



Etude Odoo V11 Opérations Documentation

Version 1.0

Didier Stadelmann

oct. 01, 2018

Table des matières

1	Introduction	1
1.1	Cadre de l'étude	1
1.2	Déroulement de l'étude	2
1.3	Odoo, Open Source et utilisateurs	4
1.4	La logistique, les opérations, la fabrication	6
1.5	Licence	10
2	Le fonctionnement logistique du système Odoo V11	11
2.1	Le processus standard	11
2.1.1	Le processus intégré depuis la vente	11
2.1.2	Démonstration	11
2.2	La structure logistique	12
2.2.1	Entrepôt (Warehouse)	12
2.2.2	Emplacement de stock (Stock Location)	12
2.2.3	Schéma des flux	13
2.3	Les flux internes et externes	14
2.4	Les routes et les règles	15
2.4.1	Les routes	15
2.4.2	Les règles	16
2.4.2.1	La méthode d'approvisionnement	16
2.4.3	Séquence de détermination de la route et de la règle applicable	16
2.5	La fabrication	17

3	Etude et tests	19
3.1	La société fictive « Open Bike SF »	19
3.2	A) Approvisionnement « Juste-à-temps »	21
3.3	B) Traçabilité	21
3.4	C) Usabilité du MPS	22
3.5	Les scénarios de test	25
3.5.1	A1 - MTS : Approvisionnement « Juste à temps » MTS	25
3.5.1.1	Objectifs du test	25
3.5.1.2	Conditions de test	25
3.5.1.3	Conditions de réussite	27
3.5.1.4	Procédure de test	27
3.5.1.5	Résultats	28
3.5.1.6	Faiblesses identifiées	28
3.5.1.7	Commentaire	29
3.5.2	A1 - MTO : Approvisionnement « Juste à temps » MTO	29
3.5.2.1	Objectifs du test	29
3.5.2.2	Conditions de test	29
3.5.2.3	Conditions de réussite	30
3.5.2.4	Procédure de test	30
3.5.2.5	Résultats	30
3.5.2.6	Faiblesses identifiées	31
3.5.2.7	Commentaire	32
3.5.3	B1 : Quality Control	32
3.5.3.1	Objectifs du test	32
3.5.3.2	Conditions de test	32
3.5.3.3	Conditions de réussite	34
3.5.3.4	Procédure de test	34
3.5.3.5	Résultats	34
3.5.3.6	Faiblesses identifiées	34
3.5.3.7	Commentaire	35
3.5.4	C1 - MPS : Approvisionnement basé sur une prévision de vente	35
3.5.4.1	Objectifs du test	35
3.5.4.2	Conditions de test	36
3.5.4.3	Conditions de réussite	36

3.5.4.4	Procédure de test	36
3.5.4.5	Résultats	37
3.5.4.6	Faiblesses identifiées	37
3.5.4.7	Commentaire	37
4	Résultats de l'étude	39
4.1	Faiblesses identifiées	39
4.2	Modules complémentaires à étudier	41
4.2.1	OCA - Odoo Community Association	41
4.2.2	Odoomrp.com	41
4.3	Décisions	42
5	Annexes	43
5.1	Documents	43
5.1.1	Maintenance de la documentation	43
5.1.1.1	Fonctionnement de la chaîne d'édition	43
5.1.1.2	Prérequis	44
5.1.1.3	Mise en place d'une nouvelle documentation	45
5.1.1.4	Installation du système pour l'édition locale	46
5.1.2	Glossaire	48
5.2	Routes et emplacements de stock	50
5.3	Routes et Qualité (avec points de contrôle)	52
5.4	Fonctionnement des routes	54
5.5	Séquence de détermination de la règle applicable	56
5.6	Situation AS-IS	58
6	Licence	61

CHAPITRE 1

Introduction

1.1 Cadre de l'étude

Cette documentation est établie dans le cadre de mon travail de diplôme de l'Ecole Supérieure d'Informatique de Gestion¹ de Delémont² (ESIG) , et porte sur le fonctionnement logistique du système ERP Odoo V11e.

Etudier le fonctionnement logistique d'un système ERP et proposer des solutions d'amélioration nécessite une expérience terrain importante, sachant que l'équivalent sur un corps humain reviendrait à étudier l'ensemble du système sanguin, probablement quelques mécanismes du système nerveux, voire le système lymphatique.

Longtemps, on a par exemple soigné la fièvre par des saignées³, jusqu'au jour où on s'est aperçu que la situation du malade se dégradait plus qu'elle ne s'améliorait.

<http://www.esig-jura.ch>

<https://en.wikipedia.org/wiki/Delémont>

[https://fr.wikipedia.org/wiki/Saignée_\(médecine\)](https://fr.wikipedia.org/wiki/Saignée_(médecine))



Fig. 1 – Pratique d’une saignée (Crédit Wikipedia)⁴

Les systèmes de gestion ERP d’aujourd’hui, que ce soit le plus imposant [SAP](#)⁵ ou le « petit » [Odoo](#)⁶, ne sont pas vraiment encore intelligents. Ils appliquent des schémas préconçus (paramétrage) suite à des actions d’utilisateurs ou de job planifiés. Quelques algorithmes bien pensés permettent aux différents responsables métiers d’obtenir une vue en temps réel sur la situation de l’entreprise.

Mais lorsque le système ERP subit une « poussée de fièvre », il est important de rechercher la cause profonde du problème, et d’appliquer la solution avec tact et parcimonie, surtout lorsque le système est en production. La saignée doit être évitée à tout prix. Cela est également le cas lorsque des fonctionnalités doivent être ajoutées au système.

La tâche est donc vaste et hardue.

1.2 Déroulement de l’étude

Le mandat du travail de diplôme porte sur l’analyse du système logistique « tel quel », sur l’établissement de propositions d’améliorations en relation avec le manufacturing complexe, et le codage d’un ou plusieurs modules de correction.

https://commons.wikimedia.org/wiki/File:Pratique_d'une_saignee.jpg

<https://www.sap.com/index.html>

<https://www.odoo.com>

A mon sens, la bonne pratique en terme d'implémentation et d'adaptation d'un ERP à une entreprise est :

1. Etudier et comprendre la philosophie du système ERP
2. Etudier et comprendre les besoins du client en terme de gestion, au sens large
3. Identifier les objets du système ERP répondant aux besoins
4. Vérifier si les flux entre les objets répondent aux exigences

Si ce n'est pas le cas, nous aurons le choix entre :

1. Rester dans la philosophie du système,
 - en proposant un flux alternatif standard au client, ou
 - adapter le système par du paramétrage
2. Adapter / Raccourcir le flux
 - par du code sur un user-exit
 - par un module de code dédié

Vu l'étendue d'un système tel que SAP, il est rare de devoir coder au coeur du système, mais cela est plus fréquent en périphérie. Sur un système tel que Odoo, cela est plus fréquent, vu que le nombre d'objets à disposition est plus restreint.

Dans tous les cas, on privilégiera autant que faire se peut d'éviter de coder - même si cela est parfois tentant. Ces ajouts de code n'étant pas connus de l'éditeur, il peuvent fragiliser la cohérence du système, et certainement ralentir les montées de version, puisque l'ensemble des flux devront être revalidé, avec un soin particulier pour ces éléments ajoutés.

Ne pas coder implique par contre une connaissance parfaite et large des capacités de l'outil en terme de paramétrage. Egalement, on ne peut pas se satisfaire d'un flux qui fonctionne parfaitement. . . Une modification peut avoir provoqué un dommage collatéral sur le flux voisin.

Dans le cadre de ce travail, **je vais donc d'abord m'appliquer à étudier et à m'imprégner de la philosophie** de fonctionnement de Odoo.

N'ayant pas encore une connaissance approfondie de l'outil, cette phase **est à risque** du point de vue **du délai** des phases de projet, puisque, vu ce qui a été évoqué

à l'instant, il n'est **pas question de proposer** un module de correction, **sans avoir la certitude** que du paramétrage pourrait corriger la situation.

Une autre question se pose : si la phase de compréhension se prolonge, je ne pourrai pas montrer de code lors de la défense. A mon sens, ceci n'est pas grave puisque appuyer du code à un autre endroit que la cause première du problème, revient à construire une magnifique villa sur une falaise qui s'érode. Je m'en tiendrai dans ce cas aux plans d'architecte.



Fig. 2 – Californie: l'érosion d'une falaise oblige des familles à évacuer leurs habitations (Crédit RTBF)⁷

1.3 Odoo, Open Source et utilisateurs

Odoo est un système ERP pour les PME, développé par Odoo SA en Belgique. La base du système est constituée de modules de base et optionnels [Open Source](#)⁸

⁷ https://www.rtbef.be/info/insolites/detail_californie-l-erosion-d-une-falaise-oblige-des-familles-a-evacuer-leurs-habitati
id=9197405

⁸ <https://www.odoo.com/documentation/user/11.0/legal/licenses/licenses.html#licenses>

(Community). Une version Enterprise ajoute des fonctionnalités par des modules supplémentaires et un support de l'éditeur contre paiement.

Une communauté OCA Odoo Community Association propose et structure des modules complémentaires Open Source.

Les principales technologies sur lesquelles s'appuie Odoo sont des standards Open Source éprouvés, notamment :

- Système d'exploitation Linux
- Base de données [PostgreSQL](https://www.postgresql.org) ⁹
- Language [Python 3](https://www.python.org) ¹⁰
- Un [Mapping objet-relationnel \(ORM\)](https://www.odoo.com/documentation/11.0/reference/orm.html#common-orm-methods) ¹¹ comme proxy entre la base de données et le code/interface
- Languages et frameworks Web (HTML, CSS, Javascript, Bootstrap) pour l'interface
- [API XMLRPC](https://www.odoo.com/documentation/11.0/webservices/odoo.html) ¹² ou JSONRPC pour l'interfaçage et communication avec des systèmes externes
- Longpolling pour la communication interne et le chat

De part la nature ouverte du système, tout un chacun peut créer gratuitement **sa propre installation** en suivant la [documentation d'installation](https://www.odoo.com/documentation/11.0/setup/install.html) ¹³ de l'éditeur. Il suffit de disposer d'un [petit serveur Linux](https://www.odoo.com/documentation/11.0/setup/install.html) ¹⁴, de type VPS par exemple. Le code communautaire est disponible sur les [serveurs de Github.com](https://github.com/odoo/odoo). ¹⁵

Si ce travail porte sur une **utilisation professionnelle** d'Odoo, quelques éléments d'informations seront fournis de manière vulgarisée dans la mesure du possible à l'attention d'un **public moins aguerri**. Je pense ici notamment **aux étudiants en informatiques**, pour qui l'approche initiale d'un système ERP n'est pas aisée.

<https://www.postgresql.org>

<https://www.python.org>

<https://www.odoo.com/documentation/11.0/reference/orm.html#common-orm-methods>

<https://www.odoo.com/documentation/11.0/webservices/odoo.html>

<https://www.odoo.com/documentation/11.0/setup/install.html>

<https://www.odoo.yenthevg.com/installing-configuring-odoo-digitalocean/>

<https://github.com/odoo/odoo>

1.4 La logistique, les opérations, la fabrication

Dans le cadre de ce travail, nous vérifierons si le système Odoo permet à une entreprise manufacturière de fournir des biens industriels à un client, en s'approvisionnant en matière premières, en les transformant, en les stockant et en les livrant.

Nous nous intéresserons principalement aux flux, à la transformation et au stockage des articles. Nous regrouperons l'ensemble de ces activités sous le terme générique d'**opérations**. Nous associerons le terme **fabrication** ou **manufacturing** aux opérations de transformation d'éléments vers un composé de nature différente.

Avant de nous lancer dans l'étude à proprement parler, nous allons encore définir quelques éléments de compréhension.

On ne fait pas d'omelette sans casser des oeufs !

Arrêtons-nous quelques instants sur cette *locution populaire*¹⁶, qui signifie que lorsque on veut entreprendre quelque chose, il faut en accepter les risques. Cela s'applique bien entendu à la fabrication et à la probabilité que le résultat final ne soit pas toujours celui escompté.

Intéressons nous maintenant à la forme littérale de cette phrase. . .

— On apprend tout d'abord que pour pour fabriquer une omelette, il faut des oeufs !

