
POLYTECHNIQUE
MONTRÉAL

# LOG2410
# Conception Logicielle

Travail Personnel

Question1 : Formalisme UML 2.0 et BPMN 2.0

Soumis par :
Ryma Messedaa- 1953336
Groupe 2

Le 10 novembre 2019

# Introduction

Le déroulement d'un projet quelconque suit toujours un nombre défini de phases et d'étapes et c'est le cas aussi pour un projet technologique. En temps normal, la réalisation d'un projet technologique nécessite plusieurs étapes, dont une pour la définition des besoins et des exigences, une étape pour l'analyse du système, une troisième étape pour la conception du système et la programmation de celui-ci, après viennent les tests, le déploiement et pour finir la maintenance du système. Il est important de préciser que les étapes citées peuvent être organisées de différentes manières, et ce dépendamment du cycle de vie que l'on décide d'employer pour le projet. L'analyse du système est l'une des plus importantes étapes du processus de développement d'un logiciel. Lors de cette phase, les besoins et les exigences définis dans l'étape précédente seront filtrés et détaillés afin de comprendre davantage le fonctionnement interne du logiciel que l'on cherche à concevoir, et pour pouvoir réaliser cette phase, il est essentiel d'utiliser des principes, des méthodes, ainsi que des notations prévues pour cet effet. À la fin de cette étape, les besoins et les exigences du client ou des utilisateurs pourront être visualisés.

Il y a plusieurs manières pour représenter les résultats de la phase d'analyse, cela peut se faire par une description textuelle ou encore de manière visuelle et graphique. La première option, qui est la description textuelle, risque d'être longue et couteuse en terme de temps car un document de texte qui décrit de façon précise contiendrait plusieurs pages et laisse généralement place à différentes interprétations, tandis que la représentation graphique des résultats, qui est aussi nommée Modélisation, est plus facile à comprendre et reflète mieux les différents liens et interactions qui puissent exister entre les éléments résultants. La modélisation des besoins et exigences d'affaires peut se faire à l'aide de différents formalismes, on peut citer ANSI, VSM, BPMN ou encore UML.

L'analyse des besoins et des exigences d'affaires est une étape indispensable et plus celle-ci est détaillée, mieux les intérêts d'affaires seront couverts. Le niveau de détails ainsi que la densité de l'information qu'on trouve dans une modélisation donnée est fortement lié au formalisme utilisé pour produire cette dernière. Il est donc très important de faire le bon choix de formalisme pour la modélisation des besoins et des exigences d'affaires et c'est sur ce même point que va porter notre travail. Nous allons donc nous attarder sur la comparaison de deux formalisme UML et BPMN et lequel est privilégié pour une conception logicielle orientée objet.

## La modélisation

L'analyse des besoins est un processus en soit qui regroupe différentes étapes afin d'exprimer les divers besoins qui sont, en réalité, une satisfaction des exigence d'affaires d'un projet technologique. Pour pouvoir produire un document d'analyse et de spécification qui contient les résultats de cette analyse, il faut passer par plusieurs étapes ; il faut d'abord déterminer les besoins qui seront négociés, gérés et validés. Une fois validés, ces besoins seront modélisés et inscrits dans le document d'analyse. Cette modélisation permettra aux équipes de développement de concevoir un logiciel qui réponds aux attentes du client. On en déduit que plus la modélisation est bien faite, mieux les exigences seront couvertes. Cette modélisation nécessite l'usage d'un formalisme et le choix du bon formalisme favorise la qualité de la modélisation et donc la qualité du produit final. Il existe plusieurs formalismes pratiques pour la modélisation des besoins et exigences d'affaires, mais dans le cadre de ce travail nous allons en traiter deux seulement : Unified Modeling Language (UML), créé au début des années 1990, et Business Process Model and Notation (BPMN), apparu aux alentours de 2004. Dans les pages qui suivent, nous allons décrire et comparer les deux formalismes cités ci-haut.

