POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Automate codeur d'amélioration continue : application à l'implantation d'un réseau d'entraide avec un ERP libre

MATHIEU BENOIT
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie industriel

mars 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Automate codeur d'amélioration continue : application à l'implantation d'un réseau d'entraide avec un ERP libre

présenté par **Mathieu BENOIT**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

PTODO NTODO, présidente
Samuel BASSETTO, membre et directeur de recherche
Martin TRÉPANIER, membre
PTODO NTODO, membre externe

DÉDICACE

REMERCIEMENTS

Samuel Bassetto

Directeur de recherche en génie industriel, aide à l'amélioration continue en contexte industriel, aide dans la création de lien avec le projet d'étude de l'Accorderie et aux projets similaires.

Alexandre Benoit

Relecture

Célia Lignon

Pour la maquette du projet Espace Membre Accorderie fait en collaboration avec DOMUS de l'université de Sherbrooke.

Centre d'excellence sur le partenariat avec les patients et le public

Projet d'étude 2

Fondation Trottier

Financement

Hassan Kassi

Relecture

Marie-Michèle Poulin

Relecture

Nadia Mohamed-Azizi

Directrice du Réseau Accorderie.

Réseau Accorderie

Projet d'étude 1

Simon Montigny

Relecture

TechnoLibre

Prêt d'équipement et d'investissement en salaire pour avancer le projet ORE pour le projet d'étude.

RÉSUMÉ

ABSTRACT

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AGPL	GNU Affero General Public License
AST	Abstract Syntax Tree
CEPPP	Centre d'excellence sur le partenariat avec les patients et le public
DB	Database
ERP	Progiciel de gestion intégré
i18n	Internationalisation et la localisation
JSON	JavaScript Object Notation
LCNC	Low-Code-No-Code
LGPL	GNU Lesser General Public License
MVC	Modèle-Vue-Contrôleur
OCA	Odoo Community Association
ORM	Object-Relational Mapping
PEP	Python Enhancement Proposal
PEP8	Python Enhancement Proposal 8
PHP	PHP : Hypertext Preprocessor
PME	Petite et moyenne entreprise
SQL	Structured Query Language
USD	United States Dollar
XML	Extensible Markup Language

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte et problématique

La valeur du marché des solutions ERP s'établissait autour de 40 milliards USD mondialement en 2020 [1, 2]. Le coût moyen par utilisateur, sur 5 ans, s'élevait à 9 000$, pour une PME, en 2022 [3].

Le développement de système ERP est complexe, nécessite une maintenance exigeante et le risque d'introduire des erreurs est important. Comment accélérer le développement de fonctionnalités de la plateforme ERP Odoo [1] 12 communautaire ?

La plateforme ERPLibre [4] a été créée dans l'objectif d'accélérer le développement de la plateforme Odoo 12 communautaire. Ce mémoire va mettre l'accent sur la génération de code par des techniques de rétro-ingénierie et la gestion d'une communauté dans un contexte d'un projet logiciel libre.

1.1.1 Choix de la plateforme ERP

Choisir une plateforme ERP libre peut offrir des avantages significatifs en termes de coût, de flexibilité, de sécurité, de communauté et d'indépendance. Odoo a été choisi puisqu'il répondait à ces critères, cependant quelle est la version qui offre le plus de fonctionnalité ?

En janvier 2023, les versions 9.0 à 12.0 ne sont plus supportées officiellement par la compagnie Odoo, voir tableau 1.1, mais elles le sont encore par OCA. La version 16.0 est la version stable actuelle. La démonstration commence à partir de 9.0, là où débute la divergence entre une version communautaire et entreprise.

Au printemps 2020, Odoo version 12.0 a été choisi par ERPLibre [2]. Une recherche de module par version d'Odoo a été effectué sur 11 Go de code et de données sur le projet ERPLibre version 1.5.0, voir le tableau 1.2. Ainsi, en date du 1 janvier 2023, la version 12.0 est encore le bon choix avec 2 977 modules, puisqu'il est celui qu'il a le plus de module sur les 133 répertoires gérés par ERPLibre. Cette tendance pourrait changer en 2024 selon l'évolution.

Pour obtenir les résultats du tableau 1.2, un script a été développé pour chercher la quantité de modules en cherchant dans les 133 répertoires Git [3], puis pour toutes les versions d'Odoo, pour tous les modules qui contiennent le fichier «manifest» et que ceux-ci inclus le paramètre

1. Anciennement OpenERP, ERP libre web, lien du projet https://github.com/OCA/OCB
2. Première version de ERPLibre : https://github.com/ERPLibre/ERPLibre/releases/tag/v0.1.0.
3. Logiciel de gestion de versions décentralisé

Tableau 1.1 Tableau des dates de lancement du logiciel Odoo

Légende	Version actuelle	Anciennes versions avec maintenance étendu	Anciennes versions ou fin de maintenance
Odoo version	Date de lancement	Commentaire	
6.0/6.1	octobre 2009	Première publication sous `AGPL`, premier client web	
7.0	décembre 2012		
8.0	septembre 2014	Changement de nom pour Odoo, anciennement OpenERP	
9.0	novembre 2015	Première publication des éditions «Community» sous licence `LGPLV3` et «Enterprise» sous licence propriétaire.	
10.0	octobre 2016		
11.0	octobre 2017		
12.0	octobre 2018	Version utilisé dans ERPLibre 1.5.0	
13.0	octobre 2019		
14.0	octobre 2020		
15.0	octobre 2021		
16.0	octobre 2022		

qu'il est installable, à date précédente du 1 janvier de chaque année.

Des fois, la quantité de module diminue d'une année à l'autre. Il y a création d'une nouvelle branche lors d'une nouvelle version qui est la suite de la version précédente. Par exemple, dans le tableau 1.2, la version 10.0 entre 2017 et 2018, il y a une réduction de 171 modules dans les répertoires d'entreprise, mais il y a eu seulement 4 mois pour faire le nettoyage, les méthodes de mises à jours ont évolués depuis.

De plus, les chiffres du tableau 1.2 semblent démontrer que les versions paires d'Odoo sont plus populaire que les versions impaires. Cependant la communauté d'Odoo est bien plus grosse que 133 répertoires.