On ne nous dit pas toutefois combien d'oeufs sont nécessaire, ni le temps de cuisson pour que l'on puisse parler d'omelette. En cuisine, la méthode d'obtention d'une omelette s'appelle une *recette*.

https://fr.wiktionary.org/wiki/on_ne_fait_pas_d\T1\textquoterightomelette_sans_casser_des_\T1\oeufs

<https://www.bioalaune.com/fr/actualite-bio/12067/non-au-gaspillage-alimentaire-cinq-astuces-utiliser-blancs-doeufs>
<https://www.cuisineaz.com/recettes/omelette-simple-56045.aspx>



Fig. 3 – Oeuf cassé (Crédit bioalaune.com)¹⁷

GUIDE DE PRÉPARATION : OMELETTE SIMPLE

1 Pers.

10 min

7 min

11,21

Facile

INGRÉDIENTS	PRÉPARATION
3 oeufs entiers	ÉTAPE 1 Battez les oeufs entiers dans un bol. Ajoutez sel et poivre à votre goût puis mélangez bien au fouet ou à la fourchette.
30 g de gruyère râpé	
6 branches de ciboulette ciselée	
2 c. à soupe d'huile d'olive	
sel, poivre	
	ÉTAPE 2 Faites chauffer l'huile d'olive dans une poêle et lorsqu'elle est bien chaude, versez les oeufs battus dans la poêle et laissez cuire 1 minute tout en remuant constamment, jusqu'à obtenir une omelette baveuse ou bien ferme selon vos goûts. Quand l'omelette est cuite à votre goût, ajoutez le fromage râpé et roulez-la. Dressez-la dans une assiette de service, et dégustez avec une salade verte assaisonnée et des mouillettes de pain beurrées ou tartinées de fromage.

Fig. 4 – Recette d'une omelette simple (Crédit <https://www.cuisineaz.com>)¹⁸

On réalise que pour cuisiner (fabriquer) une omelette de qualité et de taille satisfaisante pour **une personne**, il faut des ingrédients dans **une quantité** (30) et **unité de quantité** (grammes) bien déterminée.

D'autre part, la préparation des ingrédients se réalise en **deux étapes distinctes** impliquant des opérations précises :

- Etape 1 : Battez. . . ,
- Etape 2 : Faites chauffer. . .

Cette recette est probablement suffisante pour le repas du soir, mais dans un milieu industriel, elle est insuffisamment précise. En effet, on obtiendra une omelette de taille différente selon la taille des oeufs.

En termes industriels, on parlera de fabrication en process dans le cas où le produit est obtenu sur la base d'une formule ou d'une recette, et que les ingrédients qui le compose ne peuvent plus être facilement dissociés après la transformation. Par opposition, on parlera de fabrication discrète lorsque le produit obtenu peut être facilement touché, vu ou compté, voire redémonté.

Dans l'étude, nous nous intéresserons **uniquement à la fabrication discrète**, qui traite de produits tels que montres, voitures, machines, etc.

Pour fabriquer un objet discret, nous parlerons de nomenclatures (Bill of Materials - BOM en anglais), qui précisent que pour fabriquer 1 voiture, il faut :

- 1 châssis,
- 1 moteur et
- 4 roues.

et de gammes de fabrication ou gammes opératoires (routings en anglais), qui définissent la séquence des opérations d'usinage/assemblage, les ressources nécessaires (machines, outils, main d'oeuvre, qualification, énergie, fluides, etc.).

Une nomenclature multi-niveaux est le résultat de nomenclatures « imbriquées ». Par exemple, pour produire une voiture, il faut un moteur, lui même composé d'un bloc moteur et de 4 pistons. On comprendra ainsi qu'avant de pouvoir assembler la voiture, il aura fallu au préalable obtenir de l'acier pour réaliser les 4 pistons, puis assembler les pistons et le bloc pour former le moteur, et finalement assembler le moteur,

les roues et le châssis.

Cette séquence est jalonnée de contraintes, de nature temporelles, physiques (volumes, quantité disponible, ressources) et financières (cash flow).

On comprendra également que l'on ne produira pas forcément les moteurs dans les mêmes quantités (taille de lot) et au même endroit que les voitures. Ainsi les notions de stockage, d'emplacement de stock, d'entrepôt et de transport interviennent.

D'autre part, une entreprise maximisera ses ventes en proposant ses produits à sa clientèle, à un délai acceptable par celle-ci. Elle pourra être tentée d'anticiper les besoins des clients et de produire et stocker massivement. Mais dans ce cas, si les prévisions s'avèrent surévaluées, le stock deviendra obsolète et ne pourra plus être vendu.

Pour éviter cela, on recherchera constamment à minimiser les valeurs en stock, en standardisant les produits et en approvisionnant au dernier moment. Citons encore deux techniques fondamentales d'approvisionnement :

- le flux poussé : dans ce contexte, une prévision de vente va engendrer l'approvisionnement de tous les composants nécessaires, lesquels vont remonter progressivement vers les stocks de produits semi-finis, puis vers les stocks de produits finis. Dans le cas de notre omelette, si le délai de fabrication est de trois jours, il faudra prévoir ce que nous voulons manger dans trois jours, et nous y tenir !
- le flux tiré : dans cette situation, le réapprovisionnement n'a lieu que lorsque le stock a été consommé. Dans le cas de notre omelette, cela signifie dans nous maintenons un petit stock d'oeufs (stock intermédiaire), et que lorsque nous les utilisons pour réaliser une omelette, nous recomplétons ce stock, par un appel sur un stock à disposition chez le fournisseur. On parlera également de juste-à-temps lorsque le stock arrive au moment de sa consommation. Notre omelette est cuisinée juste-à-temps pour le repas du soir !

Bon appétit !

<https://www.bbcgoodfood.com/recipes/1669/ultimate-french-omelette>



Fig. 5 – Omelette (Crédit bbcgoodfood.com)¹⁹

1.5 Licence

Creative Commons Licence²⁰

Cette œuvre est mise à disposition selon les termes de la Licence - This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License



<https://creativecommons.org/licenses/by-nc-nd/4.0/>

CHAPITRE 2

Le fonctionnement logistique du système Odoo V11

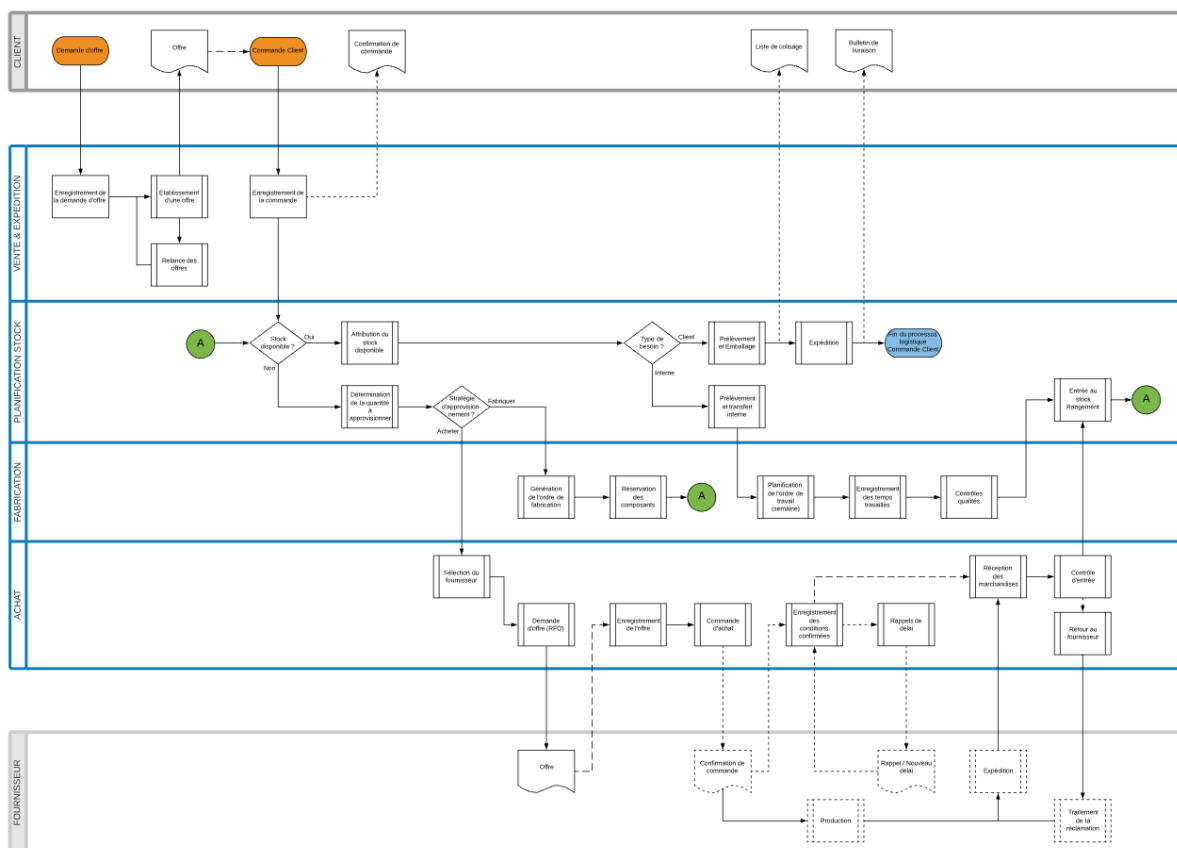
2.1 Le processus standard

2.1.1 Le processus intégré depuis la vente

Processus logistique et commercial standard (pdf) (png) (svg)

2.1.2 Démonstration

Démonstration "Enregistrement d'une commande pour un vélo"



2.2 La structure logistique

2.2.1 Entrepôt (Warehouse)

Un entrepôt est le bâtiment où les articles sont stockés.

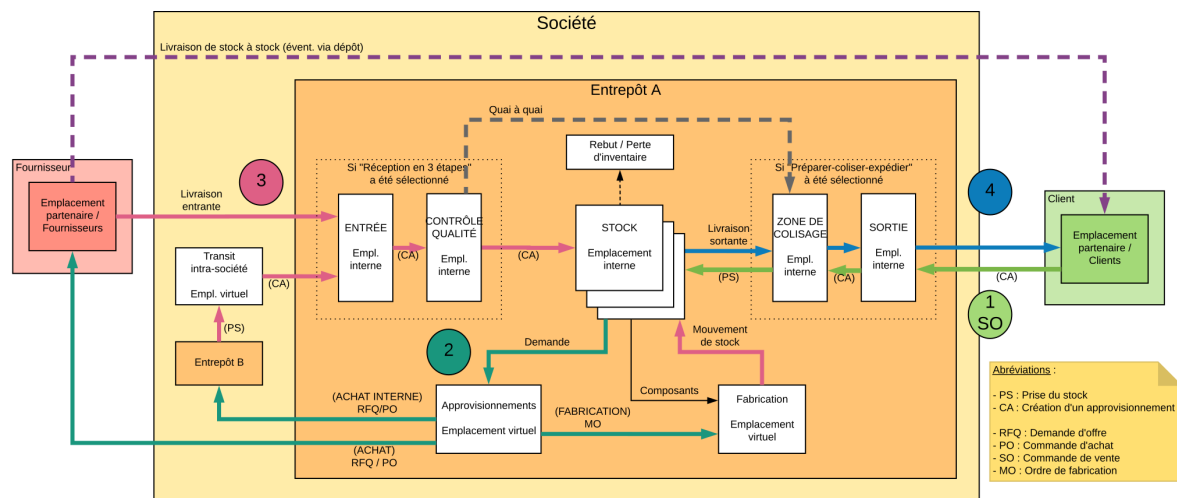
2.2.2 Emplacement de stock (Stock Location)

Un emplacement est un espace spécifique dans l'entrepôt. Il peut être considéré comme une sous-localisation de l'entrepôt, ça peut être une étagère, un plancher, une allée, etc... Par conséquent, un emplacement fait partie d'un seul entrepôt et il est impossible de relier un emplacement à plusieurs entrepôts. Il est configuré autant d'emplacements que souhaité dans un entrepôt.

2.2.3 Schéma des flux

Ce schéma décrit les emplacements de stock de stocks standards du système Odoo de base, avec leurs parents respectifs (Société, Entrepôt, Emplacement parent). Les routes entre ces emplacements sont également spécifiées.

Deux entrepôts de la même société peuvent échanger du stock en passant par une zone de transit, appartenant à la société, mais hors entrepôt.



