|  |  |
| --- | --- |
| BUREAU D’ETUDE – SIMULATION DE CURE THERMALE  UV - 5.8 - Ingénierie Systèmes | RÉSUMÉ  Ce document présente la simulation évènementielle de la cure thermale de M Jabbalehut sur Tatooine. Elle présente une étude de son fonctionnement réalisée par les spécialistes de la société DarkForce.    Fabrice LALLEMENT - Rémi RIGAL - Noëlie RAMUZAT |

# Objectif de l’etude

Notre étude a pour but de déterminer l’origine des insatisfactions des clients de la cure thermale de M Jabbalehut et de proposer des solutions pour améliorer leur contentement. Les critiques des curistes se portent sur le temps d’attente avant leurs ateliers ainsi que sur la mauvaise planification de leurs créneaux, les amenant parfois à ne pas pouvoir les réaliser. M Jabbalehut est prêt à diminuer la fréquentation maximale par jour au sein de sa cure mais préfèrerait d’autres alternatives.

Afin de réaliser l’étude demandée, notre société DarkForce décide de simuler de façon évènementielle la cure thermale. Le logiciel créé se doit d’être flexible afin de pouvoir couvrir différents types de scénario et tester plusieurs configurations possibles pour permettre de choisir les solutions optimales aux problèmes des curistes. De plus notre outil se doit d’être adaptable à d’autres contextes afin de pouvoir le réutiliser pour d’autres types d’institut en difficulté. L’implémentation du logiciel de simulation se fait en langage Java.

# Analyse du problème

## Diagramme de classe

La modélisation d’une architecture logicielle par un diagramme de classe permet de manière aisée de saisir la structure de représentation. Le diagramme de classe suivant correspond donc à la simulation que nous avons réalisée et on peut y observer deux parties. L’une d’entre elle correspond à la partie moteur de simulation et est décorrélée du problème considéré. La seconde partie implémente le problème en se basant sur le moteur de simulation.

**INCLURE DIAGRAMME DE CLASSE**

## Moteur de simulation

Dans le cadre de ce bureau d’étude, nous avons fait le choix de reconstruire un moteur de simulation intégralement. Cette volonté a plusieurs raisons. Tout d’abord cela nous permettait d’avoir un contrôle intégral sur le fonctionnement interne du moteur. De plus, d’un point de vue pédagogique, cela nous a permis de comprendre de manière plus approfondie les choix intrinsèques à un moteur de simulation (e.g. différence entre temps réel et événementiel) tout en nous permettant d’améliorer nos connaissances en Java. La structure de notre moteur de simulation se base sur trois éléments fondamentaux :

* un agenda contenant l’ensemble des événements prévus et se chargeant de les trier dans l’ordre chronologique,
* un scénario visant à fournir les particularités du problème considéré, il doit donc faire le lien entre moteur et données de simulation,
* le moteur en lui-même qui à partir de l’agenda et du scénario fait avancer la simulation, c’est lui qui a pour tâche de fournir le même temps logique à tous les éléments de la simulation ainsi que d’implémenter le générateur de nombre aléatoire ayant une seed contrôlable.

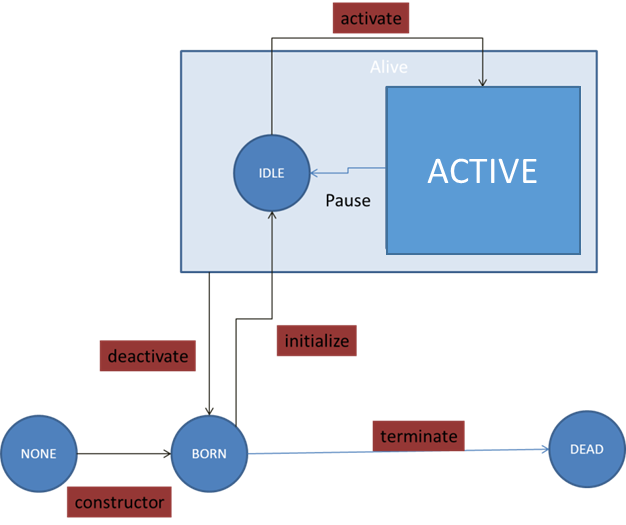
## Implémentation du problème considéré

### Notion d’entité

Les entités de simulation sont des éléments ayant un rôle prépondérant dans la simulation. L’une de leur caractéristique majeure est leur persistance dans le temps, par conséquent il est important de pouvoir les gérer finement. Elles suivent donc un cycle de vie strict permettant de gérer de manière séquentiel leur interaction avec le monde. Le cycle de vie est décrit dans les tableaux et le schéma suivant :