Dans la section total du tableau 1.2, la section unique signifie que la somme va ignorer les doublons. En date du 1 janvier 2023, il y a au total 17 309 modules, mais 6 063 modules uniques. Ça signifie qu'il y a 11 246 modules en doublon. Hors, le code diffère d'une version à l'autre même si c'est un doublon, ils peuvent avoir des bogues ou des fonctionnalités différentes entre eux.

Tableau 1.2 Nombre de modules par version Odoo à partir du 1 janvier minuit par année sur la plateforme ERPLibre 1.5.0.

Légende	Total OCA Entreprise	Version actuelle	Anciennes versions avec maintenance étendu	Anciennes versions ou fin de maintenance

17309/10728/6581 modules à supporter le 1 janvier 2023
17465/10952/6513 modules le 15 février 2023
156/132/24 nouveaux modules en 31 jours durant janvier 2023

Odoo version	2016	2017	2018	2019	2020	2021	2022	2023
6.1	295 269 26	299 270 29	299 270 29	299 270 36	299 270 36	299 270 36	299 270 36	299 270 36
7.0	637 597 40	633 592 41	634 593 41	635 594 41	669 619 50	669 619 50	669 619 50	671 619 52
8.0	741 597 144	1092 907 185	1215 996 219	1265 1027 238	1290 1036 254	1297 1043 254	1340 1049 291	1341 1050 291
9.0	135 46 89	456 346 110	725 602 123	776 643 133	796 659 137	803 666 137	844 666 178	850 669 181
10.0		523 111 412	954 713 241	1 537 953 584	1 647 1 047 600	1 685 1 085 600	1 754 1 109 645	1 765 1 120 645
11.0			288 77 211	1 398 658 740	1 710 929 781	1 797 1 000 797	1 860 1 023 837	1 869 1 032 864
12.0				**784** **137** **647**	**1 837** **993** **844**	**2 503** **1 464** **1 039**	**2 851** **1 633** **1 218**	**2 977** **1 693** **1 284**
13.0					617 115 502	1 445 844 601	2 024 1 310 714	2 241 1 506 735
14.0						906 129 777	2 150 1 143 1 007	2 648 1 698 950
15.0							786 96 690	1 669 865 804
16.0								972 206 766
Total								
Somme	1 808 1 509 299	3 003 2 226 777	4 115 3 251 864	6 701 4 282 2 419	8 872 5 668 3 204	11 411 7 120 4 291	14 584 8 918 5 666	17 309 10 728 6 581
Support Odoo	1 808 1 509 299	2 704 1 956 748	3 182 2 388 794	4 495 2 391 2 104	5 811 3 084 2 727	6 651 3 437 3 214	7 811 4 182 3 629	7 530 4 275 3 255
Unique	1 244 1 047 197	1 995 1 435 560	2 260 1 845 415	3 214 2 172 1 042	3 927 2 610 1 317	4 676 3 080 1 596	5 452 3 572 1 880	6 063 3 980 2 083

133/72/61 répertoires de modules dans ERPLibre 1.5.0

1.1.2 Introduction Accorderie

L'Accorderie, un réseau d'entraide Québécois, était à la recherche d'une plateforme améliorée, avec des technologies plus récentes pour contrer l'effet des réseaux sociaux qui est devenu un intermédiaire intéressant pour échanger entre les membres, ainsi que d'automatiser les processus d'échange de temps, qui demande actuellement une gestion manuelle.

«Le 3 juin 2002, l'Accorderie de Québec est officiellement constituée en tant qu'organisme à but non lucratif. Sa mission : lutter contre la pauvreté et l'exclusion sociale ainsi que favoriser la mixité sociale» [5].

Nous avons décelé que le présent projet pourrait répondre à leur besoin avec l'automate programmeur qui permet de facilité, entre autres la maintenance dans le temps. De plus, la plateforme a le potentiel de leur éviter des coûts, du développement redondant et leur donner des fonctionnalités personnalisées. Ainsi, nous avons obtenu accès au code source `PHP` de la plateforme Espace Membre dont le copyright mentionne l'année 2007 par la compagnie GRF Ressource Informatique. De plus, nous avons aussi eu accès à la base de données et selon les archives, le premier échange tracé est le 1 janvier 2003. La plateforme aurait eu plusieurs mises à jour au fil du temps. Cela nous donnait un cas réel empirique, avec des données et des utilisateurs réels, pour rendre concret une plateforme d'échange de temps dont le présent mémoire fait objet.

1.1.3 Introduction CEPPP

Le CEPPP, Centre d'excellence sur le partenariat avec les patients et le public, était à la recherche d'une plateforme pour la gestion du partenariat avec les patients et le public. Le Portail des partenaires ("Portail") du Centre d'excellence sur le partenariat avec les patients et le public (CEPPP) est issu de la fusion de communautés de patients partenaires, entre autres de la Direction collaboration et partenariat patient (DCPP) de la Faculté de médecine de l'Université de Montréal, et celle du CEPPP du Centre de recherche du Centre Hospitalier de l'Université de Montréal (CR-CHUM). Le Portail est un outil qui vient soutenir les activités de recrutement et de recherche sur les pratiques de partenariat. Ils avaient donc déjà une solution [6], mais elle était incomplète, il manquait la gestion de l'anonymisation et l'interface ne répondait pas à des critères esthétiques.

Le présent projet venait pallier aux problèmes rencontrés en facilitant le développement, en accélérant ce dernier, en permettant une personnalisation et en développant l'anonymisation des données. À l'instar de l'Accorderie, cela permettait aussi un cas réel d'utilisation empirique du projet afin d'améliorer et de comprendre ce dont le générateur de code a besoin.

1.2 Définitions et concepts

1.2.1 Caractéristique de la plateforme Odoo

Internationalization et localisation

Dans un réseau d'entraide, il est nécessaire de supporter plusieurs langues pour faciliter la compréhension de l'outil informatique à l'utilisation. Pour y arriver, il existe un standard `i18n`, qui a été référencé [7], qui permet d'adapter les logiciels informatiques à plusieurs langues sur plusieurs localisations [8]. Odoo rend accessible plusieurs bibliothèques pour permettre l'extraction de chaînes de caractères au moment de l'exécution, que ce soit dans du Python, Javascript ou `XML`. Le système permet de générer un fichier «pot» qui contient ces chaînes de caractères. Pour supporter une nouvelle langue et une localisation, on copie le fichier «pot» pour faire un fichier «po» sous la nomenclature `i18n` et on peut faire la traduction ou l'adaptation linguistique. Le système peut aussi générer la langue existante pour faire un fichier «po» avec les traductions déjà connu par le système, il suffit de le mettre à jour.

