

Méthodologie d'intégration

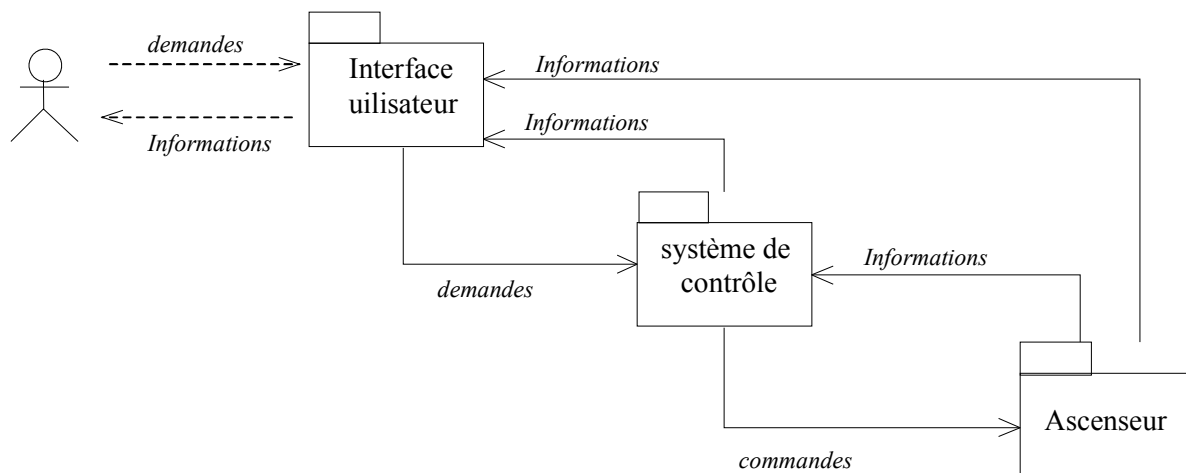
Étude de cas : Système de contrôle d'un ascenseur

1. Objectif de l'étude de cas

Cette étude de cas a pour objectif l'utilisation du formalisme UML pour réaliser un système informatique. Le déroulement de cette réalisation sera basé sur le processus présenté dans le cours « Architecture Fonctionnelle ». Cette étude de cas décrit l'état du problème, point de départ de ce processus.

2. Présentation de l'étude de cas.

Nous allons nous intéresser au système de contrôle d'un ascenseur. Le but de ce système est d'optimiser les déplacements d'un ascenseur, afin de limiter l'attente des usagers. Ce système est un logiciel qui reçoit en entrée des demandes provenant des utilisateurs et des informations provenant de l'ascenseur ; il émet en sortie des commandes pour le déplacement de l'ascenseur. On peut représenter¹ l'environnement de ce système en considérant deux autres systèmes : l'interface utilisateur qui permet de recueillir les demandes de l'utilisateur, l'ascenseur lui-même.



Sur ce schéma, nous voyons que l'interface utilisateur est à même de donner des informations sur le système de contrôle et sur l'ascenseur. Par exemple : informer l'utilisateur que sa demande a été traitée ou donner la position de l'ascenseur.

Vous voyez que le système est relativement simple ; en particulier, il ne gère pas la commande des dispositifs physiques annexe de l'ascenseur : ouverture des portes, système de sécurité, etc. Ces dispositifs sont très bien gérés par les