[Routes et Emplacements de stock \(pdf\) \(svg\)](#)

Emplacement fournisseur Un emplacement virtuel qui correspond à l'emplacement d'origine des produits issus de vos fournisseurs.

Vue Emplacement virtuel utilisé pour créer des structures hiérarchiques pour votre entrepôt, en agrégeant ses emplacements enfants ; ne peut pas directement contenir de produits.

Emplacement interne L'emplacement physique dans votre entrepôt.

Emplacement client Un emplacement virtuel qui correspond à l'emplacement de destination des produits envoyés à vos clients.

Perte de stock Emplacement virtuel servant de contre-valeur aux opérations d'inventaire utilisées pour corriger les niveaux de stock (inventaires physiques).

Approvisionnement Emplacement virtuel servant de contre-valeur temporaire pour les opérations d'approvisionnement lorsque la source (four-

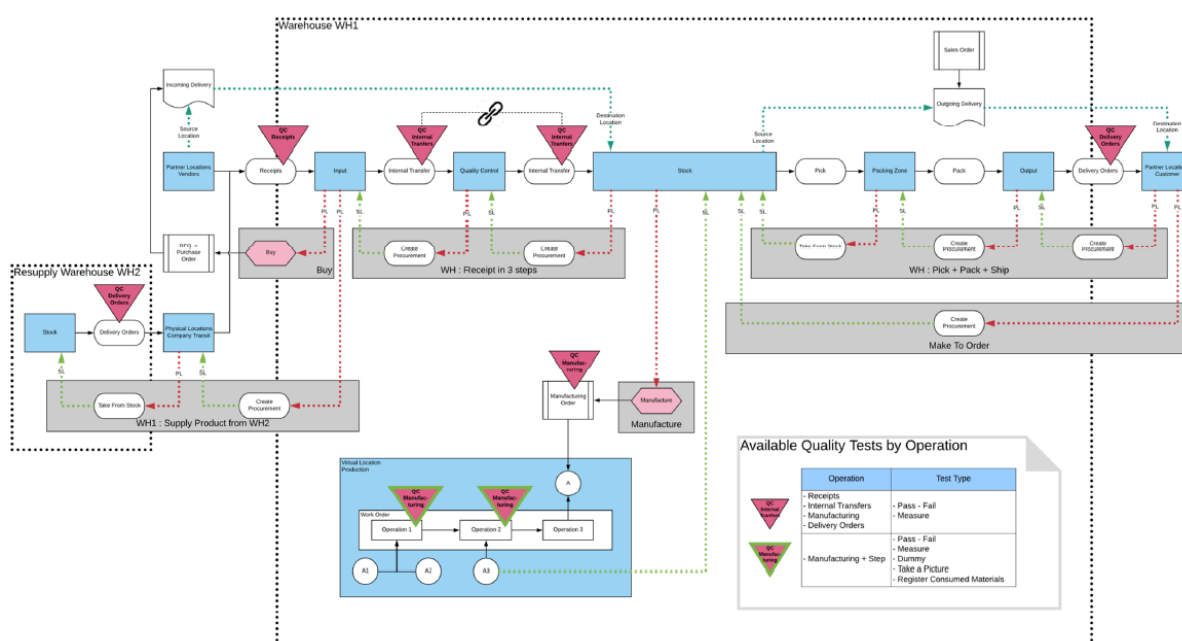
nisseur ou fabrication) n'est pas encore connue. Normalement, cet emplacement est vide une fois que le planificateur d'approvisionnement a fini de s'exécuter.

Fabrication Emplacement virtuel de contre-valeur pour les opérations de fabrication. Cet emplacement consomme des matières premières et fabrique des produits finis.

Emplacement de transit emplacement physique de contre-valeur à utiliser pour les opérations inter-entreprises et inter-entrepôts.

2.3 Les flux internes et externes

Ce schéma représente une vue détaillée des flux opérationnels internes et externes, y compris achats, inter-entrepôts et production. Les points de contrôle (QC) ainsi que les tests associés sont indiqués.



Routes et Qualité (pdf) (svg)

2.4 Les routes et les règles

2.4.1 Les routes

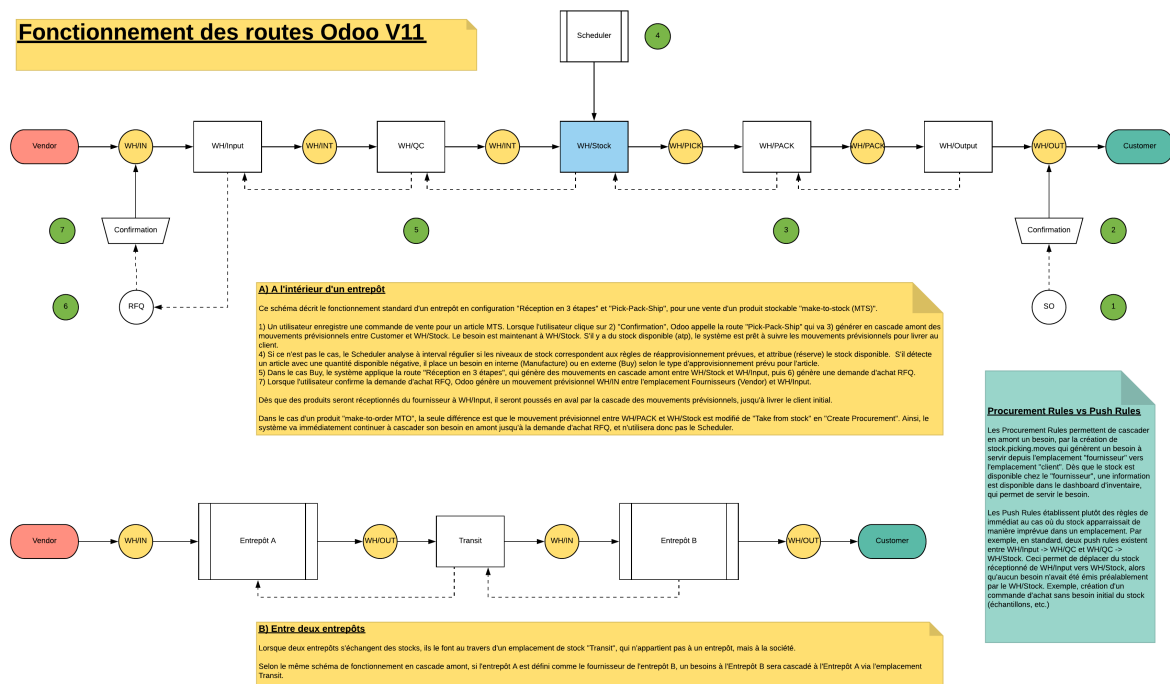
Sous Odoo, une route représente en ensemble de règles d'approvisionnement (Procurement Rules) et de règles de flux poussés (Push Rules).

Des routes peuvent être associées à un produit par :

- association directe (fichier produit)
- association par la catégorie du produit
- dans le poste de la commande client

Ce schéma représente une vue des emplacements de stock et routes/règles pour un flux transverse « Commande client » vers « Achat » et retour. Il décrit les différentes étapes pour la réalisation du besoin du client.

Fonctionnement des routes Odoo V11



Fonctionnement des routes (pdf) (svg)

2.4.2 Les règles

Les règles sont définies au niveau des routes. Il en existe deux types :

1. **Règles d'approvisionnement (Procurement Rules)** Ce type de règle crée un besoin (move_lines) entre l'emplacement de stock « demandeur » (à l'origine du besoin - Emplacement d'origine) et un emplacement « fournisseur » (qui comblera le besoin Emplacement d'approvisionnement). Lorsque l'emplacement « fournisseur » obtient du stock pour ce besoin, il le « pousse » vers son origine.
2. **Règles de flux poussés (Push Rules)** Ce type de règle permet de pousser du stock d'un emplacement à un autre, sans besoin préalable. Il peut être automatique ou nécessiter une action manuelle.

2.4.2.1 La méthode d'approvisionnement

La méthode d'approvisionnement définit si une règle :

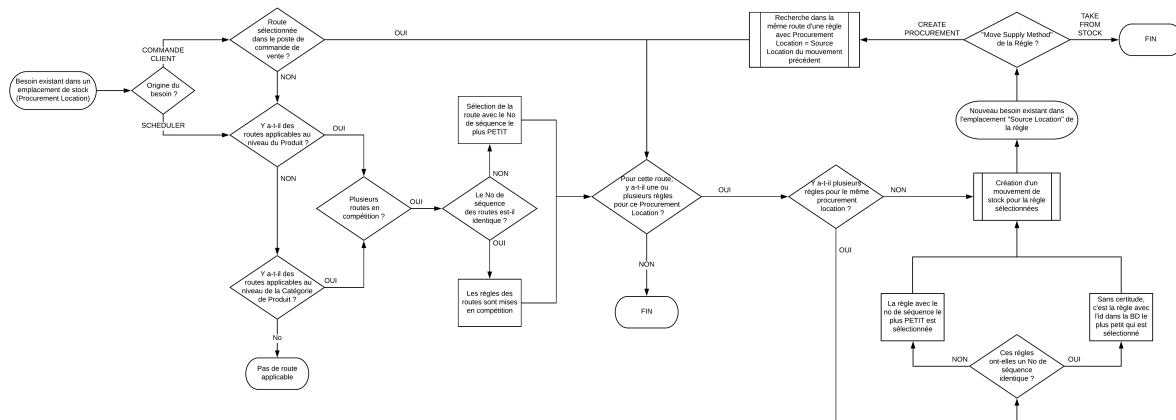
- « cascade » son besoin automatiquement en amont (Créer un approvisionnement) à une règle ayant un emplacement d'origine identique à son emplacement d'approvisionnement (MTO), ou
- puise dans le stock de l'emplacement d'approvisionnement (Prendre du stock - MTS).

2.4.3 Séquence de détermination de la route et de la règle applicable

La route sélectionnée et l'emplacement du besoin détermineront les règles applicables. Il peut toutefois arriver que des routes et règles soient en concurrence pour le même binôme « Emplacement demandeur » et « Emplacement fournisseur ». Dans ce cas, c'est la règle avec le No de séquence le plus petit qui sera sélectionnée.

Le schéma ci-dessous décrit cette séquence :

Séquence de détermination de la règle application (pdf) (svg)



2.5 La fabrication

Sous Odoo et dans sa version la plus simple, la fabrication est matérialisée par des documents de type MO (Manufacturing Order). Ces documents permettent principalement de réserver les composants dans les quantités nécessaires pour la réalisation du produit transformé.

Ces ordres de fabrication peuvent être créés manuellement, mais dans la majorité des cas, ils seront générés automatiquement par un besoin posé sur un produit associé avec la route « Manufacture » et une nomenclature.

Si la saisie des temps et le routage entre départements sont nécessaires, on créera en plus au préalable des gammes opératoires détaillant la séquence des opérations, et pour chaque opération, les composants à associer et le poste de travail concerné.

Associée à la nomenclature du produit à fabriquer/assembler, la gamme opératoire permettra d'établir des ordres de travail en relation avec un ordre de fabrication et notamment de timbrer les temps de réalisation.

En version Entreprise, des documents multimédia (instruction de montage, dessins techniques, etc.) pourront être associés aux opérations, et des points de contrôle pourront être définis afin de vérifier que les opérations ont bien été réalisées conformément aux spécifications (Module QC).

CHAPITRE 3

Etude et tests

3.1 La société fictive « Open Bike SF »

Dans le but d'assurer une certaine **cohérence des tests** à réaliser, nous utiliserons un **jeu de test** basé sur une **société fictive « Open Bike SF (OBSF) »**.

OBSF commercialise des **vélos haut de gamme**, principalement en Europe. Elle a été fondée il y a 5 ans par trois ingénieurs passionnés de cyclisme.