## Formalisme UML

**UML en gros :**

UML est un langage de modélisation graphique qui permet de visualiser la conception d'un système, autrement dit, il permet de mieux comprendre les résultats de la phase d'analyse, donc mieux comprendre les besoins et les exigences d'un système. Comme tout autre langage, il existe plusieurs versions de UML et nous allons nous attarder ici sur la version 2.0.
UML se divise en plusieurs parties ; les vues, les diagrammes ainsi que les modèles d'élément. Les vues permettent de définir le système, les diagrammes décrivent le contenu des vues à l'aide de composants graphiques, et les modèles d'élément c'est les parties graphiques des diagrammes. Il y a donc une forte dépendance entres les différentes parties de UML.

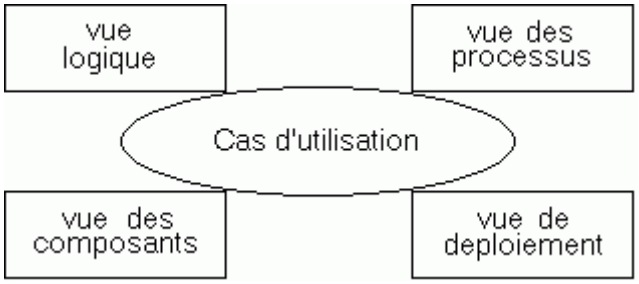


Figure 1 : Les différentes vue pour définir un système © IEEE 1997 [1]

Pour ce qui est des digrammes, UML couvre une variété de diagrammes dont le diagramme de contexte, un diagramme de composants, un diagramme de déploiement, un diagramme de paquetage, diagramme de cas d'utilisation, diagramme de classes…etc. La version la plus récente de UML contient 14 types de diagrammes, mais la version 2.0 n'en contient que 13. La réalisation d'un projet informatique nécessite un plan sur lequel les développeurs devraient se baser pour arriver au résultat souhaité. Le plan est un résultat de la phase d'analyse et de conception du projet, il contient donc toute la description et les détails concernant les exigences auxquelles doit répondre le produit final et ce sous forme de diagrammes. Par exemple, pour comprendre les différentes interactions qu'il y a entre l'utilisateur et le système, il faut se référer au diagramme de cas d'utilisation.

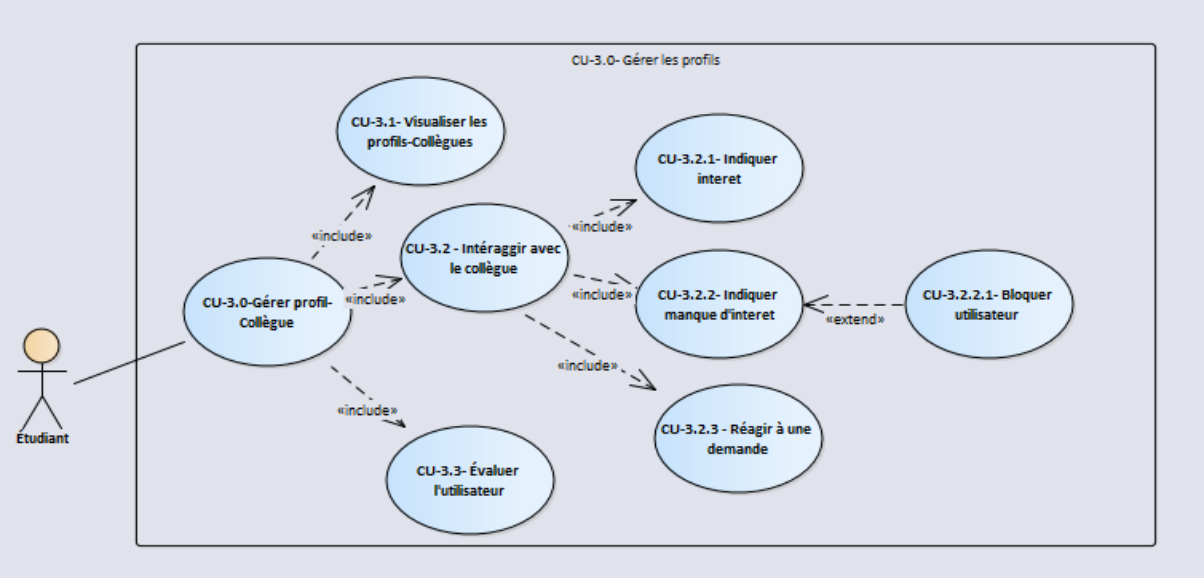


Figure2 : Diagramme d'un cas d'utilisation de l'application EquiPoly.

