|  |  |
| --- | --- |
| 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.

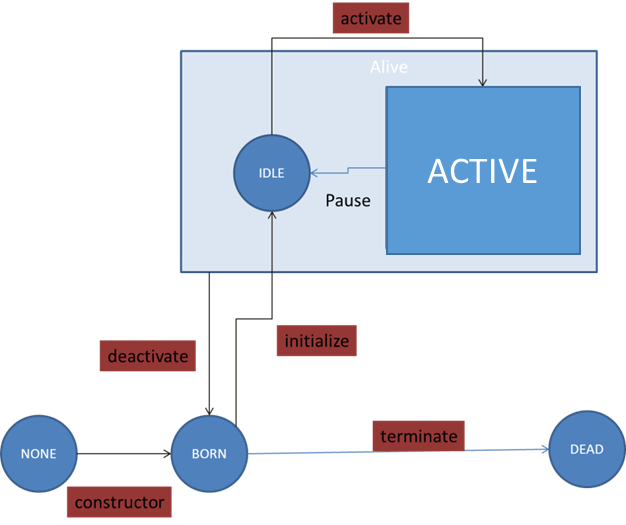
## Implémentation du problème considéré

### 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 ce qui a pour conséquence que le fait de pouvoir les gérer plus finement est important. 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. |



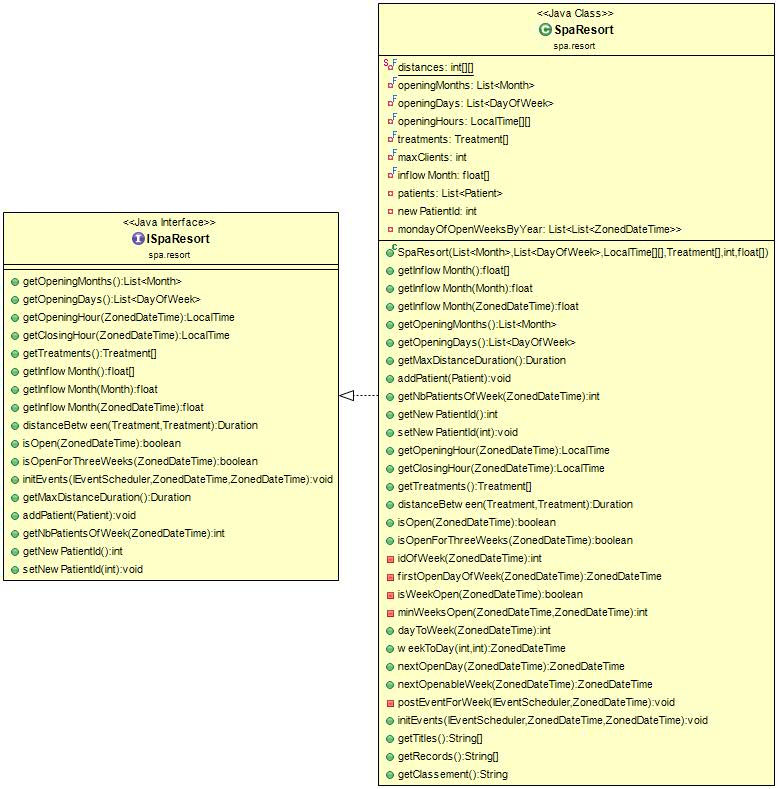
Dans notre implémentation actuelle, l’intégralité des options rendues disponibles par les entités ne sont pas encore tirées à profit. Par exemple, un utilisateur ne peut pas interagir avec la simulation pour mettre en pause certaines entités. Dans notre vision du problème, les entités sont les classes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

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

### Etablissement

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.

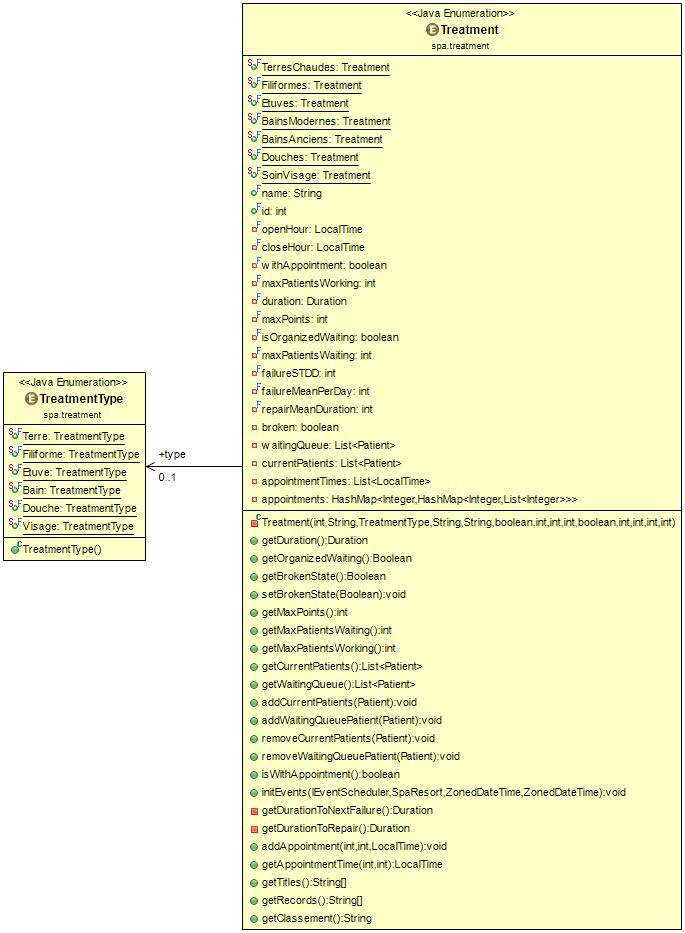


### Ateliers

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

# 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’un loi modélisant une usure est 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…