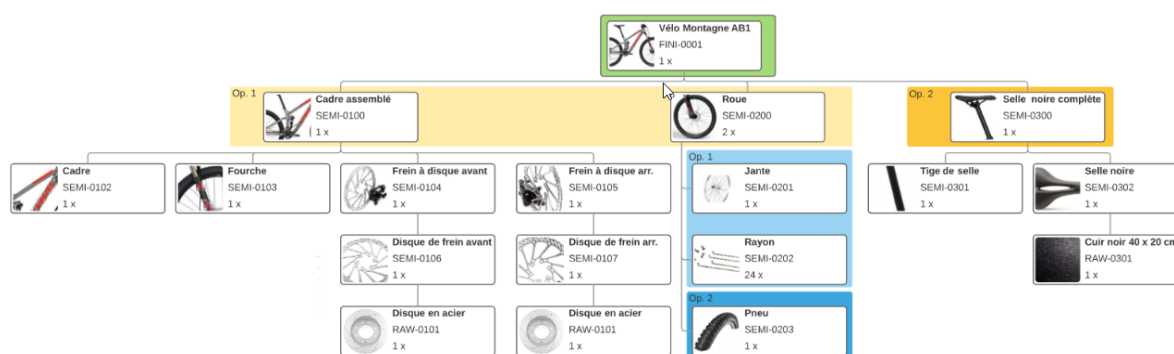
Les grands avantages des vélos OBSF sont :

- une conception robuste
- un look sportif et jeune
- un service après-vente impeccable

Les ventes connaissent une **forte croissance** depuis la création de la société. Ceci nécessite un **suivi constant des ressources** disponibles, notamment du **cash-flow** et de la **disponibilité des produits**.



Les vélos de montagne sont fabriqués entièrement dans les ateliers de Romont et de Bussigny. Les vélos de ville, par contre, sont entièrement sous-traités en Asie en mode « Fabless Manufacturing ».



Les dirigeants de OBSF souhaitent acquérir un nouveau système de gestion pour leur entreprise. Ils nous ont contacté avec un cahier des charges précis, en nous demandant d'éviter spécifiquement les points faibles de leur ancien système ERP :

3.2 A) Approvisionnement « Juste-à-temps »

A1) Le système doit permettre de commander et d'être livré « au plus tard » ou « **juste-à-temps** », c'est-à-dire au moment précis des besoins de l'ordonancement. Ceci permet notamment de **préserver les liquidités** et d'éviter les **risques d'obsolescence** du stock. A2) OBSF **maîtrise difficilement les délais** de ses produits en « **Fabless Manufacturing** », les fournisseurs **s'impliquant souvent trop peu**. Ce mode de « production à distance » implique une synchronisation et une communication constante avec les fournisseurs.

3.3 B) Traçabilité

Les vélos et certains composants doivent être **identifiés et tracés spécifiquement par des Nos de série ou de lot**. Plusieurs raisons à cela, entre autre :

- **Service après-vente :**

- Livrer la bonne pièce de remplacement, ou une alternative compatible.
- Offrir les prestations sous garantie uniquement lorsque celle-ci s'applique et éviter la fraude.
- Evaluer le taux de respect des spécifications techniques par les fournisseurs, par la remontée statistique des problèmes.

- **Rappel de produits non conformes**

- Etre en mesure d'informer et de rappeler les produits, en cas de découverte de produits non conformes, ou présentant un risques de sécurité pour l'utilisateur.

- **Contrôle des circuits de distribution**

- Etre en mesure d'identifier les revendeurs qui vendraient hors du territoire qui leur a été attribué.

B1) Le système doit permettre de connaître la **localisation d'un article** sérialisé **en tout temps**, depuis sa **première réception** chez OBSF jusqu'à la **livraison au client**, y compris à l'intérieur des stocks, des ateliers, et en

transit. Ceci est bien sûr également valable si l'article a été intégré dans un groupe ou transformé.

B2) Les employés des ateliers de montage souhaitent que leurs écrans de saisie soient **les plus conviviaux possible**, et qu'en cas d'erreur de saisie ou de composant, la **procédure de correction soit simple, rapide et intuitive**. Ceci évitera que des données erronées soient saisies dans la précipitation, et par conséquent la perte de la trace des composants.

3.4 C) Usabilité du MPS

Les ventes des produits OBSF se réalisent durant toute l'année, mais fluctuent fortement

- à certaines **périodes** et
- selon les **régions**.

Ainsi, les ventes de vélos sont

- très fortes au printemps,
- connaissent un pic avant les vacances d'été, puis
- s'affaiblissent durant l'automne et l'hiver.

Les ventes en Italie connaissent toutefois un pic de vente durant les fêtes de Noël.

Le **service après-vente** a également identifié **des pics de demande de pièces de rechange et de consommables au printemps** et dans une moindre mesure en automne. Ceci s'explique par le fait que les clients révisent leurs vélos en prévision des vacances, ou réparent après les vacances.

OBSF souhaite un outil pour gérer proprement ses prévisions de vente :

- éviter de perdre des ventes du fait de rupture de stock (vélos et composants SAV)
- éviter l'insatisfaction des clients du fait du manque de composants (SAV)
- éviter les conflits entre les différents départements de vente de vélo et le SAV en cas de stocks insuffisants ou
- faire appel à la responsabilité de chacun en cas de stocks trop importants, en conservant les prévisions de chacun

— mise à disposition d'écrans de contrôle et d'alerte, voire capable de proposer des solutions.

C1) OBSF a entendu parler du [module MPS d'Odoo Enterprise V11](#)²¹ et souhaiterait savoir dans quelle mesure il permettrait de se passer d'Excel pour couvrir les besoins précités, C2) Et comment ce système MPS réagit en cas de

- déviation de la prévision,
- de retard de production, ou
- d'imprévus.

Note : Les produits OBSF s'adressent à des clients connaisseurs. Certains sont prêts à attendre quelques semaines pour recevoir leur nouveau vélo.

OBSF a toutefois remarqué que lorsque un modèle n'est pas livrable du stock, les **ventes chutent** de près de 30%. Il est donc crucial pour OBSF d'analyser constamment les tendances du marché et de **faire des prévisions de vente** proche de la réalité. Ceci a pour conséquence l'achat anticipé de composants et le stockage de produits finis.

Le **SAV** requiert également des stock de pièces détachées. En effet, lorsque un article de réparation n'est pas disponible, les clients se fâchent souvent car ils ne peuvent pas disposer de leur vélo durant le délai de livraison.

Les **financiers** ne cessent de rappeler que le stock coûte cher ! Ceci est correct si l'on considère que le coût du stock est évalué à 15% (infrastructure, manutention, perte, obsolescence, assurances, etc). D'autre part, le cash flow peut souffrir rapidement d'un stock trop important, sachant que OBSF emprunte à 4%.

Nous avons donc ici un dilemme à traiter : - Pas de stock : moins de ventes et un risque sur l'image de marque - Trop de stock : risque financier, augmentation du prix des produits.

Nous ne nous intéresserons pas en totalité des calculs financiers liés aux stratégies de stockage, ce n'est pas le but direct de ce travail. Toutefois, afin de pouvoir

<https://www.odoo.com/slides/slide/mrp-ii-scheduler-and-master-production-schedule-422>

trouver un point optimum entre la maximisation des ventes (et du bénéfice) et la minimisation du blocage des ressources de l'entreprise, notamment financières, il est important de distinguer :

- les différents enjeux du stock :
 - cyclique : correspond au cycle « quantité demandée » * « temps de réappro »
 - sécurité : couvre les retards de livraison des fournisseurs ou une demande temporaire trop forte
 - protégé : nécessite une autorisation élevée afin d'être livré. Dans le cas d'une rupture de stock importante, on conservera une quantité minimale à la disposition exclusive des demandes les plus pressantes.
 - de synchronisation : lors de l'assemblage d'un groupe, il peut arriver qu'un composant soit en retard. Le stock des autres composants « à l'heure » forment le stock de synchronisation.
 - spéculatif : acheter plus que strictement nécessaire, afin de bénéficier d'un avantage financier ou stratégique (profiter d'un cours de change favorable, d'un rabais spécial, etc.)
 - anticipé : produire de manière anticipée afin de lisser la charge de l'entreprise (fabriquer des skis également en été afin d'être prêt à livrer dès le début de l'hiver)
 - obsolète : produit en fin de vie, peu de chance d'être vendu
 - WIP : (Work-In-Progress / Work-In-Process) : stocks qui sont actuellement dans le processus de fabrication
- les différentes typologies du stock
 - achetés/catalogue : articles du marché, revendu sans transformation par l'entreprise.
 - soutraité : article fabriqué par un tiers, sur la base de spécifications de l'entreprise.
 - matières premières : articles achetés auprès de fournisseurs, dans le but de les transformer ultérieurement.
 - semi-finis : produits en cours de fabrication, non-vendables en l'état
 - finis : produits destinés à la vente.
 - emballage : destiné à l'emballage des marchandises en prévision de leur transport.

On relevera que OBSF reçoit les demandes de clients par deux canaux distinct :

- **des commandes de vélos finis.**
 - planifiable (prévisions des vente, stratégie de pénétration)
 - fluctue en fonction des saisons
 - **des commandes de pièces détachées.**
 - rupture de stock a un fort impact négatif sur la renommée de la marque
 - partiellement planifiable (consommation passées, estimation du niveau d'usure du parc de vélos en clientèle, alerte sur faiblesse identifiée, etc.)
-

3.5 Les scénarios de test

3.5.1 A1 - MTS : Approvisionnement « Juste à temps » MTS

3.5.1.1 Objectifs du test

- Vérifier que le système achète et fabrique au plus tard en fonction des conditions suivantes :
 - Aucun stock
 - Article en gestion MTS
 - Règles de réapprovisionnement à zéro (Min : 0 / Max : 0 / Multiple : 1 / 0 days to purchase)
 - Pas de délai commercial standard (délai indicatif pour le client)
- Vérifier globalement la mécanique d'achat / fabrication.

3.5.1.2 Conditions de test

- Odoo Enterprise Version Odoo 11.0+e

- Modules installés
 - Inventory
 - Manufacturing
 - Sales
 - Purchase Management
- Paramètres activés
 - Sales/Delivery Date : Activé, de manière a pouvoir placer une ligne de commande à la date souhaitée
 - Inventory/Reservation : pas modifié, pour info sur Immediately after sales order confirmation
 - Inventory/Traceability
 - Lots & Serial Number : Activé
 - Expiration Dates : Non
 - Consignment : Non
 - Inventory/Warehouse
 - Storage Locations : Activé, pour que le scénario soit réaliste
 - Multi-warehouses : Non
 - Multi-step Routes : Non
 - Inventory/Advanced Scheduling
 - Security Lead Time for Sales : Non
 - No Rescheduling Propagation : Non
 - Manufacturing :
 - Work Orders & Quality : Activé
 - Purchases :
 - Invoicing/Bill Control : Delivered quantities
 - Products/Vendor Pricelists : Activé (pour pouvoir importer les informations des fournisseurs)
 - Dropshipping : Activé
- Données de base selon fichiers ci-dessous :
 - Produits avec routes MTS (product_template)
 - Partenaires (res_partner)
 - Infos Fournisseurs
 - Centres de travail
 - Gammes opératoires

- Nomenclatures
- Points de commande
- Remarque : pas de stocks !

3.5.1.3 Conditions de réussite

- L'ordonnancement du produit commandé est réalisé par un calcul amont des délais cumulés de **fabrication** / **assemblage** ou **achat** en fonction de la date requise par le client et des stocks disponibles (en l'occurrence pas de stocks !).
- Si la date requise par le client est trop courte, un ordonnancement aval est réalisé et une date réaliste de livraison est calculée.
- Les délais des approvisionnements sont positionnés en fonction des dates des besoins qui les concernent (date du début de l'opération de l'ordre de fabrication MO).
- Les quantités à approvisionner sont correctes (qté en besoin, min. qté minimale d'achat)

3.5.1.4 Procédure de test

- Base de données : dst-td-scenario_a1_v01-test
- Enregistrement d'une commande de vente pour 1x vélo de montagne rouge AB1 (ref. FINI-0001) pour livraison dans 102 jours calendaires.
 - Date de commande : 13.07.2018
 - Date de livraison requise : 23.10.2018 (+102 jours)
- INFO : Odoo identifie que le stock de vélos est à 0 et informe l'utilisateur
- INFO : Une livraison est générée à la date le livraison requise
- ACTION : Run Scheduler (« When you run the schedulers, Odoo tries to reserve the available stock to fulfill the existing pickings and verify if some reordering rules should be triggered. »)
- INFO : Un Ordre de Fabrication a été généré pour le vélo, début planifié dans -10 jours !