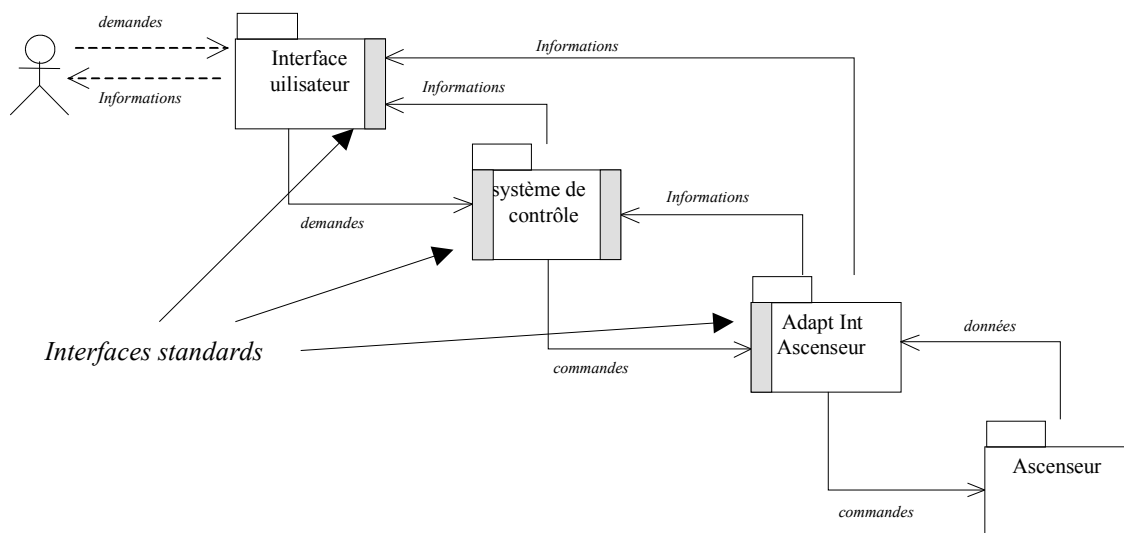
¹Ce schéma n'est pas de l'UML, car il ne montre pas des dépendances, mais les échanges de données entre sous systèmes. Nous avons représenté les sous systèmes par le symbole des paquets, cette représentation est purement conventionnelle.

systèmes classiques intégrés dans les ascenseurs. On ne veut en fait, réaliser qu'un optimiseur de déplacements intégrable sur tous les modèles de la gamme d'ascenseurs.

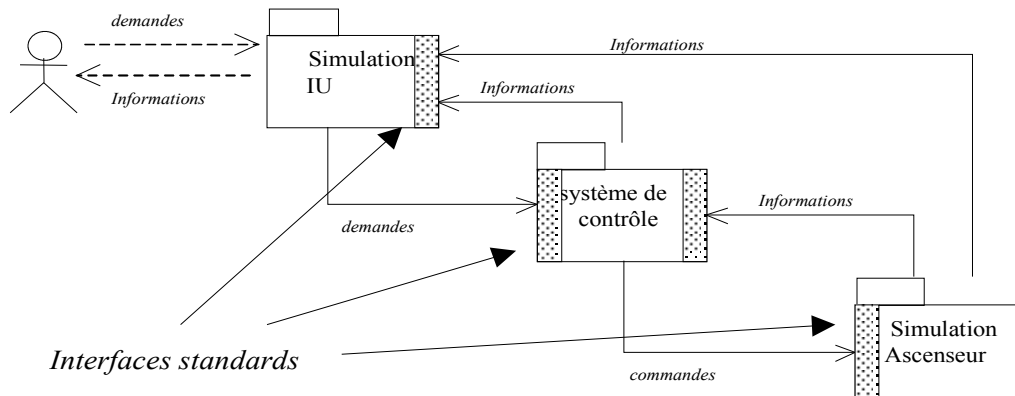
Le fait que cet optimiseur soit réalisé sous forme de logiciel est particulièrement intéressant, car on pourra le faire évoluer et l'adapter. Le faire évoluer pour suivre l'évolution de la gamme d'ascenseurs tant en performance qu'en fonctionnalité. L'adapter aux segments de marché que l'on veut atteindre (installation mono ascenseur, installation multi ascenseur, etc.), aux configurations spécifiques (très grands immeubles, accès privilégié, etc.).

Naturellement, vous n'avez pas d'installation pour tester votre système, mais le constructeur ne souhaite pas non plus tester son système sur des installations réelles. Il vous faudra donc réaliser un simulateur, pour pouvoir valider votre système. Ce simulateur sera aussi un outil d'étude pour mettre au point votre algorithme d'optimisation ou pour adapter votre système à de nouvelles configurations.

