Application Logic - Cheatsheet

Filpe Fortuanto, Pierre Kohler & Jonathan Zaehringer

November 25, 2019

Histoire

L'histoire regorge de problème de logique qui a causé parfois d'énormes dommages. L'un des plus connus et des plus martelés dans les formations informatiques est `Ariane 5`. Lors de son premier lancement, un problème de spécification récupéré de `Ariane 4` a été utilisé, ce qui s'apparente à de la réutilisation de logique. Par l'ajout d'autres problèmes ont permis à cette erreur d'être catastrophique et causé l'explosion de cette dernière. # Logique d'application

Définition

La logique d'application, ou *application logic* en anglais, est le **concept** à la base même de l'informatique. En effet, c'est ce concept qui permet l'implémentation d'une solution digitale à un problème humain.

En pratique

En pratique, dans le contexte du développement applicatif, il existe d'innombrables techniques préétablies de *découpage* d'application en fonction du besoin du domaine. Ce découpage sera influencé par une multitude de facteurs plus ou moins concrets, tel que les besoins directs de l'application.

Ainsi, une application de vente en ligne deviendra dans un premier temps l'association d'un panier d'achat, d'un catalogue, d'un système de livraison / paiement et d'un système d'authetification. Vous l'aurez compris, cela ne s'arrête pas là. Le catalogue lui-même ne sera en fait qu'un ensemble de **produits**, à l'instar du panier d'achat. Mais ces deux systèmes ne remplissent pas la même fonction, et nécessite donc une implémentation différente.

Rôle

Le rôle de la logique d'application est de permettre la décomposition d'un processus complexe en une série de tâches simples et atomiques.

Un processus complexe peut prendre des formes très variées. Il peut s'agir par exemple :

- … d'un algorithme mathématique. Dans ce cas, les "tâches simples" deviennent typiquement des opérations arithmétiques.
- … d'une solution applicative, comme une boutique en ligne ou autre plateforme proposant un espace client. Ici, on divisera d'en un premier temps le processus principal en sous-processus, et ainsi de suite jusqu'à arriver à une tâche suffisamment simple pour pouvoir être implémentée.
- … de tout autre protocole, même non informatique, pouvant faire l'objet d'une décomposition en activités élémentaires.

Risques

Si les tâches atomiques peuvent contenir des erreurs de logique en elles-mêmes, il ne s'agit pourtant pas des causes majeures de logique applicative en échec. En effet, une telle erreur est généralement plus facile à trouver à l'aide de tests unitaires bien conçus par exemple.

Les échecs de logique applicative trouvent leur origine dans l'étape de réunification des sous-processus. Chaque processus dispose d'un contexte propre, soit d'un espace individuel aussi bien qu'une fonction définie. L'application en elle-même ne contient pourtant généralement qu'un seul contexte, que devront plus ou moins tacitement se **partager** les différentes parties du code.

NB : le contexte d'une application est à prendre ici au sens large. Il peut s'agir aussi bien du contexte d'exécution à proprement parler que d'éléments de conception, tel que la modélisation d'une entité quelconque.

Bad code smell

Les `mauvaises odeurs` sont des *indicateurs* facilitant la **découverte** de problème de logic. Elles décrivent des *mauvaises habitudes*, des *erreurs de conception* ou des *erreurs de code* pouvant amener des problèmes qui seront **exploitable** par un attaquant. Les plus intéressantes dans ce contexte sont :

- **Duplicate code** : défini une duplication de code
- **Switch statements** : définis une complexité trop grande dans des conditions (switch case / if)
  - *Large class, Long methode, Long parameter list* : définis une trop grande complexité dans une classe, méthode ou paramètre
- **Refused Bequest** : définis une utilisation de l'objet pour sa capacité d'éviter la duplication de code sans prendre en compte l'évolution du programme

On peut ajouter dans la catégorie *Duplicate code*, la **réutilisation de logique** qui n'est pas une `mauvaises odeurs` en tant que telle. Mais le principe est proche et généralement problématique.

Analyse concolique

L'analyse concolique consiste à rechercher automatiquement des valeurs permettant de **parcourir tous les chemins** dans un code donné, afin de **trouver les effets de bord**. Cette méthode n'est **pas exhaustive**, et est relativement difficile à mettre en place pour des projets de grande envergure.