

Conception et mise en place d'un système Workflow pour l'environnement cloud

Workflow et Cloud

Thèse présentée par
Zerrouki Djamel

Encadre par : **M.Lahcen aid**

Une thèse présentée pour le diplôme de
Master 2 en Génie logiciel



Département Informatique
Université IBN KHALDOUN

Tiaret

2018-2019

Given a set of numbers, there are elementary methods to compute its Greatest Common Divisor, which is abbreviated GCD. This process is similar to that used for the Least Common Multiple (LCM).

Résumé

..... Les batailles de la vie ne sont pas gagnées par les plus forts, ni par les plus rapides mais par ceux qui n'abandonnent jamais

Abstrait

.....

Déclaration

.....

Liste Des Abréviations

EC2	Elastic Compute cloud.
GAE	Google App Engine
IBM	International Business Machines .
ALS	Service Level Agreement .
CPU	Central Processing Unit
QoS	Quality Of Service
VMM	Virtual Machine Monitor
XaaS	X as-a-Service
HaaS	Hardware as-a-Service
IaaS	Infrastructure as-a-Service
PaaS	Platform as-a-Service
SaaS	Software as-a-Service
BPaaS	Business Process as-a-Service
NIST	National Institute of Standards and Technology
VPN	Virtual Private Network
AWS	Amazon Web Services
WFMS	WorkFlow Management System
SGBD	Système de Gestion de Bases de Données
UIMS	User Interface Management Systems
WfMC	Workflow Management Coalition
WFMC-TC	Workflow Management Coalition Terminology and Glossary
BPR	Business Process Reengineering
WAPI	Workflow Application Programming interface

Introduction Général

Contexte

Problématique

Objectifs

Organisation du mémoire

Table des matières

I	Étude bibliographique	16
1	Concepts fondamentaux de cloud et de workflow	17
1.1	Introduction	17
1.2	Cloud computing	17
1.2.1	Concept du cloud computing	17
1.2.1.1	Vers une définition du cloud computing	18
1.2.2	Caractéristiques principales du cloud computing	19
1.2.2.1	Accès en libre-service à la demande	19
1.2.2.2	Accès réseau universel	19
1.2.2.3	Mutualisation de ressources (Pooling)	19
1.2.2.4	Scalabilité et élasticité	19
1.2.2.5	Autonome	19
1.2.2.6	Paieement à l’usage	19
1.2.2.7	Fiabilité et tolérance aux pannes	20
1.2.2.8	Garantie Quality Of Service (QoS)	20
1.2.2.9	Basé-SLA	20
1.2.3	Technologies connexes	20
1.2.4	Modèles du cloud computing	22
1.2.4.1	Modèles de service du cloudcomputing	22
1.2.4.2	Modèles de déploiement	23
1.2.5	Les principaux grands fournisseurs cloud pour 2019	24
1.2.5.1	Amazon Web Services	24
1.2.5.2	Microsoft Azure	24
1.2.5.3	Google Cloud Platform	24
1.2.5.4	IBM	25
1.3	Workflow	25
1.3.1	Introduction au Workflow	25
1.3.2	Origines	25
1.3.3	Définitions et terminologies	26
1.3.4	Définitions de base du Workflow	26
1.3.4.1	Définition d’un Workflow	27
1.3.4.2	Dimensions	28
1.3.5	Description des Systèmes Workflow	28
1.3.5.1	Build time	29
1.3.5.2	Run time	29
1.3.5.3	Utilisateurs, outils et applications	30
1.3.6	Concepts et Terminologie Workflow fondamentaux	30
1.3.6.1	Procédure Workflow (Workflow Process)	30

1.3.6.2	Activité (Process Activity)	30
1.3.6.3	Acteur, Ressource (Workflow Participant)	30
1.3.6.4	Rôle (Role)	31
1.3.6.5	Données (Workflow Relevant Data)	31
1.3.6.6	Application externe (Invoked Application)	31
1.4	Classification des systèmes Workflow	32
1.4.1	Processus collaboratifs	32
1.4.2	Workflow administratif	33
1.4.3	Workflow de production	33
1.4.4	Workflow adaptable ou Workflow ad hoc	34
1.4.5	Modèle de référence des systèmes Workflow	34
1.4.5.1	Interface avec les Outils de définition de procédures	34
1.4.5.2	Interface avec les applications clientes Workflow	35
1.4.5.3	Interface avec les applications invoquées	35
1.4.5.4	Interface avec les autres Workflow	36
1.4.5.5	Interface avec les outils de contrôle et d'administration	36
1.5	Intérêt du cloud pour les workflows	36
1.5.1	L'approvisionnement de ressources	36
1.5.2	L'allocation dynamique de ressources à la demande	37
1.5.3	L'élasticité	37
1.5.4	La garantie des QoS via des SLA	37
1.5.5	Le faible Coût d'exploitation	37
1.6	Conclusion	38
2	Réseau de Petri	39
2.1	Réseau de Pétri	39
2.1.1	Qu'est-ce que les réseaux de Pétri	39
2.1.2	L'aspect structurel	40
2.1.2.1	Représentation d'un réseau de Pétri	40
2.1.3	L'aspect comportemental	42
2.1.3.1	L'état dans un réseau de Pétri	43
2.1.3.2	Franchissement d'une transition	43
2.1.3.3	L'exécution d'un réseau de Pétri	43
2.2	Réseaux particuliers	45
2.3	Propriétés des réseaux de Petri	45
2.3.1	Réseau K-borné	45
2.3.2	Réseau vivant	45
2.3.3	Réseau réinitialisable	46
2.4	Réseaux de Pétri de haut niveau	47
2.4.1	Les réseaux de Petri colorés	47
2.4.2	Les réseaux de Pétri à Objet	48
2.4.3	Les réseaux de Pétri temporels	48
2.5	Réseaux de Petri et workflow	49
2.5.1	Modélisation de workflow avec les réseaux de pétri	49
2.5.1.1	Une sémantique formelle	49
2.5.1.2	Techniques d'analyse	49
2.5.1.3	Expressivité	49

2.5.2	Workflow à base de réseaux de pétri (WF-net)	50
2.5.3	Constructions du routage dans un WF-net	50
2.5.3.1	Séquentiel :	50
2.5.3.2	Parallèle :	51
2.5.3.3	Conditionnel :	51
2.6	Conclusion	52

II Partie pratique 53

3 Analyse et Conception 54

3.1	Unified Modeling Language (UML) :	54
3.1.1	Définition UML	54
3.1.2	Contenu UML	54
3.1.2.1	Diagramme structurel :	54
3.1.2.2	Diagramme comportemental :	55
3.2	Identification des objectifs	55
3.2.1	Spécifications fonctionnelles	55
3.2.2	Spécifications techniques	56
3.2.3	Diagramme de cas d'utilisation	56
3.2.3.1	les acteurs de système	56
3.2.3.2	Cas d'utilisation	57
3.2.4	Documentation des cas d'utilisation fonctionnels	58
3.2.4.1	S'authentifier	58
3.2.4.2	cas d'utilisation par l'utilisateur réception ou initial	58
3.2.4.3	Créer Dossiers	59
3.2.4.4	Activer Dossiers	59
3.2.4.5	cas d'utilisation par l'utilisateur simple	60
3.2.4.6	Consulter état des Dossiers	60
3.2.4.7	Finir le traitement des Dossiers	60
3.2.4.8	Gérer Bordereau	61
3.3	Représentation des informations	62
3.3.1	Diagramme de Classe	62
3.3.1.1	Conception détaillée	62
3.3.2	Diagramme d'activité	65
3.3.2.1	les rôles de l'acteur administrateur :	67
3.3.2.2	les rôles de l'acteur utilisateur rescription :	67
3.3.2.3	les rôles de l'acteur utilisateur simple :	67
3.4	Conclusion	67

4 Réalisation 68

4.1	Introduction	68
4.2	Outils et Technologies utilisées pour le développement	68
4.2.1	Spring	68
4.2.1.1	Spring Framework	68
4.2.1.2	Spring boot	68
4.2.1.3	Spring Data	68
4.2.1.4	Spring Cloud	68
4.2.1.5	Spring Cloud Netflix	69

4.2.2	Qu'est-ce que Micro Service ?	69
4.2.2.1	Pourquoi l'architecture de Microservices ?	69
4.2.2.2	Patterns dans l'architecture des microservices	69
4.2.2.3	Architecture des microservices via les composants Netflix	70
4.2.3	Composantes majeures de Netflix	70
4.2.3.1	Service Discovery Server	70
4.2.3.2	Routage dynamique et équilibreur de charge	70
4.2.3.3	Serveur Edge	71
4.2.4	Spring Boot et Spring Cloud Netflix OSS – Micro Service Ar- chitecture	71
4.2.4.1	Micro Services avec Spring Boot	71
4.2.4.2	Spring Cloud Netflix	71
4.2.5	Maven	71
4.2.5.1	Qu'est-ce que Maven ?	71
4.2.5.2	Qu'est-ce que le POM ?	72
4.2.5.3	Qu'est-ce qu'un archetype ?	73
4.2.5.4	Qu'est-ce qu'une dépendance ?	73
4.2.5.5	Qu'est-ce qu'un artefact ?	73
4.2.5.6	Qu'est-ce que le groupId/artifactId ?	73
4.2.5.7	Qu'est-ce qu'un SNAPSHOT ?	73
4.2.5.8	Qu'est-ce que le repository local, distant ?	74
4.2.6	bpmn js	74
4.2.7	Camunda BPM	74
4.2.8	Thymeleaf	74
4.3	Architecture Modèle/Vue/Contrôleur	75
4.3.1	Rôle du modèle	75
4.3.2	Rôle de la vue	75
4.3.3	Rôle du contrôleur	75
4.3.4	Interactions	76

Table des figures

1.1	Prévisions de la taille du marché du cloud computing public (Ried, 2011).	18
1.2	Convergence des différentes avancées menant à l'avènement du cloud computing.	20
1.3	Les services XaaS du cloud computing	22
1.4	Modèles de déploiement du cloud computing	24
1.5	Les systèmes de gestion de Workflow dans une perspective historique	26
1.6	Environnement du système Workflow	27
1.7	Dimensions des systèmes Workflow	28
1.8	Caractéristiques du système Workflow	29
1.9	Différentes classes des systèmes Workflow	32
1.10	Modèle de référence des systèmes de gestion de workflow (WfMC, 95).	35
2.1	Réseau de Pétri simple	40
2.2	Réseau de Pétri marqué	42
2.4	Graphe de Marquage	44
2.5	Réseau de Petri non borné	45
2.6	Réseau de Petri vivant	46
2.7	Exemple de réseau de Petri bloquant	46
2.8	Exemple de réseau de Pétri temporel	48
2.9	Procédure modélisée avec le WF-net	50
2.10	Routage séquentiel.	51
2.11	Routage parallèle.	51
2.12	Routage conditionnel non-déterministe	51
2.13	Choix explicite entre B et C à base de l'attribut x	52
2.14	Itération : B peut être exécutée plusieurs fois	52
3.1	Diagramme de Cas d'utilisation	57
3.2	Diagramme de Classe	62
3.3	Diagramme d'activité qui montre interaction des utilisateurs en fonction de leurs rôles dans le système en général.	66
4.1	Interactions entre le modèle, la vue et le contrôleur	76

Liste des tableaux

Première partie
Étude bibliographique

Chapitre 1

Concepts fondamentaux de cloud et de workflow

1.1 Introduction

Le cloud computing, traduit le plus souvent en français par ” informatique dans les nuages”, ” informatique dématérialisée ” ou encore ” infonuagique ”, est un domaine qui regroupe un ensemble de techniques et de pratiques consistant à accéder, en libre-service, à du matériel ou à des logiciels informatiques, à travers une infrastructure réseau (Internet). Ce concept rend possible la distribution des ressources informatiques sous forme de services pour lesquels l'utilisateur paie uniquement pour ce qu'il utilise. Ces services peuvent être utilisés pour exécuter des applications scientifiques et commerciales, souvent modélisées sous forme de workflows.

Ce chapitre présente une introduction au cloud computing et au workflow, nécessaire pour la compréhension générale de ce mémoire.

1.2 Cloud computing

1.2.1 Concept du cloud computing

L'idée principale du cloud est apparue dans les années 60, où le professeur John McCarthy avait imaginé que les ressources informatiques seront fournies comme des services d'utilité publique (Garfinkel, 1999). C'est ensuite, vers la fin des années 90, que ce concept a pris de l'importance avec l'avènement du grid computing (Foster, 1999). Le terme cloud est une métaphore exprimant la similarité avec le réseau électrique, dans lequel l'électricité est produite dans de grandes centrales, puis disséminée à travers un réseau jusqu'aux utilisateurs finaux. Ici, les grandes centrales sont les Datacenter, le réseau est le plus souvent celui d'Internet et l'électricité correspond aux ressources informatiques. Le cloud computing n'est véritablement apparu qu'au cours de l'année 2006 (Vouk, 2008) avec l'apparition d'Amazon Elastic Compute cloud (EC2). C'est en 2009 que la réelle explosion du cloud survint avec l'arrivée sur le marché de sociétés comme Google (Google App Engine), Microsoft (Microsoft Azure), IBM (IBM Smart Business Service), Sun (Sun cloud) et Canonical Ltd (Ubuntu Enterprise cloud). D'après une étude menée par Forrester (Ried, 2011), le marché du cloud computing s'élevait à environ 5,5 milliards de dollars en

2008, il devrait atteindre plus de 150 milliards d'ici 2020, comme l'illustre la figure 1.1.

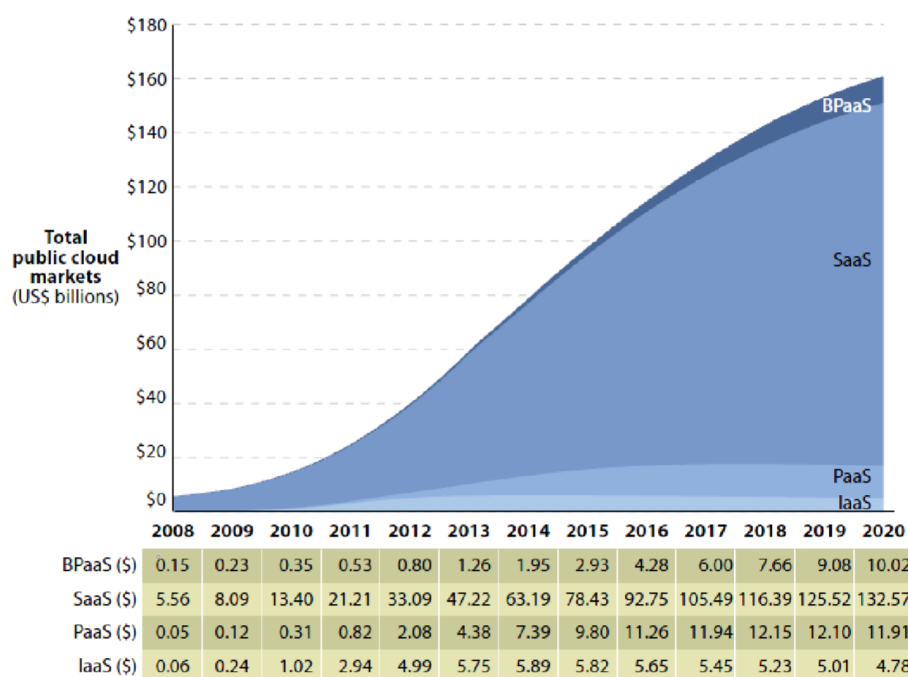


FIGURE 1.1 – Prévisions de la taille du marché du cloud computing public (Ried, 2011).

1.2.1.1 Vers une définition du cloud computing

Beaucoup de chercheurs ont tenté de définir le cloud computing (Geelan, 2008 ; McFedries, 2008 ; Buyya, 2009 ; Armbrust, 2010). La plupart des définitions attribuées à ce concept semblent se concentrer seulement sur certains aspects technologiques. L'absence d'une définition standard a généré non seulement des exagérations du marché, mais aussi des confusions. Pour cette raison, il y a eu récemment des travaux sur la normalisation de la définition du cloud computing, à l'exemple de Vaquero et coll (Vaquero, 2009) qui ont comparé plus de 20 définitions différentes et ont proposé une définition globale. En guise de synthèse des différentes propositions données dans la littérature, nous introduisons une définition mixte, qui correspond aux différents types de cloud considérés dans les travaux réalisés dans ce mémoire.

Définition 1.2.1. le cloud est comme un modèle informatique qui permet d'accéder, d'une façon transparente et à la demande, à un pool de ressources hétérogènes physiques ou virtualisées (serveurs, stockage, applications et services) à travers le réseau. Ces ressources sont délivrées sous forme de services reconfigurables et élastiques, à base d'un modèle de paiement à l'usage, dont les garanties sont offertes par le fournisseur via des contrats de niveau de Service Level Agreement (ALS).

1.2.2 Caractéristiques principales du cloud computing

1.2.2.1 Accès en libre-service à la demande

Le cloud computing offre des ressources et services aux utilisateurs à la demande. Les services sont fournis de façon automatique, sans nécessiter d'interaction humaine (Mell, 2011).

1.2.2.2 Accès réseau universel

Les services de cloud computing sont facilement accessibles au travers du réseau, par le biais de mécanismes standard, qui permettent une utilisation depuis de multiples types de terminaux (par exemple, les ordinateurs portables, tablettes, smartphones) (Mell, 2011).

1.2.2.3 Mutualisation de ressources (Pooling)

Les ressources du cloud peuvent être regroupées pour servir des utilisateurs multiples, pour lesquels des ressources physiques et virtuelles sont automatiquement attribuées (Mell, 2011). En général, les utilisateurs n'ont aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies. Toutefois, ils peuvent imposer de spécifier l'emplacement à un niveau d'abstraction plus haut.

1.2.2.4 Scalabilité et élasticité

Des ressources supplémentaires peuvent être automatiquement mises à disposition des utilisateurs en cas d'accroissement de la demande (en réponse à l'augmentation des charges des applications) (Geelan, 2008), et peuvent être libérées lorsqu'elles ne sont plus nécessaires. L'utilisateur a l'illusion d'avoir accès à des ressources illimitées à n'importe quel moment, bien que le fournisseur en définisse généralement un seuil (par exemple : 20 instances par zone est le maximum possible pour Amazon EC2).

1.2.2.5 Autonome

Le cloud computing est un système autonome et géré de façon transparente pour les utilisateurs. Le matériel, le logiciel et les données au sein du cloud peuvent être automatiquement reconfigurés, orchestrés et consolidés en une seule image qui sera fournie à l'utilisateur (Wang, 2008).

1.2.2.6 Paiement à l'usage

La consommation des ressources dans le cloud s'adapte au plus près aux besoins de l'utilisateur. Le fournisseur est capable de mesurer de façon précise la consommation (en durée et en quantité) des différents services (Central Processing Unit (CPU), stockage, bande passante, ...); cela lui permettra de facturer l'utilisateur selon sa réelle consommation (Armbrust, 2009).