Si on veut connaitre les différentes interactions entre l'acteur et le système dans un ordre chronologique, on peut directement aller voir le diagramme de séquences qui décrit parfaitement les échanges qui se passent au cœur du système et ceux qui se passent avec les acteurs.

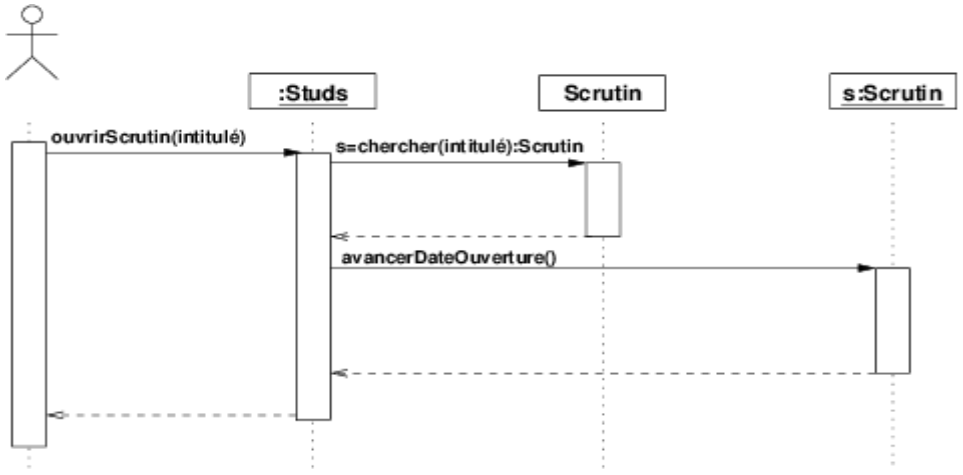


Figure 3 : Diagramme de séquence UML © IEEE 2015 [2]

Une fois rendu à l'étape de la programmation, les programmeurs peuvent se baser sur le diagramme de classes qui regroupe toutes les classes, les interfaces et les différents liens qui existent entre elles.

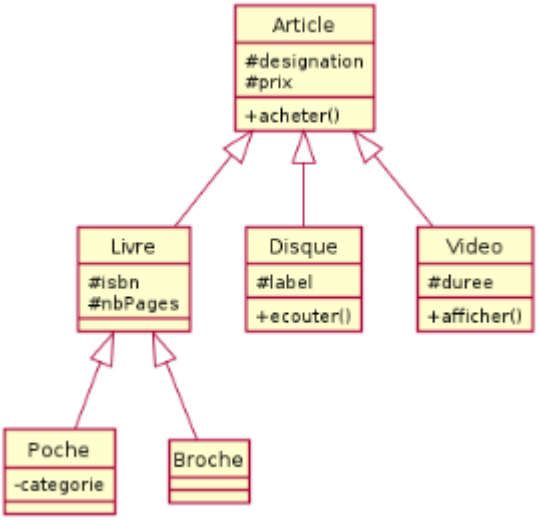


Figure 4 : Diagramme de classe en UML © IEEE 2013 [3]

UML couvre encore d'autres diagrammes, mais nous ne les décrirons pas tous dans ce travail par mesure de temps et de place. Même si UML 2.0 contient 13 diagrammes différents, les trois diagrammes décrits ci-haut sont les plus utilisé par les développeurs.

**Avantages et inconvénients :**

Comme tout autre outil, UML a des avantages et des inconvénients qui le rendent plus ou moins apprécié par ses utilisateurs.

Un des plus grands avantages de UML c'est qu'il estl considéré comme étant l'outil le plus utilisé et le plus flexible pour la conception de logiciels. Dans UML les éléments de modélisation et les interactions peuvent être modifié dans les diagramme, et ce en fonction du domaine ou des technologies utilisées. De plus, ce formalisme a une portée étendue, ce qui fait de lui un langage parfait pour communiquer les informations concernant l'architecture d'un logiciel quelconque. Un autre avantage de UML c'est que l'utilisateur n'a pas besoin de connaître toutes les fonctionnalités du langage pour pouvoir l'utiliser. Cela signifie que l'utilisateur peut exprimer une bonne partie de ses besoins en modélisation, soit 80%, rien qu'en connaissant 20% du langage UML. En plus, celui-ci contient une abondance d'outils qui couvre bien plus que les diagrammes que nous avons cités plus haut. Ces outils ne font pas juste représenter des schémas, des flèches et des étiquettes, ils peuvent également exporter les diagrammes en code et inversement, le code en diagrammes. La flexibilité de UML et l'abondance de ses outils font de lui le langage de modélisation de référence en génie logiciel, mais malgré toutes ses utilisations et avantages, UML reste critiqué par de nombreux développeurs pour certains points.