|  |  |
| --- | --- |
| **Etat** | **Description** |
| NONE | L’entité en cours de constitution (constructeur). |
| BORN | L’entité est créée mais ne peut poster d’événements. |
| IDLE | L’entité est initialisée et peut poster des événements |
| ACTIVE | L’entité est en cours de traitement des événements qui la concerne. |
| DEAD | L’entité n’a plus de raisons d’exister. Elle n’est plus connue du moteur de simulation. |

|  |  |
| --- | --- |
| **Transition** | **Description** |
| Constructor | Invocation du constructeur par le programme principal. Les paramètres du constructeur représentent les données techniques structurantes de l’entité. |
| Initialize | Invocation par le moteur. Initialisation de l’entité. Les paramètres de l’initialisation représentent des données ne structurant pas l’entité. |
| Activate | Invocation par le moteur. A l’activation, les événements de l’entité peuvent être postés à partir de cet événement. |
| Pause | Invocation par le moteur. La Pause peut être invoquée suite à demande d’utilisateur (optionnel) via le moniteur. |
| Deactivate | Invocation par le moteur. Désactive en cascade les autres entités dont elle est parente. Retire les événements ultérieurs à l’événement présent de l’échéancier. |
| Terminate | Invocation par le moteur. L’entité enlève toute référence à d’autres. Ceci permet au garbage collector de vider la mémoire. L’entité ne doit plus être référencée par aucun objet. |



Dans notre implémentation actuelle, l’intégralité des options rendues disponibles par les entités ne sont pas encore tirées à profit. Par exemple, un utilisateur ne peut pas interagir avec la simulation pour mettre en pause certaines entités. Dans notre vision du problème, les entités sont les classes suivantes :

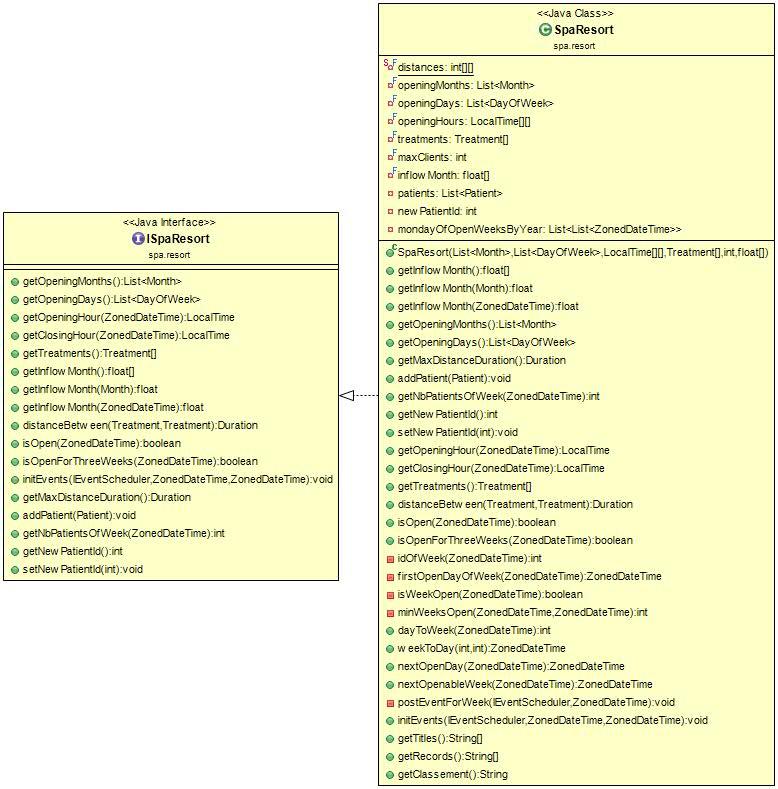
* l’établissement,
* les ateliers,
* les patients.

### Etablissement

|  |  |
| --- | --- |
| **Entité** |  |
| **Type d’Entité** |  |
| **Variables d’états** |  |
| **Variables statistiques/de scrutation** |  |
| **Paramètres techniques et Données d’initialisation** |  |
| **Evènements** |  |
| **Comportement** |  |
| **Processus Stochastique** |  |

La cure thermale est représentée sous forme de classe sous l’appellation SpaResort. Cette classe a pour vocation de regrouper les caractéristiques fondamentales de la cure à la fois en termes d’infrastructure (e.g. ateliers disponibles) que de commerce (e.g. fréquentation mensuelle).