La définition des interfaces entre les différents sous systèmes sera un des point-clés de l'étude, il sera l'objet de la partie architecture fonctionnelle de cette étude. Pour que le système de contrôle puisse être intégré facilement au système réel, après son développement dans l'environnement de simulation, il faut que ces interfaces soient les mêmes dans les deux environnements. Dans le système réel, pour utiliser ces interfaces, il faudra développer des adaptateurs d'interfaces avec les systèmes existants, nous avons figuré ce principe dans le schéma qui suit. Remarquons au passage, que l'interface utilisateur est en faite un adaptateur d'interfaces.



Dans l'environnement de simulation, le système de contrôle est rigoureusement identique, pour cela les interfaces sont les mêmes, par contre les autres sous systèmes sont des simulations qui doivent rigoureusement implémenter les interfaces standards. Nous avons résumé ce principe sur le schéma suivant :



Nous allons maintenant décrire l'état du problème qui vous donnera l'ensemble des informations que vous devrez modéliser dans votre analyse. Ces informations sont de deux types : celles concernant la stratégie d'optimisation du système de contrôle, celles concernant la définition de l'environnement de simulation, c'est à dire définition de l'interface utilisateur et de la simulation de l'ascenseur.

3. Etat du Problème

Pour décrire l'état du problème, nous allons présenter l'ensemble du système à partir du point de vue de l'utilisateur, jusqu'au fonctionnement physique de l'ascenseur. On verra de cette manière la transformation de la formation et la logique de chaque sous système qui traite un type homogène d'informations.

3.1. Les demandes utilisateurs

L'utilisateur peut faire deux types de demandes : l'appel, le déplacement.

L'appel correspond à la demande d'utilisation d'un usager situé à l'extérieur de l'ascenseur. Il y a deux types d'appels : la descente ou la montée. Pour effectuer un appel, il existe à chaque niveau² deux boutons : un pour descendre, un autre pour monter. Quand l'appel est en cours de traitement, le bouton correspondant à l'appel est éclairé, quand l'appel est traité, il s'éteint. L'appel commence à la demande d'appel et se finit dès que l'ascenseur a servi l'appel. On considère qu'un appel a été servi : si après l'appel, l'ascenseur s'est arrêté au niveau correspondant à l'appel, en se déplaçant dans le sens précisé par l'appel.

A chaque niveau, on autorise un appel pour monter et un appel pour descendre, toutes les nouvelles demandes seront donc ignorées tant que les appels existants sont en cours de traitement. Dans la mesure où un appel peut être accepté, il peut être demandé à n'importe quel moment.

Le déplacement correspond à la demande d'un utilisateur à l'intérieur de l'ascenseur pour se rendre à un autre niveau. Pour effectuer une demande de déplacement, on utilise dans l'ascenseur un tableau de bouton, chaque bouton correspond à un niveau. Quand la demande est en cours de traitement, le bouton correspondant au niveau demandé est

² Un niveau correspond au lieu où l'on peut monter ou descendre de l'ascenseur. Nous avons préféré le vocable de niveau, plutôt que celui d'étage qui est un concept d'architecture. En particulier, certains ascenseurs s'arrêtent à des niveaux qui desservent plusieurs étages.

éclairé, quand la demande est traitée, il s'éteint. Le déplacement commence à la demande et se finit dès que l'ascenseur s'est arrêté au niveau demandé. Il ne peut y avoir qu'une demande de déplacement pour un niveau. Dans la mesure où une demande de déplacement peut être acceptée, elle peut être faite à n'importe quel moment.

L'utilisateur est informé de la position de l'ascenseur par un tableau d'informations situé dans la cabine. Ce tableau est constitué par une série de chiffres correspondant à chaque niveau. Quand l'ascenseur arrive à un niveau, le chiffre correspondant au niveau s'éclaire, il ne s'éteint que quand l'ascenseur atteint le niveau suivant.

