MGL7460 – Réalisation et maintenance des logiciels

Systèmes patrimoniaux

Jacques Berger

Objectifs

Définir les systèmes patrimoniaux

Méthodes de travail avec les systèmes patrimoniaux

Prérequis

Aucun

Systèmes patrimoniaux

Qu‘est-ce qu‘un système patrimonial?

Systèmes patrimoniaux

Vieux système

Écrit par d'autres personnes

Difficile à maintenir

Code avec couplage très fort

Systèmes patrimoniaux

Code spaghetti

Incompréhension du système

Dans un vieux langage, plus supporté ou mal compris

Personne ne veut s'y aventurer

Systèmes patrimoniaux

Documentation déficiente

Système important

Tests désuets

Livre

Working Effectively with Legacy Code
Michael Feathers

Livre

Un système patrimonial est un sytème sans tests

Couverture de tests : sécurité lors du refactoring

Livre

Améliorer le code à travers le refactoring sécuritaire

Techniques :

- Harnais de tests
- Tests de caractérisation

Discussion

Que faire s'il n'existe pas de framework de test avec le langage utilisé?
Ex. C, Pascal ou COBOL

Méthodes

Méthodes de travail :

- Lorsque le refactoring sécuritaire est possible
- Lorsque le refactoring est trop risqué

Refactoring sécuritaire

Les bogues injectés vont être très difficiles à détecter et à corriger

Tout mettre en place pour ne PAS injecter de bogue (tests, révision de code, etc.)

Refactoring sécuritaire

Être très prudent dans les manipulations, plus que d'habitude

Un framework de mocking peut aider à mettre du code complexe sous test

Refactoring sécuritaire

Si on trouve un bogue déjà en place durant un refactoring, on laisse le bogue là. Du code dépend probablement de ce bogue, il faut faire une analyse plus approfondie avant de le corriger.

Refactoring trop risqué

Constuire une interface entre l'ancien code et le nouveau code, faire communiquer le nouveau code avec l'ancien code à travers l'interface

Refactoring trop risqué

Hériter des anciens objets pour leur ajouter de nouveaux comportements

Éviter de toucher à l'ancien code, autant que possible limiter les changements à l'ajout d'une ligne, un appel à l'interface du nouveau code

Refactoring trop risqué

La séparation entre le nouveau code et l'ancien code favorise l'ajout de nouvelles fonctionnalités mais augmente la complexité globale du système et nuit à la maintenabilité à long terme