En conséquence, la plupart des données caractéristiques de la cure thermale étant vouée à être constantes, elles sont définies en tant que « final » lors de la construction de l’instance. L’un des rôles prépondérants de cette entité lors de la simulation, outre fournir des données chiffrées comme la distance entre ateliers, consiste à initialiser au début les jours d’ouvertures du spa dans l’agenda du moteur de simulation. Cette initialisation respecte les hypothèses simplificatrices actuellement considérées.

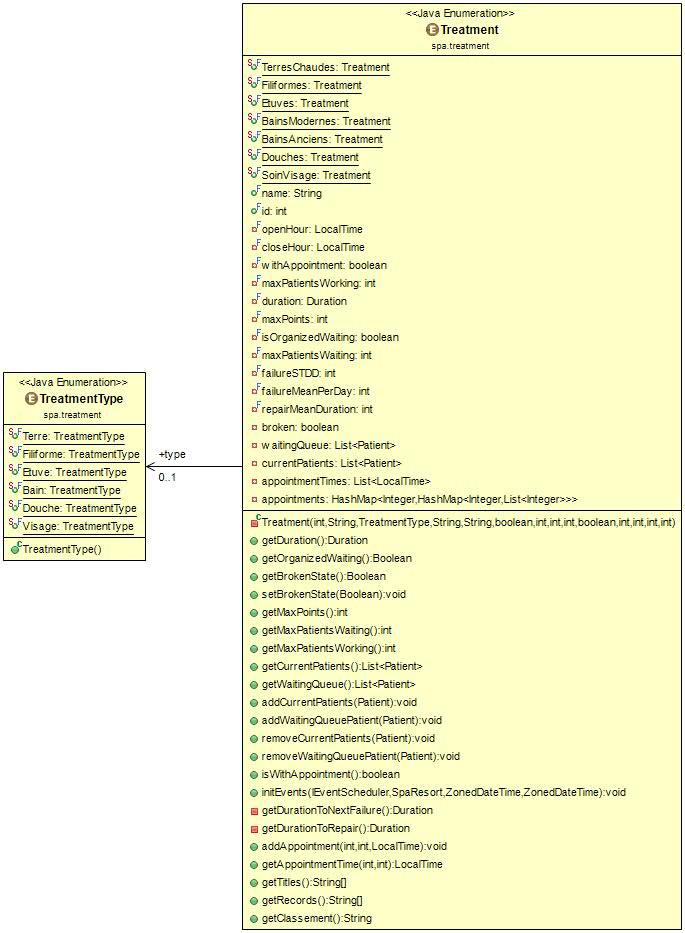


### Ateliers

Les ateliers disponibles pour les patients sont modélisés par la classe Treatment. Les ateliers étant définis à l’avance et ne souffrant pas de changements, nous avons convenu de réaliser une énumération. L’avantage est de disposer d’une base immuable pour les ateliers tout en ayant la possibilité d’en rajouter simplement.

Les ateliers ont à gérer leur file d’attente ainsi que les patients actuellement en cours de traitement. La dernière partie importante réside dans la gestion des éventuels problèmes entraînant la fermeture temporaire des ateliers. Pour cette partie, nous avons donc réfléchi à la manière de modéliser ces aléas. L’origine de ces fermetures étant principalement dû à de l’usure de matériel, nous voulions utiliser une loi représentant correctement l’usure. De ce fait, le risque se devait d’être faible au début avant d’arriver vers une valeur moyenne de défaillance à partir de laquelle la probabilité augmentait grandement. Une répartition uniforme n’était donc par exemple pas envisageable.

La première loi que nous avons voulu considérer était la loi de Weibull, couramment utilisée dans la modélisation des problèmes d’usure. Cependant, l’implémentation informatique était compliquée pour un gain négligeable en cohérence en comparaison d’autres lois. De ce fait, nous avons finalement opté pour une loi exponentielle facilement implémentable et fournissant des résultats similaires à la loi de Weibull.



### Patients

**TO DO**

### Notion d’évènements

Afin de réaliser une simulation événementielle de la cure thermale nous implémentons une liste contenant les évènements à venir. Cet agenda les trie par ordre chronologique en fonction du temps logique simulé. Tout au long de la simulation ces évènements sont retirés de l’agenda puis exécutés. Ils sont identifiés et répartis entre les différentes entités comme suit:

### Etablissement

#### OpenSpaEvent

Cet évènement est créé pour chaque jour d’ouverture du spa à son heure d’ouverture (7h). Il permet de logger les horaires du spa et de vérifier le bon fonctionnement du moteur de simulation. Il notifie les managers du changement d’état du spa.

