Etude Odoo V11 Opérations

Documentation

Version 1.0

Didier Stadelmann

sept. 29, 2018

Table des matières

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 Gestion [1] de Delémont [2], 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 limphatique.

Longtemps, on a par exemple soigné la fièvre par des saignées [3], jusqu'au jour où on s'est aperçu que la situation du malade se dégradait plus qu'elle ne s'améliorait.

[1] http://www.esig-jura.ch
[2] https://en.wikipedia.org/wiki/Delémont
[3] 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 (paramètrage) suite à des actions d'utilisateurs ou de job réguliers. Quelques algorithmes bien pensés permettent aux différents reponsables 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.

[4] https://commons.wikimedia.org/wiki/File:Pratique_d'une_saignée.jpg
[5] https://www.sap.com/index.html
[6] 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. Etude et compréhension de la philosophie du système ERP
2. Etude et compréhension des besoins du client en terme de gestion, au sens large
3. Identification des 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 paramètrage
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 privilériera autant que faire ce peu 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.

Ne pas coder implique par contre une connaissance parfaite et large des capacités de l'outil en terme de paramètrage. 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 paramètrage 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 grâve 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 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.

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

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 [8], 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.

Fig. 3 – Oeuf cassé (Crédit bioalaune.com)[9]

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.

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

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`.

GUIDE DE PRÉPARATION : OMELETTE SIMPLE

1 Pers. | 10 min | 7 min | 11,21 | Facile

INGRÉDIENTS

- 3 oeufs entiers
- 30 g de gruyère râpé
- 6 branches de ciboulette ciselée
- 2 c. à soupe d'huile d'olive
- sel, poivre

PRÉPARATION

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.

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)[10]

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

[10] https://www.cuisineaz.com/recettes/omelette-simple-56045.aspx

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 chassis,

— 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, qualitification, 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 chassis.

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 temptée 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. 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 !

Fig. 5 – Omelette (Crédit bbcgoodfood.com)[11]