1.2.2.7 Fiabilité et tolérance aux pannes

Les environnements cloud tirent parti de la redondance intégrée du grand nombre de serveurs qui les composent en permettant des niveaux élevés de disponibilité et de fiabilité pour les applications qui peuvent en bénéficier (Buyya, 2008).

1.2.2.8 Garantie QoS

Les environnements de cloud peuvent garantir QoS la qualité de service pour les utilisateurs, par exemple, la performance du matériel, comme la bande passante du processeur et la taille de la mémoire (Wang, 2008).

1.2.2.9 Basé-SLA

Les clouds sont gérés dynamiquement en fonction des contrats d'accord de niveau de service (SLA) (Buyya, 2008) entre le fournisseur et l'utilisateur. Le SLA définit des politiques, telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, la performance, l'exploitation, ou autres attributs du service, comme la facturation, et même des sanctions en cas de violation du contrat. Le SLA permet de rassurer les utilisateurs dans leur idée de déplacer leurs activités vers le cloud, en fournissant des garanties de QoS. Après avoir présenté les caractéristiques essentielles d'un service cloud, nous présentons, brièvement, dans la section suivante, quelques technologies connexes aux clouds.

1.2.3 Technologies connexes

Le cloud computing utilise des technologies telles que la virtualisation, l'architecture orientée services et les services web. Le cloud est souvent confondu avec plusieurs paradigmes informatiques, tels que le Grid computing, l'Utility computing et l'Autonomic computing, dont chacun partage certains aspects avec le cloud computing. La figure 1.2 montre la convergence des technologies qui ont contribué à l'avènement du cloud computing.

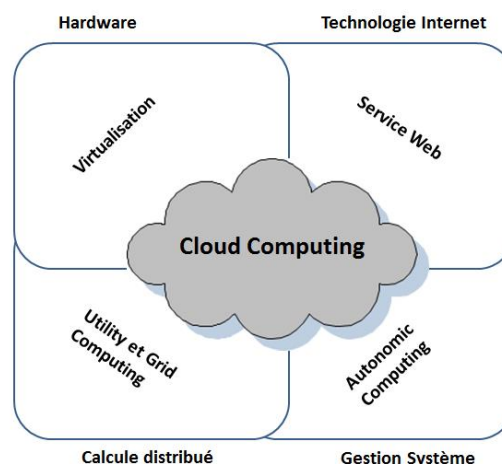


FIGURE 1.2 – Convergence des différentes avancées menant à l'avènement du cloud computing.

1. **Grid computing** : Le grid computing est un paradigme de calcul réparti qui coordonne des ressources autonomes et géographiquement distribuées pour atteindre un objectif de calcul commun. Le grid est basé sur le partage dynamique des ressources entre des participants, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications scientifiques, qui sont généralement des calculs intensifs ou des traitements de très gros volumes de données. Le cloud computing est similaire au grid computing : les deux adoptent le concept d'offrir les ressources sous forme de services. Toutefois, ils sont différents dans leur finalité : tandis que la technologie des grilles vise essentiellement à permettre à des groupes différents de partager l'accès en libre service à leurs ressources, le cloud a pour objectif de fournir aux utilisateurs des services « à la demande », sur le principe du « paiement à l'usage ». De plus le cloud exploite les technologies de virtualisation à plusieurs niveaux (matériel et plateforme d'application) pour réaliser le partage et l'approvisionnement dynamique des ressources.
2. **Utility computing (informatique utilitaire)** : L'utility computing est simplement une délocalisation d'un système de calcul ou de stockage. Il présente un modèle d'affectation des ressources à la demande et facturation des utilisateurs en fonction de leur usage. Le cloud computing peut être perçu comme une réalisation de l'informatique utilitaire. Il adopte un système de tarification basé sur l'utilité, entièrement pour des raisons économiques. Avec l'approvisionnement des ressources à la demande et le paiement à l'usage, les fournisseurs de services peuvent maximiser l'utilisation des ressources et minimiser leurs coûts d'exploitation.
3. **L'autonomic computing** : L'autonomic computing vise à construire des systèmes informatiques capables de s'auto-administrer en s'adaptant à des changements internes et externes sans intervention humaine. Le but de l'informatique autonome est de surmonter la complexité de la gestion des systèmes informatiques d'aujourd'hui. Bien que le cloud computing présente certaines caractéristiques autonomes, telles que l'approvisionnement automatique des ressources, son objectif est de diminuer le coût des ressources, plutôt que de réduire la complexité du système.
4. **Virtualisation** : La virtualisation est une technologie qui fait abstraction des détails du matériel physique et fournit des ressources virtualisées pour les applications de haut niveau. En effet, la virtualisation regroupe l'ensemble des techniques matérielles ou logicielles permettant de faire fonctionner, sur une seule machine physique, plusieurs configurations informatiques (systèmes d'exploitation, . . .), de sorte à former plusieurs machines virtuelles, qui reproduisent le comportement des machines physiques. La virtualisation constitue le socle du cloud computing, car elle offre la possibilité de mettre en commun des ressources informatiques à partir de clusters de serveurs, et ainsi d'affecter ou réaffecter dynamiquement des machines virtuelles aux applications à la demande. La figure 1.3. montre l'architecture en couches de la technologie de virtualisation. Le Virtual Machine Monitor (VMM) en français moniteur de machine virtuelle, également appelé hyperviseur, partitionne la ressource physique du serveur physique sous-jacente en plusieurs machines virtuelles différentes, chacune fonctionnant sous un système d'exploitation distinct et une pile de logiciels utilisateur.

1.2.4 Modèles du cloud computing

1.2.4.1 Modèles de service du cloud computing

X as-a-Service (XaaS) représente la base du paradigme du cloud computing, où X représente un service tel qu'un logiciel, une plateforme, une infrastructure, un Business Process Business Process as-a-Service (BPaaS), etc. Nous présentons, dans cette section, quatre modèles de services (Rimal, 2009), à savoir : (1) Logiciel en tant que services Software as-a-Service (SaaS), où le matériel, l'hébergement, le framework d'application et le logiciel sont dématérialisés, (2) Plateforme en tant que service Platform as-a-Service (PaaS), où le matériel, l'hébergement et le framework d'application sont dématérialisés, (3) Infrastructure en tant que service Infrastructure as-a-Service (IaaS) et (4) Matériel en tant que service Hardware as-a-Service (HaaS), où seul le matériel (serveurs) est dématérialisé dans ces deux derniers cas. La figure 1.3 montre le modèle classique et les différents modèles de service de cloud

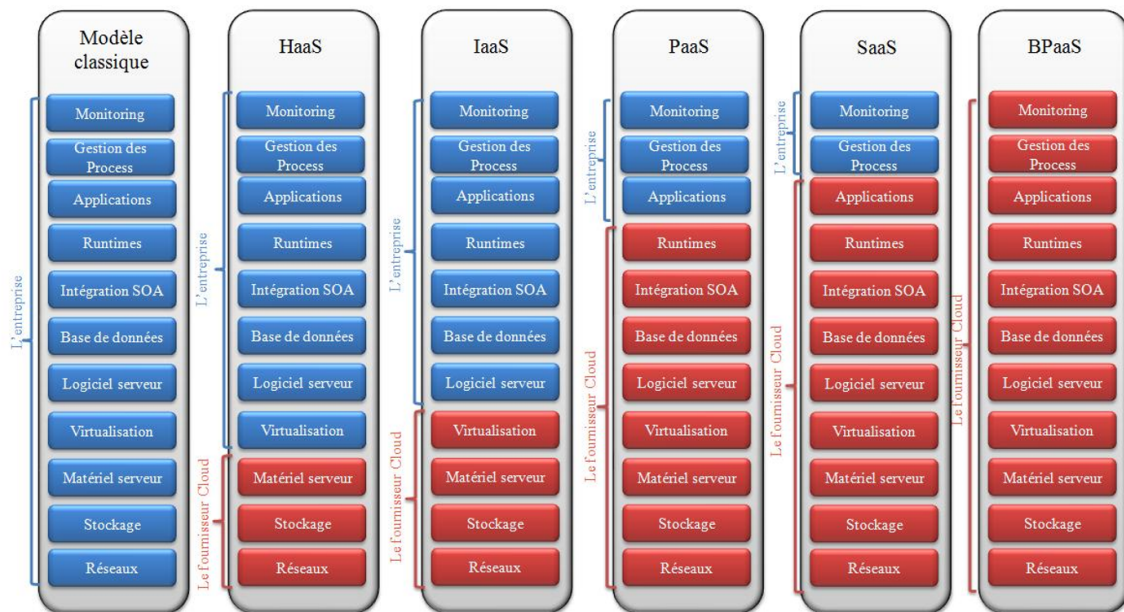


FIGURE 1.3 – Les services XaaS du cloud computing

1. Software as a Service (SaaS) :

Ce modèle de service est caractérisé par l'utilisation d'une application partagée qui fonctionne sur une infrastructure Cloud. L'utilisateur accède à l'application par le réseau au travers de divers types de terminaux (souvent via un navigateur web). L'administrateur de l'application ne gère pas et ne contrôle pas l'infrastructure sous-jacente (réseaux, serveurs, applications, stockage). Il ne contrôle pas les fonctions de l'application à l'exception d'un paramétrage de quelques fonctions utilisateurs limitées. On prend comme exemple les logiciels de messagerie au travers d'un navigateur comme Gmail ou Yahoo mail.

2. Platform as a Service (PaaS) :

L'utilisateur a la possibilité de créer et de déployer sur une infrastructure Cloud PaaS ses propres applications en utilisant les langages et les outils du fournisseur. L'utilisateur ne gère pas ou ne contrôle pas l'infrastructure Cloud

sous-jacente (réseaux, serveurs, stockage) mais l'utilisateur contrôle l'application déployée et sa configuration. Comme exemple de PaaS, on peut citer un des plus anciens -IntuitQuickbase- qui permet de déployer ses applications bases de données en ligne ou -Google App Engine (GAE)- pour déployer des services Web.

Dans ces deux cas l'utilisateur de ces services n'a pas à gérer des serveurs ou des systèmes pour déployer ses applications en ligne et dimensionner des ressources adaptées au trafic.

3. Infrastructure as a Service (IaaS) :

L'utilisateur loue des moyens de calcul et de stockage, des capacités réseau et d'autres ressources indispensables (partage de charge, pare-feu, cache). L'utilisateur a la possibilité de déployer n'importe quel type de logiciel incluant les systèmes d'exploitation. L'utilisateur ne gère pas ou ne contrôle pas l'infrastructure Cloud sous-jacente mais il a le contrôle sur les systèmes d'exploitation, le stockage et les applications. Il peut aussi choisir les caractéristiques principales des équipements réseau comme le partage de charge, les pare-feu, etc. L'exemple emblématique de ce type de service est Amazon Web Services qui fournit du calcul (EC2), du stockage (S3, EBS), des bases de données en ligne (SimpleDB) et quantité d'autres services de base. Il est maintenant imité par de très nombreux fournisseurs.

1.2.4.2 Modèles de déploiement

Selon la définition du cloud computing donnée par le National Institute of Standards and Technology (NIST) (Mell, 2011), il existe quatre modèles de déploiement des services de cloud, à savoir : cloud privé, cloud communautaire, cloud public et cloud hybride, comme illustré dans la figure 1.4.

1. Cloud privé :

L'ensemble des ressources d'un cloud privé est exclusivement mis à disposition d'une entreprise ou organisation unique. Le cloud privé peut être géré par l'entreprise elle-même (cloud privé interne) ou par une tierce partie (cloud privé externe). Les ressources d'un cloud privé se trouvent généralement dans les locaux de l'entreprise ou bien chez un fournisseur de services. Dans ce dernier cas, l'infrastructure est entièrement dédiée à l'entreprise et y est accessible via un réseau sécurisé (de type Virtual Private Network (VPN)). L'utilisation d'un cloud privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

2. Cloud communautaire :

L'infrastructure d'un cloud communautaire est partagée par plusieurs organisations indépendantes ayant des intérêts communs. L'infrastructure peut être gérée par les organisations membres ou par un tiers. L'infrastructure peut être située, soit au sein des dites organisations, soit chez un fournisseur de services.

3. Cloud public :

L'infrastructure d'un cloud public est accessible à un large public et appartient à un fournisseur de services. Ce dernier facture les utilisateurs selon la consommation et garantit la disponibilité des services via des contrats SLA.

4. Cloud hybride :

L'infrastructure d'un cloud hybride est une composition de plusieurs clouds (privé, communautaire ou public). Les différents clouds composant l'infrastructure restent des entités uniques, mais sont reliés par une technologie standard ou propriétaire permettant ainsi la portabilité des données ou des applications déployées sur les différents clouds.

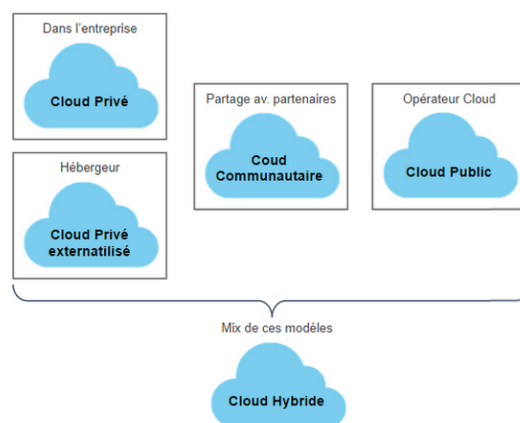


FIGURE 1.4 – Modèles de déploiement du cloud computing

1.2.5 Les principaux grands fournisseurs cloud pour 2019

Comment choisir la solution de cloud la plus adaptée à votre entreprise, quelles sont les différentes offres et les tarifs pour stocker les données informatiques sur un serveur à distance. Comparatif des principales solutions du marché.

1.2.5.1 Amazon Web Services

Avec un chiffre d'affaires de 25.65 milliards de dollar (Zdnet.fr) réalisé en 2018, le géant de l'IaaS considère 2019 comme une année d'investissement afin d'accélérer le développement de sa technologie et renforcer sa force de vente. En ce début de 2019, Amazon Web Services (AWS) est le leader incontesté dans le domaine de l'IaaS (EPSI 2019).

1.2.5.2 Microsoft Azure

Le service cloud, Microsoft Azure a réalisé un chiffre d'affaires annuel de 11 milliards de dollar. En 2019, le service cloud Microsoft Azure vient en deuxième position derrière AWS (IaaS et PaaS). A la différence d'AWS, Microsoft opère plutôt dans le commercial cloud et propose un service très diversifié grâce à une plateforme familiarisée auprès des utilisateurs (EPSI 2019).

1.2.5.3 Google Cloud Platform

L'année 2018 a été particulièrement fructueuse chez le service cloud de Google. Avec un chiffre d'affaires de près de 4 milliards de dollar, l'entreprise a décroché en 2018 les contrats les plus importants dans le secteur. Pour 2019, elle va constituer

un véritable contrepoint à AWS et à Microsoft même si l'entreprise a toujours du mal à publier convenablement ses rapports financiers jusqu'à présent (EPSI 2019).

1.2.5.4 IBM

Très présent dans le monde du multcloud et du cloud hybride, il dispose actuellement d'un chiffre d'affaires de 12.2 milliards de dollar. Même s'il reste un peu derrière AWS et Microsoft, il est en train de se focaliser sur le développement d'une plateforme capable de gérer plusieurs outils d'intelligence artificielle pour les principaux fournisseurs cloud. En plus, l'acquisition de Red Hat est un signal fort pour le cloud hybride (EPSI 2019).

1.3 Workflow

1.3.1 Introduction au Workflow

Un Workflow est la modélisation et la gestion assistée par ordinateur de l'accomplissement des tâches composant un processus administratif ou industriel, en interaction avec divers acteurs (humains, logiciels, ou matériels) invoqués [COURTOIS 96]. Outil informatique d'origine industrielle, le Workflow est l'adaptation de la GED24 adjoint de la faculté à gérer l'échange de messages. Le Workflow propose des solutions d'optimisation et de rationalisation des flux d'informations ; que ces informations soient associées à des documents, des procédures ou des messages complétant les systèmes de gestion électronique de documents et d'informations.

A l'heure actuelle plus de 250 Systèmes de Gestion de Workflow (WFMS) sont utilisés ou en développement. Cela signifie que le terme « gestion de Workflow » n'est pas simplement une nouvelle expression à la mode. Ce phénomène de gestion de processus (Workflow) aura certainement un fort impact sur la génération suivante de systèmes informatiques [COURTOIS 96, HAYES 91, KOULOPOULOS 95, SCHAEEL 97].

1.3.2 Origines

Il est intéressant de considérer l'évolution des systèmes informatiques au cours des quatre dernières décennies [VAN DER AALST 02] pour prendre conscience de la pertinence d'une gestion électronique de processus (Workflow) et apprécier l'impact de la gestion de Workflow dans un avenir proche.

La Figure 1.5 présente le phénomène de gestion de Workflow dans une perspective historique. Cette figure décrit l'architecture d'un système informatique classique en termes de composants. Dans les années soixante, un système informatique était composé d'un certain nombre d'applications autonomes. Pour chacune de ces applications une interface utilisateur et un système de base de données spécifique étaient développés, chaque application possédait donc ses propres routines pour interagir avec l'utilisateur, stocker et récupérer les données. Dans les années soixante-dix, le développement des Système de Gestion de Bases de Données (SGBD) a permis d'extraire les données des applications. En utilisant les SGBD, les applications ont ainsi été libérées du fardeau de la gestion de données. Dans les années quatre vingts,

l'apparition de systèmes de gestion d'interface utilisateur User Interface Management Systems (UIMS) a permis aux développeurs d'application d'extraire l'interaction avec les utilisateurs des applications. Enfin, les années quatre-vingt-dix sont marquées par l'apparition de logiciels de Workflow, permettant aux développeurs d'application d'extraire les procédures de travail des applications. La Figure 1.5 fait apparaître le système de gestion de Workflow comme une composante générique pour représenter et manipuler les processus d'entreprise¹.

Ainsi, à l'heure actuelle, beaucoup d'organisations commencent à considérer l'utilité d'outils avancés pour soutenir la conception et l'exécution de leurs processus d'entreprise.

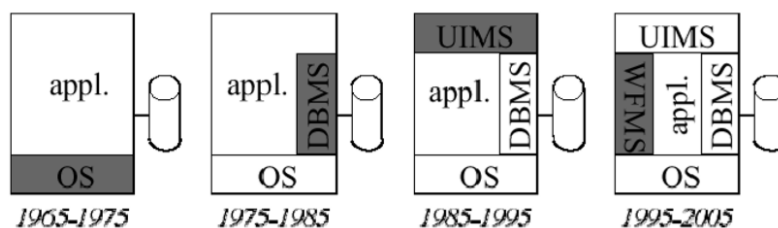


FIGURE 1.5 – Les systèmes de gestion de Workflow dans une perspective historique

1.3.3 Définitions et terminologies

Les définitions sont, pour la majorité, issues de la Coalition de Gestion de Workflow Workflow Management Coalition (WfMC). La WfMC a été fondée en 1993 par un regroupement d'industriels de l'informatique, de chercheurs et d'utilisateurs, associée à l'essor du développement des Workflows. Cette coalition a pour but de promouvoir les Workflow et d'établir des standards pour les Workflow Management System (WFMS). Elle a en particulier publié un glossaire de référence contenant les terminologies employées dans ce domaine [WfMC03 95 ; WfMC11 99]. Ces standards servent notamment à résoudre les problèmes d'interopérabilité entre systèmes Workflow mais également à définir les caractéristiques fondamentales de ces systèmes. Les documents publiés par la WfMC, qui couvrent plusieurs aspects, peuvent être considérés comme des références en la matière.