Architecture `MVC`

Pour générer des modules Odoo, le générateur de code a besoin de se reposer sur l'architecture `MVC` pour permettre une séparation claire des responsabilités et une structure de code facile à maintenir. Le générateur doit ainsi générer chacune des sections de cette architecture pour faire un tout qui permet d'échanger les données entre la base de données et l'interface utilisateur.

L'architecture `MVC` permet de séparer les composantes logicielles comme suit :

1. Le modèle : représente les données et les règles de l'application. Le modèle est responsable de la manipulation des données, de leur stockage et de leur récupération.
2. La vue : représente la présentation de la donnée. La vue est responsable de l'affichage des données stockées dans le modèle à l'utilisateur final.
3. Le contrôleur : gère les interactions entre le modèle et la vue. Le contrôleur est responsable de la réception des demandes de l'utilisateur et de leur transmission au modèle, puis de la récupération des données du modèle pour les transmettre à la vue.

Il est possible de modifier une de ces composantes sans affecter les autres composantes, ce qui facilite sa maintenance.

Website builder

Le «Website builder» est un outil nécessaire dans un réseau d'entraide pour permettre une autonomie aux utilisateurs lambda [4] (augmenter l'accessibilité) dans la création et mise à jour de contenus des pages web sur le site vitrine de l'organisation. C'est un mécanisme `LCNC` dans Odoo 12 pour créer et modifier un site web multi-page par un mécanisme de «drag and drop» avec des «snippets» paramétrables. Il permet de modifier une page web en réduisant l'intervention d'un expert technique réduisant ainsi les coûts de développement.

Architecture ORM

L'utilisation de requête SQL pour communiquer avec des bases de données demandent un temps considérable au développeur à implémenter et il nécessite la programmation d'interface avec la base de données. L'utilisation d'un `ORM` permet d'augmenter la productivité, de faciliter la maintenance et augmente la sécurité d'un logiciel. L'objectif du `ORM` est de faciliter la manipulation des données stockées dans une base de données relationnelle en les représentant sous forme d'objets dans un langage de programmation orienté objet [5]. Cela permet la simplification de l'architecture en l'écrivant en langage haut niveau au lieu de directement en `SQL`, réduisant le nombre d'erreurs et facilite la maintenance.

Architecture modulaire par héritage

Le développement de système `ERP` est complexe par l'ensemble des fonctionnalités nécessaire à la gestion des procédures et ressources des organisations. Pour réduire la complexité du développement logiciel, utiliser une architecture modulaire permet la réutilisation de code et créer des relations fonctionnelles personnalisables aux contextes des organisations.

L'héritage dans Odoo 12 peut se faire de deux manières :

1. L'héritage par extension : Cela permet d'ajouter des fonctionnalités ou de modifier le comportement d'un module existant sans toucher au code source d'origine. Les nouvelles fonctionnalités peuvent être ajoutées en créant un nouveau module qui hérite du module d'origine et en y ajoutant des vues, des modèles ou des contrôleurs supplémentaires.
2. L'héritage par substitution : Cela permet de remplacer complètement le comportement d'un module existant en créant un nouveau module qui hérite du module d'origine et

4. Utilisateur semblable à la majorité dans son comportement. `https://fr.wiktionary.org/wiki/utilisateur_lambda`

5. `https://fr.wikipedia.org/wiki/Mapping_objet-relationnel`

en y modifiant les vues, les modèles ou les contrôleurs existants.

En utilisant l'architecture modulaire avec héritage dans Odoo 12, les développeurs peuvent facilement personnaliser l'application pour répondre aux besoins spécifiques de leur organisation, sans toucher au code source d'origine et sans compromettre la compatibilité avec les mises à jour futures.

Fonctionnalité du hook lors de l'installation d'un module

Au moment de l'installation d'un module Odoo, il y a des opérations qui peuvent être effectué, comme par exemple migrer des données pour les adapter à un nouveau modèle de données. Ainsi, il y a une fonctionnalité qui se nomme le hook pour «pre-install», «post-install» et «uninstall». Ce sont des méthodes qui sont exécutés soit au moment de l'installation, avant ou après l'initialisation de la plateforme, puis à la désinstallation. En «post-install», il devient possible d'exécuter du code et accéder à la plupart des fonctionnalités du `ORM` au moment d'installer un module. C'est une manière pour exécuter des scripts dans la plateforme au moment de l'installation d'un module, que ce soit en ligne de commande ou via l'application «Application» dans Odoo.

ERPLibre

Depuis la version 9.0 d'Odoo, une version communautaire et entreprise ont été créés causant une divergence sur les fonctionnalités créant le modèle d'affaires «Open Core» [6]. L'adaptation d'un `ERP` est complexe et nécessite l'intervention d'un ou des experts pour bien répondre aux besoins de l'utilisateur pour la personnalisation de la plateforme au réalité de l'organisation. Bien que l'`OCA` travaille pour rendre accessible librement ces fonctionnalités, cela vient limiter les réseau d'entraides à pouvoir se débrouiller de manière souveraine.

La plateforme ERPLibre a ainsi été créé en début 2020 encapsulant la version Odoo 12.0 sous licence AGPLv3 en offrant une alternative 100% libre, et dans le but de faciliter le déploiement et le développement pour une organisation en permettant la gestion des dépendances avec Poetry, automatisation du déploiement avec les Docker, la gestion de tous les répertoires de module 1.2 avec Git Repo [7], et plusieurs scripts pour le développement et une documentation propre pour son utilisation. Cela permet de rendre accessible à la même endroit toutes l'information nécessaire à la gestion de son ERP pour une organisation.

6. `https://fr.wikipedia.org/wiki/Open_core`
7. `https://gerrit.googlesource.com/git-repo`

1.2.2 Cadre conceptuel

Nous proposons un modèle formel de la génération de code qui guide le travail proposé dans ce mémoire. Soit C, un code informatique exécutable (qui pourrait aussi être un script interprétable), soit μ_C les méta-données du code, et soit M, une machine disposant de deux modes d'opération : M^d le mode direct, et M^i le mode inverse, voir figure 1.1. Lorsque M opère en mode direct sur μ_C, on doit obtenir C ; en opérant en mode inverse sur C, on doit obtenir μ_C.