[11] https://www.bbcgoodfood.com/recipes/1669/ultimate-french-omelette

1.4 Licence

CHAPITRE 2

Le fonctionnement logistique du système Odoo V11

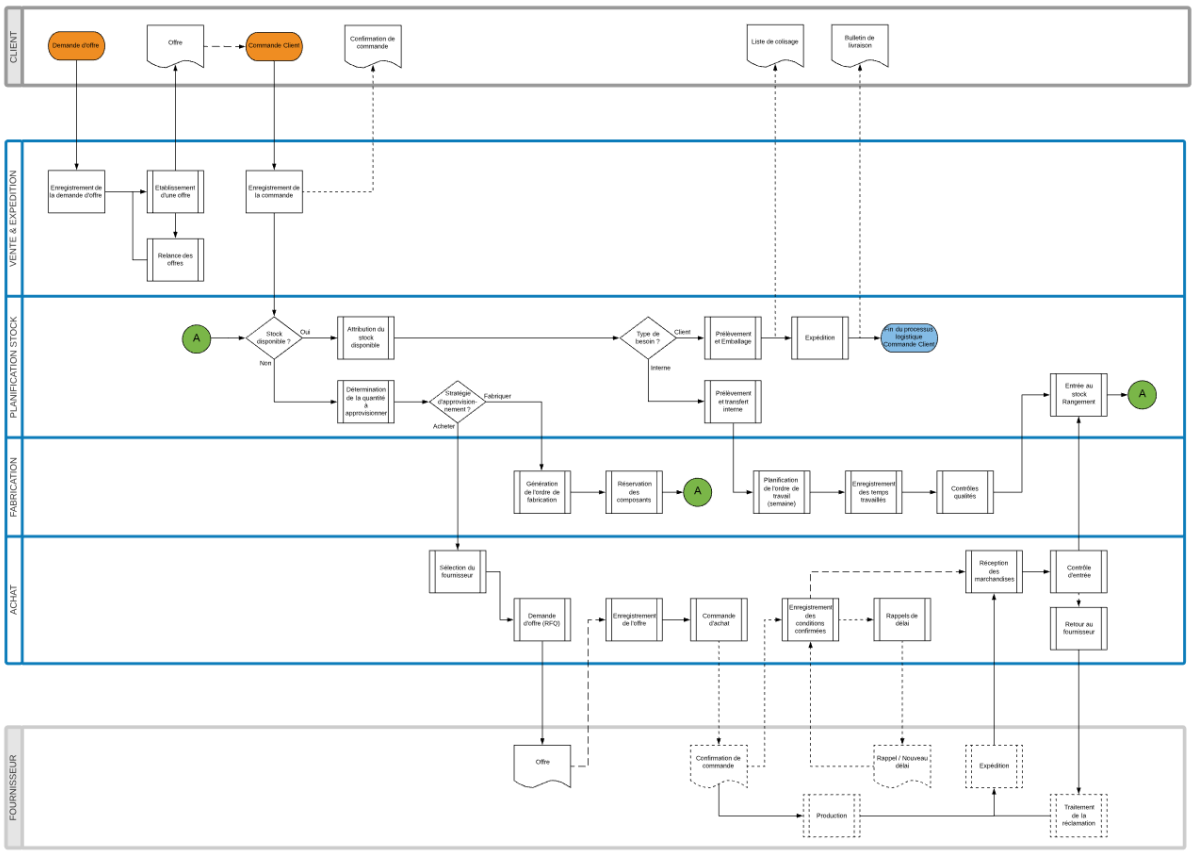
2.1 Le processus standard

2.1.1 Le processus intégré depuis la vente

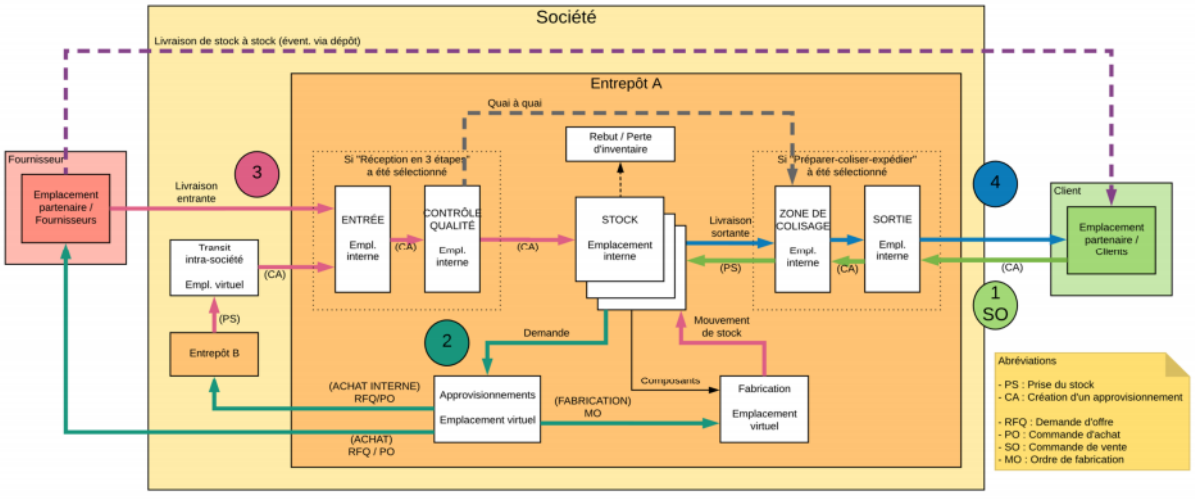
Processus logistique et commercial standard (pdf)

2.1.2 Démonstration

Démonstration "Enregistrement d'une commande pour un vélo"