1.3.4 Définitions de base du Workflow

Le sens du mot Workflow peut varier en fonction du contexte. Pour plus de clarté, les définitions les plus communément admises sur les concepts et les termes du Workflow sont rappelées ci dessous. Ces définitions sont principalement issues du Workflow Management Coalition Terminology and Glossary (WfMC-TC)-1011 [WfMC11 99], dont il existe une traduction à usage francophone [WfMC03f 98]. L'idée première du Workflow est donc de séparer les processus, les ressources et les applications, afin de se recentrer sur la logistique des processus travail et non pas sur le contenu des tâches individuelles. Un Workflow est donc le lien entre ces trois domaines comme précise la Figure 1.6.

1. Les procédures d'entreprise représentent l'organisation et la politique de l'entreprise pour atteindre certains objectifs [WfMC11 99]

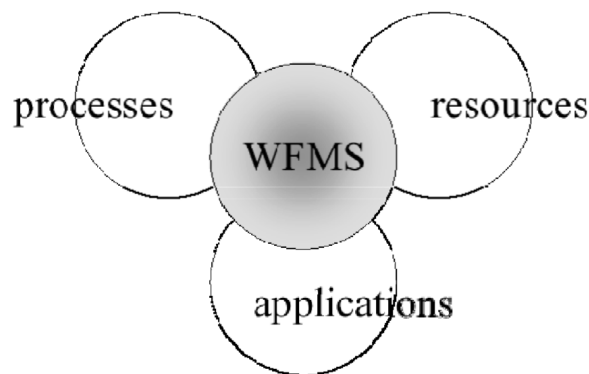


FIGURE 1.6 – Environnement du système Workflow

1.3.4.1 Définition d'un Workflow

Le Workflow est une technologie informatique ayant pour objectif la gestion des processus d'organisations ou d'entreprises : les termes suivants sont également employés pour qualifier cette technologie « Système de Gestion Electronique de Processus », « Gestion de Workflow » ou « Gestion de processus » [COURTOIS 96].

Le Workflow est l'ensemble des moyens mis en œuvre pour automatiser et gérer les processus d'une organisation. Cette gestion est rendue possible par la représentation sous forme d'un modèle, de tout ou partie des processus considérés. Le Workflow doit ensuite transcrire les modèles obtenus en une forme exécutable. Enfin, ces modèles sont exécutés et gérés. Il est ainsi possible de suivre l'évolution de leur état au fil du temps. La gestion de processus inclut également, au cours de l'exécution, la coordination et la synchronisation des différents acteurs des processus en fonction de l'état actuel des modèles.

Pour résumer, la Gestion de Processus permet donc d'attribuer à chacun et au bon moment, les tâches dont il a la responsabilité et de mettre à disposition les applications, les outils et les informations nécessaires pour leurs réalisations. Dans un contexte d'acteurs humains, le Workflow permet de décharger les acteurs de certaines tâches de gestion administrative, en leur laissant la possibilité de se concentrer sur les contenus des tâches techniques en rapport avec leurs compétences. De plus, le Workflow donne la possibilité d'effectuer une activité de monitoring sur le déroulement des Workflow de l'entreprise, permettant en particulier de connaître, en fonction de la date, l'état des activités, des acteurs, des applications et quelles sont les prochaines activités planifiées.

En synthèse, La WfMC présente le Workflow comme l'automatisation d'un processus d'entreprise, en intégralité ou en partie, pendant laquelle on définit les transmissions des documents, de l'information ou des tâches d'un participant à un autre pour agir, selon un jeu de règles procédurales [WfMC11 99]. Un Système Workflow définit, gère et exécute des procédures en exécutant des programmes dont l'ordre d'exécution est prédéfini dans une représentation informatique de la logique de ces procédures - les Workflow [WfMC11 99].

1.3.4.2 Dimensions

Associé aux définitions précédentes, il a été défini une représentation tridimensionnelle des systèmes Workflow [VAN DER AALST 98a] (Figure 1.7).

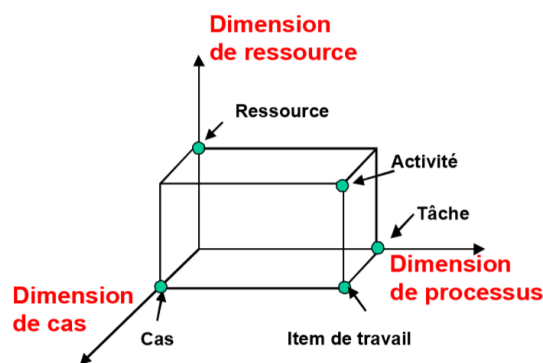


FIGURE 1.7 – Dimensions des systèmes Workflow

La Figure 1.7 donne une représentation tridimensionnelle d'un Workflow : avec une dimension de cas, une dimension de processus et une dimension de ressource. La dimension de cas représente l'instanciation du Workflow, chaque cas est un modèle à simuler et traiter individuellement. Les cas ne s'influencent donc pas directement. Ils peuvent éventuellement s'influencer indirectement via les ressources et les données. Le processus de Workflow est spécifié dans la dimension de processus, c'est-à-dire, la définition des tâches et de leur enchaînement. Enfin, dans la dimension de ressource, les ressources sont regroupées en rôles et en unités organisationnelles. En résumé, nous pouvons donc visualiser un certain nombre de termes Workflow dans la vue tridimensionnelle de la Figure 16. En particulier, une entité de travail est définie par la coordonnée cas + tâche et une activité par le cas + la tâche + la ressource. Cette représentation met en relief la gestion de Workflow comme le lien entre les cas, les tâches et l'organisation.

1.3.5 Description des Systèmes Workflow

Après avoir présenté la terminologie Workflow, il est maintenant possible de présenter le principe de fonctionnement des systèmes Workflow. A un haut niveau, ce type de système est composé de trois parties fonctionnelles [WFMC03 95]. La Figure 1.8 illustre un Système de gestion de Workflow et les relations entre ses différentes fonctions.

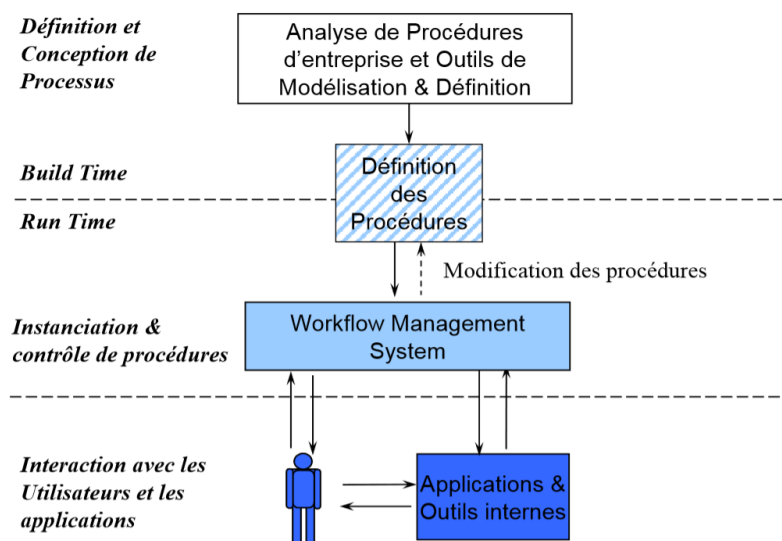


FIGURE 1.8 – Caractéristiques du système Workflow

1.3.5.1 Build time

Cette première phase permet la définition et la modélisation des procédures Workflow, elle est nommée build time. Elle est principalement composée d'un outil permettant la modélisation des Workflow dans un formalisme existant ou propriétaire à partir d'une analyse de procédures d'entreprises, il est à noter que les outils actuels préfèrent une description graphique. Un deuxième composant transcrit les modèles obtenus dans une définition des procédures « exécutable », c'est à dire compréhensibles par la partie chargée de l'exécution (simulateur du run time). Certains systèmes permettent en retour de la partie run time des modifications dynamiques de la structure de définition de procédures comme indiqué sur la Figure 1.8.

1.3.5.2 Run time

L'environnement d'exécution (ou run time) a l'entière gestion des modèles de Workflow établis dans la première partie. Cette gestion comprend l'exécution des Workflow avec un simulateur mais aussi la distribution des tâches aux rôles appropriés en cours d'exécution, la mise à disposition de l'ensemble des données et outils nécessaires, la supervision et le contrôle, etc.

Plus en détails, cette partie se décline en sous composants, introduits ci-dessous : Le Moteur de Workflow (Workflow Engine) qui est chargé de la Gestion des Procédures à travers la simulation de leurs évolutions.

Le gestionnaire des listes de tâches (Worklist Handler), chargé de distribuer les activités dans les listes des acteurs en fonction de leurs rôles.

Les listes de tâches (Worklists), qui sont les listes associées à chaque rôle dans lesquelles le moteur place les tâches à réaliser. Les tâches sont classées par priorité ou par date avec les données et les outils à utiliser de façon appropriée.

Les outils d'administration et de contrôle (Workflow Process Monitoring) suivent le déroulement des procédures Workflow, elles peuvent fournir l'état actuel des composants du Workflow et donner l'ordre de modifier le modèle de procédures.

1.3.5.3 Utilisateurs, outils et applications

La troisième phase concerne l'ensemble des outils et des fonctions d'API (Application Programming Interface) qui permettent au système Workflow de s'interfacer avec des ressources humaines, des applications informatiques extérieures et d'autres systèmes Workflow.

1.3.6 Concepts et Terminologie Workflow fondamentaux

Les principaux termes associés aux Workflow proposés par la WfMC [WfMC11 99] sont présentés dans le diagramme du méta modèle Workflow ci-dessus, ce diagramme permet également de mettre en évidence leurs interrelations. Les termes présentés ci-dessous en français avec la traduction anglaise originale associée, couvrent les notions plus importantes appartenant au Workflow et à son lexique [WfMC11 99].

1.3.6.1 Procédure Workflow (Workflow Process)

Une procédure Workflow est une procédure contrôlée par un Workflow. Une procédure est composée de plusieurs activités enchaînées pour représenter un flux de travail. Une procédure possède une structure hiérarchique et modulaire, en l'occurrence une procédure peut donc être composée de sous procédures et d'activités. Les sous-procédures peuvent être composées elles mêmes de procédures manuelles ou de procédures Workflow.

1.3.6.2 Activité (Process Activity)

Une activité est une étape d'un processus au cours de laquelle une action élémentaire est exécutée. On désigne par « action élémentaire » (ou tâche) une activité qui n'est plus décomposable en sous-procédures. La WfMC distingue une « activité manuelle », qui n'est pas contrôlée par le système Workflow, et une « activité Workflow » qui est sous le contrôle du Workflow. Un exemple d'une activité manuelle est l'ouverture d'un courrier. Une activité Workflow peut être le remplissage d'un formulaire électronique. Il existe donc des exemples d'activités manuelles intégrables dans un Workflow.

[VAN DER AALST 98a] présente l'activité Workflow comme l'intersection entre une ressource humaine ou matérielle et un bon de travail dans le cadre de l'exécution d'une tâche. Dans cette représentation, une ressource du modèle organisationnel est donc exigée pour qu'une tâche puisse être instanciée en activité et allouée à un participant de Workflow.

1.3.6.3 Acteur, Ressource (Workflow Participant)

Un acteur est une entité du modèle organisationnel participant à l'accomplissement d'une procédure. L'acteur est chargé de réaliser les activités qui lui sont attribuées via le(s) rôle(s) qui lui sont définis dans le modèle organisationnel. Les autres dénominations courantes dans la littérature de cette entité sont « ressource », « agent », « participant » ou « utilisateur ». L'acteur peut être une ressource humaine ou matérielle (machine, périphérique informatique...).

Les ressources sont organisées en classes dans le modèle organisationnel. Ces classes sont des groupes de ressources possédant des propriétés communes. Une classe est basée sur :

Rôle : défini ci dans le § suivant.

Groupe : cette classification est basée sur l'organisation (département, équipe, unité).

1.3.6.4 Rôle (Role)

Un rôle décrit en général les compétences d'un acteur dans le processus ou sa position dans l'organisation. Un rôle est associé à la réalisation d'une ou de plusieurs activités. Plusieurs acteurs peuvent tenir un même rôle. La WfMC distingue deux types de rôles [WfMC11 99] :

Les rôles organisationnels définissent un ensemble de compétences qu'un acteur possède. Ce rôle définit la position de l'acteur dans une organisation. Les rôles procéduraux définissent une liste d'activités qu'un acteur est en capacité d'exécuter.

Il est à noter que certains travaux ne différencient pas les notions d'acteur et de rôle et ne parlent que d'acteur. Cette opinion semble restreindre la clarté et la flexibilité des modèles Workflow.

1.3.6.5 Données (Workflow Relevant Data)

Une donnée pertinente pour les procédures est une information en rapport avec la réalisation des activités (en définition de la tâche, en entrée ou en sortie). Elle peut constituer l'objectif d'une tâche (manipulation de la donnée et définition de l'état de la procédure), être un élément essentiel pour activer les transitions d'état d'une instance Workflow ou être généré par la tâche et ainsi intervenir dans la détermination de la prochaine activité à déclencher. Ces données sont en général des objets au sens purement informatique mais peuvent également être une représentation d'objets physiques.

Notons qu'il existe deux autres types de données utilisées hors de la gestion de procédures :

Donnée de contrôle (Control Data) : données gérées et utilisées par le système Workflow et les moteurs Workflow.

Données Applicatives (Applicative Data) : données propres aux applications, le système de gestion de Workflow n'y a pas accès.

1.3.6.6 Application externe (Invoked Application)

Une application externe est une application informatique dont l'invocation est nécessaire à la réalisation de la tâche ou à l'exploitation des résultats générés avant de déclencher la tâche suivante ou de recommencer cette première. On tiendra compte de l'allocation de ressources, si l'application n'est pas uniquement informatique. Il faut différencier les outils (Tools), qui sont eux directement interfacés par le système Workflow, sans l'intervention d'une ressource du Système Workflow.

1.4 Classification des systèmes Workflow

Il n'existe pas de classification commune des systèmes Workflow dans la littérature, reconnue par l'ensemble de la communauté Workflow [VAN DER AALST 02]. Ceci étant essentiellement dû au nombre important de critères de classification qu'il est possible de retenir. En effet, les spécialistes adoptent différents points de vue par rapport à la notion de Workflow, les critères qui en découlent varient donc en fonction de leurs perceptions des caractéristiques présentées par ces systèmes Workflow. Ainsi, il existe plusieurs classifications, permettant de sélectionner un outil de gestion de Workflow avec différents « éclairages » sur le sujet.

Malgré ce manque d'unité, la classification proposée par [McCREADY 92] est assez répandue dans la littérature, elle est reprise par bon nombre d'auteurs [VAN DER AALST 98a], [GEORGAKOPOULOS 95]. Elle propose de distinguer quatre catégories de Systèmes Workflow. La Figure fig :classification-de-workflow présente ces différentes classes selon deux axes : Approche et Structure.

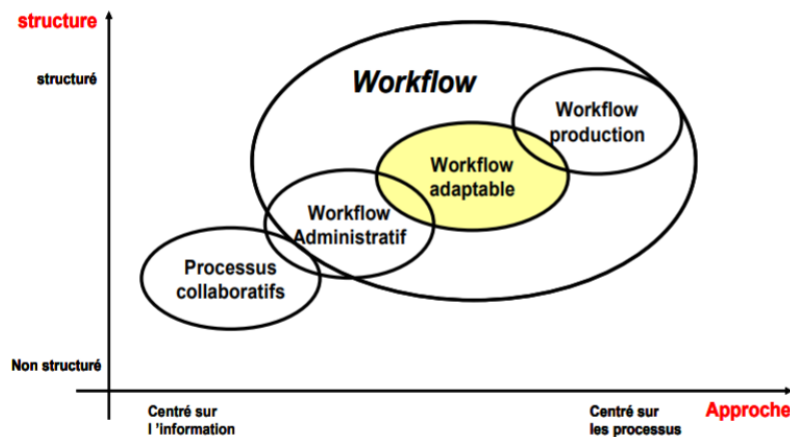


FIGURE 1.9 – Différentes classes des systèmes Workflow

1.4.1 Processus collaboratifs

Cette première classe est axée sur la communication et sur le partage d'information. Les systèmes collaboratifs sont définis pour supporter le travail en groupe, dans le cadre de la conception, de la gestion de projet ou de la résolution de problèmes faisant appel à plusieurs niveaux d'expertise. Ces systèmes permettent de réunir les intervenants d'un projet autour d'un objectif commun, les clients de la procédure y étant souvent eux-mêmes directement associés, les logiciels employés sont le plus souvent des groupwares². Les tâches des procédures gérées sont le plus souvent complexes et leur réalisation implique l'intervention de ressources aux compétences très spécifiques pour une forte valeur ajoutée. D'un autre côté, l'enchaînement des activités des procédures à traiter est faiblement structuré et peu répétitif. De part la faible structure de ces processus, ils ne font pas partie ou se situe à la frontière de ce que l'on considère comme la « sphère » Workflow comme le précise la Figure 1.9.

2. Collecticiel : classe de logiciels prévus pour être exploités de façon concurrente, sur un même projet.

1.4.2 Workflow administratif

Les systèmes Workflow administratifs (General Purpose Workflow Management Systems) ont pour objectif de décharger les ressources d'une entreprise des tâches administratives. En effet ces procédures sont répétitives, fortement prédictibles et les règles d'enchaînement des tâches sont très simples et clairement définies; ces procédures sont donc aisément automatisables, évitant ainsi un travail fastidieux où peuvent naître des erreurs souvent humaines. Les systèmes Workflow administratifs permettent de lier à une tâche administrative, les documents et les informations nécessaires à la réalisation de cette tâche par un acteur humain. Ces systèmes gèrent également le routage des documents et le remplissage de formulaires. La gestion par Workflow de procédures administratives permet un gain de l'ordre de 5% à 10% en termes de productivité et de 30 à 90% en termes de délais [ADER 99]. Enfin une dernière raison de l'automatisation de ce type de procédures en Workflow provient du fait que ces procédures possèdent une structure statique et ne sont donc pas souvent assujetties à modifications car elles possèdent une longue durée d'utilisation [SCHEER 97].