On peut représenter symbolique ces deux processus par les équations $M^d(\mu C)$=C et M^i(C)=μC. La machine peut alors être représentée par M = $\{M^d, M^i\}$.

L'ingénierie de génération est le mode direct. La rétro-ingénierie, le mode inverse, est le processus qui consiste à examiner et à analyser un système existant pour en comprendre le fonctionnement et les spécifications. Cette boucle va permettre d'intégrer des concepts d'amélioration continue sur l'évolution du code.

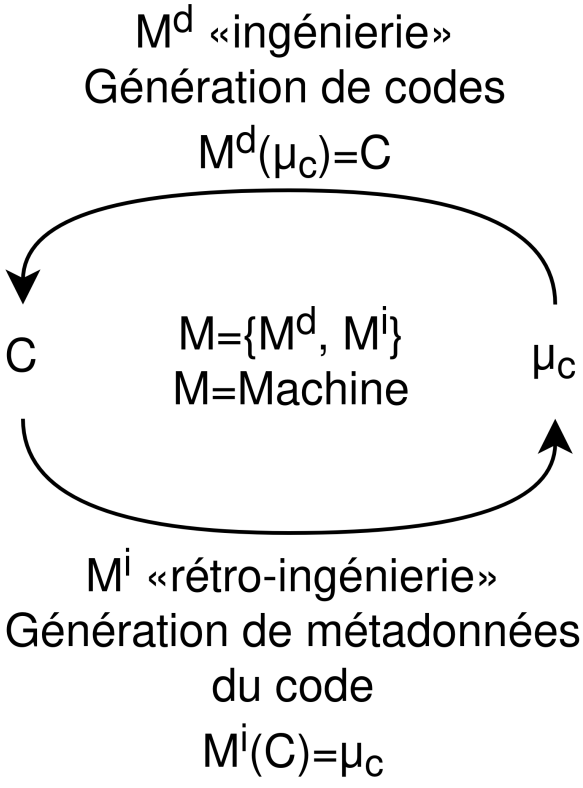

Figure 1.1 Mode direct et inverse

Pour concrétiser le sens de ce modèle formel, nous allons proposer quelques exemples simples. Ayant pour but de faciliter la compréhension, ces exemples sont triviaux et ne présentent pas

le plein potentiel de notre approche. L'interpréteur Python 3.6 et + est utilisé pour les exemples de codes.

Pour la plupart des exemples, le C, représenté par «C.py», est le code «Hello, World !», voir Code 1.1.

```
1 print("Hello, World!")
```

Code 1.1 Exemple de code Hello, World!

1.2.3 Exemples illustratifs d'auto-reproducteur

Le Quine

«Un quine [9] (ou programme auto-reproducteur, self-reproducing en anglais) est un programme informatique qui imprime son propre code source» [8] sans se lire lui-même. Il doit contenir une logique d'écriture de code et contenir ses méta-données de génération. Il est ainsi un générateur de premier niveau.

Voir l'exemple de Code 1.2, la sortie textuelle dans le console lors de l'exécution est le même que son code.

```
1 a='a=%r;print(a%%a)';print(a%a)
```

Code 1.2 Exemple de code Quine

Cependant, le Quine ne sait rien faire d'autre que de s'auto-générer. Ce qui pourrait apporter une contribution serait de faire un auto-reproducteur qui permet de dériver vers d'autres fonctionnalités et ainsi intégrer l'amélioration continue sur son propre développement.

1.2.4 Exemples illustratifs de générateur de code

La génération de code est un des moyens pour supporter le développeur dans son développement d'un logiciel. C'est la partie «création de code» de la méthodologie DevOps.

Technique de génération de code basique

Dans le code 1.3, 1.4, 1.5, la fonction «eval» en Python est dynamique, c'est-à-dire qu'il permet l'exécution à partir d'une chaîne de caractère, ce type de génération de code ne permet pas une évolution efficace ou une simplification du développement. Ça revient à lire un fichier et à l'imprimer, il n'a pas de capacité dynamique d'adaptation.

8. https://fr.wikipedia.org/wiki/Quine_(informatique)

```
print("Hello, World!")
```

Code 1.3 C

```
print('print("Hello, World!")')
```

Code 1.4 μ_C

```
eval("""print('print("Hello, World!")')""")
```

Code 1.5 M(μ_C)

Dans le code 1.6, 1.7, 1.8, cette technique basique est paramétrable, contrairement à la première exemple 1.5. Le f-strings fait office de template et il y a la capacité dynamique d'adaptation en ajoutant des itérations et des conditions sans utiliser de bibliothèque externe.

```
print("Hello, World!")
```

Code 1.6 C - fichier C.py

```
{
 "fonction": "print",
 "argument": "\"Hello, World!\""
}
```

Code 1.7 μ_C - fichier uC.json

```
import json

with open("uC.json") as f:
    metadata = json.load(f)

result = f"{metadata.get('fonction')}({metadata.get('argument')})\n"

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.8 M(μ_C)

Technique de génération de code par template

Un moteur de template est un outil de modèle structurel qui simplifie la syntaxe pour assurer une bonne maintenabilité et qui est généralement utilisé pour le développement de projet Web.

La génération de code par template est une technique de développement de logiciels qui permet de produire du code source à partir de modèles prédéfinis appelés templates.

Le code 1.9, 1.10, 1.11, 1.12 utilise la bibliothèque «Jinjer2». C'est un mécanisme similaire à code 1.8, cependant la logique est intégré directement dans le fichier template.

```
print("Hello, World!")
```

Code 1.9 C - fichier C.py

```
{
 "fonction": "print",
 "argument": "\"Hello, World!\""
}
```

Code 1.10 μ_C - fichier uC.json

```
{{ fonction }}({{ argument }})
```

Code 1.11 template - fichier template

```
import json

from jinja2 import Template

with open("template") as f:
    template = Template(f.read())

with open("uC.json") as f:
    metadata = json.load(f)

result = template.render(
    fonction=metadata.get("fonction"), argument=metadata.get("argument")
)

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.12 M(μ_C)

Les avantages du template 1.13 pour cette approche sont : une productivité accrue, une réduction des erreurs de codage, une meilleure cohérence du code et une réduction du temps de développement. Cette technique peut également faciliter la maintenance du code, puisque les modifications apportées aux templates sont automatiquement propagées à tout le code généré.

```
{% for student in students %}
  <li>
  {% if student.score > 80 %}:){% else %}:({% endif %}
```

```
  <em>{{ student.name }}:</em> {{ student.score }}/{{ max_score }}
  </li>
{% endfor %}
```

Code 1.13 Exemple de template avec logique

Générateur de code par template avec Odoo 12

La technique utilisée par Odoo 12 est le «scaffold», il gère 2 types de template : un module de base «default» et un module thème «theme».

