IDEC école supérieure d'informatique

Brevets et diplômes fédéraux ICT

Les fondamentaux du testing

Sommaire

Chapitre 6

Introduction

Souvent négligés à cause de carences dans la planification et les spécifications, les tests sont l'une des principales garanties à la mise en exploitation d'une solution IT, conjointement avec la qualité des méthodes employées lors de la conception et de la réalisation.

Au sein d'un projet de piètre qualité, déjà entaché de spécifications floues et des dépassements de budget et de délai qui en découlent, la phase de test devient fréquemment une simple variable d'ajustement.

Or, terminer un projet dans le temps et le budget impartis en ayant sacrifié les tests n'est qu'une façon de différer le problème : les coûts et délais sont reportés sur la phase d'exploitation, et ce tout au long du cycle de vie de la solution livrée.

La problématique des tests peut être résumée dans ce triangle diabolique :

- dans la majorité des entreprises et autres types d'organisations, la disponibilité et l'efficacité du système d'information (SI) est vital : un ralentissement ou un arrêt peuvent avoir de lourdes conséquences. Il n'est pas rare que le coût de la défaillance d'un système soit supérieur au coût du système lui-même.

- toute modification opérée sur l'une des composantes du SI peut entraîner des dysfonctionnements ou des pertes de fonctionnalités.

- le SI doit néanmoins s'adapter constamment à son environnement afin de continuer à satisfaire les besoins de ses utilisateurs, eux-mêmes soumis au changement.

Le premier objectif des chargés de test est de concilier au mieux ces 3 règles, en s'assurant que les modifications apportées au SI par l'introduction de nouveaux éléments matériels ou logiciels n'auront pas de répercussions négatives.

Le second objectif des chargés de tests est de préserver l'investissement de l'organisation dans son SI en s'assurant en outre que les nouveaux éléments apportent réellement les avantages qui ont motivé leur introduction.

1.1 L'enjeu des tests

Si la notion de test est courante et semble clairement définie, il convient pourtant de s'assurer d'une définition commune et correcte. Avant même de s'intéresser au « pourquoi ? » ou au « comment ? », la première question à se poser est donc « qu'est-ce que tester ? »

Tester, ce n'est pas vérifier si cela fonctionne « bien »

Tester, c'est vérifier si l'objet est conforme à ses spécifications

Ce qui implique que lesdites spécifications existent, et c'est bien souvent le cœur du problème. Car pour que les spécifications – le cahier des charges – soient correctes, il faut que l'analyse des besoins ait été correctement effectuée. Méthodiquement, l'analyste doit :

1. délimiter le domaine d'analyse
2. recueillir l'information sur les besoins auprès des utilisateurs concernés du système
3. traduire ces besoins en réponses informatiques

C'est seulement à partir du moment où l'on sait très précisément ce qu'on attend du système et ce que l'on n'en attend pas qu'il devient possible de le tester, soit de vérifier sa conformité.

Pas de spécifications = pas de test valable

La plupart des tests consistent à faire des essais d'utilisation de quelque chose dont la réalisation est terminée, quelque chose de trop complexe ou de trop vaste pour qu'une erreur saute immédiatement aux yeux.
Le rôle du test est de chercher à trouver l'erreur. Car… Comment pourrait-il ne pas y en avoir ?

Les tests visent à découvrir les défauts avant qu'ils ne causent une défaillance. Un défaut a pour origine une erreur, qui peut appartenir à deux catégories principales :

- **erreur de conception** : l'objet à tester ne répond pas aux besoins, parce que les besoins ont été mal compris ou incomplètement exprimés,

- **erreur de réalisation** : en réalisant l'objet à partir d'un concept sans défaut, on introduit des erreurs à cause d'une mauvaise interprétation d'un élément du concept.