Pour commencer, nous allons citer l'argument le plus nommé par les développeurs qui critiquent UML et c'est l'informalité, autrement-dit, ils trouvent que ce n'est pas nécessaire d'avoir un diagramme UML pour modéliser les besoins et exigences en affaires ou toute autre aspect du projet. Ils pensent que cette modélisation peut se faire dans PowerPoint, Visio ou tout autre logiciel de modélisation simple d'usage. Aussi, UML est beaucoup critiqué pour sa complexité qui fait fuir les développeurs. La grande variété de diagrammes et d'outils du langage donne une sensation d'incapacité à apprendre à l'utiliser et que la personne fera mieux de s'en passer. D'autres développeurs considère que certaines architectures logicielles simples ne nécessitent aucune modélisation complexe pour expliquer leur conception, qui peut être faite par une simple documentation et qu'il est préférable de se concentrer plus sur la phase de programmation que sur celle de la modélisation, dans des cas comme celui-ci, le langage UML est jugé comme étant inutile.

Je tiens à préciser que dans certaines situations, la modélisation en utilisant UML prend énormément de temps, et cela est fortement lié à la complexité du langage, donc pour éviter de consacrer plus de temps qu'il en faut à la modélisation, nous sommes portés à limiter le temps consacré à l'analyse logicielle avec UML, car celle-ci peut être infiniment détaillée et donc prendre beaucoup de temps.

Pour conclure, UML existe et il est utilisé par les plus grands depuis plusieurs années et il va continuer d'exister et les adhérences et les refus à ce langage resteront aussi.

**Approche orientée objet en UML :**

UML est un langage riche et complet et en plus du fait qu'il peut aider les développeurs à présenter la description de leur système de façon compréhensible, il est principalement utilisé dans le développement de logiciels orientés objet. En plus, UML n'a été créé qu'après que la programmation orientée objet est apparue. Tous les diagrammes de UML sont facilement utilisés quand il s'agit d'un logiciel orienté objet, que ce soit le diagramme de classe, dans lequel on peut clairement montrer les différentes classes et objets ainsi que les tous les liens existant entre eux, il y aussi le diagramme des composants qui est très intéressant pour comprendre les composants ainsi que les dépendances dans le système à développer. Donc, UML contribue à la décomposition du système logiciel en objets afin de mieux comprendre leurs interactions. De plus, dans les diagrammes UML, l'utilisateur peut bien représenter les concepts fondamentaux de la conception orientée objet et ce grâce aux divers outils qu'offre le langage.