Dans le code 1.14, tout est presque commenté, rien n'est utilisable, mais nous avons la structure `MVC`. Le gain d'accélération au développement est minime.

```
> ./odoo/odoo-bin scaffold module_default ./
> tree module_default/
controllers
    controllers.py
    __init__.py
demo
    demo.xml
__init__.py
__manifest__.py
models
    __init__.py
    models.py
security
    ir.model.access.csv
views
    templates.xml
  views.xml
```

Code 1.14 Commande Odoo pour générer un module avec le Scaffold

Cette fois-ci, dans le code 1.15, tout est vide, le module «theme_module» fait une erreur à l'installation. Il est plus efficace copier et modifier un thème existant que d'utiliser le «scaffold».

```
> /odoo/odoo-bin scaffold -t theme theme_module ./
> tree theme_module/
demo
    pages.xml
__init__.py
__manifest__.py
static
```

```
    src
        scss
            custom.scss
views
    options.xml
    snippets.xml
```

Code 1.15 Commande Odoo pour générer un thème avec le Scaffold

Le gain d'accélération au développement est minime, tellement que c'est négligeable. Copier un module fonctionnel et le modifier en enlevant ce qu'on ne veut pas est plus efficace que d'utiliser cette technique. Disons que cette technique est utile pour un débutant pour comprendre à quoi ressemble une architecture vide, mais il va apprendre plus vite en regardant le fonctionnement de module fonctionnel.

Technique de génération de code par rétro-ingénierie

Lorsqu'on veut faire de la rétro-ingénierie [10] avec un générateur le code, l'objectif est de faire de la ré-ingénierie sur la partie génération, ainsi on altère le code, voir figure 1.2.

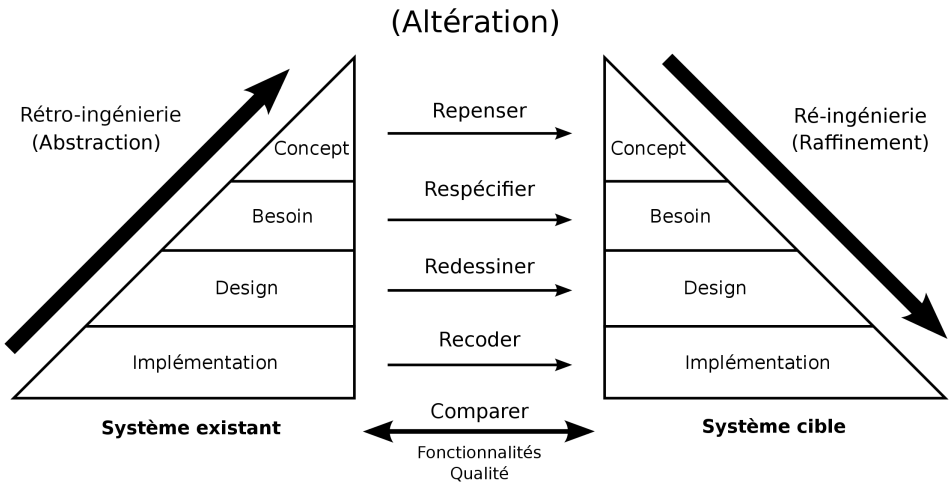

Figure 1.2 Altération du code avec la rétro-ingénierie et la ré-ingénierie. Image modifiée [11]

Dans le code 1.17, la bibliothèque AST est utilisée pour analyser l'arbre de syntaxe abstraite du code source. La difficulté avec la rétro-ingénierie est de savoir précisément ce qu'on cherche à extraire ; il faut avoir un cas d'utilisation spécifique. Ici, le cas d'utilisation est la recherche d'un nœud de type expression qui contient des paramètres. L'avantage est de permettre de corriger des problèmes de qualité logiciel entre l'extraction d'information et la génération

du code. Dans ce contexte, le résultat 1.18 du code source original 1.16 devient formaté en `PEP8 [12]`.

```
print(                        "Hello, World!"                              )
```

Code 1.16 C mal formaté - fichier C.py

```
import ast

with open("C.py", "r") as f:
    code = f.read()

# Extraction du AST
tree = ast.parse(code)
node = tree.body[0]

if (
    isinstance(node, ast.Expr)
    and isinstance(node.value, ast.Call)
    and isinstance(node.value.func, ast.Name)
):
    # Si une expression executable de type function est trouve
    fct_arg = ""
    for arg in node.value.args:
        if isinstance(arg, ast.Str):
            # Cherche un parametre
            fct_arg = arg.s
            break
    # Template
    result = f"""{node.value.func.id}("{fct_arg}")\n"""

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.17 M qui extrait μ_C pour générer C.py

