Aix Marseille Université - Campus Luminy

UFR des Sciences

Rapport de Projet

Master Informatique

Module Génie Logiciel

Projet Ascenseur :

Spécification du système.

Réalisé par :

ZEMMOURI Yasmine.

BEKHEDDA Hadjira.

2023-2024.

Introduction :

L'ascenseur, symbole de modernité et d'efficacité, est devenu un élément essentiel de notre quotidien. Que ce soit dans les gratte-ciel imposants ou les modestes immeubles d'appartements, les ascenseurs offrent un moyen pratique et rapide de se déplacer entre les différents niveaux d'un bâtiment. Cependant, derrière cette apparente simplicité se cache une ingénierie complexe et un logiciel sophistiqué qui régissent le mouvement fluide et sécurisé de la cabine.

Au cœur de ce projet se trouve la gestion des interactions entre les utilisateurs et l'ascenseur, ainsi que la coordination des mouvements de la cabine pour satisfaire ces demandes de manière optimale.

Ce document de spécification servira de guide pour le développement du système de contrôle-commande de l'ascenseur. Il détaillera les spécifications fonctionnelles du système, les interactions entre les utilisateurs et l'ascenseur, les protocoles de fonctionnement et même l'interface utilisateur graphique de test.

Dans les sections suivantes, nous présenterons en détail la manière dont les demandes des utilisateurs seront gérées, le protocole de fonctionnement de l'ascenseur, et bien d'autres aspects essentiels de ce système.

Ce projet représente une opportunité de mettre en pratique les principes du génie logiciel dans un contexte réel, en contribuant au développement d'un système qui améliore notre vie quotidienne en rendant les déplacements verticaux plus efficaces et sécurisés.

Objectif :

Notre objectif est de développer un système de contrôle-commande d'ascenseur capable de répondre aux besoins des utilisateurs, et d’assurer un fonctionnement sûr et efficace de l'ascenseur en minimisant les déplacements inutiles de la cabine.

Le système doit gérer les commandes provenant à la fois des utilisateurs à l'intérieur de la cabine et des utilisateurs à l'extérieur à chaque niveau. Il doit également inclure une interface utilisateur graphique de test pour simuler le fonctionnement de l'ascenseur.

Fonctionnalités principales du système :

- Le système doit pouvoir gérer les commandes "monter", "descendre", "arrêter au prochain niveau" et "arrêter d'urgence" de la partie opérative.
- Le système doit prendre en compte les demandes des utilisateurs à l'intérieur de la cabine et à l'extérieur à chaque niveau.
- Le système doit minimiser les déplacements inutiles de la cabine en optimisant la séquence des requêtes.

Fonctionnement de l’ordonnanceur :

Le fonctionnement d'un ascenseur implique un ordonnancement des requêtes pour déterminer à quelle demande la cabine d’ascenseur va répondre.

Nous avons mis en place une fonction prenant en compte la position actuelle de l’ascenseur ainsi que la direction dans laquelle il se déplace, à savoir UP (en montée) ou DOWN (en descente). Elle renvoie l’étage suivant où l'ascenseur doit s'arrêter.

Nous avons trois types de requetes :

i. Les requetes issues de la cabine de l'ascenseur : un utilisateur rentre dans la cabine de l'ascenseur et appuie sur l'un des boutons mis à disposition pour se rendre à l'étage voulu, et ceci peu importe la direction de l'ascenseur.
ii. Les requetes issues des palliers : On y trouve deux types de requetes :
    1. Les requetes pour monter (UP).
    2. Les requetes pour descendre (DOWN).

L'ordonnanceur reçoit ces requetes et suit un algorithme pour spécifier le prochain étage à atteindre.

L'algorithme proposé est le suivant :

- Si des demandes en cours sont détectées à l'étage actuel dans la direction actuelle, l'ascenseur devra s'arreter immédiatement à cet étage pour satisfaire la demande. L'ordonnanceur renvoie l'étage actuel.
- Si aucune demande n'est trouvée à l'étage actuel dans la direction actuelle et qu'un changement de direction est nécessaire, l'ordonnanceur bascule vers la direction opposée. Par exemple :
    - S'il montait (Direction.UP) et atteint le dernier étage, il change de direction pour descendre (Direction.DOWN).
    - S'il descendait (Direction.DOWN) et atteint le rez-de-chaussée, il change de direction pour monter (Direction.UP).
- Une fois la direction changée, l'ordonnanceur recherche une nouvelle demande dans cette nouvelle direction. Il parcourt les étages dans cette direction jusqu'à ce qu'une demande soit trouvée ou qu'il atteigne une extrémité (par exemple, le dernier étage en montant ou le rez-de-chaussée en descendant).
- Si aucune demande n'est formulée, l'ordonnanceur ne fait rien (-1 est retourné).