2.2 La structure logistique

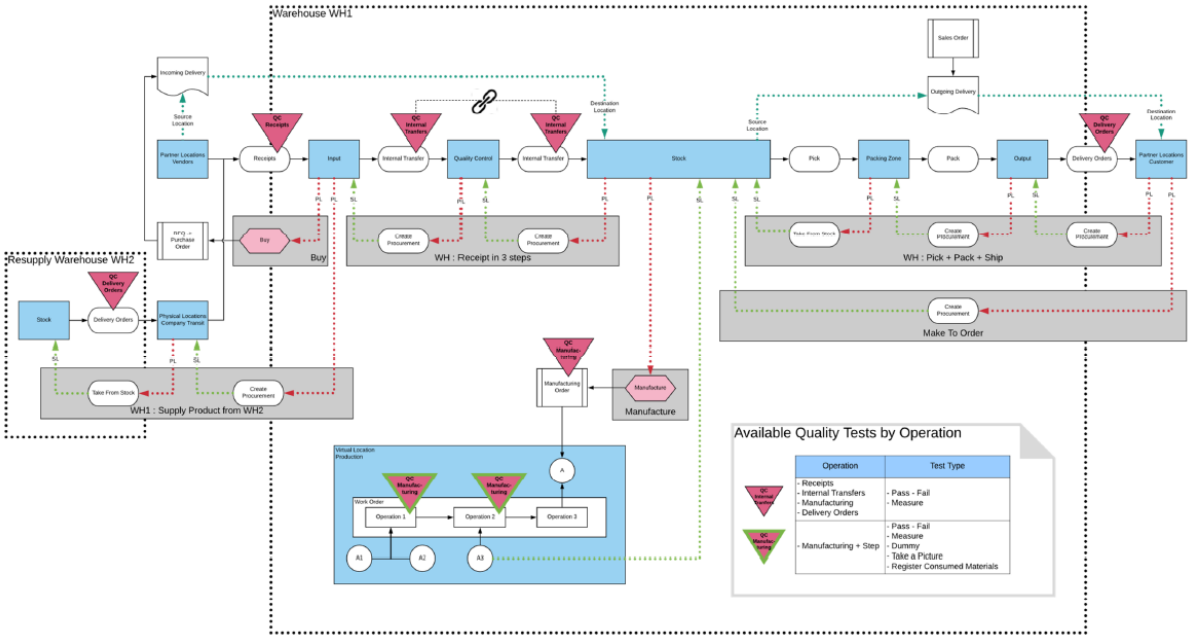


Routes et Emplacements de stock (pdf)

2.2.1 Les entrepôts

2.2.2 Les emplacements de stock

2.3 Les flux internes et externes



Routes et Qualité (pdf)

2.3.1 Les routes

2.3.2 Les règles

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 fabriqué 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'ordonnancement. 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 erronnées 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 [12] 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

[12] https://www.odoo.com/slides/slide/mrp-ii-scheduler-and-master-production-schedule-422

d'un stock trop important, sachant que OBSF emprunte à 4%.

Nous avons donc ici un dilemne 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 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 un 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 transformés 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
  - Régles 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 livraision 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

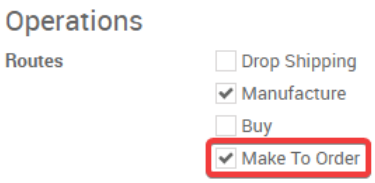
3.5.2.1 Objectifs du test

— Vérifier que le système achète et fabrique au plus tard en fonction des conditions suivantes :
  — Aucun stock
  — Article en gestion MTO
  — Régles 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

      — `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 livraision 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 à 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] [13] 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] [14] 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] [15] La suppression d'une commande de vente pour article

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

MTO ne réactualise pas le forecast de l'article.

— En cas de suppression d'un PO pour un article MTO, le système ne le régénèra que si l'article dispose d'une règle de réapprovisionnement et 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.

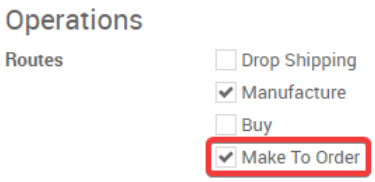
3.5.3 A1 - MTO-MTS_V01 : Test du module OCA mrp_mto_with_stock

3.5.3.1 Objectifs du test

— Vérifier que le système achète et fabrique au plus tard en fonction des conditions suivantes :
  — Aucun stock
  — Article en gestion MTO-MTS
  — Régles 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.

3.5.3.2 Conditions de test

— Système identiques au test A1 - MTS, sauf
  — Module OCA mrp_mto_with_stock [16] installé
  — Tous les articles en MTO_MTS



— Produits avec routes MTO

3.5.3.3 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)

3.5.3.4 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

[16] https://github.com/OCA/manufacture/tree/11.0/mrp_mto_with_stock

  - Date de livraison requise : 10.11.2018 (+102 jours)
- INFO : Une livraison est générée à la date le livraision 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 !

3.5.3.5 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.
- [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 à regroupé 3 besoins sous un RFQ au même fournisseur, alors que leurs dates de besoin sont différentes. Quelles sont les règles de regroupement ?
- [OK] Les quantités en besoin sont correctes.