1.4.3 Workflow de production

Les systèmes Workflow de production impliquent des procédures prévisibles et assez répétitives. Leurs principales différences avec les Workflow administratifs résident dans la complexité des tâches et de la structure des procédures, dans leur capacité à faire appel à des informations provenant de systèmes d'information variés et dans l'enjeu que représente leur réussite. En effet, la procédure Workflow correspond directement au travail effectué par l'entreprise. En d'autres termes, la performance de l'entreprise est directement liée à l'exécution de la procédure managée par le Workflow. On dit dans ce cas qu'il est mission critical ³ **Un système est dit critique lorsque : les vies de personnes sont tributaires de son fonctionnement ou le coût économique d'un dysfonctionnement est catastrophique.** [INCONCERT 97]. C'est par exemple le cas des organismes financiers, des compagnies d'assurances, des usines de production manufacturières. La réalisation des procédures est donc associée à une forte valeur ajoutée et un volume d'informations traitées important. La complexité des procédures traitées est également due à la répartition de leurs activités sur plusieurs sites. Dans ce cas, les tâches exécutées nécessitent souvent l'interrogation de plusieurs systèmes informatiques, hétérogènes et distribués. Il est donc nécessaire que les systèmes Workflow de production fournissent un ensemble d'outils ou de fonctions d'API permettant de se connecter à plusieurs systèmes. Enfin, même si les procédures traitées sont assez répétitives, elles sont susceptibles d'être modifiées plus souvent que les procédures administratives, car associées à la modification des objectifs du métier. Ces modifications peuvent par exemple avoir lieu dans le cadre d'une restructuration de BPR29 ou d'un CPI30.

Les systèmes Workflow de production doivent donc pouvoir évoluer. Par ailleurs, l'exécution de certaines procédures ne peut pas toujours se poursuivre de manière automatisée, suite à l'occurrence d'un ou de plusieurs événements qui font aboutir le système dans un état particulier. Dans ce cas, il est nécessaire de faire intervenir des acteurs humains pour la prise de décision. Pour ce faire, le système Workflow

3.

de production peut faire appel à un autre système, de type collecticiel ou un autre système Workflow ad hoc, qui servira d'interface pour l'exécution dirigée par un acteur humain de la suite de la procédure. Ce type de Workflow est dit « composite » [EDER 96]. Enfin, dans la littérature, les systèmes Workflow de production sont également appelés case-based [VAN DER AALST 98a].

1.4.4 Workflow adaptable ou Workflow ad hoc

L'impossibilité pour les systèmes de gestion de Workflow traditionnels de traiter les différents changements dynamiques dans les flux de travail est une limite à dépasser. A ce titre, il a été introduit les concepts de Workflow adaptable (adaptive Workflow) [VAN DER AALST 98c] et de Workflow ad hoc [VOORHOEVE 97]. La nuance entre ces deux nouveaux termes provient du fait qu'ad hoc désigne un acte spécialement fait pour un objet déterminé alors qu'adaptable prévoit un changement définitif de la procédure.

Les Workflow ad hoc se situent à la frontière gauche de la représentation Figure 1.9 dans la « sphère » Workflow adaptable. Ils régissent des procédures dont la structure est déterminée pendant l'exécution en fonction des décisions humaines prises suite à la réalisation d'une tâche, plus concrètement, la structure se construit par pas en suivant le rythme de l'exécution. En effet, la réalisation d'une procédure non structurée peut impliquer à chaque fois l'exécution d'un nouvel enchaînement des tâches, voire la création de nouvelles tâches. Il n'y a pas a priori de persistance de l'enchaînement de ces tâches.

Les Workflow adaptables sont, quant à eux, des supports comparables aux Workflow de production classiques possédant une structure préétablie, mais pouvant traiter certains changements de structure « en ligne ». Ces changements peuvent aller des changements individuels/ad hoc (gestion d'exception), c'est à dire d'un aiguillage pour déterminer l'activité suivante, jusqu'à la reconception par Business Process Reengineering (BPR) de processus [VAN DER AALST 98d]. En conclusion, ils ont une action globale pouvant inclure la définition du Workflow ad hoc

Il est intéressant de classifier les différents changements possibles par un Workflow adaptables, dans le but de mieux les anticiper [SADIQ 99]. Les changements sont envisageables selon plusieurs perspectives : la ressource, le contrôle, la procédure, la tâche et le système [VAN DER AALST 98d].

1.4.5 Modèle de référence des systèmes Workflow

Le modèle de référence, Figure 18, présente l'architecture générale de l'environnement proposée par la WfMC, il identifie les interfaces couvrant cinq domaines de fonctionnalités entre le système Workflow et son environnement.

1.4.5.1 Interface avec les Outils de définition de procédures

Cette interface, située entre les outils de modélisation/définition et le logiciel de gestion du Workflow pendant l'exécution, est nommée interface d'import/export de définition de processus. Cette interface définit le format d'échange et d'appels des APIs, qui permettent l'échange d'informations de définition de procédures sur une variété de médias d'échange : physiques ou électroniques. Cette interface permet l'échange d'une définition de processus complète ou d'un sous-ensemble. Par

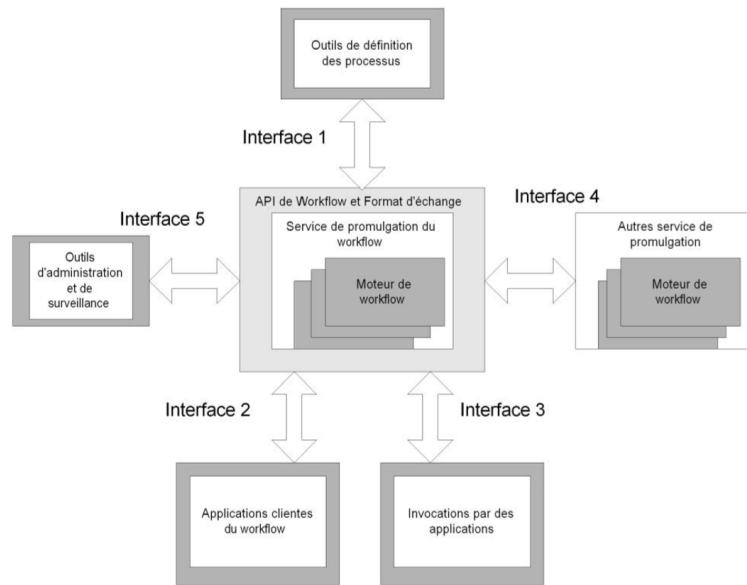


FIGURE 1.10 – Modèle de référence des systèmes de gestion de workflow (WfMC, 95).

exemple le changement de définition d'un ensemble de procédures ou plus simplement la modification des attributs d'une activité particulière dans une définition de procédures.

1.4.5.2 Interface avec les applications clientes Workflow

La liste des tâches (Worklist) à exécuter par une ressource est généralement définie et gérée par le service d'exécution du Workflow. Cette liste doit pouvoir déclencher des appels à des applications clientes diverses et des ressources. La solution retenue pour respecter la susdite exigence, consiste à encapsuler la variété d'application qui peut être utilisée derrière un jeu standard d'API le Workflow Application Programming interface (WAPI). Ce jeu permet ainsi d'utiliser une communication standardisée entre les applications clientes, le moteur de Workflow et les Worklist, indifféremment de la nature de l'implémentation réelle des produits clients.

1.4.5.3 Interface avec les applications invoquées

Il est évident que le système Workflow ne peut pas intégrer l'invocation automatique de toutes les applications qu'il peut être amené à utiliser pendant l'exécution d'un Workflow. Par exemple les applications dont les données sont fortement typées. Dans ce cas un composant externe supplémentaire, nommé agent d'application, est ajouté, il est chargé de la traduction des informations dans un format compréhensible par le standard WAPI.

Dans le cas le plus simple, l'invocation d'application est traitée localement par un moteur de Workflow, mais les applications invoquées peuvent être utilisées par plusieurs moteurs de Workflow et peuvent se situer sur des machines distantes, il convient donc de définir un format commun d'utilisation des ces applications entre les Workflow dans le but de communiquer correctement et de synchroniser l'appel à

ces applications.

1.4.5.4 Interface avec les autres Workflow

Un des objectifs de la normalisation dans la définition de Workflow est de pouvoir transmettre des WorkItem entre deux systèmes Workflow conçus par des concepteurs de systèmes Workflow différents. Trois principaux types d'interopérabilité ont été identifiés :

Workflow chaînés : La dernière activité d'un Workflow A doit pouvoir fournir un item à la première activité d'un Workflow B.

Workflow hiérarchiques : une activité d'un Workflow A doit pouvoir être vu comme un Workflow B.

Workflow Peer to Peer : Une procédure globale est composée d'activités gérées en partie par un Workflow et en partie par un autre Workflow, sans système de supervision de la procédure complète.

Workflow Synchronisés : Deux Workflow s'exécutent en parallèle et doivent pouvoir se synchroniser sur certaines activités.

Pour résumer, il est possible d'identifier deux aspects principaux nécessaires à l'inter fonctionnement de Workflow :

- L'interprétation commune de la définition de procédures (ou d'un sous-ensemble).
- L'appui pendant l'exécution de l'échange des divers types d'information de contrôle et le transfert des données appropriées et/ou d'applications entre les services d'exécution Workflow différents.

1.4.5.5 Interface avec les outils de contrôle et d'administration

L'objectif de cette interface est de permettre à un logiciel de Monitoring de Workflow de s'interfacer avec plusieurs Workflow différents et ainsi regrouper la supervision d'un ensemble de systèmes Workflow dans un logiciel.

L'interface 5 permet à une application de gestion indépendante d'interagir avec des Workflow de différents domaines. L'application de gestion peut aussi se charger d'autres fonctions de gestion, au-delà de celles-ci. Par exemple, elle peut aussi gérer des définitions de procédures de Workflow, agissant comme un dépôt d'information commun à plusieurs systèmes et distribuant des définitions de processus aux divers Workflow via des opérations au travers de leurs interfaces 1. Malgré cela, des scénarii d'implémentations moins modulaires sont aussi envisageables ; par exemple l'application de gestion peut être une partie intégrante du service d'exécution.

1.5 Intérêt du cloud pour les workflows

Les clouds offrent plusieurs avantages pour les applications à base de workflows. Ces avantages facilitent :

1.5.1 L'approvisionnement de ressources

Dans les grilles, l'ordonnancement est basé sur un modèle en best-effort, dans lequel l'utilisateur spécifie la quantité de temps nécessaire et délègue la responsabilité de l'allocation des ressources et d'ordonnancement de tâches à un ordonnanceur

fonctionnant en mode batch utilisant des files d'attentes. Dans le cloud, au lieu de déléguer l'allocation au gestionnaire de ressources, l'utilisateur peut provisionner les ressources nécessaires et ordonnancer les tâches en utilisant un ordonnanceur contrôlé par l'utilisateur. Ce modèle d'approvisionnement est idéal pour les workflows, car il permet au système de gestion de workflow d'allouer une ressource une seule fois et de l'utiliser pour exécuter de nombreuses tâches.

1.5.2 L'allocation dynamique de ressources à la demande

Contrairement aux grilles, les clouds donnent l'illusion que les ressources informatiques disponibles sont illimitées. Cela signifie que les utilisateurs peuvent demander, et s'attendre à obtenir des ressources suffisantes pour leurs besoins, à tout moment. L'approvisionnement à la demande est idéal pour les workflows et d'autres applications faiblement couplées, car il réduit le surcoût (overheads) d'ordonnancement total et peut améliorer considérablement les performances du workflow (Singh, 2005 ; Juve, 2008)

1.5.3 L'élasticité

Outre l'approvisionnement des ressources à la demande, les clouds permettent aussi aux utilisateurs de libérer des ressources à la demande. La nature élastique de clouds facilite le changement des quantités et des caractéristiques de ressources lors de l'exécution, permettant ainsi d'augmenter le nombre de ressources, quand il y a un grand besoin, et d'en diminuer, lorsque la demande est faible. Cela permet aux systèmes de gestion de workflow de répondre facilement aux exigences de qualité de service (QoS) des applications, contrairement à l'approche traditionnelle, qui nécessite de réserver à l'avance des ressources dans les environnements de grilles.

1.5.4 La garantie des QoS via des SLA

Avec l'arrivée des services de cloud computing provenant de grandes organisations commerciales, les accords de niveau de service (SLA) ont été une préoccupation importante pour les fournisseurs et les utilisateurs. En raison de compétitions entre les fournisseurs de services émergents, un grand soin est pris lors de la conception du SLA qui vise à offrir (i) de meilleures garanties de QoS aux utilisateurs, et (ii) des termes clairs pour l'indemnisation, en cas de violation du contrat. Cela permet aux systèmes de gestion de workflow de fournir de meilleures garanties de bout en bout en "mappant" les utilisateurs aux fournisseurs de services selon les caractéristiques des SLA.

1.5.5 Le faible Coût d'exploitation

Économiquement motivés, les fournisseurs de cloud commercial s'efforcent d'offrir de meilleures garanties de services par rapport aux fournisseurs de grille. Les fournisseurs de cloud profitent également des économies d'échelle, en fournissant des ressources de calcul, de stockage et de bande passante, à un coût très faible grâce, à la virtualisation. Ainsi l'utilisation des services de cloud public pourrait être économique et une alternative moins coûteuse, par rapport à l'utilisation de

ressources dédiées, qui sont plus chères. Un des avantages de l'utilisation des ressources virtuelles pour l'exécution de workflow, plutôt que d'un accès direct à la machine physique, est le besoin réduit pour sécuriser les ressources physiques des codes malveillants. Cependant, l'effet à long terme de l'utilisation de ressources virtuelles dans les clouds qui partagent efficacement une "tranche" de la machine physique, plutôt que d'utiliser des ressources dédiées pour les workflows de calculs intensifs, est une question de recherche intéressante.

1.6 Conclusion

....lk

Chapitre 2

Réseau de Petri

Introduction

Les Réseau de Pétri représentent un outil mathématique puissant dans le domaine de la modélisation et de la vérification des systèmes. En plus de leur force d'analyse, ils offrent une représentation graphique simple qui aide à la modélisation des systèmes complexes.

Donc Le modèle des réseaux de Pétri est un outil graphique de modélisation et d'analyse des systèmes parfaitement adapté à l'étude des structures de contrôle. Il permet notamment de maîtriser et d'assurer la sûreté de fonctionnement de logiciels complexes (aéronautique, transports, industrie...).

Le formalisme formel des Réseau de Pétri (RdP), adapté à la prise en compte des problèmes de concurrence, de synchronisme et de parallélisme, constitue un excellent outil de spécification fonctionnelle d'un problème et de mise en évidence des contraintes. Les propriétés mathématiques qui découlent de l'analyse des RdPs permettent une étude comportementale et structurelle essentielle à la validation d'une spécification. Les possibilités de simulation offertes par les outils informatiques supportant le formalisme contribuent également à cette validation.

En général, les méthodes de l'étude de systèmes par réseau de pétri se composent de trois étapes : premièrement on écrit le système en termes de réseau, pour obtenir un modèle en réseau ; deuxièmement on analyse le modèle obtenu, pour en déduire des propriétés comme l'absence de blocage, existence d'une solution, etc. Finalement, on fait la révision des propriétés obtenues pour montrer si le système est bon. Le résultat de cette méthode nous indique une analyse qualitative du système. Elle constitue une approche très importante pour avoir une bonne évaluation des systèmes.

2.1 Réseau de Pétri

2.1.1 Qu'est-ce que les réseaux de Pétri

Les réseaux de Pétri sont définis comme étant un formalisme qui permet la description et l'analyse du comportement des systèmes concurrents, introduit par Carl Adam Pétri en 1962. [1] Les définitions concernant les réseaux de Pétri portaient sur deux aspects :

— **Un aspect structurel**

Quelles sont les actions, quels sont les sites, quelles sont les conditions pour qu'une action soit possible et quelles sont les conséquences d'une action ?

— **Un aspect comportemental**

Comment représenter le fonctionnement d'un réseau de Pétri ? C.-à-d. ce qui se passe quand une action ou plusieurs actions sont exécutées.

2.1.2 L'aspect structurel

Définition 2.1.1. (Définition d'un réseau de pétri :

Un réseau de pétri (R) est un triple $R = (P, T, W)$ où P est l'ensemble des places (les places représentent les sites) et T l'ensemble des transitions (les transitions représentent les actions) tel que :

- P est un ensemble final de places,
- T est un ensemble final de transition ($P \cap T = \emptyset$),
- $W : (P \times T) \cup (T \times P) \rightarrow N = \{0, 1, 2, \dots\}$.

2.1.2.1 Représentation d'un réseau de Pétri

Représentation graphique

L'un des aspects les plus agréables des réseaux de Pétri est qu'il est extrêmement aisé de les visualiser ; c.-à-d., donner une interprétation graphique à sa structure qui peut être représentée à travers un graphe bipartite fait de deux types de sommets : les places et les transitions reliées alternativement par des arcs orientés qui portent des poids entiers positifs, si un poids n'est pas porté alors il est égal à 1 (RdP ordinaire). Généralement, les places sont représentées par des cercles et les transitions par des rectangles, le marquage d'un RdP est représenté par la distribution de jetons dans l'ensemble de ses places telle que chaque place peut contenir un ou plusieurs jetons représentés par des points dans le cercle représentant la place. [2]

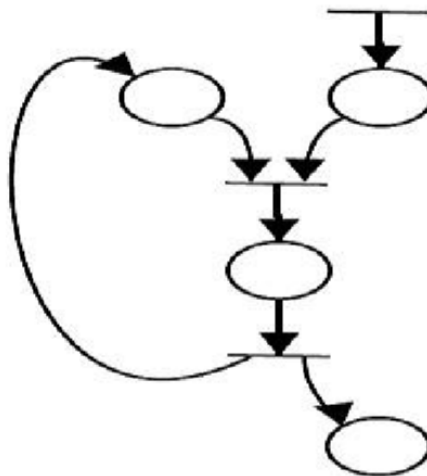


FIGURE 2.1 – Réseau de Pétri simple

Représentation matricielle

Une représentation matricielle d'un RdP est offerte afin de simplifier les Tâches d'analyse et de vérification effectuée sur un modèle RdP. Agir sur une représentation graphique d'un modèle RdP est une Tâche délicate en comparant avec une représentation matricielle. Il est possible de représenter la fonction W (fonction de poids) par des matrices.[2]

Définition 2.1.2. (Réseau de pétéri) : Soit Un réseau de pétéri $R = (P, T, W)$ avec $P = \{p1, p2, \dots, pm\}$ et $T = \{t1, t2, \dots, tn\}$, on appelle matrice des pré conditions pré la matrice $m \times n$ à coefficients dans N tel que $pre(i, j) = W(p_i, t_j)$, elle indique le nombre de marques que doit contenir la place p_i pour que la transition t_j devienne franchissable, de la même manière on définit la matrice des post conditions post la matrice $n \times m$ tel que $post(i, j) = W(t_j, p_i)$ contient le nombre de marques déposées dans p_i lors du franchissement de la transition t_j . La matrice $C = post - pré$ est appelée matrice d'incidence du réseau (m représente le nombre de places d'un réseau de Pétri et n le nombre de transitions.).

La représentation Graphique d'un marquage dans un RdP marqué est présentée par des marques dans la place appelées jetons.

Le marquage d'un réseau de Petri est représenté par un vecteur de dimension m à coefficients dans N . La règle de franchissement d'un réseau de Petri est définie par : $M'(p) = M(p) + C(p, t)$.

Représentation d'un RdP marqué

Un réseau de Pétri marqué est le couple $N = \langle R, M \rangle$ où :

- R est un réseau de Pétri .
- M est une application de marquage .
- $M : P \rightarrow N$.