#### CloseSpaEvent

Cet évènement est créé pour chaque jour de fermeture du spa lors de son heure de fermeture (14h). Il notifie les managers du changement d’état du spa et créer un évènement de fin d’atelier pour chaque patient en cours de traitement ou d’attente au sein de ces derniers. Un évènement pour quitter le spa est ajouté pour chaque patient en cours de déplacement.

### Ateliers

#### FailureEvent

Ces évènements sont créés au début de la simulation, lors de la création des différents ateliers permettant le calcul de leurs jours de fermeture. Ils sont créés par atelier en fonction du jour et de l’heure renvoyés par la loi exponentielle simulant leur probabilité de panne. Cet évènement met à jour la variable d’état *broken* de l’atelier pour signaler la panne de celui-ci. Chaque patient effectuant le traitement ou attendant son tour se voit ajouter un événement de fin d’atelier au même horaire que le *failureEvent* afin d’évacuer les patients.

#### RepairEvent

Comme pour *failureEvent*, ces événements sont créés au début de la simulation, lors de la création des différents ateliers permettant le calcul de leur temps de réparation et après la création du *failureEvent* associé. Cet évènement met à jour la variable d’état *broken* de l’atelier pour signaler la réparation de celui-ci.

#### AvailableTreatmentEvent

Cet évènement est créé lors du départ d’un patient d’un atelier sans rendez-vous après que ce patient ait fini son traitement. Il cherche le curiste suivant dans la file d’attente de l’atelier si elle n’est pas vide. Ce curiste est choisi en fonction de s’il est le premier dans la file d’attente et est assez rapide pour rentrer dans l’atelier ou s’il existe un patient malhonnête qui arrive à prendre sa place. Puis l'évènement ajoute le patient choisi à la liste des patients en cours de traitement dans l’atelier et créer un évènement fin d’atelier pour ce patient. Il instancie également l’horaire de commencement de l’atelier par le patient pour le calcul de ses points.

### Patient

#### CreatePatientsEvent

Cet évènement est créé à chaque fin de semaine (dimanche) afin de créer les patients manquant pour atteindre le taux d’affluence correspondant au mois de l’année. Le patient créé par cet évènement a 5% de chance d’être malhonnête et sa date de commencement de cure est instancié au lundi suivant. Initialisation des évènements du nouveau patient est ensuite effectuée (prise de rendez-vous, arrivée dans le spa).

#### PatientArrivalEvent

Cet évènement est créé lors de l’initialisation de chaque patient, pour chaque matin des neuf semaines sur les trois ans à venir. Il est déclenché à l’arrivée d’un client dans le spa à chaques jours de sa cure. Il remet à zéros la liste des ateliers de la journée déjà effectuées par le patient, change l’état du patient pour le mettre à “en mouvement” et créer un nouvel évènement de recherche d’action à effectuer par le curiste.

#### SearchForActionEvent

Cet évènement est créé suite à l’arrivée d’un patient dans le spa ou à la fin du traitement dans un atelier ou s’il n’y a plus de place dans un atelier et que le patient doit en chercher un autre. Cet évènement choisit le prochain atelier que le patient va faire, ou s’il va en zone de repos. Il change l’état du patient pour le mettre à “en mouvement”. Il élit l’atelier en fonction de sa proximité, de s’il est avec rendez-vous et s’il est en panne. Puis il créé l’évènement arrivée dans un atelier (et zone de repos) pour le patient et l’atelier choisi.

#### ArrivedTreatmentEvent

Cet évènement est créé suite à la recherche d’action par le patient et représente le patient arrivant dans un atelier après s’être déplacé. Il change l’état du patient pour le mettre “en cours de traitement”. S’il y a de la place dans l’atelier il ajoute le patient dans la liste des curistes en cours de traitement, et créer un évènement fin de traitement. Il instancie également l’horaire de commencement de l’atelier par le patient pour le calcul de ses points. S’il n’y a pas de place dans l’atelier mais de la place dans la file d’attente le patient est ajouté à cette dernière. L’évènement instancie l’horaire de début d’attente pour calculer le temps d’attente des patients. Sinon il créer un nouvel évènement de recherche d’action à effectuer par le curiste.

