Complexité des logiciels

Classes de logiciels

- Logiciels amateurs
- Logiciels jetables (consommables)
- Logiciels de qualité industrielle
- Logiciel vitaux

La complexité des logiciels

- “Dieu ne joue pas aux dés”, Einstein
- La nature doit s’expliquer simplement, car il n’y a pas de place pour le caprice ou l’arbitraire
- Pour le logiciel ceci n’est malheureusement pas vrai
- Il faut lutter contre la complexité inutile

Origines de la complexité des logiciels

- La problématique du domaine
- Le processus de développement
- L’ inhérente flexibilité du logiciel
- La caractérisation du comportement des systèmes discrets

Conséquence de la complexité

- Le risque de déficiences graves est élevé
- La mise au point est lente et chaotique
- La maintenance est disproportionnée
- Le coût est astronomique
- Crise du logiciel (OTAN Garmish 1968)

Caractéristiques des systèmes complexes

- La complexité s'organise en hiérarchie
- La détermination des composants primitifs est arbitraire et dépend du point de vue
- Le couplage entre composants est faible
- La cohésion au sein d'un composant est forte
- La nature est économe, la diversité des composants est faible
- Un système complexe qui fonctionne est une évolution d'un système plus simple qui a fonctionné

Forme canonique d'un système complexe

- Multiplicité de la notion de hiérarchie
- Hiérarchie de type "être une sorte de"
- Hiérarchie de type "avoir pour composants"
- Orthogonalité des deux hiérarchies

Limites des capacités de l'être humain

- Nombre de concepts manipulés (7 éléments plus ou moins 2)
- Vitesse de transfert (5 secondes entre paquets)
- Vitesse de traitement
- Vitesse et limitations des dispositifs d'entrée-sortie

Examen des techniques de gestion de la complexité

- A défaut de réduire la complexité il faut la maîtriser
- Donner une illusion de simplicité
- Application de critères de partionnement (accidentel, fonctionnel, logique, ...)

Rôle de la décomposition

- Diviser pour régner
- Raffinage récursif jusqu'à obtention d'éléments compréhensibles
- Division intelligente de l'espace des états d'un système

Décomposition algorithmique

- Approche traditionnelle
- Chaque module représente une étape du processus global
- Décomposition fonctionnelle depuis le cahier des charges jusqu'au sous-programme

Décomposition orientée-objet

- Approche plus récente (en informatique)
- Chaque module représente un objet du domaine de l'application
- Les objets sont des entités autonomes qui collaborent afin de réaliser un projet global

Comparaison entre techniques de décomposition

- Les deux techniques de décomposition sont intéressantes
- Elles sont complètement indépendantes (orthogonales)
- Il faut d'abord en choisir une pour commencer à décomposer
- Puis se servir du résultat comme cadre pour exprimer l'autre point de vue

Décomposition / Composition

- Le terme décomposition orientée-objets est réducteur
- L'approche orientée-objet n'est pas seulement descendante
- Approche descendante, ascendante, récursive, itérative, incrémentale, ...

Rôle de l'abstraction

- L'abstraction permet de gérer la complexité en réduisant la quantité d'information
- L'abstraction va à l'essentiel (suivant un point de vue)
- L'abstraction met en avant les caractéristiques principales
- L'abstraction dissimule les caractéristiques secondaires

Rôle de la hiérarchie

- Il est possible de faire des abstractions d'abstractions
- Recherche des similitudes au sein d'un groupe d'abstractions
- Les hiérachies d'abstractions ne sont pas faciles à déterminer

La complexité des logiciels : conclusion

- Le logiciel est complexe par nature
- Il existe des hiérarchies de complexité
- Les systèmes peuvent être décomposés selon ce qu'ils font ou selon ce qu'ils sont
- L'approche orientée objet conduit à une décomposition exprimée à partir des éléments du domaine de l'application