```
print("Hello, World!")
```

Code 1.18 C corrigé - fichier C.py

1.2.5 Tester un générateur de code

Il existe le test «Output comparison testing» [13] qui est le principe que le générateur de code crée du code (une sortie en texte lisible par l'humain) et que l'humain valide cette sortie textuel.

Pour valider si le générateur utilise sa pleine capacité, il suffit de faire des tests de toutes les combinaisons de ses techniques de génération et d'utiliser un outil de couverture de code pour déterminer les lignes qui sont opérés.

CHAPITRE 2 REVUE DE LITTÉRATURE

CHAPITRE 3 MÉTHODE

L'objectif général de ce projet a été de créer et de valider le fonctionnement d'un automate générateur de code qui servir à l'implantation d'un ERP libre. Plus spécifiquement, les travaux ont été concentré sur les cinq sous-objectifs suivants :

SO-1 développer un générateur de code de module sur Odoo,

SO-2 développer une logique d'amélioration continue sur l'écriture du code,

SO-3 développer une interface permettant de paramétrer la génération de code,

SO-4 développer un système de gestion de communauté.

Nous adoptons une méthodologie Agile pour le développement de ces composantes d'un automate générateur de code. Les fonctionnalités de génération de code seront validées par des tests de comparaison entre les codes générés et les codes révisés par l'humain.

3.1 SO-1 - Générateur

Développer une logique d'écriture de module sur une architecture de `MVC` avec support de plateforme web. Mise en place d'un concept de gabarit de code qui génère du code. Mise en place de la génération de code à partir de données. Générer un module à partir d'une base de données externes.

3.2 SO-2 - Rétro-ingénierie

Développer la capacité de comprendre une structure de code et de la reproduire. Définir ce que c'est de l'amélioration continue et son application dans un contexte d'automatisation. Mise en place de test de validation de code sur des critères de qualité mesurables. Intégration de règles de codage standardisées pour favoriser le réseau d'entraide.

3.3 SO-3 - Interface

Proposer une classification des techniques que l'automate peut réaliser en programmation. Développer un interface permettant le contrôle de l'automate pour l'orienter dans la programmation de fonctionnalités. Rendre disponible une interface `LCNC` pour permettre aux utilisateurs de programmer leurs fonctionnalités.

3.4 SO-4 - Déploiement

Développer un système de distribution de l'automate.

3.5 SO-5 - Réseau d'entraide

Documenter les processus de développement pour amener les utilisateurs à contribuer et les faire participer à la maintenance. Mettre des guides pour permettre le sentiment d'appartenance. Mettre en place une politique tolérance zéro avec un système de communication non violente et créer un lieu de discussion public. Élaboration du prototype pour les spécifications du réseau de l'Accorderie du Canada. Élaboration du prototype pour les spécifications de l'organisme CEPPP du Canada.

3.6 Environnement informatique

La plateforme ERPLibre 1.5.0 contenant Odoo 12 communauté, qui utilise les langages Python 3.7, XML, Javascript et SCSS. Nos tests et développement ont été effectués sur le système d'exploitation Ubuntu 20.04. Pour les temps d'exécution des tests, ils ont été effectués sur une machine avec le CPU AMD Ryzen 9 5950X, mémoire ram 2x«32GiB DIMM DDR4 Synchronous Unbuffered (Unregistered) 2667 MHz (0.4 ns)», et disque 1To wd-black-sn850-nvme. Toutes les commandes utilisées sont faites à partir de la racine du projet ERPLibre.

3.7 Méthodologie de test

Les tests ont été codés directement dans ERPLibre dans un script Python avec un ensemble de script pour valider les différences dans le Git après exécution, et cacher des différences.

Le tout a été programmé avec la technique parallélisme avec la bibliothèque Asyncio et des nouveaux processus et non des «thread» [1]. Seul le test de nouveau projet est exécuté après le parallélisme ajouter plus de temps à l'exécution.

Puisque le test d'un générateur de code consiste à valider ce qu'il a généré, ainsi vérifier la couverture de code est un bon indicateur pour déterminer le code utilisé et validé ce qui est déprécié dans le générateur.

1. Processus légé

3.7.1 Couverture du code

La couverture du code a été faite avec la bibliothèque «Coverage» version 7.0.1 en Python et configurée sur le répertoire qui contient le générateur de code pour faire le suivi des lignes exécuter dans la machine. Ça devient une métrique de la performance d'utilisation sur la quantité de fonctionnalité généré.

