Introduction

Qualité de Développement — R5.A.08

C. Raïevsky

2026-2027

IUT A Valence
Université Grenoble Alpes

Département Informatique

BUT Informatique 3ème année

Organisation Générale

14 séances d'une heure trente

- Une partie théorique éventuelle sur :
  - La qualité de développement
  - Le langage Rust
- Une partie pratique, en Rust donc
  - Pratique du langage
  - Sensibilisation aux enjeux de qualité
  - Mise en application des principes, méthodes et outils vus en cours

Parties pratiques

Travaux Pratique

- Une remise en route sur Rust
- Mise en place d'une chaîne d'intégration continue

Projet de programmation répartie

- Sujet libre
- Avec contraintes techniques
- En groupe
- Soutenances

Plan

Qualité de développement :

1. Définition
2. Utilité - Objectifs
3. Critères
4. Quand ? Comment ?

Qualité de Développement

Mise en contexte

Qualité de développement : un aspect de la qualité logicielle

Qualité Logicielle

Qu'est-ce que la qualité logicielle ?

Qualité Logicielle

Qu'est-ce que la qualité logicielle ?

Est-ce la même chose pour :

- Un script de conversion pdf $\rightarrow$ svg ?
- Un TP de programmation système ?
- Le logiciel d'évitement automatique de collision aérienne ?
- Le logiciel qui contrôle un robot chirurgical ?
- Un jeu vidéo ?

Qualité Logicielle - critères

Principaux critères de qualité logicielle

- Fonctionnalité
- Fiabilité
- Sûreté
- Sécurité
- Conformité
- Compatibilité
- Portabilité
- Performance
- Maintenabilité
- Utilisabilité
- Testabilité

Qualité Logicielle

Lors de la conception, il faut choisir :

- Les critères retenus
- Les niveaux d'exigence

Ces choix dépendent :

- De la réglementation
- Des normes
- Du contexte client
- Des ressources disponibles
- De la politique de l'entreprise

Qualité de Développement

Lors du développement, il faut mettre en place :

- Des méthodes
- Des outils

Pour garantir que les choix sont respectés

Qualité de Développement

Démarche de professionalisation

Durant la formation (TP) ou pour des projets personnels

- Peu d'exigence
- "Ça fonctionne"

Quand le logiciel est un produit

- Le client a son mot à dire
- Norme ?
- Réglementation ?
- Peut-être qu'il ne veut pas payer 2 semaines de travail pour gagner 1 seconde au démarrage
- L'entreprises aussi peut avoir des exigences

Exemples

Valve's Steam update script

```
1 STEAMROOT="$(cd "${0%/*}" && echo $PWD)"
2
3 # Scary!
4 rm -rf "$STEAMROOT/"*
5
```

Eve Online

Supression du fichier `boot.ini`...

Exemples cuisants

Ariane 5 - Vol 501

- Dépassement d'entier dans les centrales inertielles
- 370 millions de dollars de charge utile perdus
- Réutilisation des centrales d'Ariane 4 sans refaire les procédures de test

Thérac 25 - Irradiations mortelles

- Réutilisation du code de machines antérieures sans tests
- Codé en assembleur
- Pas d'organisme externe de vérification
- Au moins 5 morts par irradiation

Pourquoi ?

On adopte et respecte une démarche de qualité logicielle pour :

- Répondre aux besoins et aux contraintes de l'utilisateur
- Répondre aux critères de qualité définis par l'entreprise

Enjeux externes au projet

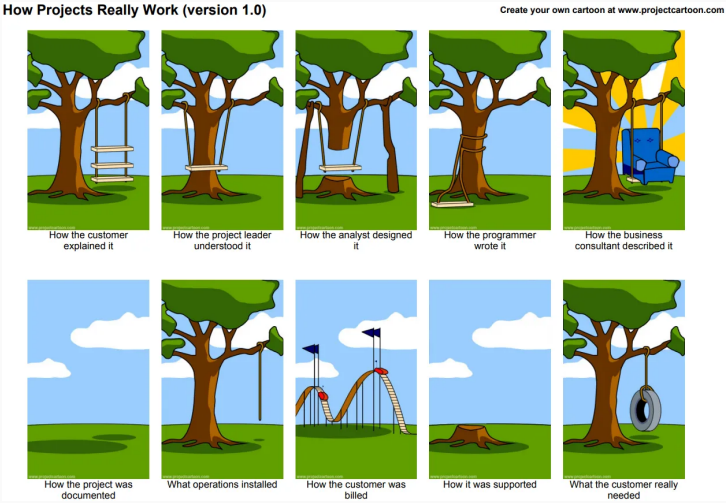
- Image de l'entreprise
- Coût de développement et de maintenance
- Réutilisabilité

Quoi ?

Principaux critères de qualité logicielle

- Fonctionnalité
- Fiabilité
- Sûreté
- Sécurité
- Conformité
- Compatibilité
- Portabilité
- Performance
- Maintenabilité
- Utilisabilité
- Testabilité

Critères : Fonctionnalité



Le logiciel

- Fait ce qui a été spécifié
- Ce dont l'utilisateur a besoin (idéalement)

"To replace programmers with AI, clients will need to accurately describe what they want. We're safe."
M. Milanovič

Critères : Fiabilité - Sécurité - Sûreté

Fiabilité - Le logiciel ne fait pas d'erreur et ne s'arrête pas :

- En utilisation normale
- En cas d'utilisation inappropriée

Sécurité - En cas d'attaque, le logiciel :

- Ne donne pas accès à des informations
- Ne fait pas d'erreur et ne s'arrête pas

Sûreté - En cas de défaut le logiciel :

- Ne cause pas de dommage
  - À l'utilisateur
  - À d'autres systèmes

Critères : Conformité - Compatibilité - Portabilité

Conformité - Le logiciel respecte :

- La réglementation
- Les normes

Compatibilité - Le logiciel peut :

- Échanger des données et
- s'intégrer avec les autres systèmes de l'application

Portabilité - Le logiciel peut :

- S'exécuter sur différents systèmes,
- Différentes machines

Critères : Performance - Maintenabilité - Utilisabilité

Performance - Le logiciel :

- Donne les résultats attendu dans le temps imparti
- Réagit dans les délais définis
- Supporte la charge de traitement déterminée

Maintenabilité - Un coût raisonnable est associé à :

- La correction de défauts
- L'ajout de fonctionnalités

Utilisabilité - Le logiciel peut être utilisé :

- De manière intuitive
- De manière efficace

Critères : Testabilité

Testabilité - Il est possible de tester le logiciel

- Sans le modifier
- Sans nécessairement avoir accès à son code

Quand et comment assurer la qualité de développement ?

Quand ?

Avant le développement :

- Analyse - Conception

Pendant le développement :

- Méthodes de développement (TDD, Agile, Cycle en V)
- Tests automatiques + Couverture de code
- Outils d'analyse du code, *code review*
- Structuration du code

Après le développement :

- Tests automatiques
- Déploiement et intégration continus

Comment assurer la qualité d'un logiciel ?

Dans ce cours

- Tests automatiques,
  - Unitaires, d'intégration
  - Métriques de couverture de code
- Documentation
- *Clean Code*
- Structuration du code, principes SOLID