$M(p)$ est le nombre de marques (jetons) contenus dans la place p . Le marquage d'un réseau de Pétri est une opération qui consiste à assigner des jetons dans les places.

On appelle marquage M d'un Réseau de Pétri le vecteur du nombre de marques dans chaque place : la I^{eme} composante correspond au nombre de marques dans la I^{eme} place. Il indique à un instant donné l'état du RdP.[2]

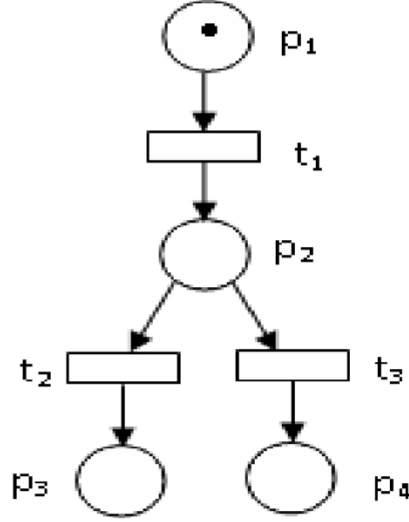


FIGURE 2.2 – Réseau de Pétri marqué

Exemple 2.1.1. Pour le réseau de la (Figure 2.2) .

$$P = \{p1, p2, p3, p4\} \text{ et } T = \{t1, t2, t3\}$$

$$pre = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

(a) La matrice pré

$$C = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(c) La matrice d'incidence

$$post = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

(b) La matrice post

$$M = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(d) Le vecteur de marquage M

Le marquage du RdP présenté à la Figure 3 est donné par :

$$M = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

On appelle marquage initial, noté M0, le marquage à l' instant initial (t = 0).

2.1.3 L'aspect comportemental

Le comportement d'un réseau de Pétri est déterminé par sa structure et par son état. Pour exprimer l'état d'un réseau de Pétri, les places peuvent contenir des jetons qui ne sont que de simples marqueurs. .[1]

2.1.3.1 L'état dans un réseau de Pétri

Dans la théorie des réseaux de Pétri, l'état d'un réseau est souvent appelé marquage du réseau qui est défini par la distribution des jetons sur les places. Le marquage d'un réseau de Pétri $R = (P, T, W)$ est défini par la fonction de marquage $M : P \rightarrow N$.

Un réseau de Pétri marqué est dénoté par $\sum = (P, T, W, M_0)$ où M_0 est le marquage initial. Le comportement d'un réseau de Pétri marqué est déterminé par ce qu'on appelle règle de franchissement.

2.1.3.2 Franchissement d'une transition

Une règle de franchissement est une simple relation de transition qui définit le changement d'état dans un réseau marqué lors de l'exécution d'une action. Afin de définir une règle de franchissement, il est nécessaire de formaliser quand le réseau peut exécuter une action : on dit qu'une transition $t \in T$ peut être franchie à partir d'un marquage M (qui représente l'état du système à un instant donné) si et seulement si chaque place d'entrée $p \in {}^*t$ de la transition t contient au moins un nombre de jetons qui est supérieur ou égal au poids de l'arc reliant cette place d'entrée p avec la transition t tel que : $M(p) \geq W(p, t) \forall p \in P$. Une règle de franchissement est définie par $M'(p) = M(p) - W(p, t) + W(t, p)$ pour tout $p \in P$, ce qui veut dire que lorsque la transition t est franchie à partir d'un marquage M , il faut saisir $W(p, t)$ jetons à partir de chaque place d'entrée à la transition t et déposer $W(t, p)$ jetons dans chaque place de sortie de la transition t ce qui permet de produire un nouveau marquage M' .

Le franchissement d'une transition t dénoté par $M[t > M'$ est dite l'occurrence de t . On dit que deux transitions t_1, t_2 (pas nécessairement distinctes) sont franchies en concurrence par un marquage M si et seulement si $M(p) \geq W(p, t_1) + W(p, t_2)$ pour toute $p \in P$. [3]

Cette vision de l'exécution concurrente de deux transitions dans un RdP est contradictoire avec celle qui impose que deux occurrences de transition sont parallèles si et seulement si : elles sont causalement indépendantes et n'ont pas une relation de conflit entre eux. Deux occurrences sont en conflit si l'un des deux peut avoir lieu mais pas toutes les deux.

2.1.3.3 L'exécution d'un réseau de Pétri

Exécution séquentielle

1. **Séquence de franchissement** : Une séquence de franchissement « s » est une suite de transitions (t_1, t_2, \dots, t_n) qui permet, à partir d'un marquage « M », de passer au marquage « M' » par le franchissement successif des transitions définissant la séquence.
2. **Marquage accessible** : Le marquage d'un Réseau de Pétri à un instant donné est une vectrice colonne dont la valeur de la i ème composante est le nombre de marques dans la place P_i à cet instant.

Le franchissement d'une transition conduit à un changement du marquage. Le passage du marquage M_k au marquage M_l par franchissement de la transition T_j est noté : $M_k(T_j) > M_l$. Le nombre de marques dans la place P_i pour

le marquage M_k est noté $M_k(P_i)$. A partir d'un même marquage, il peut être possible de franchir plusieurs transitions, menant ainsi à des marquages différents. L'ensemble des marquages accessibles à partir du marquage M_0 est l'ensemble des marquages obtenus à partir de M_0 par franchissements successifs d'une ou plusieurs transition(s). Cet ensemble est noté $A(R; M_0)$. [3]

3. **Graphe de marquage** : On peut représenter l'ensemble des marquages accessibles par un graphe si ce dernier est fini. Le graphe de marquage a comme sommet l'ensemble des marquages accessibles $A(R, M_0)$. Un arc orienté relie deux sommets M_i et M_j s'il existe une transition t franchissable permettant de passer d'un marquage à un autre $M_i[t > M_j]$. Les arcs du graphe sont étiquetés par les transitions correspondantes. [3]

Exemple 2.1.2.

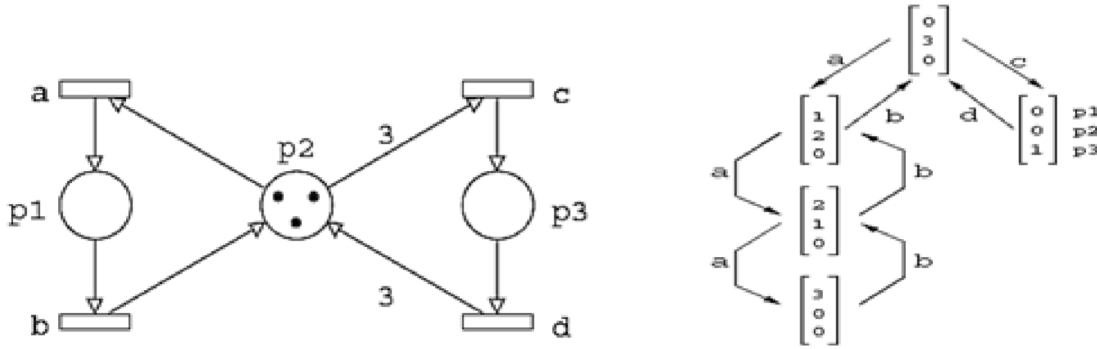


FIGURE 2.4 – Graphe de Marquage

4. L'exécution séquentielle d'un réseau de Pétri

L'exécution séquentielle d'un réseau de Pétri est définie en termes d'un ensemble de séquences d'occurrences. Une séquence d'occurrences est une séquence de transitions franchissables dénotée par $\sigma = M_0 t_1 M_1 t_2 \dots$ tel que $M_{i-1}[t_i > M_i]$. Une séquence $t_1 t_2 \dots$ est une séquence de transitions (commencée par le marquage M) si et seulement si il existe une séquence d'occurrences $M_0 t_1 M_1 \dots$ avec $M = M_0$. Si la séquence finie $t_1 t_2 \dots t_n$ conduit à un nouveau marquage M' à partir du marquage M , on écrit $M[t_1 t_2 \dots t_n > M']$ ou simplement $M[t_1 t_2 \dots t_n >]$ si on ne veut pas spécifier le marquage résultat. [3]

Exécution concurrente

Une exécution concurrente d'un réseau de Pétri est une exécution dans laquelle plusieurs transitions peuvent se franchir en même temps, elle est souvent déterminée par la notion de processus. Ceci permet de donner une interprétation de la concurrence dans un réseau de Petri selon la sémantique basée sur la vraie concurrence (sémantique d'ordre partiel) qui est interprétée dans la théorie des réseaux de Pétri par un type spécial de réseaux appelés réseaux d'occurrences. [3]

2.2 Réseaux particuliers

2.3 Propriétés des réseaux de Petri

il existe un certain nombre de propriétés qui ont été définies pour les réseaux de Petri, à savoir, le caractère borné, la réinitialisation la vivacité, la conservation, la terminaison (Diaz, 2001). Certaines de ces propriétés sont dites propriétés dynamiques car elles dépendent du marquage initial et sont liées à l'évolution du réseau, alors que d'autres sont dites propriétés statiques du fait qu'elles soient liées à la typologie du réseau et indépendantes du marquage initial.

2.3.1 Réseau K-borné

Définition 2.3.1. (Réseau de Petri borné) :

Une place P_i est bornée pour un marquage initial M_0 si pour tout marquage accessible à partir de M_0 , le nombre de jetons dans P_i reste borné. Elle est dite k-bornée si le nombre de jetons dans P_i est toujours inférieur ou égal à k. Un RdP est (k) borné si toutes les places sont (k) bornées.

Exemple 2.3.1. Un RdP peut ne pas être borné. Sur l'exemple représenté à la (figure 2.5), la transition T_1 admet la place P_1 comme unique place d'entrée. La place P_1 a un jeton : la transition T_1

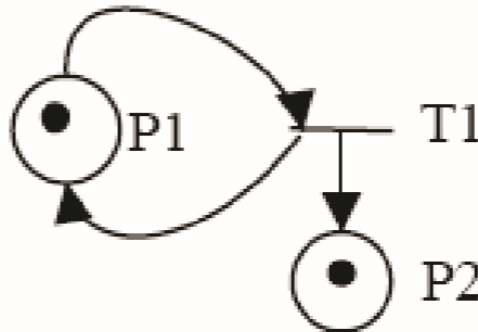


FIGURE 2.5 – Réseau de Petri non borné

est franchissable. Comme P_1 est aussi place de sortie de T_1 , le franchissement de T_1 ne change pas le marquage de P_1 . La transition T_1 est donc franchissable en permanence et peut donc être franchie un nombre de fois infini. Chaque franchissement de T_1 ajoute un jeton dans P_2 dont le marquage va donc tendre vers l'infini.

Définition 2.3.2. Réseau de Petri sauf :

Un RdP est sauf ou binaire pour un marquage initial M_0 s'il est un borné.

2.3.2 Réseau vivant

Définition 2.3.3. La vivacité :

Une transition T_j est vivante pour un marquage initial M_0 si pour tout marquage

accessible M_k , il existe une séquence de franchissement S à partir de M_k contenant $T_j : M_k \in^* M_0, \exists S, M_k | S >$ et $S = ...T_j...$

Si une transition T_j est vivante alors, à tout instant, on sait que T_j peut être franchie dans le futur. Dans le cas d'un réseau de Petri modélisant un système fonctionnant en permanence, si une transition n'est pas vivante et si une fonction du système est associée au franchissement de cette transition, ce la veut dire qu'à partir d'un certain instant, cette fonction ne sera plus disponible dans le futur, ce qui peut traduire une erreur ou une panne.

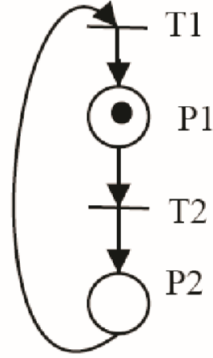


FIGURE 2.6 – Réseau de Petri vivant

Définition 2.3.4. Blocage :

Un blocage (ou état puits) est un marquage pour lequel aucune transition n'est validée.

Un réseau de Petri est dit sans blocage pour un marquage initial M_0 si aucun marquage accessible n'est un blocage.

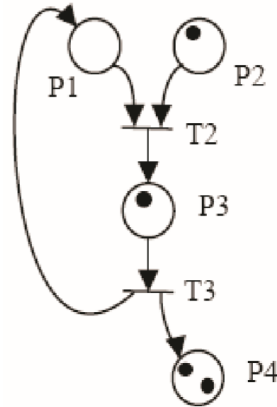


FIGURE 2.7 – Exemple de réseau de Petri bloquant

Le réseau de Petri de la (Figure 2.7), par exemple, a pour blocage le marquage : $M = (1, 0, 0, 1)$.

2.3.3 Réseau réinitialisable

Définition 2.3.5. (états d'accueil et Réseau de Petri réinitialisable)

Un RdP a un état d'accueil M_a pour un marquage initial M_0 si pour tout marquage

accessible M_k à partir de M_0 , il existe une séquence de franchissement permettant d'atteindre le marquage $M_a : \forall M_k \in {}^* M_0, \exists S_j, M_k | S_j > M_a$

Un RdP est réinitialisable pour un marquage initial M_0 si M_0 est un état d'accueil.

Si un réseau de Petri présente un état d'accueil, il est facile de vérifier s'il est sans blocage et d'étudier sa vivacité.

2.4 Réseaux de Pétri de haut niveau

Dans la pratique, on est amené à modéliser toute sorte de système, à savoir les protocoles de communication, les systèmes de production, les systèmes réactifs, les systèmes temps réel, etc. Cette variété de systèmes a poussé les chercheurs à étendre la théorie des RdPs en introduisant beaucoup de concepts relatifs à chaque domaine d'application. Ces efforts ont donné naissance aux RDPs de haut niveau tels que les RDPs colorés, hiérarchiques, temporisés et autres. [6]

2.4.1 Les réseaux de Petri colorés

Les réseaux colorés (Jensen, 1997) ont été introduits afin de modéliser des systèmes complexes tout en gardant les possibilités de vérification. Lorsque le nombre d'entités du système à modéliser est important, la taille du réseau de Petri devient rapidement énorme, et si les entités présentent des comportements similaires, l'usage des réseaux colorés permet de condenser le modèle. En effet, une couleur est une information attachée à un jeton. Cette information permet de distinguer des jetons entre eux et peut être de type quelconque. Par conséquent, une place peut contenir des jetons de différentes couleurs et une transition peut être franchie de différentes manières, selon la couleur. Ceci est réalisé en attachant un domaine de couleur à chaque place et à chaque transition. Ainsi, les arcs ne sont pas seulement étiquetés par le nombre de jetons mais aussi par leurs couleurs. Le franchissement d'une transition est alors conditionné par la présence dans les places en entrée du nombre de jetons nécessaires, qui en plus satisfont les couleurs qui étiquettent les arcs. Après le franchissement d'une transition, les jetons qui étiquettent les arcs d'entrée sont retirés des places en entrée tandis que ceux qui étiquettent les arcs de sortie sont ajoutés aux places en sortie de cette transition.

Ainsi, pour un même système, le nombre de comportements qui peuvent être exprimés par un réseau coloré est nettement plus élevé qu'avec un réseau simple. Ce sont des réseaux très adaptés aux architectures distribuées. D'autant plus qu'à tout réseau coloré correspond un réseau de Petri simple qui lui est isomorphe. Ce ci permet donc d'exploiter les mêmes techniques d'analyse que celles développées pour les réseaux simples en plus d'autres qui ont été complétées et adaptées aux réseaux colorés telle que le support de la hiérarchisation.

Définition 2.4.1. Réseau de Petri Colorés :

Un réseau de Petri Coloré (CPN) et un tuple $(\Delta, P, T, Arc, Noeud, Couleur, Grade, E, M_0)$ où :

- Δ : est un ensemble de domaines de couleurs (chaque domaine est un ensemble fini et non vide).

- *Arc* : est un ensemble fini d'arcs tel que $P \cap Arc = T \cap Arc = \emptyset$
- *Nœud* : est la fonction $Nœud, Nœud : Arc \longrightarrow P \times T \cup T \times P$.
- *Couleur* : $P \longrightarrow PowerSet(\Delta)$. *Couleur*(*p*) est la fonction couleur qui associe à chaque place un domaine de couleur.
- *Garde* : est une garde, qui fait correspondre à chaque transition une expression booléenne. Les variables de la garde appartiennent à Δ .
- *E* : est l'application qui associe à chaque arc, un élément de *Couleur*(*p*)MS où *p* est une place appartenant à l'arc. *E* indique le nombre de jetons colorés à recevoir de la place qui se trouve en entrée de la transition, et le nombre de ceux à produire dans la place qui se trouve en sortie.
- M_0 : est l'application qui associe à chaque place *p*, un élément de *Couleur*(*p*) M_S . M_0 (*p*) indique la distribution initiale des jetons colorés dans la place *p*.

De manière générale, un marquage *M* d'un réseau coloré est une application qui associe à chaque place *p*, un élément de *Couleur*(*p*)MS. *M*(*p*) est un multi-ensemble sur *Couleur*(*p*) qui indique les marques colorées présentes dans la place *p* au marquage *M*. L'état du modèle est défini par un marquage coloré.

2.4.2 Les réseaux de Pétri à Objet

Les réseaux de Pétri objet (OPN) étendent le formalisme des réseaux de Pétri colorés avec une intégration complète des propriétés orientées objet y compris l'héritage le polymorphisme et la liaison dynamique, l'orientation objet fournit des primitives de structuration puissantes permettant la modélisation des systèmes complexes [6]

2.4.3 Les réseaux de Pétri temporels

Les réseaux de Pétri temporels sont obtenus depuis des réseaux de Pétri en associant des dates min et max aux transitions.

Supposons qu'une transition *t* est devenue sensibilisée pour la dernière fois à l'instant *x*, alors elle ne peut l'être encore qu'après l'instant $x + \min(t)$ et avant l'instant $x + \max(t)$, sauf si le tir d'une autre transition a désensibilisé *t* avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle. Les réseaux de Pétri temporels expriment nativement des spécifications en «délai». Ils peuvent aussi exprimer des spécifications en «durées». Leur domaine d'application est donc large. [6]

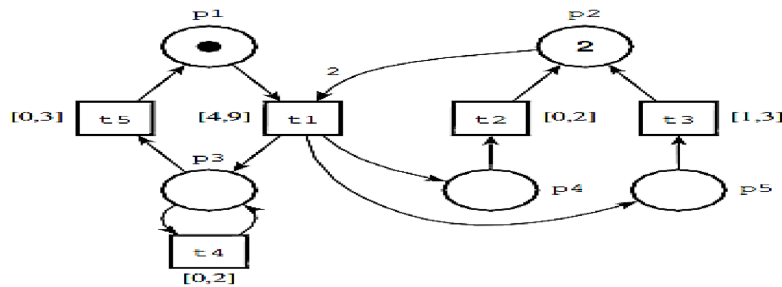


FIGURE 2.8 – Exemple de réseau de Pétri temporel

2.5 Réseaux de Petri et workflow

Les Réseaux de Petri (RdPs) constituent un formalisme majeur pour modéliser les processus workflows. Une des forces des RdPs est la base mathématique forte qu'ils offrent avec une représentation graphique. Dans cette section, nous résumons la projection topographique entre concepts de workflow et de RdP .