Analogie

*Vous aviez besoin d'une voiture pour transporter votre kayak et on vous livre un cabriolet à deux places : c'est une **erreur de conception**.*

*Si vous recevez un break doté d'une galerie mais que celle-ci n'est pas assez solidement fixée, c'est une **erreur de réalisation**.*

Par défaut, on entend toute divergence entre les spécifications de l'objet et son comportement réel. Un élément du réseau qui fonctionne sans panne et sans retard mais ne répond pas à l'un des besoins exprimés par ses utilisateurs souffre d'un défaut, de même qu'une application qui calcule faux. Un défaut est une non-conformité.

Tester, c'est chercher l'erreur

A noter qu'il est impossible de prouver l'absence d'erreur au sein d'un élément dont la complexité dépasse celle d'un bouton on/off : il faudrait essayer toutes les possibilités afin de s'assurer qu'aucune d'entre elles ne présente de défaut. Or un algorithme de quelques instructions – 5 ou 6 lignes de code – implique déjà plusieurs milliards de possibilités.

Il faut donc admettre que l'objet sera mis en service avec des défauts. Les tests servent à ramener le nombre de défauts à une proportion jugée acceptable et à les détecter au plus tôt.

On considère en effet que la durée de vie moyenne d'une application est de 7 ans, sachant que beaucoup seront exploitées plus de dix ans. Les défauts perdurent d'autant plus longtemps qu'on préfère généralement faire une mise à jour de l'application plutôt que la remplacer complètement. Pour un élément matériel (et sa partie logicielle), la durée de vie varie entre 3 ans et 5 ans mais de nombreux éléments sont conçus et utilisés pour des durées supérieures.

En conséquence, les coûts de maintenance représentent entre 40% et 80% du coût total.

Plus une erreur est détectée tardivement, plus elle est coûteuse à éliminer. Parfois même, il n'est plus envisageable de l'éliminer, et elle devra être contournée durant toute la phase d'exploitation.

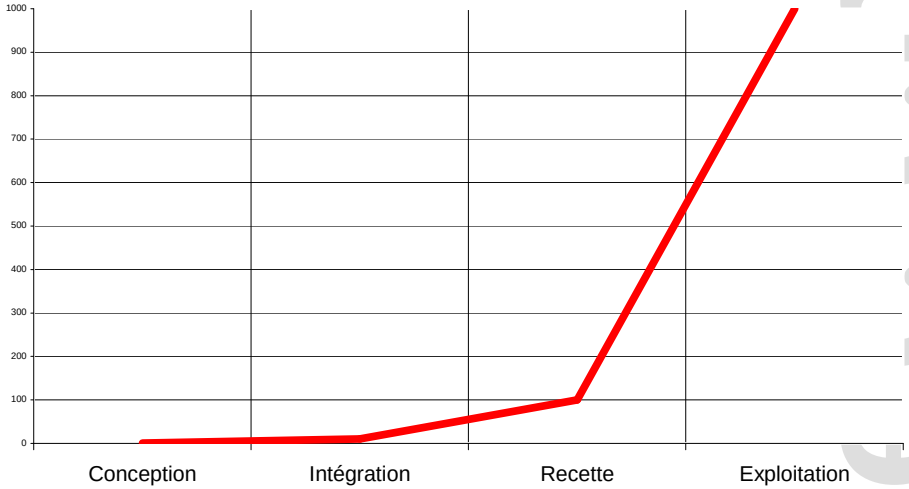

Figure 1 : le coût d'un défaut est proportionnel au retard dans sa détection

Quelques cas récents où les tests ont clairement été insuffisants :

- **2015**, 2 heures durant, impossible d'encaisser le moindre cent dans les 14'000 points de vente nord-américain de Starbucks. La liste des points de vente avait été supprimée lors de la mise à jour du logiciel de caisse : plus aucun d'entre eux ne pouvait donc se connecter au système.

- **2017**, c'est un bug qui a cloué au quai les pendulaires des CFF. Neufs bugs informatiques sont responsables d'un certain nombre d'interruptions totales de circulation durant 5 jours. Les logiciels avaient été testés en laboratoire mais la simulation n'a pas pris en compte une situation réelle de trafic.