#### EndTreatmentEvent

Cet évènement est créé suite à l’arrivée d’un patient dans un atelier ayant de la place ou suite à la sélection d’un patient dans la file d’attente. Il correspond à la fin du traitement du patient dans l’atelier donné. L’évènement enlève le patient de sa liste de patient en cours de traitement et calcule ses points en fonction du temps resté dans l’atelier. Il met à jour la liste des ateliers effectués par le patient ce jour-ci et créer un nouvel évènement de recherche d’action à effectuer par le curiste.

#### AppointmentTimeoutEvent

Cet évènement est créé lors de l’initialisation des évènements des patients suite à leur création. Les rendez-vous des curistes pour les ateliers en proposant sont calculés au début de la première semaine pour les neuf semaines prévues dans la cure. Cet évènement est prévu cinq minutes avant l’heure de rendez-vous et fonctionne comme une alarme qui dit au patient d’aller à son atelier. Il créé un évènement d’arrivée dans un atelier avec le patient et l’atelier de son rendez-vous.

#### LeaveSpaEvent

Cet évènement est créé suite à la fermeture du spa qui amène les patients à quitter l’institut et à notifier les managers du changement d’état du spa. L’évènement change l’état du patient pour le mettre à “est parti”.

# Modélisation du système

## Hypothèses simplificatrices

Certaines hypothèses simplificatrices ont été utilisées pour la modélisation de ce bureau d’étude. Ces hypothèses permettent de simplifier grandement l’implémentation du système tout en n’influant selon nous que peu sur les résultats obtenus. En conséquence, aucune des hypothèses énumérées ici ne nous semble dénaturer l’analyse à postériori des résultats.

|  |  |
| --- | --- |
| Hypothèse | Cause |
| Le spa n’ouvre que par semaines entières | L’ouverture de la cure thermale lors de semaines partiellement non travaillées n’a pas de sens étant donné que les cures se comptent en semaine. Cela engendrerait de l’impossibilité de finir les cures sans que cela soit dû au fonctionnement interne de l’établissement. |
| Chaque année comporte le même nombre de semaines d’ouverture | Cette hypothèse ne diminue que très peu le temps d’ouverture de l’établissement. Elle permet par contre de simplifier le retour de patients d’une année sur l’autre. En effet, si une année comporte plus de semaines ouvrables que l’année suivante, il ne sera pas possible de refaire venir tout le monde sans dépasser l’affluence maximale. |
| Les managers ne sont pas actuellement implémentés | La présence de managers par atelier, bien que rajoutant du réalisme, apporte peu à la simulation si ce n’est des phases de recherche à l’aveugle de la part des curistes. La replanification des rendez-vous est donc simplifiée et cela reviendrait à supposer que le manager est perpétuellement présent. |
| Aucune distinction n’est faîte pour la fermeture des ateliers entre problème d’infrastructure ou maladie du personnel | Du point de vue du rendement de la cure thermale, peu importe la raison de fermeture d’un atelier. Nous avons donc choisi de nous focaliser sur la durée de fermeture. Qui plus est, bien qu’une loi modélisant une usure soit facile à trouver, il est bien plus compliqué de modéliser le fait de tomber malade. |

# Implémentation du modèle

Description du code. Ceci peut être aidé par le placement de commentaires pertinents dans le code.

Manuel utilisateur succinct : description du fonctionnement, paramétrage.

Le code source, fichiers de données et exécutables seront fournis sous forme électronique dans des sous-répertoires *src*, *data*, *bin* respectivement.

# Compte Rendu de v&v

Les ingénieurs veilleront à effectuer les tests requis pour s’assurer du bon fonctionnement de leur logiciel de simulation.

Les données issues de chaque simulation seront examinées et critiquées si nécessaire.

Revue critique du fonctionnement et des résultats.

# Présentation des résultats

L’analyse des résultats demandée nécessitera de votre part un peu d’imagination. Bien des choses sont envisageables pour améliorer la situation. Toutes ne sont pas à évaluer. Il vous est demandé d’en évaluer de manière chiffrée au moins une.

## Résultats de la simulation

Synthèse des résultats. On utilisera des graphiques (histogrammes, camemberts, etc.) pour faciliter la lecture.

*Les résultats seront fournis sous forme de fichiers Excel de préférence sinon dans un format type Word les schémas étant insérés dedans de manière propre.*

## Analyse des résultats

Commentaire des résultats et réponses apportées au problème posé.

Perspectives à apporter à M Jabbalehut.

## Perspectives d’évolution

Suggestions d’amélioration du logiciel de simulation, du modèle…