Un processus définit les tâches aussi bien que les conditions pour leur exécution. En utilisant les RdPs, un processus est représenté en conversant sa seule entrée (i.e., un nœud du début) dans une place sans arcs entrants, et sa seule sortie (i.e., nœud de la fin) dans une place sans arcs sortants. Les conditions sont représentées par des places, et les tâches par des transitions. Habituellement, un processus spécifié par RdP devrait accomplir deux exigences : (1) il doit être possible d'accéder à tout moment un état pour lequel il y a un jeton dans la place finale, et (2) quand il y a un jeton dans la place finale, tous les autres jetons auraient dû disparaître.

Dans un processus modélisé par un RdP, une transition active correspond à un workitem , et le tir d'une transition à une instance de l'activité. Certains work-items peuvent seulement être transformés dans une instance d'activité une fois ils sont déclenchés. Un déclencheur pourrait correspondre à une initiative du participant, à un événement externe ou à un signal du temps initié par l'environnement. A chaque transition correspondante à une tâche qui exige un déclencheur, une autre place d'entrée est ajoutée. Une occurrence du déclencheur apporte un jeton dans cette place supplémentaire. Le jeton est consommé une fois la transition appropriée est franchie. Un échec pendant l'exécution d'une tâche exige un 'rollback' (i.e., revenir à l'état antérieur au début de l'activité). Quand une activité sera complétée avec succès, des changements deviennent définitifs.

2.5.1 Modélisation de workflow avec les réseaux de pétri

Plusieurs recherches ont recommandé l'utilisation des réseaux de pétri pour la modélisation de workflow, et cela à cause de plusieurs facteurs

2.5.1.1 Une sémantique formelle

Rend la spécification de workflow non ambiguë. L'interprétation de ces spécifications peut être définie mathématiquement.

2.5.1.2 Techniques d'analyse

Ils existent plusieurs techniques d'analyse de propriétés qualitatives et quantitatives de modèle de workflow à base des réseaux de pétri, une des propriétés qualitatives est de vérifier l'occurrence d'un inter blocage en vérifiant l'accessibilité de toutes les tâches du réseau

2.5.1.3 Expressivité

Les réseaux de pétri peuvent exprimer la structure des processus ainsi que leurs dynamisme, y compris la concurrence (le parallélisme). Ils servent aussi comme langage pour la définition de la structure d'un processus, simulation de processus et la gestion de workflow

2.5.2 Workflow à base de réseaux de pétri (WF-net)

La modélisation de workflow en terme de réseaux de pétri consiste à modéliser le comportement dynamique du système en représentant les tâches sous forme de transitions, les conditions sous forme de places et les cas sont représentés par les jetons. Les réseaux de pétri qui permettent de modéliser les workflows sont appelés WF-net (réseaux de workflow).

Définition 2.5.1. (Wf-nets) : Le réseau de pétri qui permet de modéliser le contrôle de flux d'un workflow est appelé un WF-net (réseaux workflow) Ssi :

1. Il existe une seule place source $i \in P$ avec $*i = \emptyset$. (i début du processus).
2. Il existe une seule place puits $o \in P$ avec $*o = \emptyset$ (o est la fin du processus).
3. Chaque nœud $x \in (P \cup T)$ est sur un chemin de i à o . C'est à dire : $\text{Chemin}(i, x) \wedge \text{Chemin}(x, o)$.

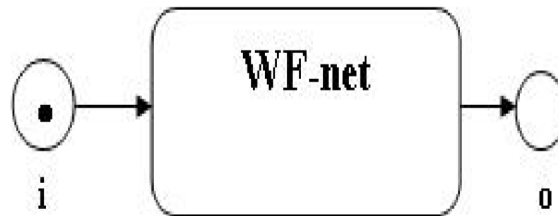


FIGURE 2.9 – Procédure modélisée avec le WF-net

Un réseau de workflow (WF-net) est un réseau de pétri qui possède une seule place d'entrée (i) et une seule place de sortie (o) et puis que n'importe quel cas traité avec la procédure représentée par un WF-net est crée au moment de début de son traitement par le système de gestion de workflow (SGWF) est supprimé une fois que sont traitement est complètement achevé par le SGWF. Cela veut dire que le WFnet spécifie le cycle de vie d'un cas du processus modélisé .

Remarque 2.5.1. *Un chemin C dans un WF-net d'un nœud n_1 au nœud n_k est une séquence $\langle n_1, n_2, \dots, n_k \rangle$ tel-que $\langle n_i, n_{i+1} \rangle \in F$, pour $i = 1 \dots k - 1$. F est la relation de flux.*

2.5.3 Constructions du routage dans un WF-net

Un réseau de workflow (WF-net) est utilisé pour spécifier le routage de flux dans les cas du workflow, quatre types de routage ont été identifiés :

2.5.3.1 Séquentiel :

Utilisé pour traiter la relation de causalité entre les tâches. Soit deux tâches A et B, si la tâche B est exécutée après l'accomplissement de A, ce comportement peut être modélisé avec un RDP en ajoutant les places pour capter les relations de causalité entre les tâches A et B. La figure 2.10 illustre ce comportement.

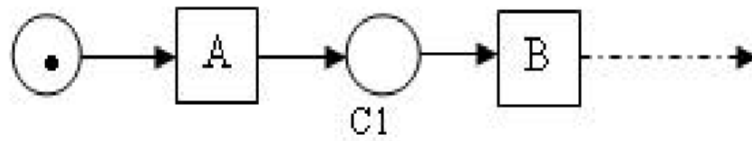


FIGURE 2.10 – Routage séquentiel.

2.5.3.2 Parallèle :

Utiliser dans le cas où l'ordre d'exécution est moins strict. Par exemple si on a deux tâches B et C qui doivent être exécutées mais dans un ordre arbitraire, la modélisation de ce comportement de routage parallèle comporte deux constructions le AND-split et AND-join.

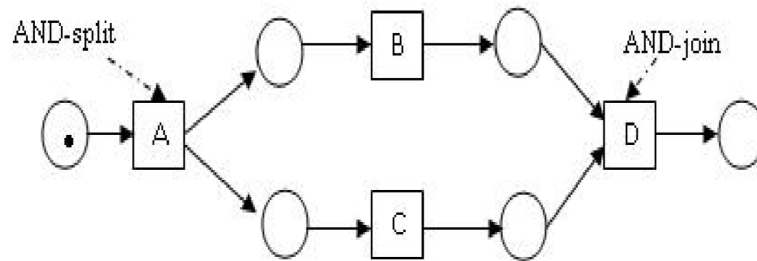


FIGURE 2.11 – Routage parallèle.

2.5.3.3 Conditionnel :

Utilisé pour traiter les cas où le routage de flux peut dépendre des données d'un cas. Il existe deux types de routage conditionnel : **le choix non-déterministe** et **le choix déterministe**.

1. **Choix non-déterministe** : Pour modéliser ce comportement deux constructions sont utilisées OR-Split et OR-join. La figure 2.12 représente un choix entre l'exécution des deux tâches B ou C grâce à un jeton dans la place c2, mais cela permet un choix non-déterministe (l'exécution de B ou de C car les deux tâches sont permises).

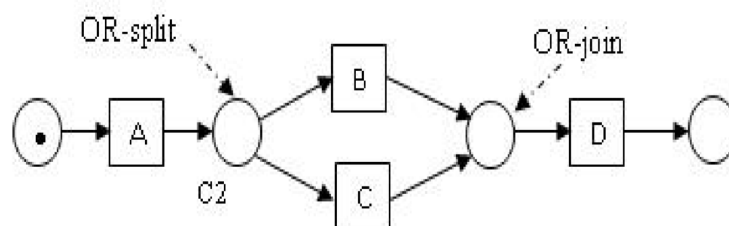


FIGURE 2.12 – Routage conditionnel non-déterministe

2. **Choix déterministe** : Le choix entre plusieurs alternatives peut être guidé par les données (ou les attributs) du workflow, dans ce cas c'est un choix déterministe.

Nous pouvons prendre le schéma de la figure précédente et ajouter une pré-condition pour chacune des tâches B et C (en se basant sur un ou plusieurs attributs du workflow).

Il y a une autre façon de modéliser le choix déterministe comme dans la figure suivante. La transition A comporte deux places de sortie c2 et c3 et produit un seul jeton (soit dans c2 ou dans c3). Le choix entre les deux places est basé sur la valeur de l'attribut x. Un symbole spécial est utilisé pour représenter que la tâche A est un OR-split (exclusive OR).

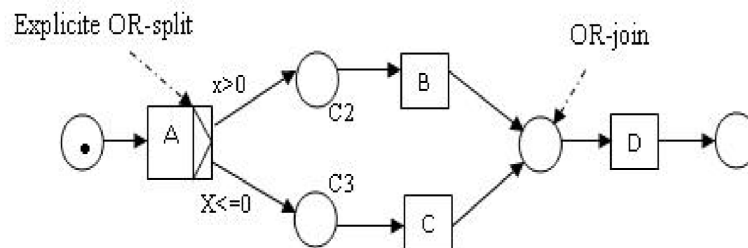


FIGURE 2.13 – Choix explicite entre B et C à base de l'attribut x .

3. **Itération** : L'itération peut être modélisée en utilisant l'implicite (OR-split, OR-join), l'explicite (OR-split, OR-join). Il est utilisé dans le cas où une ou plusieurs tâches peuvent être exécutées plusieurs fois. La figure suivante montre un exemple d'un modèle de workflow en utilisant un routage itératif.

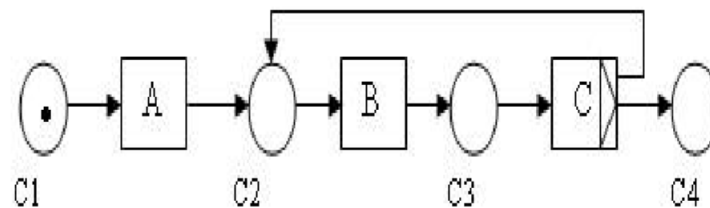


FIGURE 2.14 – Itération : B peut être exécutée plusieurs fois

2.6 Conclusion

Deuxième partie

Partie pratique

Chapitre 3

Analyse et Conception

Introduction

Dans cette partie de pratique et dans ce chapitre nous parlons sur l'analyse à la conception de travail d'abord on va parler sur UML (sa définition , contenu ... ex). puis identifier les objectifs de projet spécifications des besoins fonctionnaire et les besoins technique ,ensuite la description du diagramme de cas d'utilisation et pour la conception :une représentation des informations par le diagramme de classe et le diagramme d'activité

3.1 Unified Modeling Language (UML) :

Dans ce qui suit, nous allons donner une brève description d'UML.

3.1.1 Définition UML

L'OMG définit l'UML comme un langage visuel dédié à la spécification, la construction et la documentation des artefacts d'un système logiciel. Aussi la façon dont tout le monde modélise non seulement la structure de l'application, le comportement et l'architecture, mais aussi des processus d'affaires et la structure des données. Ce langage est conçu pour modéliser divers types de systèmes et de taille quelconque. Il possède une approche entièrement objet couvrant tout le cycle de développement. Le système est décomposé en un ensemble d'objets collaborant [GUIBERT, 2010].

3.1.2 Contenu UML

L'UML comporte 13 diagrammes qui se répartissent en deux catégories [GUIBERT, 2010]. Nous ne mentionnons que les diagrammes que nous utilisons, les autres sont en annexe G.

3.1.2.1 Diagramme structurel :

- **Diagramme de classes (Class Diagram)** : ce diagramme décrit la structure statique du système, il définit les classes, leurs attributs et leurs relations. Il est considéré comme le diagramme le plus important.

- **Diagramme de paquetages (Package Diagram) :** définit les dépendances entre les paquets (groupement d'éléments UML) constituant un modèle.

3.1.2.2 Diagramme comportemental :

- **Diagramme de cas d'utilisation (Use Case Diagram) :** pour décrire les besoins des utilisateurs.
- **Diagramme de séquence (Sequence Diagram) :** décrit comment chaque objet interagit avec l'autre et dans quel ordre, sur un axe temporel donné. Ce diagramme sont associés aux diagrammes de cas d'utilisation. .

3.2 Identification des objectifs

Au début du projet, nous nous sommes concentrés sur les besoins qui pourraient normalement être considérés comme un système général et nous avons interrogé les ingénieurs de la société nationale de retraités "CNR", qui ont exprimé des besoins importants .

Nos entretiens ont été complétés en suivant les étapes ci-dessous

Étape 1 : Choisir les entretiens Afin d'identifier les besoins, nous avons contacté des personnes pouvant fournir des informations utiles et fournir des explications. Les personnes responsables de la production des différentes unités ou entreprises sont les plus appropriées pour répondre à nos questions.

Étape 2 : Planification du développement du programme À ce stade de la planification des entretiens, nous avons étudié deux points essentiels :

- Déterminer le contexte général de l'entretien : "CNR",
- Réglage de la date et de l'heure de l'entretien (date) .

Étape 3 : Préparez-vous pour l'entrevue Préparez les questions de développement et les supports pour un processus sans faille (stylos, cartons de réponses, papiers blancs supplémentaires, clé USB).

Étape 4 : Conduisez l'entretien : Commencez le processus d'entretien en vous présentant d'abord, puis en présentant brièvement notre projet, puis en passant l'entretien en interrogeant la personne concernée et en enrichissant sa conversation d'observations et de questions. Intermédiaire.

Étape 5 : Après l'entretien Après avoir pris les informations collectées et conclu l'entretien, nous avons décidé de procéder à d'autres entretiens en développant et en nous réunissant à chaque fois.

Sur la base de ces entretiens et de l'entretien avec l'enseignant, nous avons pu identifier les spécifications suivantes :

3.2.1 Spécifications fonctionnelles

1. Le système doit permettre à l'administrateur de Gérer Modale .
2. Le système doit permettre à l'administrateur de Gérer Sous-direction .
3. Le système doit permettre à l'administrateur de Gérer les service.
4. Le système doit permettre à l'administrateur de Gérer workflow pour les dossier .
5. Le système doit permettre de Gérer les tâches des service .

6. Le système doit permettre à l'administrateur de Gérer les utilisateur .
7. Le système doit permettre à l'administrateur de consulte les historique et la recherche par tâche .
8. Le système doit permettre à l'administrateur de consulte les historique et la recherche par dossier et les jours pour chaque tâche .
9. Le système doit permettre à l'administrateur de consulte bordereau .
10. Le système doit permettre à l'utilisateur de Gérer les dossier .
11. Le système doit permettre à l'utilisateur de Activé le dossier .
12. Le système doit permettre à l'utilisateur de finir le traitement des dossiers par tâche .
13. Le système doit permettre à l'utilisateur de Gérer les bordereaux .
14. Le système doit permettre à l'utilisateur de accepte le bordereau .
15. Le système doit permettre à l'utilisateur de refuse le bordereau .
16. Le système doit permettre à l'utilisateur de imprimer le bordereau .

3.2.2 Spécifications techniques

1. Le système doit être déployé sur le cloud .
2. la configuration et la germanisation de code dynamique .
3. la modélisation de workflow par dessiner BPMN

3.2.3 Diagramme de cas d'utilisation

3.2.3.1 les acteurs de système

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système (réception d'information, etc.). Pour notre application, on a trois acteurs : le premier acteur c'est l'administrateur et le deuxième c'est utilisateur simple et le troisième c'est l'utilisateur simple .

Acteur	Type Acteur	Descriptions
Administrateur	Acteur Principale	S'authentifier
		Gérer le Modale
		Gérer Sous Direction
		Gérer les Services
		Gérer les Tâches
		Gérer les Utilisateurs
		Gérer Workflow par BPMN
		Consulter les Bordereaux
		Consulter les historique des dossiers
		Gérer les Utilisateurs
Utilisateur réception	Acteur Principal	Créer et Gérer les dossiers
		Activé les dossiers et ajoute sur wf
Utilisateur Simple	Acteur Principal	finir le traitement de dossier
		Gérer les Bordereaux
		Gérer les historique des dossiers

3.2.3.2 Cas d'utilisation

Le diagramme des cas d'utilisation suivant illustre les fonctionnalités qu'un simple utilisateur du système, également l'administrateur, peut faire. Ce diagramme a été inspiré des spécifications citées ci-dessus :

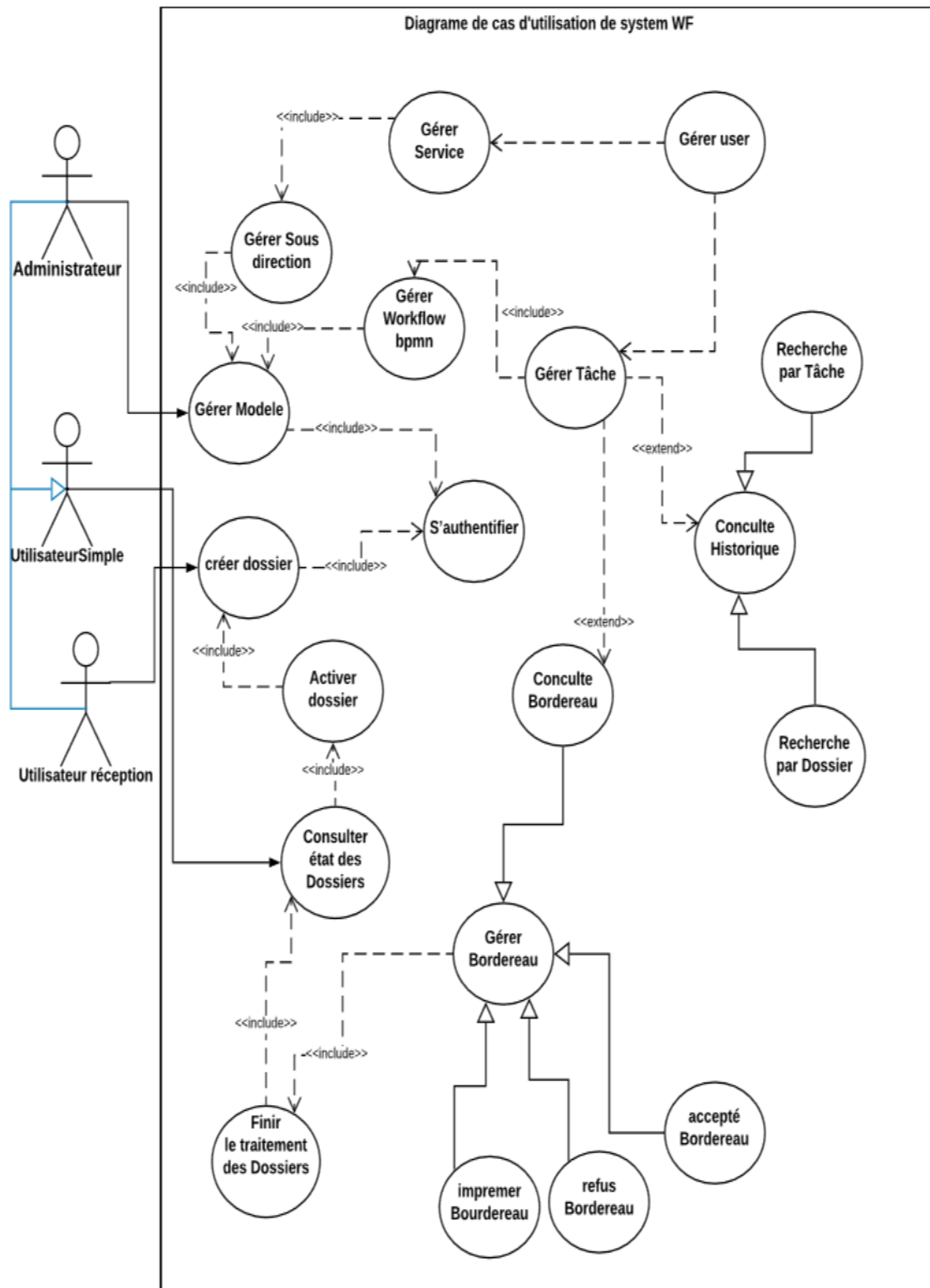


FIGURE 3.1 – Diagramme de Cas d'utilisation

3.2.4 Documentation des cas d'utilisation fonctionnels

Nous détaillerons chaque cas d'utilisation avec une description brève et spécifie les acteurs principal et les acteurs secondaires, ainsi que les Post-conditions et les pré-conditions pour le faire sont la séquence principal et la séquence alternative .