CHAPITRE 4 RÉSULTAT

Les résultats sont présentés en commençant d'abord par décrire l'implémentation du modèle conceptuel d'automate présenté dans la sous-section 1.2.2. Ensuite, ce sera présenté successivement les résultats propres à chacun des sous-objectifs décrits dans le chapitre 3.

4.1 Implémentation : du modèle conceptuel au modèle opérationnel

L'implémentation de la machine (conceptuelle) est passée par la programmation manuelle d'une première interface et d'un premier noyau, en mettant à jour une version préliminaire de générateur basé sur une GUI [14] en lui intégrant la capacité de générer du code à partir d'un module externe via son «hook» au moment de l'installation. La version à ce jour, ainsi que ses guides d'utilisation et ses scripts associés, sont disponibles sur le site de ERPLibre [1].

L'interface permet l'interactivité avec un utilisateur ou le système-cible. L'interface est découplée en deux ensembles de méta-données : μ_C^A et μ_C^B. μ_C^B est l'ensemble qui paramétrise le passage de méta-données au code pour un module spécifique. μ_C^A est l'ensemble qui paramétrise le passage de code aux méta-données tout en préparant un μ_C^B adapté à ce module spécifique. Ce découplage a permis l'adaptation de l'interface au contexte de l'installation de modules sur une instance Odoo via des «hooks». Par la suite, un ensemble supplémentaire de méta-données, noté μ_C^0, a été dégagé de la programmation manuelle de versions successives de μ_C^A. μ_C^0 sert à initialiser une version de départ de μ_C^A.

Le noyau prend les paramètres issus de l'interface pour créer des méta-données, générer l'ensemble des fichiers des modules désirés (mode direct) et faire de la rétro-ingénierie (mode indirect) sur des modules existants.

4.1.1 Développement et amélioration continue

Dans la figure 4.1, μ_C^0, μ_C^A, μ_C^B, C et M sont tous des modules installables sur Odoo. M^i et M^d sont des sections de code dans le module M. μ_C^0, μ_C^A et μ_C^B dépendent de M. μ_C^A, c'est les macro-méta-données, que μ_C^B, c'est les micro-méta-données.

Au départ d'un nouveau module code, μ_C^0 génère μ_C^A qui génère μ_C^B qui génère C. Il existe un script qui automatise un nouveau code, le développeur peut paramétrer le nom des modules et leurs emplacements. Ensuite, le développement commence en itération agile, les actions de

1. `https://erplibre.ca`

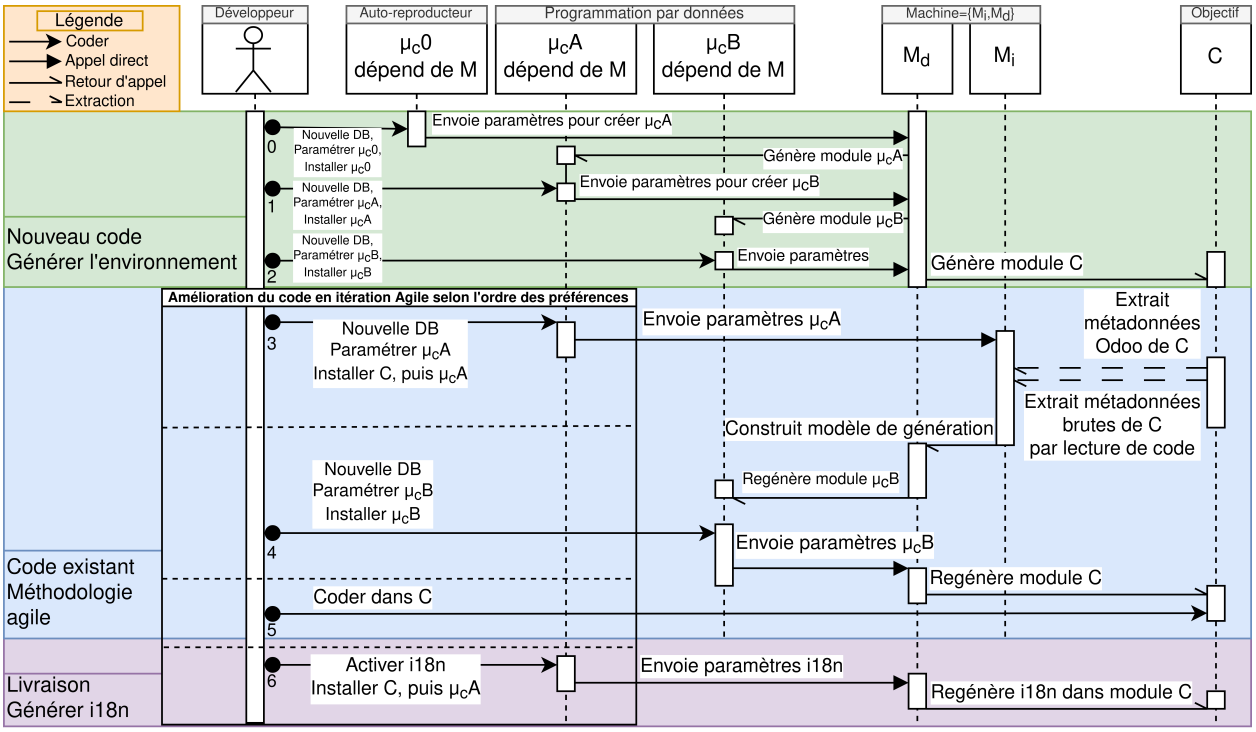

Figure 4.1 Interaction du développeur avec le générateur de code

3 à 6 peuvent être exécutées dans l'ordre du choix du développeur.

Passer par l'étape 3 permet de mettre à jour l'étape 4 selon l'état du code via la rétro-ingénierie. Passer par l'étape 4 permet de mettre à jour le code selon le générateur. Il est possible de générer de nouvelles sections, comme la vue portail. Passer à l'étape 5 permet de personnaliser le code directement. Passer par l'étape 6 permet de mettre à jour le i18n de manière automatique.

La livraison sert à générer le i18n. C'est Odoo qui le génère, mais le générateur envoie les commandes, la liste des langues désirées à supporter et place les fichiers aux bons endroits dans le module. La raison pour laquelle c'est μ_C^A qui doit le générer, c'est parce que le module doit être fini d'être généré et rechargé pour ensuite générer les langues, sinon ils sont corrompus par les traces de μ_C^B.

4.1.2 Architecture

La figure 4.2, elle démontre un développeur qui utilise l'interface de la machine qui opère dans le noyau de la machine.

1. L'interface machine permet à l'utilisateur de créer un modèle de données pour indi-

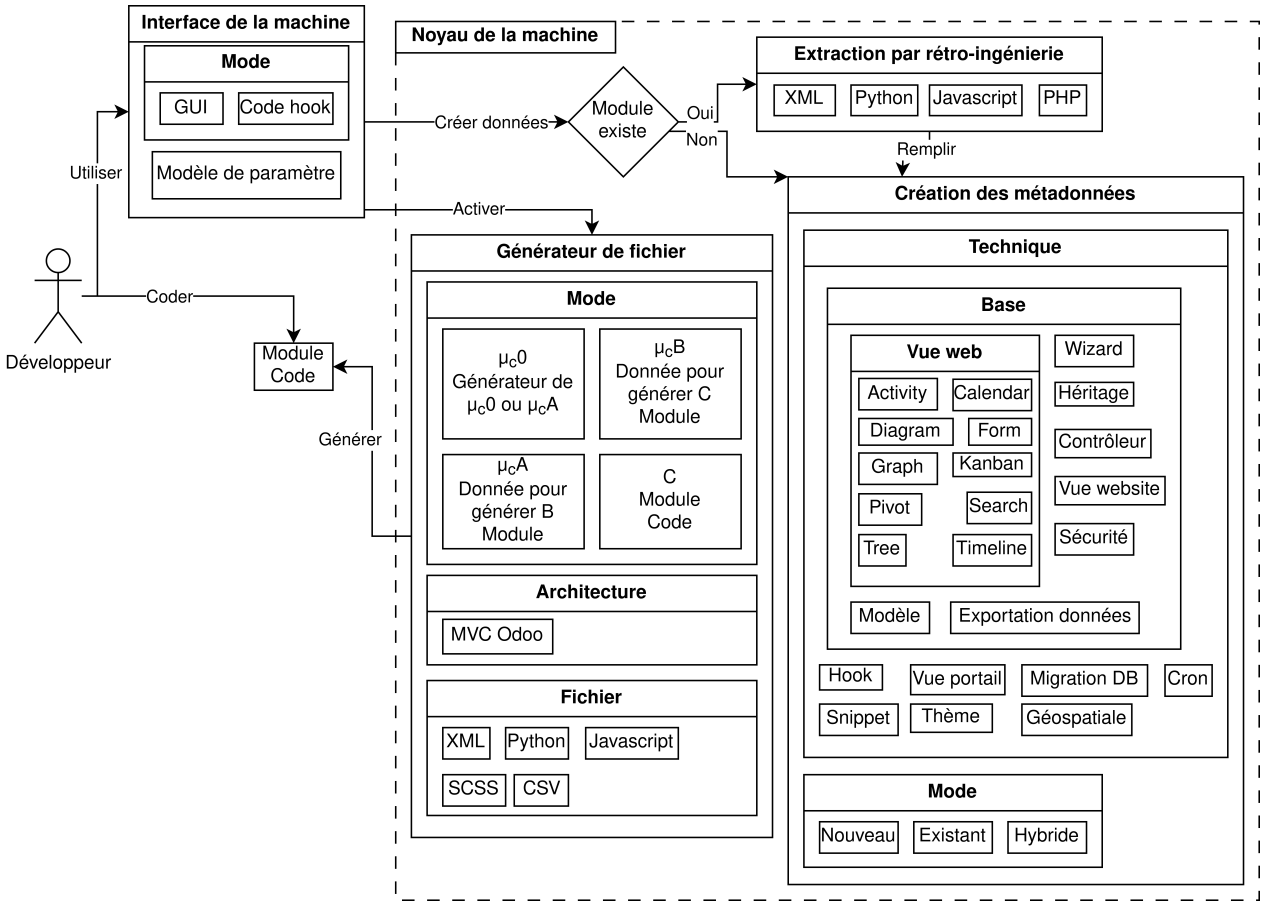

Figure 4.2 Architecture de l'automate

quer à la machine quelle opération effectuée avec leurs données associées. Plusieurs combinaisons, héritables, sont possibles selon les différentes techniques. De plus, c'est ici qu'on vient activer la génération de code.

2. L'extraction par rétro-ingénierie permet de remplir le modèle de méta-données sans passer par la paramétrisation. Il extrait des informations qui ne sont pas accessibles dans le modèle de données d'Odoo sur le module.