3.5.1.5 Résultats

- Fichier de résultats
- [OK] La date de livraison déterminée par le système est correcte, étant entendu que la date de livraison requise par le client (+102 jours) était supérieure au délai de production du vélo sans stocks (chemin critique de 93 jours).
- [KO] L'ordre de fabrication principal (pour le vélo FINI-0001) a été positionné dans le passé à -10 jours, au lieu de +102 jours.
- [KO] Le scheduler ne traite que le premier niveau de besoin. Dans le cas de notre vélo, il faut relancer le scheduler 5 fois afin de générer l'ensemble des MO et PO.
- [KO] Tous les besoins sont planifiés dans le passé, alors qu'ils pourraient être positionnés au plus tard, en fonction de la hiérarchie des besoins.
- [OK] Les quantités en besoin sont correctes.

3.5.1.6 Faiblesses identifiées

- Si un besoin est posé pour un article MTS qui n'a pas de Règle de réapprovisionnement, il semblerait que ce besoin soit ignoré par le scheduler, sans aucune information à l'utilisateur. Il est bien entendu possible de générer un ordre d'approvisionnement manuellement, mais le risque est important que le besoin soit perdu de vue et provoque des perturbations dans l'ordonnancement.
- Le scheduler doit « tourner » plusieurs fois avant que l'ensemble des ruptures de stock ne soient identifiées. Ceci a pour conséquence que si le scheduler est exécuté automatiquement toutes les 24 heures, les composants de dernier niveau de notre vélo ne seront commandés que 4 jours plus tard.

3.5.1.7 Commentaire

- Le mode de gestion Make-To-Stock MTS d'Odoo est dédié au réapprovisionnement du stock « au plus tôt ». Ce n'est pas un besoin à une date qui réclame du stock, c'est une quantité insuffisante dans le stock. Il ne prend pas en considération la date du besoin.
- Fort de ce constat, je décide d'évaluer le système Make-To-Order MTO afin de vérifier s'il correspond mieux au besoin de produire au plus tard.

3.5.2 A1 - MTO : Approvisionnement « Juste à temps » MTO

3.5.2.1 Objectifs du test

- Vérifier que le système achète et fabrique au plus tard en fonction des conditions suivantes :
 - Aucun stock
 - Article en gestion MTO
 - Règles de réapprovisionnement à zéro (Min : 0 / Max : 0 / Multiple : 1 / 0 days to purchase)
 - Pas de délai commercial standard (délai indicatif pour le client)
- Vérifier globalement la mécanique d'achat / fabrication.

3.5.2.2 Conditions de test

- Système identiques au test A1 - MTS, sauf
 - Tous les articles en MTO

Operations

Routes

<input type="checkbox"/>	Drop Shipping
<input checked="" type="checkbox"/>	Manufacture
<input type="checkbox"/>	Buy
<input checked="" type="checkbox"/>	Make To Order

- Produits avec routes MTO

3.5.2.3 Conditions de réussite

- L'ordonnancement du produit commandé est réalisé par un calcul amont des délais cumulés de **fabrication** / **assemblage** ou **achat** en fonction de la date requise par le client et des stocks disponibles (en l'occurrence pas de stocks !).
- Si la date requise par le client est trop courte, un ordonnancement aval est réalisé et une date réaliste de livraison est calculée.
- Les délais des approvisionnements sont positionnés en fonction des dates des besoins qui les concernent (date du début de l'opération de l'ordre de fabrication MO).
- Les quantités à approvisionner sont correctes (qté en besoin, min. qté minimale d'achat)

3.5.2.4 Procédure de test

- Base de données : dst-td-scenario_a1_v02-test
- Enregistrement d'une commande de vente pour 1x vélo de montagne rouge AB1 (ref. FINI-0001) pour livraison dans 102 jours calendaires.
 - Date de commande : 30.07.2018
 - Date de livraison requise : 10.11.2018 (+102 jours)
- INFO : Une livraison est générée à la date le livraison requise
- ACTION : Run Scheduler (« When you run the schedulers, Odoo tries to reserve the available stock to fulfill the existing pickings and verify if some reordering rules should be triggered. »)
- INFO : Un Ordre de Fabrication a été généré pour le vélo, début planifié dans -10 jours !

3.5.2.5 Résultats

Fichier de résultats

- [OK] La date de livraison déterminée par le système est correcte, étant entendu que la date de livraison requise par le client (+102 jours) était supérieure au délai de production du vélo sans stocks (chemin critique de 93 jours).
- [OK] L'ensemble des MO et PO ont été générés, sans nécessité de faire tourner le scheduler.
- [OK] Les quantités en besoin sont correctes.
- [KO] La plupart des besoins ont été placés correctement dans le temps. Toutefois, la date d'achat de l'article RAW-0301 a été posée à 0 jours de la date planifiée, alors que la fiche fournisseur indique 40 jours. La raison était que la fiche fournisseur prévoyait une quantité minimale d'achat de 100 pces, alors que le besoin MTO n'était que d'une pièce. Une deuxième fiche avec quantité minimale à 1 a résolu le problème du délai.
- [A vérifier] Pour la commande PO000001, le système a regroupé 3 besoins sous un RFQ au même fournisseur, alors que leurs dates de besoin sont différentes. Quelles sont les règles de regroupement ?

3.5.2.6 Faiblesses identifiées

- [A1_MTO_F01]²² Dans le cas où la seule fiche fournisseur indique une quantité minimale d'achat supérieure au besoin MTO, le système détermine correctement le fournisseur, mais pas le délai, ni le prix. Ceci a pour conséquence que l'achat pourrait être tardif et engendrer un retard de livraison.
- [A1_MTO_F02]²³ Lors du test, une erreur de manipulation m'a contraint à supprimer la commande de vente. De manière surprenante, les MO et PO y relatifs n'ont pas été supprimés.
- [A1_MTO_F03]²⁴ La suppression d'une commande de vente pour article MTO ne réactualise pas le forecast de l'article.
- En cas de suppression d'un PO pour un article MTO, le système ne le régénère que si l'article dispose d'une règle de réapprovisionnement et

<https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36>

<https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36>

<https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36>

en faisant tourner le scheduler. Le besoin se placera alors selon les règles MTS, soit « au plus tôt »

- Le système génère les RFQ sans tenir compte des quantités économiques (paliers de prix des infos fournisseurs).
- Le système réordonne les Infos Fournisseurs à la sauvegarde du produit. Quelles sont ces règles et l'impact ?

3.5.2.7 Commentaire

- Le mode de gestion Make-To-Order MTO d'Odoo est dédié au réapprovisionnement des besoins stricts. Il ne prend pas en considération les données d'une fiche fournisseur existante.

3.5.3 B1 : Quality Control

3.5.3.1 Objectifs du test

- Vérifier que le système enregistre correctement les Nos de série et Nos de lots durant tout le processus logistique :
 - Articles sérialisés
- Vérifier globalement la succession logique des écrans de saisie.

3.5.3.2 Conditions de test

- Odoo Enterprise Version Odoo 11.0+e
- Modules installés
 - Inventory
 - Manufacturing
 - Sales
 - Purchase Management

- Paramètres activés
 - Sales/Delivery Date : Activé, de manière a pouvoir placer une ligne de commande à la date souhaitée
 - Inventory/Reservation : pas modifié, pour info sur Immediately after sales order confirmation
 - Inventory/Traceability
 - Lots & Serial Number : Activé
 - Expiration Dates : Non
 - Consignment : Non
 - Inventory/Warehouse
 - Storage Locations : Activé, pour que le scénario soit réaliste
 - Multi-warehouses : Non
 - Multi-step Routes : Non
 - Inventory/Advanced Scheduling
 - Security Lead Time for Sales : Non
 - No Rescheduling Propagation : Non
 - Manufacturing :
 - Work Orders & Quality : Activé
 - Purchases :
 - Invoicing/Bill Control : Delivered quantities
 - Products/Vendor Pricelists : Activé (pour pouvoir importer les informations des fournisseurs)
 - Dropshipping : Activé
- Données de base selon fichiers ci-dessous :
 - Produits avec routes MTS (product_template)
 - Partenaires (res_partner)
 - Infos Fournisseurs
 - Centres de travail
 - Gammes opératoires
 - Nomenclatures
 - Points de commande

3.5.3.3 Conditions de réussite

- Tous les mouvements de stock réalisés sur des éléments sérialisés sont répertoriés
- Une arborescence démontrant la trace d'un produit sérialisé permet de retrouver l'ensemble des documents parcouru par un article sérialisé, depuis le fournisseur au client.

3.5.3.4 Procédure de test

- Base de données : dst-td-raw05-test

3.5.3.5 Résultats

Vidéo de démonstration du flux²⁵

Vidéo de démonstration du flux (MP4)

Rapport de traçabilité (pdf)

- [OK] Les nos de série sont réclamés et enregistrés à chaque mouvement de stock.
- [OK] La séquence des écrans de saisie dans les ordres de travail est correcte.
- [KO] Le rapport n'est pas éclaté correctement.

3.5.3.6 Faiblesses identifiées

- Le rapport qui ne s'éclate pas correctement (n'éclate qu'un niveau apparemment)

<https://nextcloud.open-net.ch/index.php/s/HYtXQgAbbLfz3pw>

Upstream Traceability

PRODUCT	LOT/SERIAL NUMBER	DATE	QUANTITY	LOCATION
▼ [FINI-0001] Vélo de montagne AB1 rouge	0000006	2018-08-13 08:19:13	1.000 Unit(s)	Customers
WH/OUT/00001		2018-08-13 08:19:13	1.000 Unit(s)	Stock -> Customers ↓
▼ [FINI-0001] Vélo de montagne AB1 rouge		2018-08-13 08:18:19	1.000 Unit(s)	Production -> Stock ↓
▼ [SEMI-0200] Roue tous terrains		2018-08-13 08:18:19	2.000 Unit(s)	Production ↓
MO/00001		2018-08-13 08:18:19	2.000 Unit(s)	Stock -> Production ↓
▼ [SEMI-0200] Roue tous terrains		2018-08-13 08:12:28	2.000 Unit(s)	Production -> Stock ↓
▼ [SEMI-0202] Rayon		2018-08-13 08:12:28	48.000 Unit(s)	Production ↓
MO/00002		2018-08-13 08:12:28	48.000 Unit(s)	Stock -> Production ↓
WH/IN/00003		2018-08-13 08:09:32	48.000 Unit(s)	Vendors -> Stock ↓
▼ [SEMI-0201] Jante		2018-08-13 08:12:28	2.000 Unit(s)	Production ↓
MO/00002		2018-08-13 08:12:28	2.000 Unit(s)	Stock -> Production ↓
WH/IN/00003		2018-08-13 08:09:32	2.000 Unit(s)	Vendors -> Stock ↓
▼ [SEMI-0203] Pneu tous terrains		2018-08-13 08:12:28	2.000 Unit(s)	Production ↓
MO/00002		2018-08-13 08:12:28	2.000 Unit(s)	Stock -> Production ↓
WH/IN/00004		2018-08-13 08:09:48	2.000 Unit(s)	Vendors -> Stock ↓
▼ [SEMI-0100] Cadre assemblé rouge	0000005	2018-08-13 08:18:19	1.000 Unit(s)	Production ↓
MO/00001		2018-08-13 08:18:19	1.000 Unit(s)	Stock -> Production ↓
▼ [SEMI-0100] Cadre assemblé rouge		2018-08-13 08:17:12	1.000 Unit(s)	Production -> Stock ↓
▼ [SEMI-0103] Fourche télescopique jaune		2018-08-13 08:17:12	1.000 Unit(s)	Production ↓
MO/00005		2018-08-13 08:17:12	1.000 Unit(s)	Stock -> Production ↓
WH/IN/00001		2018-08-13 08:08:23	1.000 Unit(s)	Vendors -> Stock ↓
▼ [SEMI-0102] Cadre rouge	1235	2018-08-13 08:17:12	1.000 Unit(s)	Production ↓
MO/00005		2018-08-13 08:17:12	1.000 Unit(s)	Stock -> Production ↓
WH/IN/00005		2018-08-13 08:10:15	1.000 Unit(s)	Vendors -> Stock ↓
▼ [SEMI-0104] Frein à disque avant	0000003	2018-08-13 08:17:12	1.000 Unit(s)	Production ↓
MO/00005		2018-08-13 08:17:12	1.000 Unit(s)	Stock -> Production ↓
▼ [SEMI-0104] Frein à disque avant		2018-08-13 08:15:25	1.000 Unit(s)	Production -> Stock ↓
▼ [SEMI-0106] Disque de frein avant	0000002	2018-08-13 08:15:25	1.000 Unit(s)	Production ↓
MO/00006		2018-08-13 08:15:25	1.000 Unit(s)	Stock -> Production ↓
▼ [SEMI-0106] Disque de frein avant		2018-08-13 08:14:04	1.000 Unit(s)	Production -> Stock ↓
▼ [RAW-0101] Disque en acier	1234	2018-08-13 08:14:04	1.000 Unit(s)	Production ↓
MO/00007		2018-08-13 08:14:04	1.000 Unit(s)	Stock -> Production ↓
WH/IN/00002		2018-08-13 08:09:13	2.000 Unit(s)	Vendors -> Stock ↓

3.5.3.7 Commentaire

- Le mandant connaît déjà le problème sur l'éclatement du rapport. Ce phénomène est corrigé par un fix en attendant que l'éditeur ne corrige ce qui semble être un bug.