Prenons les exemples suivants :

Exemple 1 : Si l'ascenseur est à l'étage 3 en montant (`Direction.UP`) et qu'il y a une demande à l'étage 5, l'ordonnanceur devrait renvoyer 5, indiquant que l'ascenseur doit s'arrêter à l'étage 5.

Exemple 2 : Si l'ascenseur est à l'étage 5 en montant (`Direction.UP`) et qu'aucune demande n'est en cours, l'ordonnanceur devrait détecter qu'un changement de direction est nécessaire, passer en mode de descente (`Direction.DOWN`) et rechercher une demande dans cette direction.

Pour procéder à cet algorithme, plusieurs méthodes ont été implémentées :

1- **Méthodes «addGoRequest », « addUpRequest », « addDownRequest » :** permettent d'ajouter une demande d'un type particulier pour un étage donné à la liste correspondante.
2- **Méthodes « deleteDown », « deleteUP », « deleteGO » :** afin de supprimer toutes les demandes d'un type particulier (Monte, descente) pour un étage donné d'une liste correspondante.
3- **Méthode « deleteRequests »** : utilise les méthodes `deleteUP`, `deleteDOWN` et `deleteGO` pour supprimer toutes les demandes pour un étage donné.
4- **Méthode « next » :** détermine l'étage suivant où l'ascenseur doit se déplacer en fonction des demandes et de la direction actuelle pour déterminer si l'ascenseur doit continuer à monter ou à descendre, ou changer la direction si nécessaire.
    - Si la direction est "UP" l'ascenseur atteint le dernier étage. Dans ce cas l'ascenseur doit changer de direction. Sinon il continue à monter.

- Si la direction est "UP" l'ascenseur atteint le rez-de-chaussée (étage 0). Dans ce cas l'ascenseur doit changer de direction. Sinon il continue à descendre.
- Si l'ascenseur atteint le rez-de-chaussée ou le dernier étage sans trouver de demande, la méthode retourne `NO_FLOOR` pour indiquer qu'il n'y a pas d'étage à desservir dans la direction opposée.

5- **Méthode « isRequested » :** vérifie l'existence d'une demande à un étage donné dans une direction donnée. Elle est utilisée pour déterminer si l'ascenseur doit s'arrêter à un étage spécifique.
6- **Méthode « getOppositeDirection » :** retourne la direction opposée, utilisée pour changer la direction de l'ascenseur lorsque nécessaire.

L'ordonnanceur est conçu pour minimiser les temps d'attente des passagers lors des déplacements de l'ascenseur. Il n'est néamoins pas responsable des actions de l'ascenseur. C'est le role du contrôle commande.

Fonctionnement du contrôle commande :

Le contrôle-commande d'un ascenseur représente le système centralisé qui supervise et gère tous les aspects de son fonctionnement. Il réagit aux commandes des utilisateurs, surveille en temps réel la position de la cabine et les conditions de sécurité grâce à des capteurs, et prend des décisions pour contrôler la direction de l'ascenseur. Son rôle essentiel est de garantir un déplacement efficace entre les différents étages d'un bâtiment, en s'assurant que les demandes des passagers sont traitées de manière optimale via l'ordonnanceur.

Nous avons mis en place un algorithme visant à contrôler le fonctionnement de ce contrôle commande, qui gère le mouvement de l'ascenseur :

1- ***Vérification des demandes :*** Cette méthode est utilisée pour vérifier et traiter les demandes de déplacement de l'ascenseur. Elle prend en entrée le nombre d'étages `nbFloors` et un objet `panel` de type `PanelSimulator` qui représente le panneau de l'ascenseur, parcourt tous les étages [0 à nbFloors] et vérifie si le bouton d'étage a été pressé afin d'ajouter une demande de déplacement vers cet étage. Pour les buttons « DOWN », elle ajoute une demande vers le bas et les buttons de «UP » vers le haut et activant la lumière correspondante.
2- ***Descendre l'ascenseur :*** Cette méthode est appelée lorsque l'ascenseur doit se déplacer vers le bas. Elle prend en entrée l'étage de destination `suivant`.
3- ***Monter l'ascenseur :*** Cette méthode est similaire à la méthode précédente.
4- ***Arrivée de l'ascenseur :*** Lorsque l'ascenseur est arrivé à un l'étage de destination, la méthode vas arrêter l'ascenseur. Et éteint les lumières des boutons sur le panneau de contrôle.
5- ***Méthode CheckAndProcess :*** L'objectif de cette méthode est de gérer le déplacement de l'ascenseur, les boutons d'arrêt, d'initialisation et vérifier le traitement des demandes liées à l'ascenseur. Puis assurer que l'ascenseur répond correctement aux demandes de l'utilisateur tout en tenant compte l'étage actuel et de sa direction de déplacement en exécutant les étapes suivantes :
    - ***a.*** Déterminer le prochain étage vers lequel l'ascenseur doit se déplacer.
    - ***b.*** Si l'étage de destination est supérieur à l'étage actuel : appeler la méthode Monter l'ascenseur.
    - ***c.*** Si l'étage de destination est inférieur à l'étage actuel : cela signifie que l'ascenseur doit descendre.
    - ***d.*** Appelle la méthode Descendre l'ascenseur.