3.2.4.1 S'authentifier

CU : S'authentifier
ID :1
Description brève : chaque utilisateur doit s'authentifier auprès de l'application afin de pouvoir utiliser les fonctionnalités du système, tel que la consultation du tableau de bord, la manipulation les rapports ...etc.
Acteurs primaires : utilisateur, administrateur
Acteurs secondaires :
Pré-conditions : – La connexion auprès du serveur d'application doit être réussite. – L'utilisateur doit être enregistré dans le système.
Enchaînement principal : Ce cas d'utilisation commence lorsqu'un utilisateur souhaite accéder à l'application. 1. L'utilisateur saisit le lien de l'application dans barre d'adresse du navigateur. 2. Le serveur répond à l'utilisateur en renvoyant une page d'authentification. 3. L'utilisateur saisit son nom d'utilisateur et son mot de passe et les valide en appuyant sur 4. "Log in". 5. Le serveur vérifie la validité du nom d'utilisateur et du mot de passe. 6. Le serveur envoie une page d'accueil de l'application à l'utilisateur concerné.
Post-conditions : L'utilisateur est connecté.
Enchaînement alternatif : — E1 : La page d'authentification n'apparait pas à l'utilisateur. * L'enchaînement démarre après le premier point de l'enchaînement principal. * Le serveur envoie un message d'erreur à l'utilisateur. — E2 : Le nom d'utilisateur ou/et le mot de passe ne sont pas valides. L'enchaînement démarre après le quatrième point de l'enchaînement principal. * Le serveur envoie un message d'erreur à l'utilisateur. * Le serveur demande à l'utilisateur de ressaisit le nom d'utilisateur et le mot de passe.

3.2.4.2 cas d'utilisation par l'utilisateur réception ou initial

l'utilisateur de réception il fait de rôle principal le premier c'est gérer les dossiers le deuxième c'est activer le dossier.

3.2.4.3 Créer Dossiers

CU : Créer Dossiers
ID : 2
Description brève : chaque utilisateur peut Créer et enregistrer les nouveaux Dossiers
Acteurs primaires : utilisateur réception ou initiale,
Acteurs secondaires : simple utilisateur
Pré-conditions : L'utilisateur doit être connecté au système.
Enchaînement principal : Ce cas d'utilisation commence lorsqu'un utilisateur souhaite accéder à l'application. 1. L'utilisateur sélectionne de ajouter Nouveau dossier. 2. le système répond à l'utilisateur en affichant un formulaire d'enregistrement. 3. L'utilisateur saisit le code et le nom prénom et numéro de téléphone de la personne du dossier 4. L'utilisateur vérifie le dossier et sélectionne les champs présents. en appuyant sur « enregistrer ». 5. Le système envoie une page des dossiers.
Post-conditions : le dossier est enregistré et ajouté dans le système.
Enchaînement alternatif : E1 : Le code de Dossier n'est pas valide ou/et déjà existe. o L'enchaînement démarre après le quatrième point de l'enchaînement principal. o Le serveur envoie un message d'erreur à l'utilisateur. o Le serveur demande à l'utilisateur de ressaisir le code de Dossier.

3.2.4.4 Activer Dossiers

CU : Activer Dossiers
ID : 3
Description brève : L'utilisateur peut activer le statut du dossier et commencer au début de la tâche "Ajouter au flux de travail", après avoir rempli tous les documents.
Acteurs primaires : utilisateur réception ou initiale,
Acteurs secondaires : simple utilisateur
Pré-conditions : Le Dossier doit être enregistré dans le système.
Enchaînement principal : 1. L'utilisateur sélectionne de ajouter la liste des dossiers. 2. le système répond à l'utilisateur en affichant la liste des dossiers {"tous la liste", "liste activer", "liste non activer" } . 3. L'utilisateur sélectionne dossier 4. le système répond à l'utilisateur en affichant le détail, est après en appuyant sur "Activer". 5. le système vérifie l'id "code" de dossier et active leur état
Post-conditions : le dossier est activé dont la première tâche.
Enchaînement alternatif :

3.2.4.5 cas d'utilisation par l'utilisateur simple

l'utilisateur il fait de rôle principal : consulter l'état du dossier pour finir le traitement créer des bordereaux pur transfert les dossier (imprimer et refus ou accepté les bordereaux).

3.2.4.6 Consulter état des Dossiers

CU : Consulter état des Dossiers
ID : 4
Description brève : L'utilisateur peut visualiser le statut des dossiers en cours de traitement en fonction de sa tâche et voir tous les dossiers envoyés par une autre tache.
Acteurs primaires : utilisateur
Acteurs secondaires :
Pré-conditions : <ul style="list-style-type: none"> - Le Dossier doit être Activé dans le système. - L'utilisateur doit être connecte ou le système.
Enchainement principal : <ol style="list-style-type: none"> 1. L'utilisateur sélectionner le statut des dossiers . 2. system répond à l'utilisateur en afficher les liste des dossier encore de traitement ou les noves dossiers envoyés
Post-conditions : L'utilisateur visualiser le statut des dossiers par sa tâche.
Enchainement alternatif :

3.2.4.7 Finir le traitement des Dossiers

CU : Finir le traitement des Dossier
ID : 5
Description brève : L'utilisateur peut sélectionner liste des dossier et finir leur traitement
Acteurs primaires : utilisateur
Acteurs secondaires :
Pré-conditions : <ul style="list-style-type: none"> - Le Dossier doit être Activé dans la tâche de l'utilisateur . - L'utilisateur doit être connecte ou le système.
Enchainement principal : <ol style="list-style-type: none"> 1. L'utilisateur sélectionner la liste des dossiers pour finir le traitement en payent sur "terminer" , 2. system répond à l'utilisateur et terminer le traitement des dossier par changer état et la date fin de traitement .
Post-conditions : L'utilisateur finir le traitement des dossiers par sa tâche.
Enchainement alternatif :

3.2.4.8 Gérer Bordereau

CU : Gérer Bordereau
ID : 6
Description brève : L'utilisateur peut sélectionner liste des dossier et Gérer un Bordereau
Acteurs primaires : utilisateur
Acteurs secondaires :
Pré-conditions : - Le Dossier doit être terminer le traitement dans la tâche de l'utilisateur . - L'utilisateur doit être connecte ou le système.
Enchainement principal : 1. L'utilisateur sélectionner la liste des dossiers et sélectionner le suivant tache pour transfert 2. system répond à l'utilisateur et créer nounous bordereau et afficher la liste de bordereau correspondre a leur tache . 3. L'utilisateur put imprimer et accepte ou refuse le bordereau .
Post-conditions : L'utilisateur créer bordereau et transfert les dossier si le bordereau accepte par validation d'un utilisateur de la tâche reçu .
Enchainement alternatif : E1 :bordereau refuse. * renvoyer a la tâche président tous les dossier

3.3 Représentation des informations

3.3.1 Diagramme de Classe

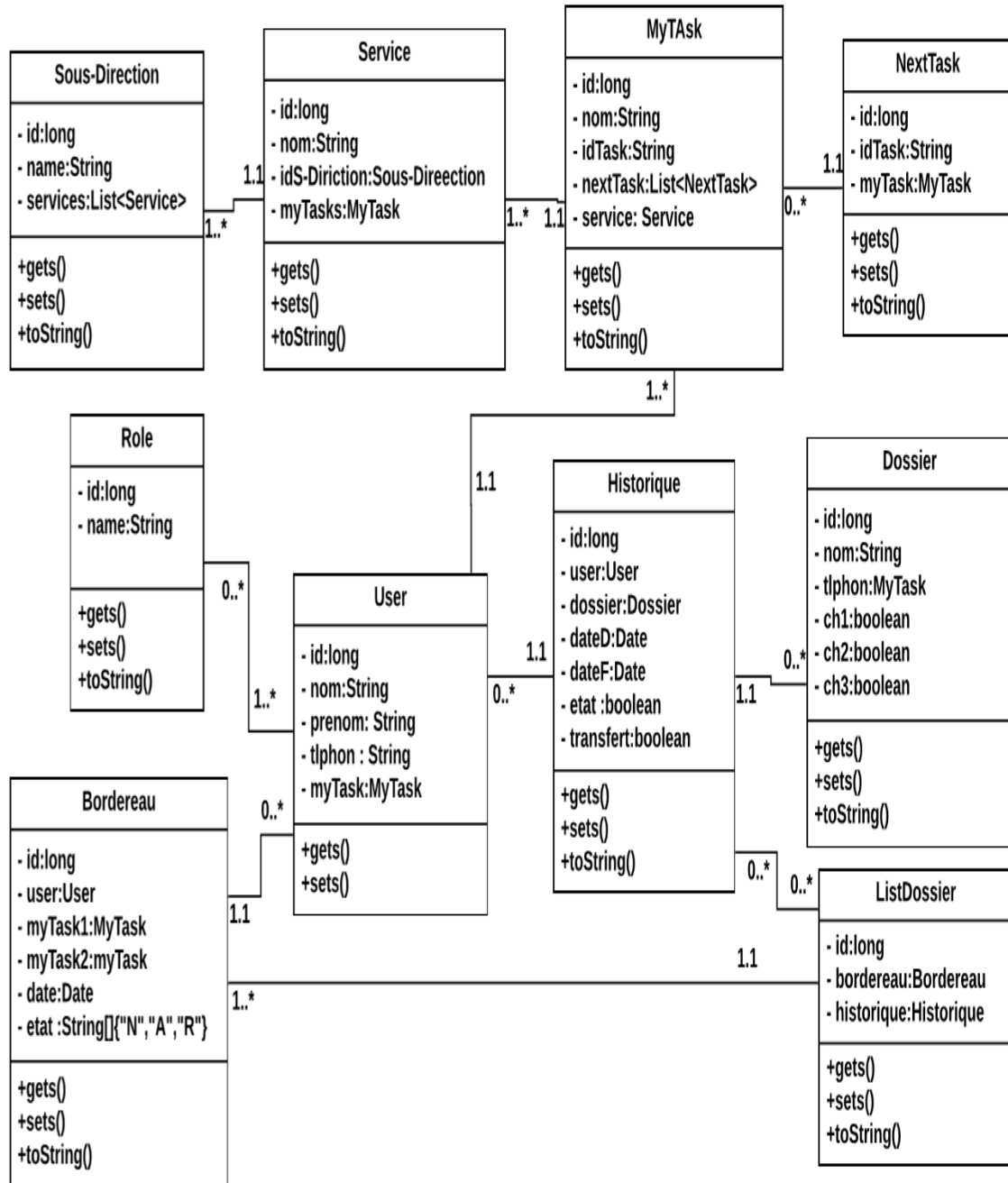


FIGURE 3.2 – Diagramme de Classe

3.3.1.1 Conception détaillée

Nous allons définir pour chaque classe ses attributs et leur types , ainsi que les méthodes qu'elle offre. Dans le but d'alléger le rapport, nous avons jugé essentiel de ne citer les classes que nous avons conçu .

La classe Sous-Direction

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getServices() et setServices()
nom	String	
services	<i>List < Service ></i>	

La classe Service

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getSous-Direction() et setSous-Direction() getMyTasks() et setMyTasks()
nom	String	
sous-Direction	Sous-Direction	
myTasks	<i>List < MyTask ></i>	

La classe MyTask

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getIdTask() et setIdTask() getService() et setService() getUsers() et setUsers() getNextMyTasks() et setNextMyTasks()
nom	String	
idTask	String	
services	Service	
users	<i>List < User ></i>	
nextMyTasks	<i>List < NextMyTask ></i>	

La classe MyTask

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getIdTask() et setIdTask() getService() et setService() getNextMyTask() et setNextMyTask()
nom	String	
idTask	String	
services	Service	
nextMyTasks	<i>List < NextMyTask ></i>	

La classe NextTask

Attribut	Type	Méthodes
id	long	getId() et setId() getIdTask() et setIdTask() getMyTask() et setMyTask()
idTask	String	
myTask	MyTask	

La classe Utilisateur "User"

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getPrenom() et setPrenom() getTlphon() et setTlphon() getMyTask() et setMyTask()
nom	String	
prenom	String	
tlphon	String	
myTask	MyTask	

La classe Role

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom()
nom	Date()	

La classe Dossier

Attribut	Type	Méthodes
id	long	getId() et setId() getNom() et setNom() getPrenom() et setPrenom() getTlphon() et setTlphon() getCh1() et setCh1() getCh2() et setCh2() getCh3() et setCh3()
nom	String	
prenom	String	
tlphon	String	
ch1	boolean	
ch2	boolean	
ch3	boolean	

La classe ListDossier

Attribut	Type	Méthodes
id	long	getId() et setId() getBordereau() et setBordereau() getHistorique() et setHistorique()
bordereau	Bordereau	
historique	Historique	

La classe Historique

Attribut	Type	Méthodes
id	long	getId() et setId() getDateD() et setDateD() getDateF() et setDateF() getDossier() et setDossier() getUser() et setUser() isEtat() et setEtat() isTransfert() et setTransfert()
dateD	Date()	
dateF	Date()	
dossier	Dossier	
user	User	
etat	boolean	
transfert	boolean	

La classe Bordereau

Attribut	Type	Méthodes
id	long	getId() et setId() getDateD() et setDateD() getUser() et setUser() getMyTask1() et setMyTask1() getMyTask2() et setMyTask2() getEtat() et setEtat() getHistorique() et setHistorique()
dateD	Date()	
user	User	
myTask1	MyTask	
myTask2	MyTask	
etat	String[]	
historique	Historique	

3.3.2 Diagramme d'activité

à partir de cas d'utilisation et de diagramme de classe va exprimer le Diagramme d'activité qui montre interaction des utilisateurs en fonction de leurs rôles dans le système en général.

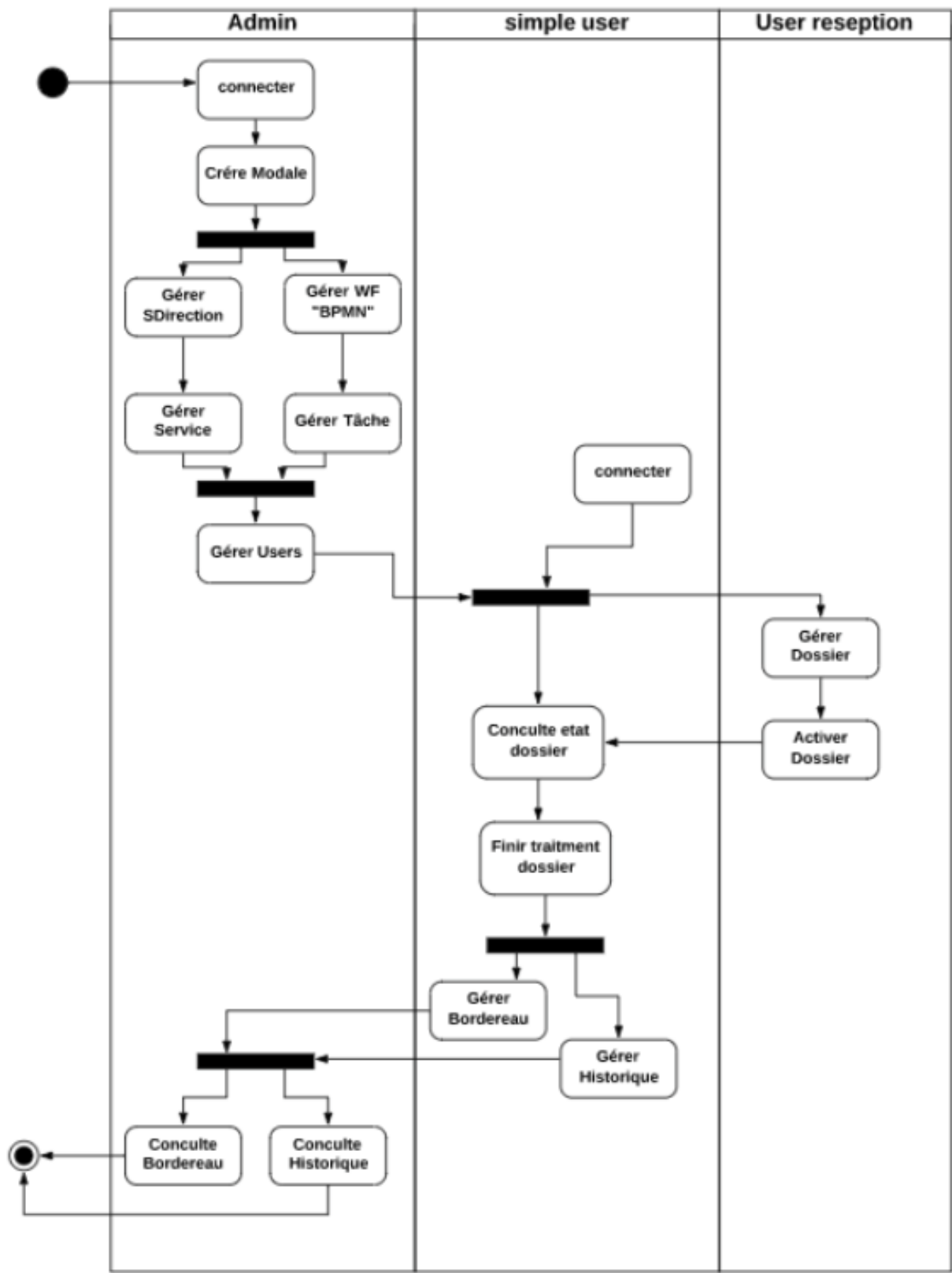


FIGURE 3.3 – Diagramme d'activité qui montre interaction des utilisateurs en fonction de leurs rôles dans le système en général.

