

Le génie Logiciel (suite)

Lors du cours précédent, on a étudié différents cycles de vie, dont la cascade, ou la spirale.

Analyse des besoins

L'analyse des besoins est une étape menant à l'élaboration de spécifications.

Definition

C'est le processus visant à établir quelles fonctionnalités le système doit fournir, et les contraintes auxquelles il sera soumis.

Ce processus se résume aisément ainsi : **“Quoi ?”** et non **“comment ?”**.

Objectifs

- Comprendre la nature exacte du problème, par l'analyste et par le client
- Fournir un document compréhensible par le client et les concepteurs du système, ce qui fournira une sorte de fondement du contrat
- Définir la base pour la validation des étapes ultérieures (A-t'on conçu ce qui a été demandé?)
- Préciser les contraintes de réalisation
- Prendre des décisions : Développer ? Acheter ? Louer ? Sous-traiter ?
- Réduire les coûts de développement du système. Eviter l'oubli, les allers-retours dans le cycle de vie.

Les étapes

- **Analyse des besoins (Requirement Analysis)** : Rassembler toutes les informations nécessaires à la compréhension du problème
- **Spécification des besoins (Requirement specifications)** : Spécifier de manière claire et précise le résultat de l'analyse en utilisant des méthodes de modélisation

Les besoins

Il en existe 2 catégories

- **besoins fonctionnels (exigences fonctionnelles)** : ce que l'utilisateur attend en terme de fonctionnalités. exemple : dans un DAB, on aura besoin d'implémenter la vérification de la validité de la carte bleue.

- **besoins non fonctionnels (spécifications techniques)** : conformité aux standards, contraintes sous lesquelles le système doit rester opérationnel. exemple : dans un DAB, le distributeur doit répondre au plus 2 secondes après insertion de la carte, ou encore, contrainte du système d'exploitation (on demande un travail sous UNIX).

Exemple d'étude de cas : Projet ARENA

- Développer une infrastructure pour une arène opérationnelle : enregistrer des nouveaux jeux et joueurs, organiser des tournois, gérer les scores des joueurs, etc.
- Offrir la possibilité aux organisateurs de programmer des matchs et de gérer leurs résultats.
- Fournir aux développeurs de jeux d'en développer de nouveaux ou d'adapter l'existant en utilisant ARENA.

Besoins fonctionnels

- définir un nouveau jeu, un nouveau tournoi
- Permettre à un joueur de s'enregistrer, être membre d'une ligue, jouer des matchs, etc.

Besoins non fonctionnels

- Le système doit supporter le déroulement de plusieurs tournois en parallèle (10 maximum), chacun peut impliquer au maximum 64 joueurs et plusieurs centaines de spectateurs.
- Les joueurs doivent pouvoir jouer avec au moins un modem 56k.

Les besoins non fonctionnels les plus courants sont listés ci-dessous :

- **Besoins en performance**
 - temps de réponse
 - Espace mémoire nécessaire
 - Restrictions sur la représentation des données
 - Nombre d'utilisateurs
 - Nombre de terminaux
 - Tableau de métriques
- **Contraintes de conception**
 - utilisation de méthodes standards
 - Langage de programmation
 - Système d'exploitation
 - matériels
 - structure de la documentation (formats de rapports)
- **Interfaces externes**, elles expriment les relations entre le système et :
 - Les utilisateurs (messages d'erreurs ...)
 - Le matériel

- Le logiciel (liaison avec une base de données, des bibliothèques mathématiques, graphiques, procédures d'échanges de messages).

L'analyse des besoins en pratique

Les différents acteurs lors de cette étape sont l'analyste, le client, et l'utilisateur final.

Procédé

- Séries de questions posées aux clients et aux utilisateurs
- Eventuellement, les documents fournis par le client
- Analyse de la tâche

Problèmes rencontrés

- comment organiser et structurer les informations obtenues ?
- Comment résoudre les contradictions pouvant exister entre les différents interlocuteurs ?

Elements de solution

- Utiliser des méthodes d'analyse : SA, SADT, Merise, OOA ...
- Qualités de l'analyste en matière de communication
- Application de 3 principes
 - **Décomposition** des problèmes en plusieurs parties (ex d'un système d'exploitation : difficilement appréhendable en tant que tel, on le décompose alors en sous-système).
 - **Abstraction** : appréhender les problèmes à un bon niveau d'abstraction, éviter les détails. (Dans l'exemple d'un OS : creation / destruction de fichiers quel qu'en soit le système de fichiers, par exemple ...).
 - **Projection** : Etudier le système a partir de plusieurs points de vue : utilisateur, programmeur et administrateur.

Types de spécifications

Il existe différents types de spécifications,

Spécification informelle

généralement rédigée en langage naturel

- pas de structures précises de document
- Pas de contenu imposé
- Risque d'ambiguïtés
- Pas d'outils
- Lisibles (?) par l'utilisateur final

Spécification standard

- Plan type
- contenu imposé
- Règles de conservation et mise à jour

Spécification semi-fonctionnelle

Ce type est lié aux méthodologies employées, à base de graphiques et utilisant des modèles.

Les trois méthodes citées ci-dessus sont très utilisées dans l'industrie

Spécification formelle

Elle est fondée sur des langages de spécification, et reste dans un domaine encore très universitaire.

Structure d'un document de spécification

Il est en règle générale composé des éléments suivants :

- introduction
- Description générale
 - Environnement / contexte du système
 - Modèle conceptuel (différent du MCD) : regrouper l'ensemble des fonctionnalités ...
 - Caractéristiques des utilisateurs

Cette description mènera à la rédaction d'un cahier des charges. Un détail plus poussé du cahier des charges donnera naissance aux spécifications.
- Besoins fonctionnels
- Spécification des structures de données Spécifie les entités les plus importantes du système, ainsi que les associations qui les relient (MEA ?).
- Besoins en performance (non fonctionnels)
- Contraintes de développement
- Références (bibliographie, source d'obtention ...)
- Index

- Annexes (description, tout ce qui peut être vulgarisé par le lecteur ...)

Historique des méthodes d'analyse

- Années 70 : Méthodes orientées vers les fonctionnalités des systèmes, décomposition fonctionnelle et hiérarchique d'un système.
- Années 80 : Méthodes systémiques, modélisation des données et modélisation des traitements de manière séparée.
- 1990-1995 : Emergence des méthodes objet analyser et modéliser un système en considérant de manière unifiée les données et les traitements Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE ...
Aucune méthode ne s'est réellement imposée.
- 1995-1997 : Unification et normalisation des méthodes vers une sorte de standard appelé UML.

UML est le résultat d'un travail de synthèse et de fusion des méthodes dominantes. Il devient un standard. Notons toutefois qu'UML est un **langage**, pas une méthode, bien qu'au final on puisse l'assimiler en tant que tel.

UML

UML est un langage graphique qui propose des diagrammes

- Diagrammes des cas d'utilisation
- Diagrammes d'objets
- Diagrammes de classes
- Diagrammes de déploiement
- Diagrammes de collaboration
- Diagrammes de séquences
- Diagrammes d'états-transitions
- Diagrammes d'activités

Ces diagrammes n'ont pas nécessairement une utilité dans tous les cas. Par exemple, dans le cas d'une application de gestion, des diagrammes d'états-transitions ne sert à rien.