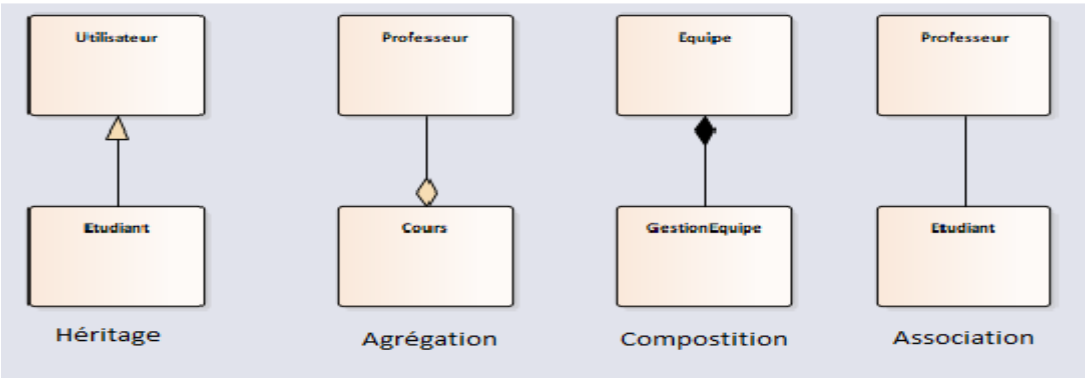


Figure 5 : Concepts orienté objet en UML

## Formalisme BPMN :

**BPMN en gros :**

BPMN est un formalisme de modélisation qui sert à représenter les différentes étapes d'un processus métier et qui vise à fournir une notation qui peut être comprise facilement par l'ensemble des personnes impliquées dans la gestion de ce processus, tel que les analystes, les techniciens responsables de l'implémentation ainsi que les autres participants du projet. Ce formalisme vise principalement à faciliter et accélérer l'expression des besoins et exigences d'un projet ainsi que les processus des différents acteurs et ce, en utilisant un langage visuel, commun et intuitif. De plus, le langage vise à faciliter le passage de la modélisation vers l'implémentation du projet. BPMN évolue très rapidement, en 2011 une version majeure du formalisme est apparue et c'est la version 2.0.

BPMN contient quatre modèles qui sont décrits par des spécifications dites formelles et ce en utilisant des symboles graphiques, une syntaxe, une sémantique ainsi que des règles d'usage. Donc, BPMN peut être considéré comme étant un langage avec son propre vocabulaire et syntaxe qui le représentent qui permettent de garder une certaine homogénéité dans l'expression des modèles. De plus, ce formalisme est assez complet pour représenter différents types de processus avec un niveau élevé de détails et d'informations

Le premier modèle de BPMN est le diagramme d'orchestration, il correspond à la définition de la séquence d'un processus du début à la fin, tout en citant les différents événements, activités et passerelles.

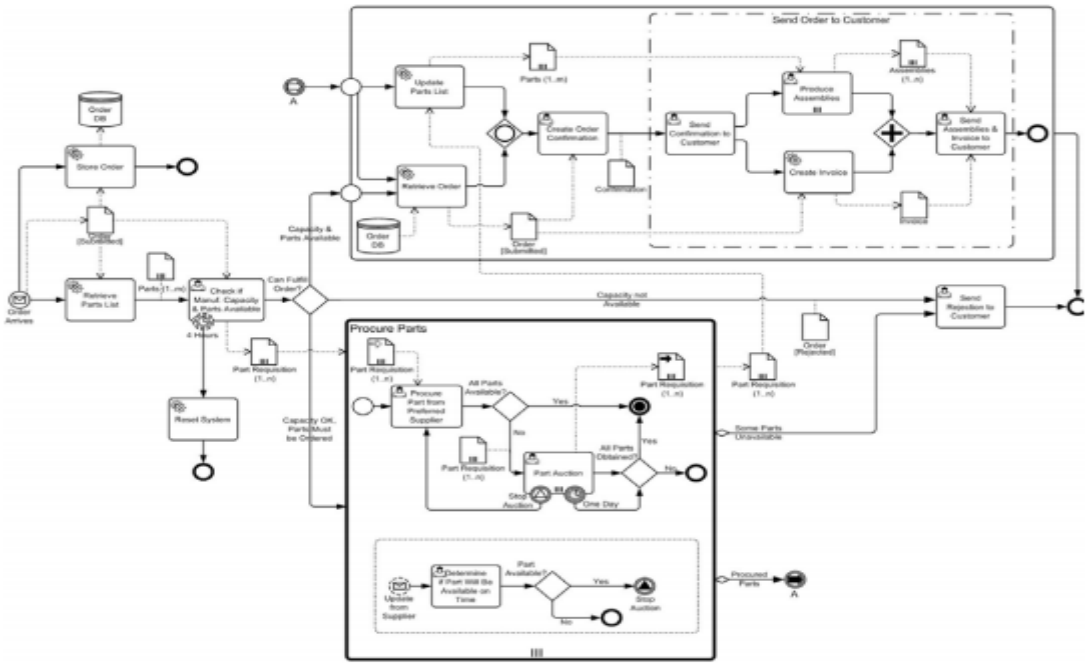


Figure 6 : Exemple de diagramme d'orchestration © IEEE 2011 [4]

Le deuxième modèle est le diagramme de collaboration, il permet d'exprimer graphiquement les interactions entre le processus en question et d'autres processus internes ou externes à l'organisation.