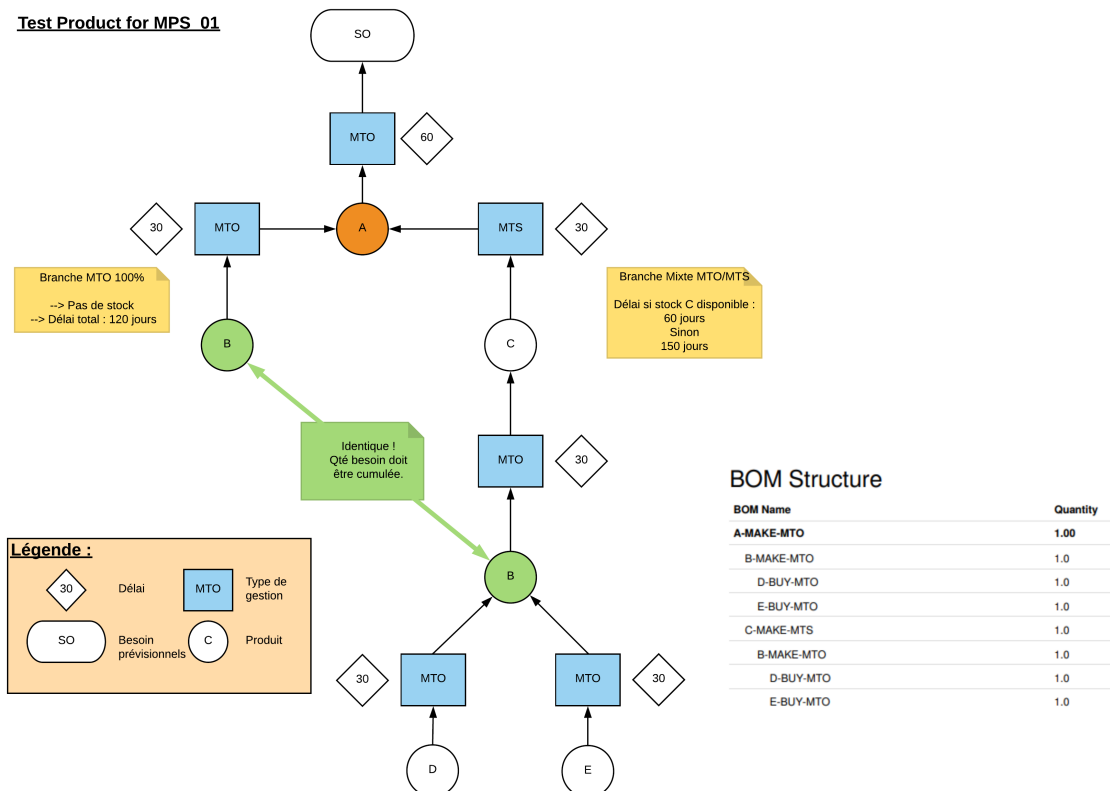
3.5.4 C1 - MPS : Approvisionnement basé sur une prévision de vente

3.5.4.1 Objectifs du test

- Vérifier que les faiblesses constatées lors les tests A1_MTO et A1_MTS sont corrigées par le module MPS
- Vérifier globalement la mécanique du module MPS.

3.5.4.2 Conditions de test

- Système identiques au test A1 - MTS, avec ajout des articles spécifiques suivants :



3.5.4.3 Conditions de réussite

- Les prévisions réalisables au délai souhaité sont validées
- Les prévisions non réalisables au délai souhaité sont repoussées à un délai réalisables
- L'utilisateur en est informé - par exemple par des codes de couleur
- Lors ordres d'approvisionnement dans l'horizon du besoin sont générés

3.5.4.4 Procédure de test

- Base de données : dst-td-mps-eval01-test

3.5.4.5 Résultats

Vidéo du test MPS²⁶

Vidéo du test MPS (MP4)

- [OK] Les prévisions sont enregistrées et transmises à la planification
- [KO] La planification est effectuée avec les mêmes faiblesses que constatées lors les tests A1_MTO et A1_MTS

3.5.4.6 Faiblesses identifiées

- Identiques à A1_MTO et A1_MTS

3.5.4.7 Commentaire

- Le module MPS nécessite que l'ensemble des articles à planifier soient enregistrés, y compris les composants. Ceci constitue un risque d'oubli, alors que seuls les articles de tête ne devraient être planifiés et les besoins de leurs composants cascades.
- Le module MPS génère des besoins prévisionnels. Il ne vérifie pas ultérieurement si cette prévision a été respectée.

<https://nextcloud.open-net.ch/index.php/s/Qi5JayK3ARzG3cw>

CHAPITRE 4

Résultats de l'étude

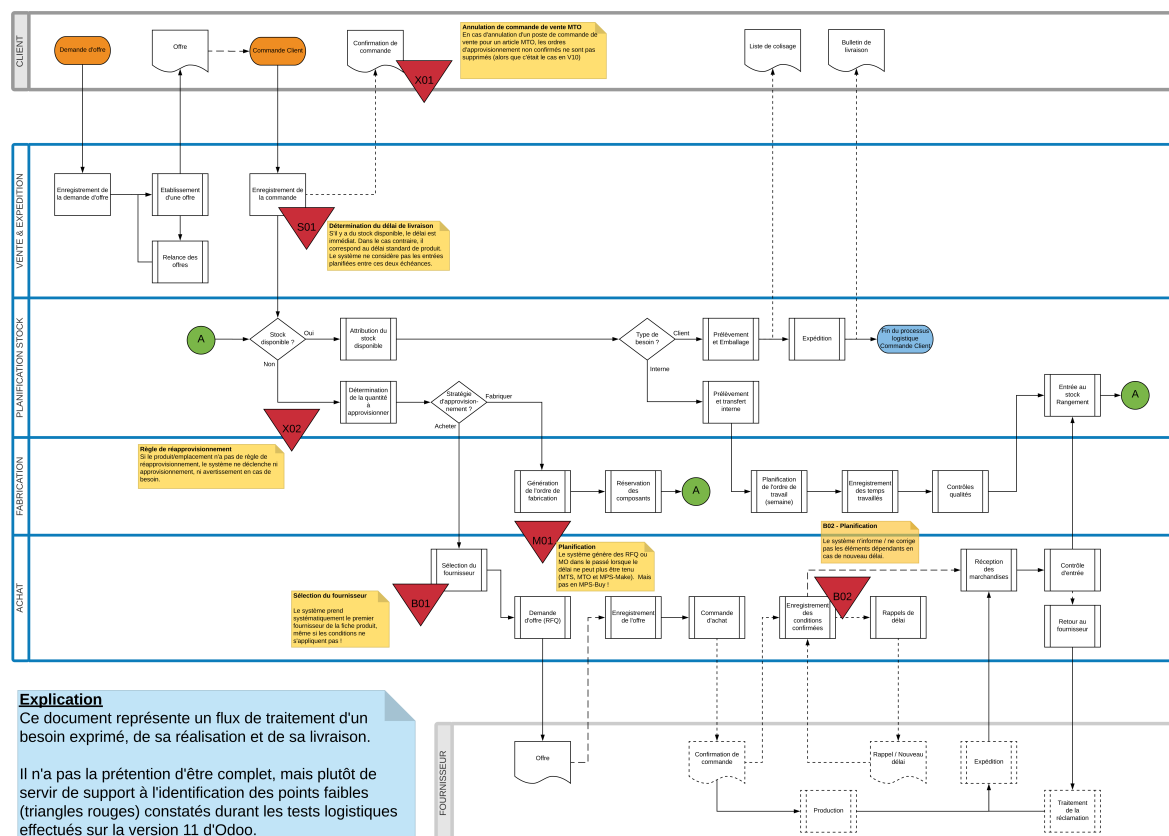
4.1 Faiblesses identifiées

Afin de pouvoir évaluer et qualifier l'importance des faiblesses identifiées durant les tests, j'ai décidé de dessiner un flux transversal client - fournisseur, que j'ai **arbitrairement** considéré comme standard.

Ceci m'a permis de placer sur ce schéma les points faibles identifiés, de les qualifier et de comparer ma compréhension de la criticité du point avec celle de mon mandant.

Voici cette carte, que nous appellerons « Situation AS-IS » (telle quelle), puisque c'est l'état constaté du système actuel. Chaque triangle rouge identifie une faiblesse constatée et précises l'élément du flux qu'elle impacte.

Situation AS-IS (pdf) (svg)



X01 - Annulation de commande de vente MTO En cas d'annulation d'un poste de commande de vente pour un article MTO, les ordres d'approvisionnement non confirmés ne sont pas supprimés (alors que c'était le cas en V10).

N.B. Il s'agit probablement d'un paramétrage manquant en version 11 standard, facilement réactivable.

S01 - Détermination du délai de livraison S'il y a du stock disponible, le délai est immédiat. Dans le cas contraire, il correspond au délai standard de produit. Le système ne considère pas les entrées planifiées entre ces deux échéances.

X02 - Règle de réapprovisionnement Si le produit à un emplacement déterminé n'a pas de règle de réapprovisionnement, le système ne déclenche ni approvisionnement, ni avertissement en cas de besoin.

M01 - Planification Le système génère des RFQ ou MO dans le passé lorsque le délai ne peut plus être tenu (MTS, MTO et MPS-Make). Mais pas en MPS-Buy !

B01 - Sélection du fournisseur Le système prend systématiquement le premier fournisseur de la fiche produit, même si les conditions ne s'appliquent pas !

N.B. Ce fonctionnement est probablement souhaité par l'éditeur, afin de permettre au client de déterminer sa propre séquence de sélection du meilleur fournisseur. Mais un filtre portant notamment sur les dates de validité serait probablement judicieux.

B02 - Planification Le système n'informe / ne corrige pas les éléments dépendants en cas de nouveau délai.

4.2 Modules complémentaires à étudier

4.2.1 OCA - Odoo Community Association²⁷

- [manufacture](#)²⁸
- [manufacture-reporting](#)²⁹
- [purchase-workflow](#)³⁰
- [stock-logistics-warehouse](#)³¹
- [stock-logistics-transport](#)³²

4.2.2 Odoomrp.com³³

- [odoomrp-utils](#)³⁴
- [odoomrp-wip](#)³⁵

<https://odoo-community.org>

<https://github.com/OCA/manufacture>

<https://github.com/OCA/manufacture-reporting>

<https://github.com/OCA/purchase-workflow>

<https://github.com/OCA/stock-logistics-warehouse>

<https://github.com/OCA/stock-logistics-transport>

<http://www.odoomrp.com>

<https://github.com/odoomrp/odoomrp-utils>

<https://github.com/odoomrp/odoomrp-wip>

4.3 Décisions

Après discussion avec le mandant, nous décidons de résoudre prioritairement le point S01, car :

- On sait que les clients sont habitués à être livré rapidement. Lors d'une demande client, si un produit n'est pas livrable du stock, la probabilité que le client n'achète pas augmente proportionnellement au délai d'attente. Offrir à ce moment un délai standard, c'est mieux que rien, mais si du stock pourrait être disponible entre ces deux dates, autant en profiter et maximiser les chances de transformation en commande ferme.
- La solution au point S01 signifie l'ajout d'une possibilité de **questionner l'arbre des encours d'approvisionnement** et des stocks intermédiaires, afin d'obtenir un délai plausible.
- Cette solution présuppose que les éléments d'approvisionnement sont placés judicieusement dans le temps. Or, les faiblesses constatées M01 et B02 précisent justement que le système est en difficulté dans ce domaine.
- Nous allons donc également au préalable résoudre les points M01 et B02, par un **module de recalcul des dates des éléments d'approvisionnement en cours**.