3. La création de méta-données se fait soit par l'utilisateur via la GUI ou le «Code Hook», il gère à la fois plusieurs techniques qui sont dans des modules. Le mode «nouveau» permet de créer de nouvelles données. Une fois qu'ils sont créés, c'est le mode «existant» qui est utilisé. Cependant le mode «hybride» permet d'écraser les données existantes en réactivant le mode «nouveau».

4. Le générateur de fichier se fait activer par l'interface, mais il prend les méta-données pour faire sa génération selon le mode qu'il doit générer et l'architecture qu'il connaît. Il fait des liaisons entre les modèles et les vues en référence aux noms des champs de

chaque modèle de données.

Chacun de ces blocs de l'architecture est modulaire, chaque technique est héritable pour modifier le comportement et ajouter des liaisons pour permettre une génération de code au final.

La sécurité dépend du modèle. Le contrôleur dépend du modèle. La vue dépend du contrôleur et du modèle.

4.1.3 Auto-générateur

Représenté par μ_C^0, c'est un auto-générateur ! Au moment de son installation, il génère le même code que lui-même au même endroit dans le système de fichier. Une légère modification va créer une autre entité qui sera une déviation dans l'objectif de démarrer une chaîne.

C'est le module `M` qui contient les méta-données de μ_C^0. Ainsi, exécuter μ_C^0 devient un test de fonctionnalité et c'est un succès lorsqu'il n'y a pas de différence. Cependant sa programmation est actuellement spécifique à sa génération, aucun autre module n'a besoin de cette fonctionnalité unique.

L'auto-générateur est utilisé pour générer des μ_C^A avec une légère modification dans les paramètres. Même s'il a la capacité de générer un μ_C^B, mieux vaut créer la chaîne proposé pour faire de l'amélioration continue.

4.2 Résultats propres à SO-1

4.2.1 Génération par gabarit

La génération par gabarit était déjà supportée dans la version initiale [14], de plus, il y a eu des améliorations :

CHAPITRE 5 DISCUSSION

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

6.2 Limitations de la solution proposée

6.3 Améliorations futures

RÉFÉRENCES

[1] "Les principaux erp du marché," Mordor Intelligence, 17 mars 2023. [En ligne]. Disponible : https://www.mordorintelligence.com/fr/industry-reports/enterprise-resource-planning-market

[2] "Marché de la planification des ressources d'entreprise – croissance, tendances, impact du covid-19 et prévisions (2023-2028)," Big Bang ERP inc., 17 mars 2023. [En ligne]. Disponible : https://bigbang360.com/fr/les-principaux-erp-du-marche/

[3] "What 1,384 erp projects tell us about selecting erp (2022 erp report)," Software Path Ltd, 18 janvier 2022. [En ligne]. Disponible : https://softwarepath.com/guides/erp-report

[4] M. Benoit et M.-M. Poulin. (2023, 19 mars) Erp libre v1.5.0 agplv3 contenant odoo 12.0. [En ligne]. Disponible : https://erplibre.ca

[5] M. Michaud et L. K. Audebrand, "Les paradoxes de la transformation d'une association en coopérative de solidarité : le cas de l'accorderie de québec," *Économie et Solidarités*, vol. 44, nº. 1-2, p. 152–168, 2014. [En ligne]. Disponible : https://id.erudit.org/iderudit/1041610ar

[6] D. Margulius. (2019, 27 décembre) Projet portail ceppp - suitecrm 7.10.9. [En ligne]. Disponible : https://github.com/lerenardprudent/ceppp_crm

[7] A. Lehtola *et al.*, *Current platform support for internationalization.* United States : Wiley, 1997, p. 229–287.

[8] Wikipédia. (2023, 17 février) Internationalisation et localisation - i18n. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Internationalisation_et_localisation

[9] A. Sarkar, "Quines are the fittest programs : Nesting algorithmic probability converges to constructors," 2020. [En ligne]. Disponible : https://arxiv.org/abs/2010.09646

[10] Wikipédia. (2022, 10 septembre) Rétro-ingénierie en informatique. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique

[11] (2021, 18 juin) Image altération du code via la rétro-ingénierie et la ré-ingénierie. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique#/media/Fichier:Retroingenierie_-_Byrne.svg

[12] G. van Rossum, B. Warsaw et N. Coghlan. (2023, 25 février) Guide de style pour le code python. [En ligne]. Disponible : https://peps.python.org/pep-0008/

[13] Wikipédia. (2023, 13 mars) Liste de tests informatiques. [En ligne]. Disponible : https://en.wikipedia.org/wiki/Software_testing

[14] B. Luis. (2019, 27 août) Modules odoo 12 - projet initiale du générateur de code et migrateur de base de données. [En ligne]. Disponible : https://github.com/bluisknot/github_odoo_apps/tree/12.0