3.2. L'optimisation des déplacements

Pour optimiser les déplacements, on applique les principes suivants :

- Quand l'ascenseur se déplace dans un sens, il prend en compte toutes les demandes qui ne nécessitent pas un changement de sens et il stocke toutes les autres.
- Quand il n'a plus de demandes à prendre en compte, il change de sens, s'il existe des demandes stockées ; Il s'arrête, s'il n'existe pas de demandes stockées.
- Les demandes qui ont été stockées ne pourront pas toutes être servies quand l'ascenseur se déplacera en sens inverse. En effet, les demandes d'appels imposent un sens qui peut ne pas être celui du déplacement courant de l'ascenseur. Ces demandes ne seront servies qu'au prochain déplacement dans l'autre sens.

Donnons un exemple :

Supposons que l'ascenseur monte au 7ème étage.

Quand il passe au 4ème étage, un utilisateur du 3ème étage demande un appel pour monter.

Son appel ne sera traité qu'à la prochaine remontée.

3.3. L'interface ascenseur

L'interface ascenseur permet d'obtenir des informations sur l'ascenseur et de lui envoyer des commandes. Ces informations et ces commandes ont été conçues afin d'implémenter facilement le système de contrôle et l'interface utilisateur ; c'est donc une interface de haut niveau.

3.3.1. Les commandes de l'ascenseur

L'ascenseur accepte deux commandes : marche, arrêt_niveau.

Marche : cette commande admet un paramètre le sens du déplacement, elle met en mouvement la cabine dans le sens indiqué.

Arrêt_niveau : cette instruction demande l'arrêt de l'ascenseur au prochain niveau.

3.3.2. Informations sur l'ascenseur

Les informations caractérisant l'ascenseur sont les suivantes : son état, sa position, son sens de déplacement.

Etat : l'ascenseur peut être dans trois états qui vont définir son comportement.

Repos : l'ascenseur est au repos et il peut recevoir une commande de déplacement.

Déplacement : l'ascenseur se déplace et il peut recevoir une commande d'arrêt.

Transitoire : l'ascenseur s'arrête, mais il n'est pas encore au repos ; dans cet état il ne peut recevoir aucune commande.

Position : cette information donne la position de l'ascenseur par rapports aux différents niveaux, sa valeur est un chiffre :

Valeur entière : l'ascenseur est exactement au niveau précisé dans la valeur.

Autre valeur : l'ascenseur est entre deux niveaux, correspondant aux deux entiers encadrant la valeur. Du point de vue pratique, il serait conseillé que cette valeur intermédiaire soit toujours la même, par exemple la valeur médiane (1.5 ; 2.5 ; etc.), pour être sûr que la donnée ne véhicule pas d'autre information.

Sens : cette information donne le sens du déplacement (montée, descente). Elle n'a de signification que dans l'état déplacement, on aurait donc pu la considérer comme un sous état. En fait, elle ne modifie pas le comportement de l'ascenseur, elle n'est donc pas représentative d'un état. Contrairement aux autres informations, elle ne donne pas réellement une information sur l'ascenseur, mais sur l'histoire de l'ascenseur. Montée signifie que la dernière instruction qu'a reçue l'ascenseur était instruction de montée, de même descente signifie que la dernière instruction qu'a reçue l'ascenseur était une instruction de descente.

Les informations concernant l'ascenseur peuvent être obtenues de deux manières : par une requête issue de l'extérieur, par un événement émis par l'ascenseur vers l'extérieur. Les requêtes sont utilisées pour obtenir les informations stables, par exemple le sens. Les événements sont utilisés pour signaler des changements d'information, par exemple l'ascenseur est passé dans l'état « au repos ». Il vous appartiendra dans l'architecture fonctionnelle à travers la définition des interfaces de préciser le mode d'obtention des informations.

3.4. L'ascenseur et ses dispositifs physiques

Le simulateur de l'ascenseur doit pouvoir implémenter l'interface ascenseur de manière réaliste, c'est à dire modifier les informations qui caractérisent l'ascenseur en fonction des commandes qu'il reçoit, cette modification pouvant entraîner l'émission d'événements vers l'extérieur.

Le plus simple pour réaliser une simulation est de se baser sur la structure physique du système que l'on veut simuler, en simulant les dispositifs élémentaires du système. De cette manière, on a une architecture d'intégration facile à comprendre et facile à réaliser.

