

The elevator problem

*A brief contextualized introduction to **finite state machines**
and some **data structures***

Xavier Van de Woestyne

Touraine Tech 2019

Hello !

- Xavier Van de Woestyne (Bruxelles, Lille, Paris) ;
- *Data Engineer* chez **margo.com**
- J'aime bien programmer (OCaml, F#, Haskell, Erlang/Elixir, Kotlin, Io, Elm) ;
- <https://xvw.github.io>, **xvw** sur Gitlab et Github ;
- **twitter.com/vdwxv** et **xvw@merveilles.town** ;
- **Phutur** : *useless software with useful languages* ;
- **LilleFP** : *Meetup* approximativement bimestriel ;
- j'ai une "technique" de préparation de présentation assez particulière :
 - ▶ j'essaie de découvrir quelque chose ;
 - ▶ j'essaie de faire la promotion de quelque chose ;
 - ▶ **je fantasme sur une "perle"** ;
 - ▶ **je rôle sur quelque chose.**


Objectifs de la présentation

- Raisonner la notion de **programme à états** (*stateful*) ;
- présenter un exemple **concret** de programme à états (les ascenseurs) ;
- proposer des perspectives d'implémentation **commodes** ;
- voir les **forces** et les **faiblesses** des machines à états par opposition aux "objets" classiques ;
- raisonner (un peu) **l'expérience utilisateur** ;
- utilisation de langages moins *mainstreams* (et statiquement typés) ;
- fantasmer l'implémentation "potentielle" d'ascenseurs... (en utilisant des langages inadaptés à l'implémentation réelle d'un ascenseur)

Hors périmètre

- Implémentation "réelle" de l'ascenseur ;
- focus sur l'implémentation logique.

Caveat emptor !

- L'objectif de la présentation est de tâcher d'être **pédagogique** ;
- c'est une présentation très **idéologique** (et subjective) ;
- le **raisonnement** sera donc privilégié à la présentation de code ;
- ce serait sûrement très bête d'implémenter un ascenseur de cette manière...
- 40 minutes... c'est très court  (+10 de questions réponses).

Une vraie histoire !

*Dans un espace de co-working parisien, il y a **un** ascenseur ...*

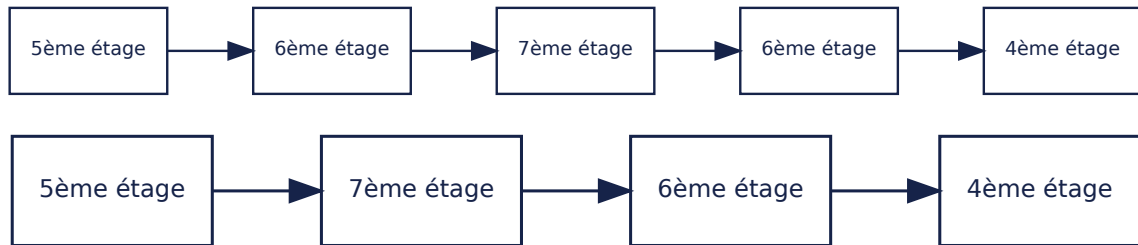
- Un ascenseur pour **8 étages** ;
- pouvant accueillir ~ 6 personnes ;
- avec **un seul** bouton d'appel ;
- tombant souvent en **panne**.

Mise en contexte : **un seul** bouton d'appel

Pouvant rendre le trajet **très frustrant**. Par exemple :

- si l'ascenseur est à l'étage 5, et qu'il se rend à l'étage 7 ;
- que j'appelle l'ascenseur à l'étage 6, pour descendre à l'étage 4 ;
- l'ascenseur s'arrêtera à l'étage 6, puis continuera vers l'étage 7 ;
- a cause d'un soucis, l'ascenseur s'arrêtera **toujours** à l'étage 6 ;
- pour ensuite descendre à l'étage 4 ...

Mise en contexte : **un seul** bouton d'appel



Optimisation du trajet

L'absence de deux boutons (monter/descendre) rend l'optimisation du trajet impossible.

Autre aléa (assez amusant)

Avant la maintenance **N** dûe à une **Nème** panne

- Si j'appuie **N** fois sur le bouton d'appel, en attendant l'ascenseur ;
- arrivé à mon étage, il ouvrira/fermera ses portes **N** fois...
- a l'époque je m'étais arrêté à 13.

Autre aléa (assez amusant)

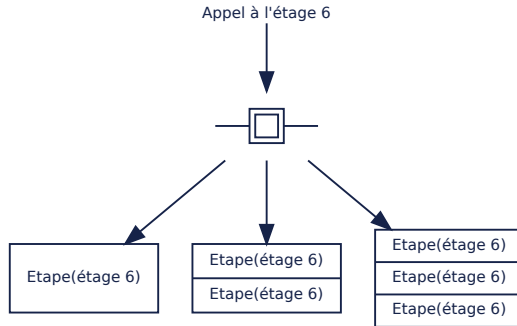


Figure 1: Stack ftw

Voici le "pourquoi" de cette présentation

A chaque fois que j'ai dû attendre cet ascenseur... j'y pensais.

Un problèmes sur plusieurs fronts

L'arbitrage

- Décider quelles stratégies adopter (attente longue ou trajet long ?) ;
- quand "rouvrir" la porte ?
- Intersection intelligente entre "la prise de décision automatique" et le "raisonnement simple". (Par exemple, comment se comporter vis à vis du poids maximum supporté par un ascenseur ?)

L'implémentation

- Modélisation plus complexe qu'il n'y paraît ;
- choix de structures de données adéquates ;
- **rendre les états impossibles... impossibles** ;
- **scalabilité** : comment augmenter le nombre d'ascenseurs ?

Raisonnement sur un programme "à états"

- Un programme qui "*se souvient*" des événements précédents ;
- et qui peut effectuer une transition vers un autre état en se basant sur son état courant.

Une collection d'ascenseurs est régie par une collection de règles et d'états

- Que se passe-t-il quand on appelle un ascenseur ?
- Peut-on ouvrir des portes qui sont déjà ouvertes ?
- Peut-on fermer des portes qui sont déjà fermées ?
- A quel étage se trouvent les ascenseurs ?

Modélisation naïve : les états implicites

- Ne pas être explicite sur les états ;
- qui deviennent définis par des environnements (des variables mutables) ;
- ce qui rend le programme **dur à raisonner**, notamment sur **l'intégrité** des transitions ;
- impose (trop) souvent des **assertions à l'exécution**.

On devrait rendre les états explicites !

- Définitions "formelles de états" ;
- rendre les transitions **explicites** ;
- rendre les états impossibles... impossible.

Machines à états finis

Support natif dans certains langages

Implémentation dans un langage avec un système de types riche

Structures algébriques et structures de données relatives à l'implémentation

Représentation de l'état du *cluster* avec une monade d'état

Heuristiques

Résumé

Conclusions