CHAPITRE 5

Annexes

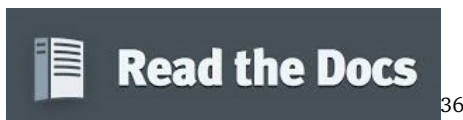
5.1 Documents

- *Routes et emplacements de stock*
- *Routes et Qualité (avec points de contrôle)*
- *Fonctionnement des routes*
- *Séquence de détermination de la règle applicable*
- *Situation AS-IS*

5.1.1 Maintenance de la documentation

5.1.1.1 Fonctionnement de la chaîne d'édition

<https://readthedocs.org>



Note : Cette documentation est générée et hébergée par le site « [Read the Docs](#) », qui permet sa mise à jour **en continu**, avec notamment les caractéristiques suivantes :

- un système de recherche intégré
 - une édition simplifiée
 - une mise en forme structurée et rigide (idéale pour des documents techniques)
 - une gestion de plusieurs versions
 - une édition en plusieurs langues
-
- Les fichiers sources sont stockés dans un répertoire de [Github.com](#).
 - Ils sont écrits en langage *reStructuredText* (extension .rst).
 - Un *webhook* est établi entre « [Read the Docs](#) » et les fichiers sources sous [Github.com](#).
 - A chaque changement d'un fichier source, le *webhook* active la régénération du site en ligne.

5.1.1.2 Prérequis

- Disposer d'un compte chez [Github.com](#) (ou similaire)
- Disposer d'un compte chez « [Read the Docs](#) » connecté aux sources du compte [Github.com](#).

Compléments

Afin d'assurer un meilleur confort d'édition, les logiciels ci-après peuvent être installés en complément :

- Un éditeur de code (*Visual Studio Code* dans notre cas)
- Le moteur d'édition *Sphinx*, qui est le moteur de génération de « *Read the Docs* ».

5.1.1.3 Mise en place d'une nouvelle documentation

- Sous *Github.com*
 - Créer un repository sous *Github.com*
 - relever L'URL DE CELUI-CI (par exemple : `https ://github.com/open-net-sarl/readthedocs.git`).
- Sous Read the Docs
 - Cliquer **Import a Project** et sélectionner un Repository
 - Cliquer **Admin > Settings**. Vérifier l'URL du champ « Repository URL ».
 - Adapter les autres champs en fonction du besoin, notamment l'édition au format PDF ou ePub, en plus de HTML
 - Cliquer **Admin > Settings > Advanced Settings**. Vérifier le niveau de confidentialité sous « Privacy Level ».

Note : Cette installation minimale permet déjà de créer une documentation en ligne en éditant des fichiers .RST (restructuredText) directement sous *Github.com*.

Ceci constitue toutefois une solution peu élégante. D'autre part, le résultat de la conversion des fichiers .rst vers html par « *Read the Docs* » est assez lente.

Afin de travailler dans de meilleures conditions, nous allons procéder en complément à l'installation d'un éditeur de code connecté à *Github.com*, ainsi qu'à l'installation du moteur de génération de documentation *Sphinx* (qui est identique à celui de « *Read the Docs* »).

5.1.1.4 Installation du système pour l'édition locale

- Installation du moteur *Sphinx*

Note : *Sphinx* a besoin de Python pour fonctionner. Sous Windows, l'installation de *Anaconda* constitue probablement l'option la plus simple.

- Télécharger et installer *Anaconda*
- Ouvrir la console « Anaconda Prompt »

```
conda install sphinx      # Installation de Sphinx
```

Pour générer un nouveau projet :

```
cd /path/to/project/
mkdir docs
cd docs
sphinx-quickstart
# Répondre aux questions en validant les valeurs proposées en cas_
↳ de doute, il est possible de reconfigurer le projet_
↳ ultérieurement.
# Vous obtiendrez un fichier de configuration de base conf.py et
# un document d'index basique, mais fonctionnel et prêt à être_
↳ édité.
```

Pour générer le site html à partir des sources existantes :

```
# pour générer les documents en html :
cd /path/to/project/docs
make html
# un dossier _build existe dorénavant, avec les fichiers sources_
↳ du site html
# vous pouvez consulter le site généré en local sous 'file:///path/
↳ to/project/docs/_build/html/index.html
```

Optionnel : pour automatiser la régénération de la documentation html à chaque modification d'un fichier source :

```
# installer les modules nécessaires
conda install -c conda-forge sphinx-autobuild
# se déplacer dans le répertoire de la documentation
cd /path/to/project/docs
# initialiser la régénération automatique
sphinx-autobuild . _build/html
# Dès maintenant, la documentation est disponible en local à l'
↳ 'adresse http://localhost:8000.
# il suffit de rafraîchir la page du navigateur (CTRL+F5) pour
↳ obtenir la nouvelle version
# après quelques secondes.
#
# interrompre cet automatisme avec CTRL+C
```

- Sous *Visual Studio Code*
 - Installer le contrôle de code source git au besoin
 - Créer un répertoire pour les documents de projet en local
 - Initialiser le répertoire pour git³⁷.
 - Connecter le git local au git distant³⁸

```
cd /home/myProject/
git remote add origin https://...
git remote show origin # if everything is ok, you will see your
↳ remote
git push -u origin master # assuming you are on the master branch.
```

Avertissement : Ne pas oublier de pousser les modifications régulièrement vers *Github.com*, sans quoi le site online ne sera pas mis à jour.

<https://www.youtube.com/watch?v=6n1G45kpU2o>

<https://stackoverflow.com/a/43364619>

5.1.2 Glossaire

Anaconda [Anaconda](#)³⁹ est une distribution libre et open source des langages de programmation Python et R, qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérés par le système de gestion de paquets conda

Voir aussi :

— [Lien de téléchargement de Anaconda Version 5.2](#)⁴⁰

Git [Git](#)⁴¹ est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.

Github [Github.com](#)⁴² est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions [Git](#).

Read The docs [Read the Docs](#)⁴³ est une plateforme d'hébergement de documentation de logiciels. Le code source est librement disponible, et l'utilisation du service est gratuit. Il produit la documentation sur la base de fichiers produits par la moteur de génération [Sphinx](#).

Voir aussi :

— [Documentation officielle Read The Docs](#)⁴⁴

reStructuredText

RST reStructuredText est un langage de balisage léger utilisé notamment dans la documentation du langage Python. Bien que sauvegardé sous un format textuel, l'extension associée est parfois indiquée comme étant RST.

Voir aussi :

— [Documentation edX très complète](#)⁴⁵

— [Documentation succincte](#)⁴⁶

<https://www.anaconda.com>

https://repo.anaconda.com/archive/Anaconda2-5.2.0-Windows-x86_64.exe

<https://fr.wikipedia.org/wiki/Git>

<https://github.com>

<https://readthedocs.org>

http://docs.readthedocs.io/en/latest/getting_started.html

<http://draft-edx-style-guide.readthedocs.io/en/latest/ExampleRSTFile.html>

<http://udig.refrains.net/files/docs/latest/user/docguide/sphinxSyntax.html>

Sphinx [Sphinx](#)⁴⁷ Sphinx est un générateur de documentation libre. Il a été développé par Georg Brandl pour la communauté Python en 2008, et est le générateur de la documentation officielle de projets tels que Python, Django, Selenium, Urwid, ou encore Bazaar.

Voir aussi :

— [Documentation officielle Sphinx](#)⁴⁸

Visual Studio code

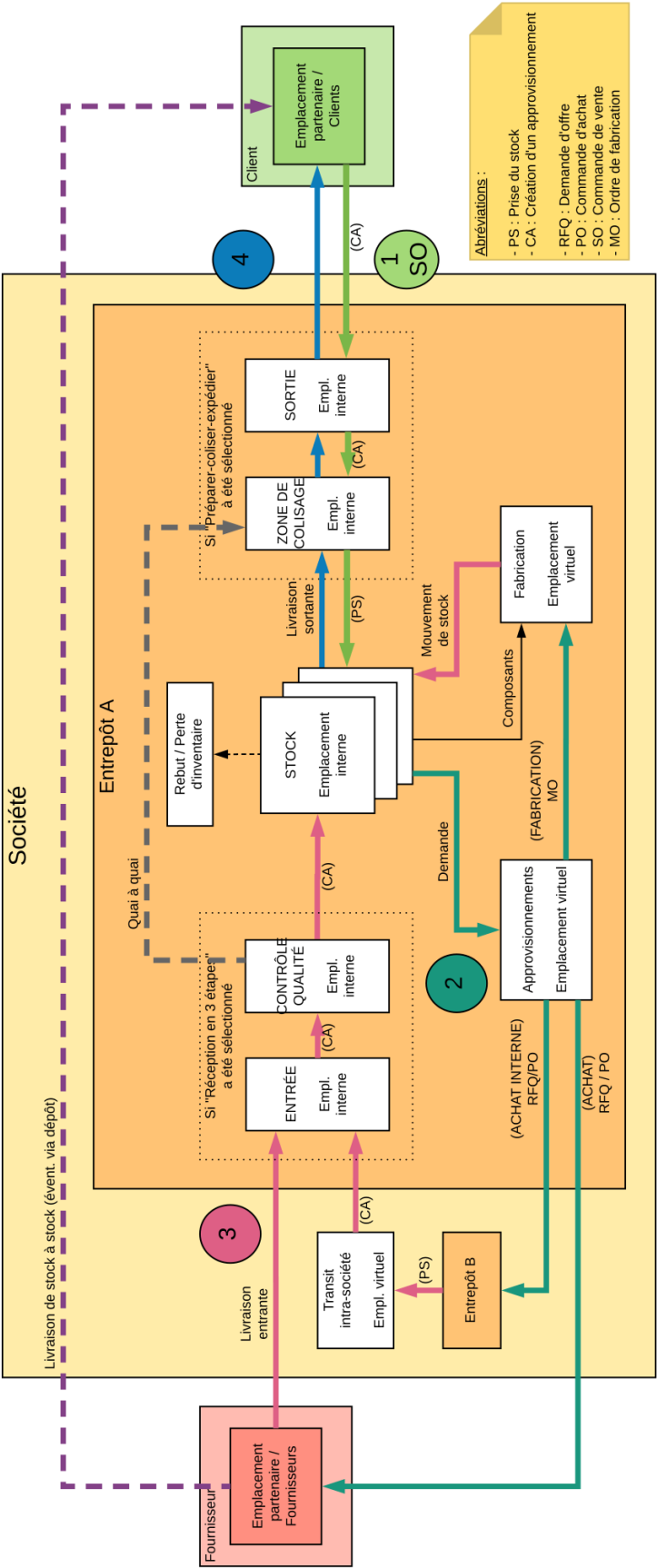
VSCode Editeur de code cross-platform, open source et gratuit, supportant une dizaine de langages. Le code source est fourni sous la licence libre MIT (plus précisément la licence Expat) sur le site du projet sur Github. En revanche, [l'exécutable](#)⁴⁹ est proposé sur le site officiel de Microsoft sous une licence privatrice.

Webhook [Définition Wikipédia](#)⁵⁰

Fonction de rappel HTTP définie par l'utilisateur, qui récupère et stocke les données issues d'un événement, en général externe à l'application.