Dans la mesure où notre système de contrôle ne s'intéresse qu'au déplacement, seul seront simulés les dispositifs physiques liés au déplacement. Nous considérons que quatre dispositifs physiques : un moteur, une cabine, des capteurs et une cage d'ascenseur.

La base de la simulation est le calcul de la position de la cabine pour détecter si le faisceau des capteurs a été coupé et rétabli. Cette information sur les capteurs nous permet de mettre à jour l'information position de l'interface ascenseur qui donne la position de l'ascenseur par rapport au niveau.

3.4.1. Le moteur

Le moteur met la cabine en mouvement, les moyens de transmission du mouvement ne nous intéressent pas. Ce moteur comporte deux commandes : marche, arrêt.

Marche : Cette commande comporte un paramètre le sens, elle met en mouvement la cabine. La cabine n'a pas d'inertie et la vitesse du moteur est constante. Ce qui signifie que la cabine passe de la vitesse 0 à la vitesse v instantanément, ce qui n'a pas de sens au point de vue physique, mais correspond à une approximation satisfaisante pour notre simulation. Vous remarquez que cette commande n'a pas de paramètre vitesse, la vitesse est une propriété du moteur. Il serait bon de pouvoir modifier cette vitesse pour valider votre système de contrôle dans différentes conditions de fonctionnement.

Arrêt : l'arrêt correspond à l'arrêt du moteur, quel que soit son sens de fonctionnement. La cabine n'ayant pas d'inertie, l'arrêt est immédiat.

Ce moteur comporte deux états : marche, attente.

Attente : quand le moteur est en attente, il peut accepter toutes les commandes.

Marche : quand le moteur est en marche, il ne peut exécuter que la commande « arrêt ».

3.4.2. La cabine

La cabine peut recevoir trois commandes : mise_en_mvt, fin_mvt, ouverture :

Mise_en_mvt : cette commande accepte en paramètre la vitesse relative, elle permet de déplacer la cabine.

Fin_mvt : cette commande stoppe la mise en mouvement de la cabine.

Ouverture : cette commande ouvre la cabine, on considère qu'elle est refermée après un laps de temps fixe.

La cabine est caractérisée par deux informations : son état, sa position.

Etat : la cabine peut être dans trois états : ouverte, fermée, en mouvement.

Ouverte : la cabine est ouverte pour laisser entrer ou descendre des usagers. Il est bien évident que dans cet état, elle ne peut être mise en mouvement.

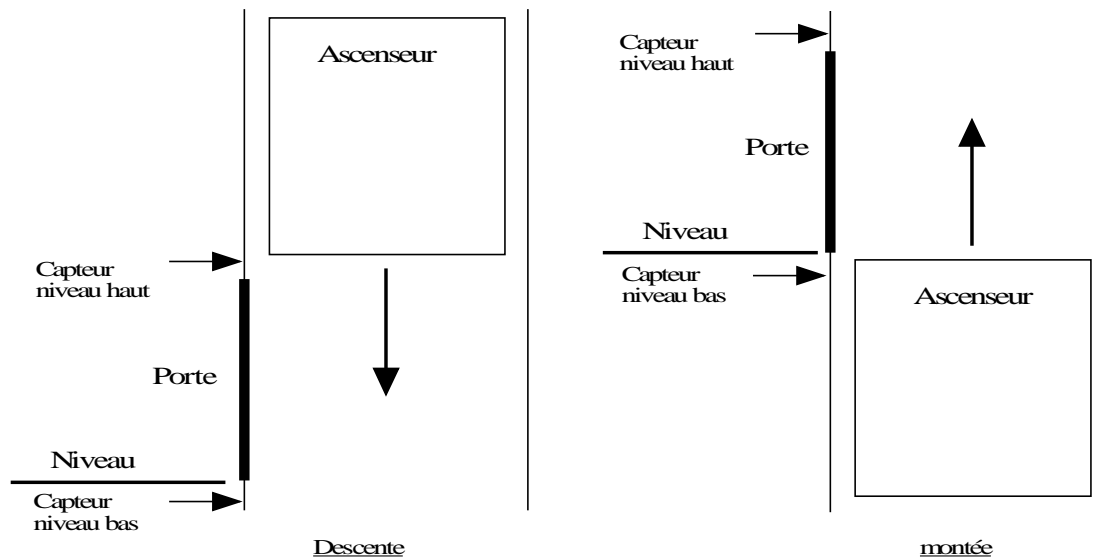
Fermée : la cabine est fermée avec ou sans usager à l'intérieur. Dans cet état, elle peut être mise en mouvement.