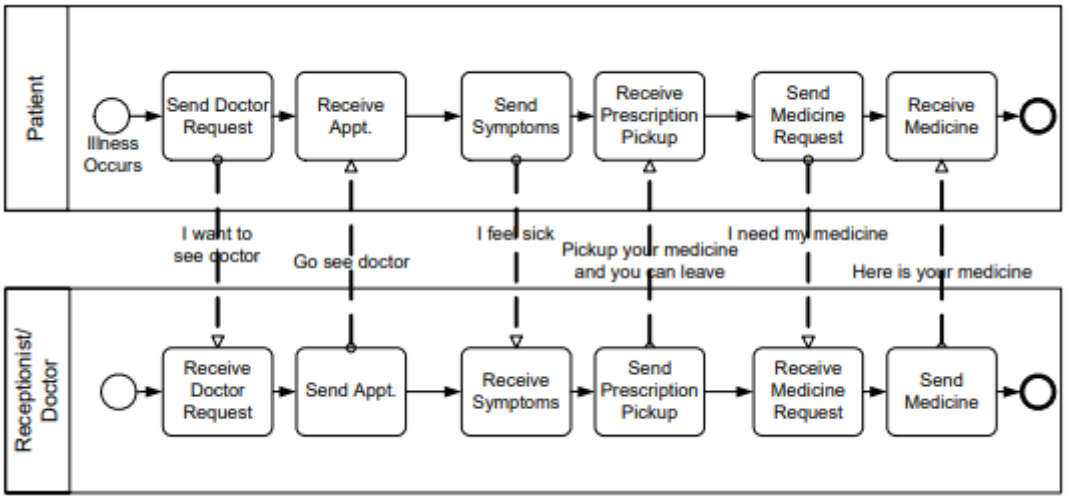


Figure 7 : Diagramme de collaboration © IEEE 2011 [4]

Le troisième modèle est le diagramme de conversation, qui est en réalité une description du diagramme de collaboration que nous avons défini ci-haut, il montre les différentes communications, représentées par des hexagones, qui arrivent entre les participants représentés par des cadrans, appelés Pools, dans l'exemple ci-dessous.

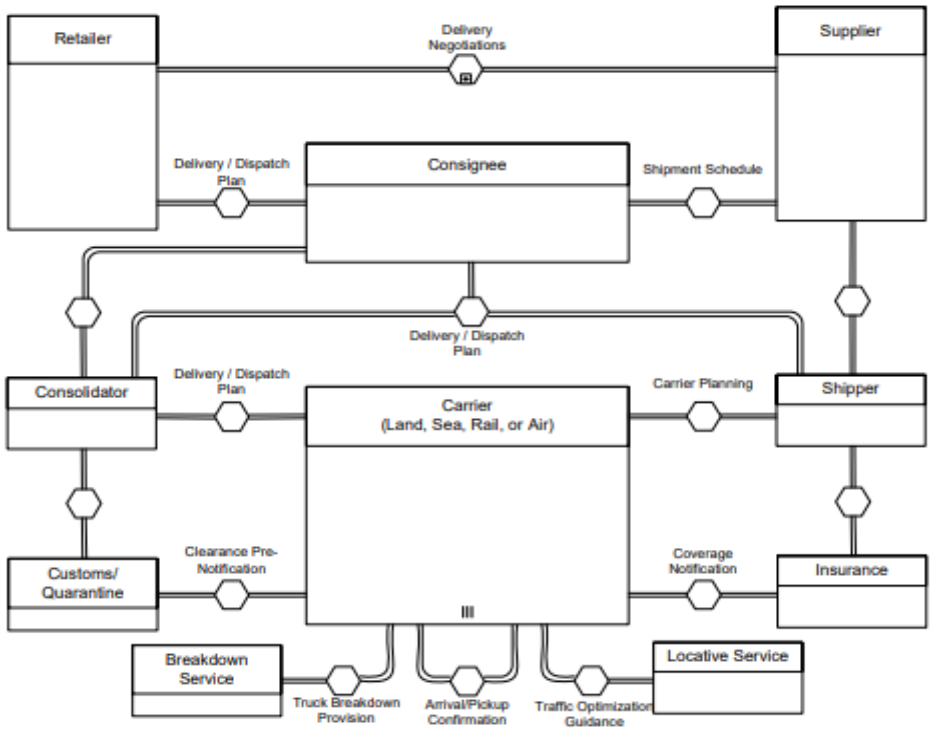


Figure 8 : Exemple de diagramme de conversation © IEEE 2011 [4]