<https://readthedocs.org>
<http://www.sphinx-doc.org/en/stable/contents.html>
<https://code.visualstudio.com/>
<https://en.wikipedia.org/wiki/Webhook>

5.2 Routes et emplacements de stock

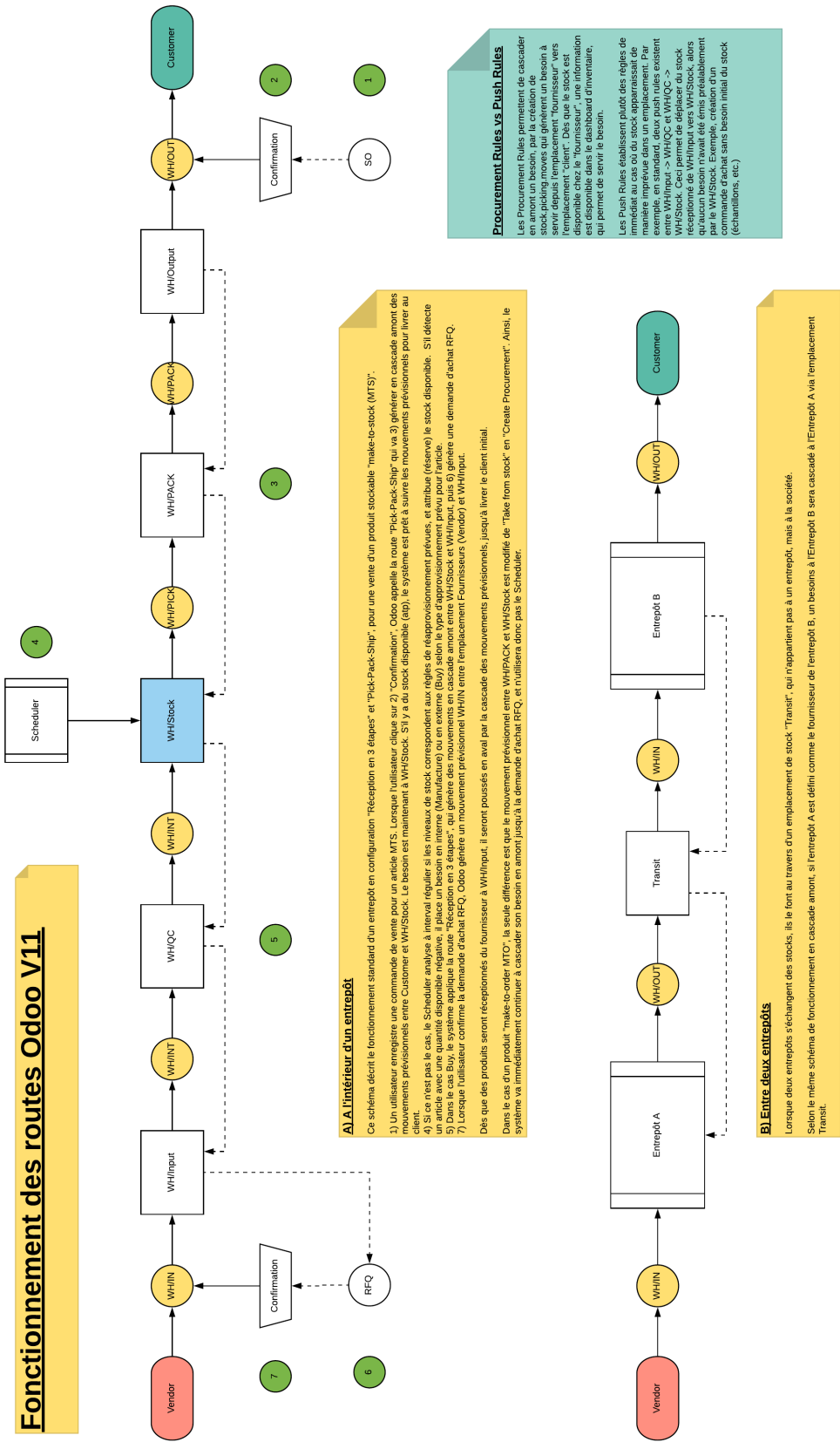


Type d'emplacement	Description
Emplacement fournisseur	un emplacement virtuel qui correspond à l'emplacement d'origine des produits issus de vos fournisseurs.
Vue	emplacement virtuel utilisé pour créer des structures hiérarchiques pour votre entrepôt, en agrégeant ses emplacements enfants ; ne peut pas directement contenir de produits.
Emplacement interne	l'emplacement physique dans votre entrepôt.
Emplacement client	un emplacement virtuel qui correspond à l'emplacement de destination des produits envoyés à vos clients.
Perte de stock	emplacement virtuel servant de contre-valeur aux opérations d'inventaire utilisées pour corriger les niveaux de stock (inventaires physiques).
Approvisionnement	emplacement virtuel servant de contre-valeur temporaire pour les opérations d'approvisionnement lorsque la source (fournisseur ou fabrication) n'est pas encore connue. Normalement, cet emplacement est vide une fois que le planificateur d'approvisionnement a fini de s'exécuter.
Fabrication	emplacement virtuel de contre-valeur pour les opérations de fabrication. Cet emplacement consomme des matières premières et fabrique des produits finis.
Emplacement de transit	emplacement physique de contre-valeur à utiliser pour les opérations inter-entreprises et inter-entrepôts.

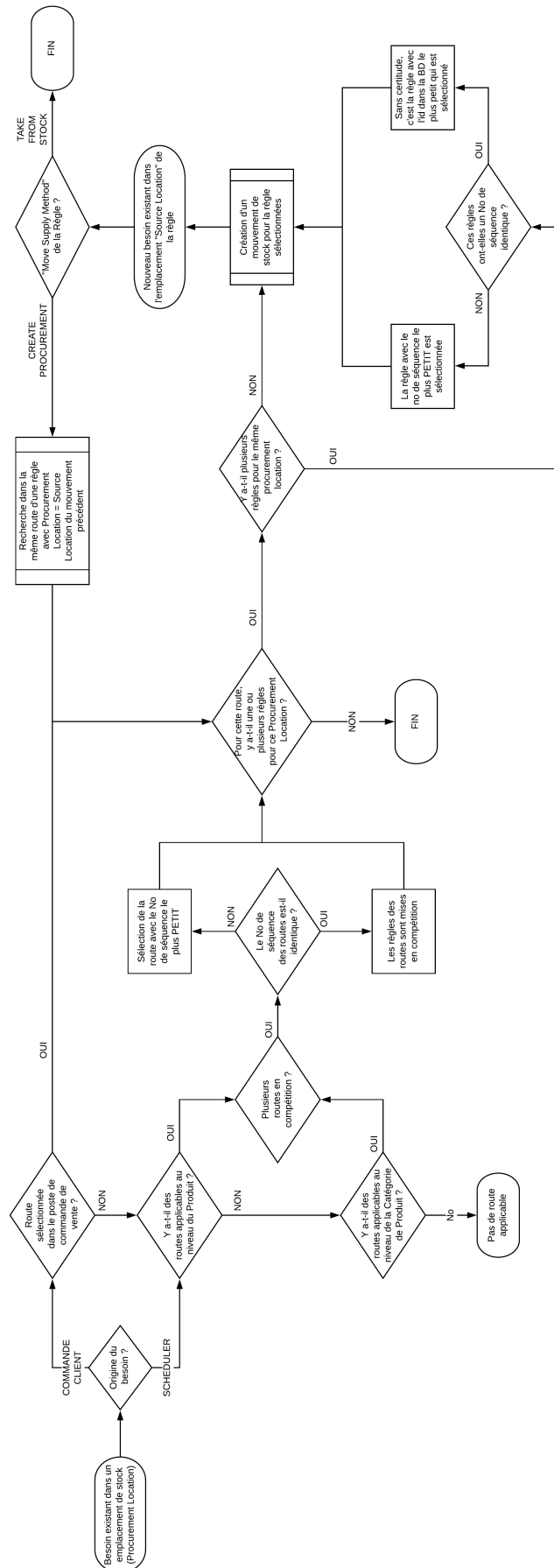
5.3 Routes et Qualité (avec points de contrôle)

5.4 Fonctionnement des routes

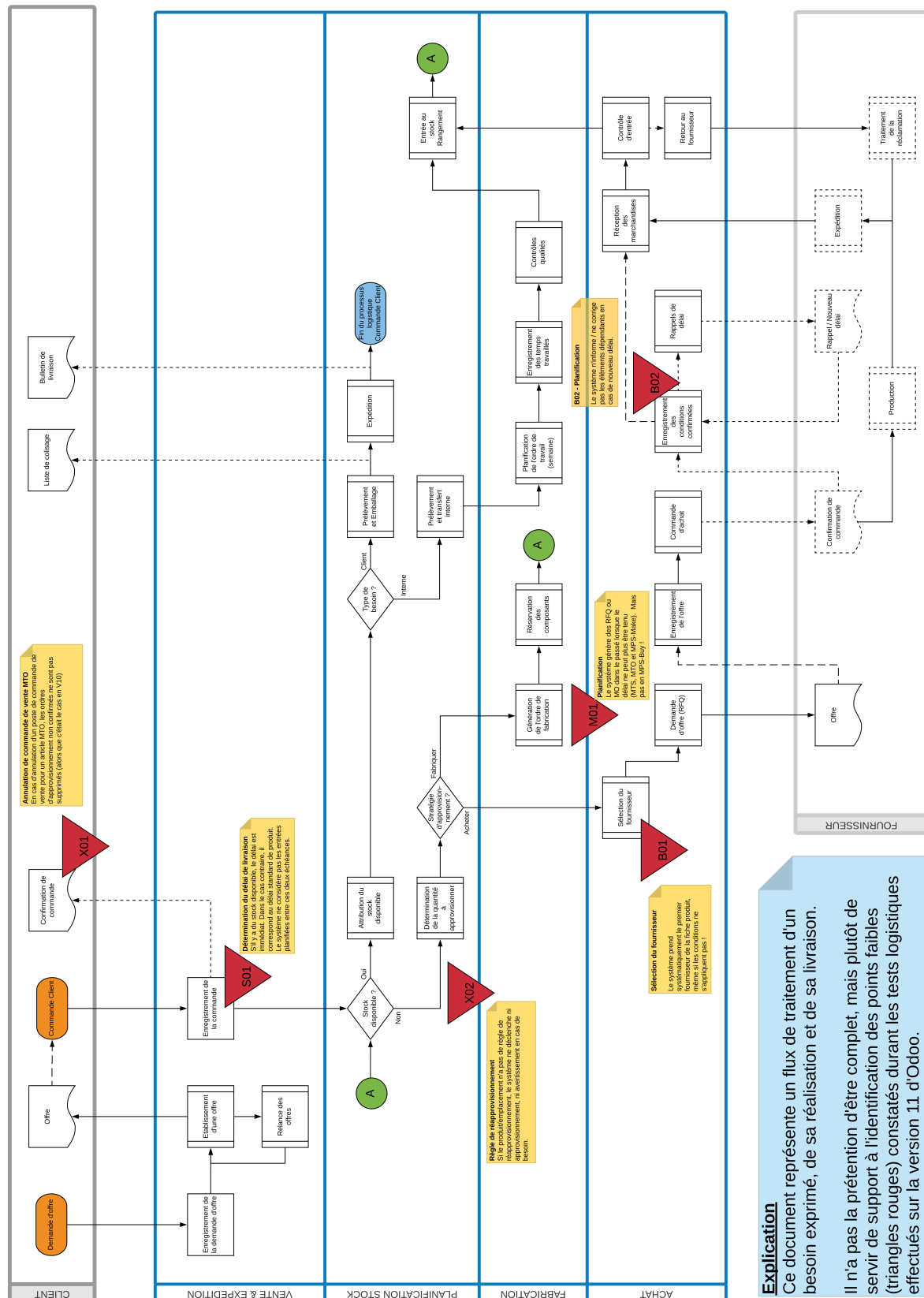
Fonctionnement des routes Odoo V11



5.5 Séquence de détermination de la règle applicable



5.6 Situation AS-IS



CHAPITRE 6

Licence

Creative Commons Licence ⁵¹

Cette œuvre est mise à disposition selon les termes de la Licence - This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License



A

Anaconda, [48](#)

C

Créer un approvisionnement, [16](#)

E

Emplacement d'approvisionnement, [16](#)

Emplacement d'origine, [16](#)

Emplacement de stock, [12](#)

Entrepôt, [12](#)

ESIG, [1](#)

F

flux, [6](#)

flux opérationnels, [14](#)

G

gammes opératoires, [17](#)

Git, [48](#)

Github, [48](#)

M

méthode d'approvisionnement, [16](#)

Manufacturing Order, [17](#)

N

nomenclature, [17](#)

O

OBSF, [19](#), [20](#)

OCA, [5](#)

Odoo, [4](#)

Odoo Community Association, [5](#)

opération, [17](#)

ordres de fabrication, [17](#)

P

paramétrage, [2](#)

points de contrôle, [14](#)

poste de travail, [17](#)

Prendre du stock, [16](#)

processus standard, [11](#)

Procurement Rules, [15](#)

Push Rules, [15](#)

R

règles d'approvisionnement, [15](#)

règles de flux poussés, [15](#)

Read The docs, [48](#)

reStructuredText, [48](#)

route, [15](#)

RST, [48](#)

S

séquence des opérations, [17](#)

Sphinx, [49](#)

Stock Location, [12](#)

stockage, [6](#)

T

transformation, [6](#)

V

Visual Studio code, [49](#)

VSCode, [49](#)

W

Warehouse, [12](#)

Webhook, [49](#)