e. Si l'étage suivant est l'étage actuel : appeler la méthode Arriver de l'ascenseur pour gérer l'arrivée à cet étage.

f. Si le bouton Init est pressé : réinitialise la simulation de l'ascenseur en appelant Reset et pour indiquer que l'ascenseur doit redémarrer depuis le rez-de-chaussée.

g. Si le bouton Stop est pressé : arrête immédiatement la simulation de l'ascenseur en appelant Stop Simulator.

h. Si aucune des conditions précédentes n'est satisfaite, la méthode détermine à nouveau le prochain étage.

Par exemple, lorsque l'utilisateur presse le bouton "Monter" depuis le cinquième étage, le Contrôle-Commande active la direction "vers le haut" et commence à surveiller la position de l'ascenseur. Il utilise la méthode goUP() pour faire monter l'ascenseur jusqu'à ce qu'il atteigne le cinquième étage, puis appelle la méthode arrived() pour ouvrir les portes. Si, pendant le trajet, l'utilisateur appuie sur le bouton d'arrêt d'urgence, le Contrôle-Commande réagit en arrêtant immédiatement l'ascenseur, en supprimant toutes les demandes en cours et en éteignant les lumières des boutons. Ensuite, il peut redémarrer l'ascenseur lorsque l'urgence est résolue. La méthode checkAndProcess() assure la gestion continue des demandes et des déplacements, en garantissant que l'ascenseur réponde efficacement aux besoins des utilisateurs tout en maintenant la sécurité.

Examinons quelques exemples :

- ✓ *Mouvement vers le bas :* Lorsque le Contrôle-Commande détecte une demande de descente depuis un étage supérieur, il active la direction "vers le bas" et commence à faire descendre l'ascenseur. Il surveille en permanence la position actuelle de l'ascenseur. Par exemple, s'il détecte que l'ascenseur a atteint l'étage 3, il arrête l'ascenseur au prochain étage disponible, supprime les demandes à cet étage et éteint les lumières des boutons correspondants.
- ✓ *Mouvement vers le haut :* De manière similaire, lorsque le Contrôle-Commande reçoit une demande de montée depuis un étage inférieur, il active la direction "vers le haut", démarre l'ascenseur et suit son mouvement. Lorsque l'ascenseur atteint l'étage demandé, il l'arrête, supprime les demandes associées à cet étage et éteint les lumières des boutons correspondants.
- ✓ *Arrêt d'urgence :* Si un utilisateur appuie sur le bouton d'arrêt d'urgence pendant un trajet, le Contrôle-Commande réagit instantanément en arrêtant l'ascenseur, quelle que soit sa position. Il supprime toutes les demandes en cours, éteint toutes les lumières de bouton et attend que la situation d'urgence soit résolue. Une fois l'arrêt d'urgence annulé, il peut redémarrer l'ascenseur en continuant à traiter les demandes restantes.
- ✓ *Boucle de contrôle principale :* La méthode checkAndProcess() représente la boucle principale du Contrôle-Commande. Elle surveille en permanence les demandes des utilisateurs et réagit en conséquence. Par exemple, si un utilisateur demande d'arrêter l'ascenseur, il s'arrête immédiatement. Si l'utilisateur demande de monter, le Contrôle-Commande fait monter l'ascenseur, et lorsqu'il atteint l'étage demandé, il gère l'ouverture des portes. Cette boucle garantit que l'ascenseur fonctionne de manière fluide, répondant aux besoins des utilisateurs tout en maintenant la sécurité.

En résumé, le Contrôle-Commande joue un rôle central en supervisant le mouvement de l'ascenseur, en répondant aux commandes des utilisateurs. Il assure la coordination efficace des déplacements de l'ascenseur et la gestion des situations d'urgence.