Le quatrième et dernier modèle est le diagramme de chorégraphie, celui-ci est consacré pour la modélisation et la représentation du comportement attendu entre les participants qui interagissent entre eux.

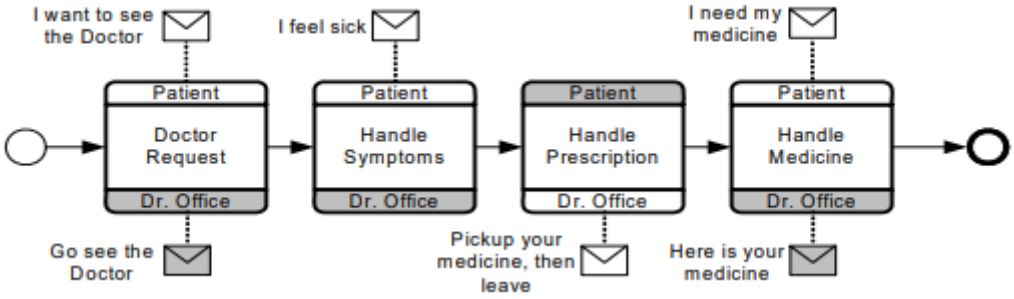


Figure 9 : Exemple de diagramme de chorégraphie © IEEE 2011 [4]

Les deux derniers modèles nommés, soient le diagramme de conversation et le diagramme de chorégraphie, sont en pratique beaucoup moins utilisés que les deux premiers.

**Avantages et inconvénients :**

BPMN est de plus en plus utilisé pour la modélisation des processus métier ce qui fait de lui un formalisme apprécié par ses utilisateurs, mais comme tout autres logiciels, il est critiqué par de nombreux développeurs.

La notation BPMN est un langage à niveau de compréhension intuitif, donc celui-ci est porté à faciliter l'intégration d'autres professionnels impliqués dans le projet pour mieux contrôler et surveiller le processus. Aussi, c'est un des rares formalismes qui prennent bien en charge la gestion des processus métier. BPMN contribue de façon remarquable à l'organisation de l'entreprise qui l'utilise, il rend l'échange de l'information traçable grâce à la documentation réalisée par celui-ci. De plus, grâce à la fluidité et la compréhension que nous procure ce formalisme, l'efficacité des équipes de travail et leur productivité sont clairement améliorées.

La version BPMN 2.0 nécessite plusieurs itérations pour arriver aux diagrammes et à la modélisation souhaitée. De plus, pour pouvoir utiliser ce formalisme, l'utilisateur doit connaître parfaitement l'interprétation et la sémantique des différents diagrammes de BPMN, car un manque de maîtrise du langage peut facilement l'induire en erreur. Un des autres inconvénients de BPMN c'est le niveau de support fourni pour la réalisation des différents

modèles, la plupart des outils de ce formalisme n'offrent qu'un support partiel pour l'exécution des diagrammes BPMN.

Pour finir, BPMN est un standard de modélisation de processus bien adopté pour la représentation graphique des processus de métier. Le formalisme s'impose au fur et à mesure des années pour devenir la référence en termes de modélisation des processus métier.

**Approche orientée processus en BPMN :**

Le formalisme de modélisation BPMN adopte une approche orientée processus, ceci dit qu'en BPMN, il est possible d'exécuter les diagrammes que nous avons décrit précédemment et de passer à des applications orientées processus.

## Différences entre les deux formalismes :

Le langage de modélisation UML est un langage de modélisation visuelle dans le domaine du génie logiciel, constitué de différents diagrammes et vise à construire et documenter un système logiciel. BPMN, quant à lui, c'est une visualisation graphique, comme UML, mais qui illustre et représente des processus métier. Donc, pour une première différence nous pouvons dire que UML est axé sur la modélisation de système logiciel et BPMN est axé sur la modélisation des processus métier.