3.5.3.6 Faiblesses identifiées

- [A1_MTO_F01] [17] Dans le cas où la seule fiche fournisseur indique une quantité minimale d'achat supérieure au besoin MTO, le système déter-

[17] https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36

mine 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] [18] 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] [19] 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 régénèra que si l'article dispose d'une règle de réapprovisionnement et 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 ?

3.5.3.7 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.

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

3.5.4.1 Objectifs du test

— Vérifier que le système achète et fabrique au plus tard en fonction des conditions suivantes :

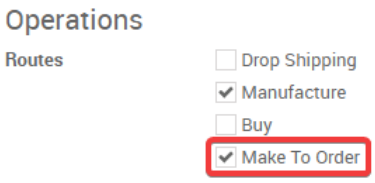
https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36
https://nextcloud.open-net.ch/index.php/s/HaqniNCeQogjq36

- Aucun stock, hormi le produit « A » avec 60 pces on Hand
- Articles en gestion MTO et MTS
- Régles de réapprovisionnement à zéro (Min : 0 / Max : 0 / Multiple : 1 / 0 days to purchase)
- Prévisions de vente enregistrées sur les produits finaux et intermédiaires

— Vérifier globalement la mécanique du module MPS.

3.5.4.2 Conditions de test

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



— `Produits avec routes MTO`

3.5.4.3 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)

3.5.4.4 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 livraision 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 !

3.5.4.5 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 à regroupé 3 besoins sous un RFQ au même fournisseur, alors que leurs dates de besoin sont différentes. Quelles sont les règles de regroupement ?

3.5.4.6 Faiblesses identifiées

— [A1_MTO_F01] [20] 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] [21] 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] [22] 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 régénèra que si l'article dispose d'une règle de réapprovisionnement et 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 ?

3.5.4.7 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.

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

CHAPITRE 4

Solutions à étudier

4.1 OCA - Odoo Community Association[23]

- manufacture [24]
- manufacture-reporting [25]
- purchase-workflow [26]
- stock-logistics-warehouse [27]
- stock-logistics-transport [28]

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

4.2 Odoomrp.com[29]

— odoomrp-utils [30]
— odoomrp-wip [31]

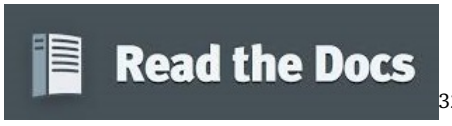
[29] http://www.odoomrp.com
[30] https://github.com/odoomrp/odoomrp-utils
[31] https://github.com/odoomrp/odoomrp-wip

CHAPITRE 5

Annexes

5.1 Maintenance de la documentation

5.1.1 Fonctionnement de la chaîne d'édition



[32]

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

[32] https://readthedocs.org

— 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 language *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 régénéraion du site en ligne.

5.1.2 Prérequis

— Disposer d'une compte chez *Github.com* (ou similaire)
— Disposer d'un compte chez *« Read the Docs »* connecté aux sources du compte *Github.com*.

5.1.2.1 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 »*.

5.1.3 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 fichier .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 »*).

5.1.4 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 proposée 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 raffraîchir la page du navigateur (CTRL+F5) pour␣
↪oubtenir la nouvelle version
```

(suite sur la page suivante)

(suite de la page précédente)

```
# 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 git [33].
  — Connecter le git local au git distant [34]

```
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 your 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.

5.2 Glossaire

Anaconda Anaconda [35] 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 [36]

https://www.youtube.com/watch?v=6n1G45kpU2o
https://stackoverflow.com/a/43364619
https://www.anaconda.com
https://repo.anaconda.com/archive/Anaconda2-5.2.0-Windows-x86_64.exe

Git Git [37] 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 [38] 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 [39] 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 [40]

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 [41]

— Documentation succinte [42]

Sphinx Sphinx [43] 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 [44]

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

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.refractions.net/files/docs/latest/user/docguide/sphinxSyntax.html
https://readthedocs.org
http://www.sphinx-doc.org/en/stable/contents.html

MIT (plus précisément la licence Expat) sur le site du projet sur Github. En revanche, l'exécutable [45] est proposé sur le site officiel de Microsoft sous une licence privatrice.

Webhook Définition Wikipédia [46]

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

https://code.visualstudio.com/
https://en.wikipedia.org/wiki/Webhook

CHAPITRE 6

Licence

Index