3.3.2.1 les rôles de l'acteur administrateur :

l'interaction de l'acteur administrateur avec le système il faut d'abord de connecté où système , puis créer et défini le modèle par la spécification de nom et les chants de dossier pour la génération de code et lancer nouveau micro service déployé sur le Cloud par leur nom de service spécifié,ensuite peut gérer le sous direction et les service et le modale de workflow par "BPMN" pour gérer le tâche ,ensuite gérer les utilisateurs et enfin il peut consulter les bordereaux qui gérer par les utilisateurs et de même consulter les historiques.

3.3.2.2 les rôles de l'acteur utilisateur rescription :

l'interaction de l'utilisateur réception ou bien initial , il faut déjà créé par l'administrateur et connecter au système, il peut de gérer les dossiers par créer des nouveaux dossiers sur la réception et vérifie le les champs de chaque dossier ,ensuite il peut activer les dossiers veut dire ajouter les dossiers sur voir flou le début de traitement de dossier sur les processus "tâches" de workflow qui déjà défini par administration.

3.3.2.3 les rôles de l'acteur utilisateur simple :

après l'authentification au système l'utilisateur sa connecter sur leur tâche et il peut voir tous les dossiers en attente qui sont en cours de traitement et les bordereaux qui déjà reçu par un autre tâche, l'utilisateur peut finir le traitement de dossier par leur tâche , peut gérer un nouveau bordereau pour transfert liste des dossiers vers d'autres tâches et de gérer les historiques des dossiers.

3.4 Conclusion

Dans ce chapitre nous avons fait une étude de notre système logiciel en utilisant la modélisation UML après la spécification des besoins.

Dans le chapitre suivant nous allons expliquer la phase de réalisation du système, son intégration et son déploiement sur le cloud par les micro service, tout en respectant les directives de la conception.

Chapitre 4

Réalisation

4.1 Introduction

Une fois la conception validée, il est temps de passer à la réalisation. Nous présentons dans ce chapitre le module réalisé en utilisant des captures d'écran pour montrer ses principales fonctionnalités. Nous commençons par présenter les outils et technologies utilisées pour le développement, ensuite nous passons à l'architecture technique de la solution puis l'architecture du code.

4.2 Outils et Technologies utilisées pour le développement

4.2.1 Spring

4.2.1.1 Spring Framework

Spring Framework fournit un modèle complet de programmation et de configuration pour les applications d'entreprise modernes basées sur Java - sur tout type de plate-forme de déploiement.

Un élément clé de Spring est le support infrastructurel au niveau de l'application : Spring met l'accent sur la «plomberie» des applications d'entreprise afin que les équipes puissent se concentrer sur la logique métier au niveau de l'application, sans liens inutiles avec des environnements de déploiement spécifiques.

4.2.1.2 Spring boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

4.2.1.3 Spring Data

4.2.1.4 Spring Cloud

Spring Cloud fournit aux développeurs des outils permettant de créer rapidement certains modèles courants dans les systèmes distribués (gestion de la confi-

guration, découverte de services, disjoncteurs, routage intelligent, micro-proxy, bus de contrôle, jetons à usage unique, verrous globaux, élection des dirigeants, distribution sessions, état du cluster). La coordination des systèmes distribués conduit à des modèles de plaque de chaudière et, grâce à Spring Cloud, les développeurs peuvent rapidement mettre en service des services et des applications mettant en œuvre ces modèles. Ils fonctionneront bien dans n'importe quel environnement distribué, y compris l'ordinateur portable du développeur, les centres de données à nu et les plateformes gérées telles que Cloud Foundry.

4.2.1.5 Spring Cloud Netflix

Spring Cloud Netflix fournit des intégrations Netflix OSS pour les applications Spring Boot via la configuration automatique et la liaison à Spring Environment et à d'autres idiomes de modèles de programmation Spring. Quelques annotations simples vous permettent d'activer et de configurer rapidement les modèles courants dans votre application et de créer de grands systèmes distribués avec des composants Netflix testés au combat. Les modèles fournis incluent la découverte de service (Eureka), le disjoncteur (Hystrix), le routage intelligent (Zuul) et l'équilibrage de la charge côté client (ribbon).

4.2.2 Qu'est-ce que Micro Service ?

L'objectif principal de la mise en œuvre des micro-services est de scinder l'application en un service distinct pour chaque fonctionnalité de base et chaque service de l'API. Elle devrait être déployée indépendamment sur le cloud. Nous avons choisi le langage de programmation réactif du projet familial spring.io avec un ensemble de composants pouvant être utilisés pour mettre en œuvre notre modèle d'exploitation. Spring Cloud intègre très bien les composants Netflix dans l'environnement Spring. Il utilise une configuration automatique et une convention de configuration similaire à celle du fonctionnement de Spring Boot.

4.2.2.1 Pourquoi l'architecture de Microservices ?

Nous avons choisi l'architecture de micro-services pour écrire chaque fonctionnalité en tant que service distinct pour les fonctionnalités de base et d'API, ce qui nous aide à réaliser la livraison et l'intégration en continu.

4.2.2.2 Patterns dans l'architecture des microservices

— Api Gateway

1. Choisissez de créer l'application en tant qu'ensemble de micro-services.
2. Décidez comment le client de l'application va interagir avec les micro-services.
3. Avec une application monolithique, il n'y a qu'un seul ensemble de points d'extrémité (généralement répliqués, à charge équilibrée).
4. Dans une architecture de micro-services, chaque micro-service expose cependant un ensemble de points finaux.

— Service registry

1. Le registre de service aide à déterminer l'emplacement des instances de service pour envoyer la demande au correspondant un service
2. Ici, nous avons utilisé Netflix Eureka pour enregistrer un service pouvant être enregistré dans le registre de services. serveur et il peut être identifié par le routeur.

— Service Discovery

1. Dans une application monolithique, les services s'appellent par le biais d'appels de méthode ou de procédure au niveau de la langue.
2. Toutefois, dans une application moderne basée sur des micro-services, elle s'exécute généralement dans des environnements virtualisés où le nombre des instances d'un service et de leurs emplacements change de façon dynamique.
3. Chaque service peut être identifié à l'aide d'un routeur enregistré auprès du serveur de registre de services.

4.2.2.3 Architecture des microservices via les composants Netflix

Nous avons utilisé les composants Netflix pour réaliser les modèles d'architecture de microservices ci-dessus. OPTISOLBUSINESS 2019.

Operations Component	Spring, Netflix OSS
Service Discovery server	Netflix Eureka
Edge Server	Netflix Zuul
Central configuration server	Spring Cloud Config Server
Dynamic Routing and Load Balancer	Netflix Ribbon
OAuth 2.0 protected API's	Spring Cloud + Spring Security OAuth2
Monitoring	Netflix Hystrix dashboard and turbine

4.2.3 Composantes majeures de Netflix

4.2.3.1 Service Discovery Server



Netflix Eureka permet aux micro-services de s'enregistrer eux-mêmes au moment de leur exécution, tels qu'ils apparaissent dans la structure du système. OPTISOLBUSINESS 2019

4.2.3.2 Routage dynamique et équilibreur de charge



Netflix Ribbon peut être utilisé par les consommateurs de services pour rechercher des services au moment de l'exécution. Le ruban utilise les informations disponibles dans Eureka pour localiser les instances de service appropriées. Si plusieurs instances sont trouvées, le Ruban appliquera un équilibrage de charge pour répartir les demandes sur les instances disponibles. Le ruban ne s'exécute pas en tant que service distinct, mais en tant que composant intégré dans chaque consommateur de service. OPTISOLBUSINESS 2019

4.2.3.3 Serveur Edge



Zuul est (bien sûr) notre gardien du monde extérieur, ne permettant pas le passage de demandes externes non autorisées. Zuul fournit également un point d'entrée bien connu aux micro-services dans le paysage système. L'utilisation de ports alloués de manière dynamique est pratique pour éviter les conflits de ports et minimiser l'administration, mais elle rend évidemment la tâche plus difficile pour tout consommateur de services donné. Zuul utilise Ribbon pour rechercher les services disponibles et achemine la demande externe vers une instance de service appropriée. OPTISOLBUSINESS 2019

4.2.4 Spring Boot et Spring Cloud Netflix OSS – Micro Service Architecture

4.2.4.1 Micro Services avec Spring Boot

Spring Boot est un tout nouveau framework de l'équipe de Pivotal, conçu pour simplifier le démarrage et le développement d'une nouvelle application Spring. Le cadre adopte une approche de configuration avisée, libérant les développeurs de la nécessité de définir la configuration standard. OPTISOLBUSINESS 2019

4.2.4.2 Spring Cloud Netflix

Spring cloud Netflix fournit des intégrations Netflix OSS pour les applications de démarrage printanier via la configuration automatique et la liaison à l'environnement Spring et à d'autres modèles de programmation Spring. Avec quelques annotations simples, nous pouvons rapidement activer et configurer des modèles courants dans une application et construire des systèmes distribués volumineux avec des composants Netflix. De nombreuses fonctionnalités sont disponibles avec le nuage de printemps Netflix. Ici, nous avons répertorié certaines des fonctionnalités communes que nous avons implémentées avec les micro-services avec Spring Boot et Netflix. OPTISOLBUSINESS 2019

- **Découverte du service :**

Les instances Eureka peuvent être enregistrées et les clients peuvent Découvrez les exemples à l'aide de haricots à gestion printanière

- **Création de service :** Le serveur Eureka intégré peut être créé avec configuration Java déclarative

- **Configuration Externel :**

Bridge from the Spring Environment (permet aux utilisateurs de configuration des composants Netflix à l'aide de Spring Boot conventions)

- **Routeur et filtre :**

Enregistrement automatique des filtres Zuul, et un simple convention sur l'approche de configuration pour la création de proxy inverse.

4.2.5 Maven

4.2.5.1 Qu'est-ce que Maven ?

Maven est essentiellement un outil de gestion et de compréhension de projet.

Maven offre des fonctionnalités de :

- construction , compilation ;
- documentation ;
- rapport ;
- gestion des dépendances ;
- gestion des sources ;
- mise à jour de projet ;
- déploiement.

4.2.5.2 Qu'est-ce que le POM ?

Le **POM** (**P**roject **O**bject **M**odel) est une façon de décrire, de manière déclarative, un projet au sens de Maven. Cette description est contenue dans le fichier **pom.xml** présent dans le repertoire de base du projet. Le fichier pom.xml contient donc tous les éléments permettant de gérer le cycle de vie du projet. JAVA.DEVELOPPEZ.COM 2019.

Code xml :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>Maven Quick Start Archetype</name>
<url>http://maven.apache.org</url>
```

```
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
```

```
</project>
```

- **project** : c'est la balise racine de tous les fichiers pom.xml.
- **modelVersion** : cette balise indique la version de POM utilisée. Bien que cette version ne change pas fréquemment, elle est obligatoire afin de garantir la stabilité d'utilisation.
- **groupId** : cette balise permet d'identifier un groupe qui a créé le projet. Cette clé permet d'organiser et de retrouver plus facilement et rapidement le projet.

- **artifactId** : cette balise indique un nom unique utilisé pour nommer les artifacts à construire.
- **packaging** : type de packaging du projet (ex. : JAR, WAR, EAR, etc.).
version : version de l'artifact généré par le projet.
- **name** : nom du projet.
- **url** : adresse du site du projet. description : description du projet.
- **dependencies** : balise permettant de gérer les dépendances.

4.2.5.3 Qu'est-ce qu'un archetype ?

Un archetype est un template de projet. Le fait d'utiliser des archetypes pour initialiser un projet permet de gagner du temps et de respecter une certaine convention. JAVA.DEVELOPPEZ.COM 2019.

4.2.5.4 Qu'est-ce qu'une dépendance ?

Une dépendance est une référence vers un artefact spécifique contenu dans un repository. Cet artefact est nécessaire pour une ou plusieurs phases du cycle de vie du projet. JAVA.DEVELOPPEZ.COM 2019.

L'exemple le plus simple est une dépendance sur une bibliothèque jar qui permet d'en utiliser le contenu dans le projet.(JAVA.DEVELOPPEZ.COM 2019).

4.2.5.5 Qu'est-ce qu'un artefact ?

Dans Maven, un artefact est un élément spécifique issu de la construction du logiciel.

Dans Java, les artefacts les plus communs sont des JARs, mais ce peut être aussi un fichier WAR, un EAR, un ZIP, etc.(JAVA.DEVELOPPEZ.COM 2019).

4.2.5.6 Qu'est-ce que le groupId/artifactId ?

Le groupId est l'identifiant du groupe, à l'origine du projet. GroupId suit les mêmes règles de nommage que les packages Java (exemple : fr.masociete.monprojet), et on choisit généralement comme groupId le nom du top package du projet. (JAVA.DEVELOPPEZ.COM 2019).

L'artifactId est l'identifiant du projet au sein de ce groupe.

L'artifactId est utilisé par défaut pour construire le nom de l'artefact final (exemple : pour un artifactId=monprojet, le nom du fichier jar généré sera monprojet-version.jar).

4.2.5.7 Qu'est-ce qu'un SNAPSHOT ?

Par convention, une version en cours de développement d'un projet voit son numéro de version suivi d'un -SNAPSHOT.

Ainsi un projet en version 2.0-SNAPSHOT signifie que cette version est une pré-version de la version 2.0, en cours de développement.

Ce concept de SNAPSHOT est particulièrement important pour Maven. En effet, dans la gestion des dépendances, Maven va chercher à mettre à jour les versions SNAPSHOT régulièrement pour prendre en compte les derniers développements.

Utiliser une version SNAPSHOT permet de bénéficier des dernières fonctionnalités d'un projet, mais en contrepartie, cette version peut être (et est) appelée à être modifiée de façon importante, sans aucun préavis. (JAVA.DEVELOPPEZ.COM 2019).

4.2.5.8 Qu'est-ce que le repository local, distant ?

Un repository local est un répertoire sur le poste du développeur permettant de stocker, suivant la même arborescence, tous les artefacts téléchargés depuis le(s) repository distant(s). JAVA.DEVELOPPEZ.COM 2019.

Un projet ayant pour POM :

Code xml :

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>fr.masociete</groupId>
<artifactId>monprojet</artifactId>
<version>1.0</version>
</project>
```

4.2.6 bpmn js

bpmn-js est une boîte à outils de rendu et un modélisateur Web BPMN 2.0. Il est écrit en JavaScript, intègre les diagrammes BPMN 2.0 dans les navigateurs modernes et ne requiert aucun serveur. Cela facilite son intégration dans n'importe quelle application Web.

La bibliothèque est construite de manière à pouvoir être à la fois un visualiseur et un modélisateur Web. Utilisez le visualiseur pour intégrer BPMN 2.0 dans vos applications et l'enrichir avec vos données. Utilisez le modélisateur pour créer des diagrammes BPMN 2.0 dans votre application.

4.2.7 Camunda BPM

Camunda BPM est un système de gestion de flux de travaux gratuit écrit en Java qui définit et exécute les processus métier dans BPMN 2.0. Camunda BPM prend également en charge les normes CMMN 1.1 et DMN 1.1 normalisées par le groupe de gestion des objets (OMG).

4.2.8 Thymeleaf

Thymeleaf est un moteur de template Java moderne côté serveur pour les environnements Web et autonomes.

L'objectif principal de Thymeleaf est d'intégrer des modèles naturels élégants à votre flux de travail de développement : un code HTML pouvant être correctement affiché dans les navigateurs et fonctionnant également en tant que prototypes statiques, ce qui permet de renforcer la collaboration au sein des équipes de développement.

Avec des modules pour Spring Framework, une multitude d'intégrations avec vos outils favoris et la possibilité d'intégrer vos propres fonctionnalités, Thymeleaf est

idéal pour le développement Web JVM HTML5 moderne - bien qu'il puisse faire beaucoup plus.

4.3 Architecture Modèle/Vue/Contrôleur

L'architecture Modèle/Vue/Contrôleur (MVC) est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface.

L'organisation globale d'une interface graphique est souvent délicate. Bien que la façon MVC d'organiser une interface ne soit pas la solution miracle, elle fournit souvent une première approche qui peut ensuite être adaptée. Elle offre aussi un cadre pour structurer une application.

Dans l'architecture MVC, les rôles des trois entités sont les suivants. (IRIF 2019)

- **modèle** : données (accès et mise à jour)
- **vue** : interface utilisateur (entrées et sorties)
- **contrôleur** : gestion des événements et synchronisation

4.3.1 Rôle du modèle

Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.

Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ses données. Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données. Si par exemple le programme manipule une base de données pour les emplois du temps, le modèle peut avoir des méthodes pour avoir, tous les cours d'une salle, tous les cours d'une personnes ou tous les cours d'une groupe de Td.

4.3.2 Rôle de la vue

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, ...). Ses différents événements sont envoyés au contrôleur.

La vue peut aussi donner plusieurs vues, partielles ou non, des mêmes données. Par exemple, l'application de conversion de bases a un entier comme unique donnée. Ce même entier est affiché de multiples façons (en texte dans différentes bases, bit par bit avec des boutons à cocher, avec des curseurs). La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

4.3.3 Rôle du contrôleur

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des

données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

Dans le cas d'une base de données des emplois du temps. Une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants. En effet si un même objet reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieux et peut conduire à un code pas très élégant (un énorme switch). C'est pour éviter ce problème que le contrôleur est réparti en plusieurs objets.

4.3.4 Interactions

Les différentes interactions entre le modèle, la vue et le contrôleur sont résumées par le schéma de la figure 4.1 suivante.

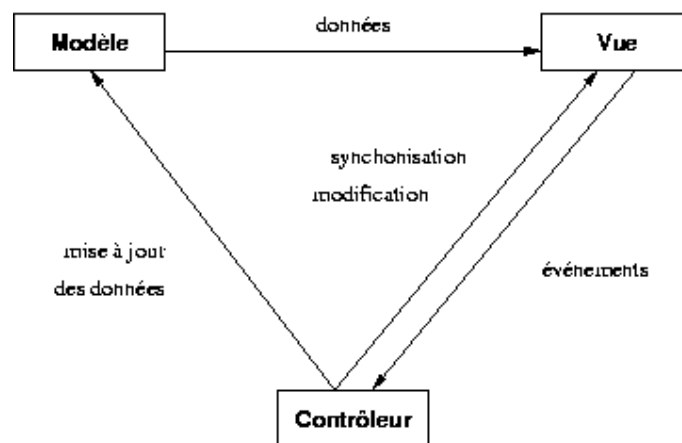


FIGURE 4.1 – Interactions entre le modèle, la vue et le contrôleur

Bibliographie

- EPSI (2019). *PALMARÈS 2019 DES FOURNISSEURS DU CLOUD*. URL : <http://www.epsi.fr/fournisseurs-cloud-2019/>.
- IRIF (2019). *Architecture Modèle/Vue/Contrôleur*. URL : <https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/mvc.html>.
- JAVA.DEVELOPPEZ.COM (2019). *FAQ Maven*. URL : <https://java.developpez.com/faq/maven/?page=Terminologie-et-documentation#Qu-est-ce-que-Maven>.
- OPTISOLBUSINESS (2019). *MicroServices Architecture – Spring Boot And Netflix Infrastructure*. URL : <https://www.optisolbusiness.com/insight/micro-services-architecture-spring-boot-and-netflix-infrastructure>.