En mouvement : la cabine est en mouvement. Dans cet état, elle peut être arrêtée.

Position : cette information donne la position de la cabine dans un repère gradué en niveaux, dont l'origine est le niveau 0. La précision de la position dépendra de la simulation, mais elle devra au moins donner des valeurs permettant de simuler la détection des capteurs.

3.4.3. Les capteurs

Nous n'avons de perception de la position de la cabine que grâce à des capteurs situés dans la cage d'ascenseur. A chaque niveau, il existe deux capteurs qui encadrent le niveau ; ils permettent d'arrêter la cabine en face du niveau. Le capteur haut sera utilisé pour l'arrêt en montant, le capteur bas sera utilisé pour l'arrêt en descendant. Avec cette solution utilisant deux capteurs, le positionnement est détecté toujours par le même changement d'état du capteur. On aurait pu utiliser un seul capteur par niveau, en distinguant pour chacun des sens, le type de changement d'état. La solution double capteurs a été retenue, car elle permet d'implémenter des solutions redondantes, ce que nous ne ferons pas dans cette étude de cas, mais qui devrait être fait dans la réalité pour des raisons de sécurité. Nous avons illustré ce dispositif dans le schéma qui suit.



Les capteurs sont caractérisés par deux états : détection, non_détection.

Détection : la détection correspond au cas où la cabine est en face du capteur.

Non_détection : la non_détection correspond au cas où la cabine n'est pas en face du capteur.

4. Conseil pour l'analyse

L'idée de base pour concevoir le système de contrôle est de gérer des arrêts et non des déplacements ; la mise en mouvement n'étant qu'une action automatique après chaque arrêt qui dépend de la stratégie générale du déplacement. Toute demande de l'utilisateur (appel ou déplacement) va donc générer un arrêt. Par exemple, une demande de déplacement au 3^{ème} par un utilisateur monté au 4^{ème} sera considéré comme une demande d'arrêt au 3^{ème}. Si l'ascenseur est entrain de monter, il stockera cette demande afin de l'honorer lors de la prochaine descente. Attention toute demande ne génère pas un arrêt, car il peut exister une demande non encore servie ayant généré le même arrêt. Pour reprendre notre exemple, si un utilisateur a déjà fait une demande d'appel au 3^{ème} pour la descente, la demande de déplacement précédente ne générera pas d'arrêt, car il existe déjà un arrêt satisfaisant la demande.

Le système de contrôle aura deux buts : classer les arrêts, générer à partir de ces arrêts les instructions pour l'ascenseur.

Le classement des arrêts consistera à déterminer les arrêts qui peuvent être servis sans que l'ascenseur change de sens ; ceux qui ne peuvent pas l'être seront mémorisés pour être traités plus tard. En sachant que plus tard ne signifie pas : quand l'ascenseur aura changer de sens, comme l'explique la section sur l'optimiseur dans le chapitre Etat du problème.

Ces demandes d'arrêt seront utilisées pour générer les instructions pour l'ascenseur. En fonction de la position de l'ascenseur et du niveau de l'arrêt, on générera une instruction d'arrêt au prochain niveau quand l'ascenseur sera en mouvement, et une instruction de déplacement quand l'ascenseur sera au repos.

Les demandes seront traitées indéfiniment, le système peut être au repos, mais ne s'arrête jamais. Il est au repos quand il n'y a pas d'arrêt à traiter, à la première demande qui génère un arrêt, il se réveille pour traiter l'arrêt. Il est bien évident que pour un système réel, cette solution n'est pas acceptable. Mais notre but ici est de traiter des problèmes d'optimisation et non ceux d'exploitation de l'ascenseur ; on considérera donc que cette solution est acceptable dans le cadre de cette étude.