée de RUP
    - http://www.enterpriseunifiedprocess.com/
- AUP Agile Unified Process
  - Version de RUP prenant en compte uniquement les aspects « agile »
    - http://www.ambysoft.com/unifiedprocess/agileUP.html

Typologie des applications



Conclusion

Nous avons survolé plusieurs modèles de cycle de vie:

- **pourquoi plusieurs modèles ?**
  - parce que aucun n'est parfait, ou meilleur que les autres
  - chacun présente des qualités et des défauts, dépendant du contexte dans lequel il est mis en oeuvre
- **conclusion ?**
  - identifier le modèle qui semble le mieux approprié à votre contexte, et suivre les grandes lignes de celui-ci, sans s'imposer des règles strictes
  - garder toujours un oeil critique sur la méthode / contexte

Quid des types de développement ?

On voit donc se dégager 2 grandes familles de pratiques pour les développements:

- codage de l'ensemble des fonctionnalités définies à partir des besoins identifiés
- codage par le biais de maquettes ou prototypes successifs permettant d'inclure petit à petit les fonctionnalités identifiées initialement ou au fil des échanges entre concepteurs et client/demandeur

Pourquoi utiliser des méthodes de développement?

Pour guider les développeurs de la phase d'analyse à celle de maintenance