De plus, UML est un langage de modélisation orienté objet qui adopte donc une approche orientée objet pour la modélisation des systèmes, tandis que BPMN adopte une approche orientée objet pour modéliser les systèmes.

UML est une notation de modélisation qui sert à documenter la conception d'un système couvrant plusieurs domaines tels que les télécommunications, la vente, le transport…etc. Tandis que BPMN est une notation de modélisation utilisée plus pour la gestion des processus d'entreprises.

## Le meilleur choix pour une conception orientée objet :

Pour répondre à la question de façon directe, nous pouvons nous baser sur ce que nous avons dit ci-haut. Le formalisme de modélisation UML est un formalisme orienté objet, tandis que le formalisme BPMN est un formalisme orienté processus, donc pour une conception orientée objet, le langage UML est à privilégier parce qu'il représente mieux les conceptions

orientée objet, que ce soit par le biais de ses différents diagrammes qui tiennent en considérations cette conception, comme le diagramme de classes qui est un bon moyen pour connaitre les différents liens entre les classes du système à concevoir, ou encore le diagramme de composants qui décrit le système en fonction de ses éléments.

## Conclusion :

En conclusion, nous pouvons constater que chacun des formalismes a des avantages et des inconvénients par rapport à l'autre. Un formalisme UML est plus approprié pour une conception orientée objet et plus utilisé dans la modélisation de système informatique, tandis qu'un formalisme BPMN est plus approprié pour une conception orientée processus. De plus, le formalisme UML modélise l'architecture et la conception des développements informatiques, tandis que le formalisme BPMN modélise et repense l'organisation dans son ensemble. Ce que nous même avons constaté lors de nos travaux dirigés c'est que la modélisation d'un système prend beaucoup de temps vu que le niveau de détails est illimité, c'est donc au développeur de gérer et définir le temps qui doit être consacré à la modélisation, car si jamais il ne le fait pas cette étape en UML pourrait lui faire perdre énormément de temps. Pour finir et pour répondre à la question, UML est un langage riche et complet qui est utilisé pour modéliser le génie logiciel orienté objet, mais aussi la structure des applications et des processus métier.

## Bibliographie :

[1] Carina, Roels (2019),Débutez l'analyse logicielle avec UML, consulté le 5 novembre 2019, tiré de https://openclassrooms.com/fr/courses/2035826-debutez-lanalyse-logicielle-avec-uml/2035851-uml-c-est-quoi

[2] D. Conan, C. Taconet, C. Bac, Télécom SudParis, CSC 4002 (2015), Conception et programmation orientée objet, consulté le 6 novembre 2019, tiré de http://www-inf.it-sudparis.eu/COURS/CSC4002/EnLigne/Cours/CoursUML/6.21.41.html

[3] Object Management Group, Inc (2011), Business Process Model and Notation (BPMN) version 2.0, consulté le 8 novembre 2019, tire de https://www.omg.org/spec/BPMN/2.0/PDF

[4] Ludovic, Arbelet (2004),BPMN : la modélisation des processus métier, consulté le 8 novembre 2019, tiré de https://bfmbusiness.bfmtv.com/01-business-forum/bpmn-la-modelisation-des-processus-metier-244098.html

## Références:

1- Kruchten, les différentes vues de l'architecture d'un projet, consulté le 5 novembre 2019, tiré de http://prive.iutenligne.net/6AnHSRtoQtJzxZjO/informatique/langages/kettaf/UML/07encadrementdeprojet/04vues.html

2- D. Conan, C. Taconet, C. Bac, Télécom SudParis, CSC 4002 (2015), Conception et programmation orientée objet, consulté le 6 novembre 2019, tiré de http://www-inf.it-sudparis.eu/COURS/CSC4002/EnLigne/Cours/CoursUML/6.21.41.html

3- Pierre, Gérard (2013), UML cours 2 : Diagramme de classe, consulté le 5 novembre 2019, tiré de https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours02.html

4- Object Management Group, Inc (2011), Business Process Model and Notation (BPMN) version 2.0, consulté le 8 novembre 2019, tire de https://www.omg.org/spec/BPMN/2.0/PDF