

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

DÉDICACE	iii
REMERCIEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE DES MATIÈRES	vii
LISTE DES TABLEAUX	ix
LISTE DES FIGURES	x
LISTE DES SIGLES ET ABRÉVIATIONS	xi
LISTE DES ANNEXES	xii
CHAPITRE 1 INTRODUCTION	1
1.1 Contexte et problématique	1
1.1.1 Choix de la plateforme ERP	1
1.1.2 Introduction Accorderie	4
1.1.3 Introduction CEPPP	4
1.2 Définitions et concepts	5
1.2.1 Caractéristique de la plateforme Odoo	5
1.2.2 Cadre conceptuel	8
1.2.3 Exemples illustratifs d'auto-reproducteur	9
1.2.4 Exemples illustratifs de générateur de code	9
1.2.5 Tester un générateur de code	15
CHAPITRE 2 REVUE DE LITTÉRATURE	16
CHAPITRE 3 MÉTHODE	17
CHAPITRE 4 RÉSULTAT	18
CHAPITRE 5 DISCUSSION	19

CHAPITRE 6 CONCLUSION	20
6.1 Synthèse des travaux	20
6.2 Limitations de la solution proposée	20
6.3 Améliorations futures	20
RÉFÉRENCES	21

LISTE DES TABLEAUX

Tableau 1.1	Tableau des dates de lancement du logiciel Odoo	2
Tableau 1.2	Nombre de modules par version Odoo à partir du 1 janvier minuit par année sur la plateforme ERPLibre 1.5.0.	3

LISTE DES FIGURES

Figure 1.1	Mode direct et inverse	8
Figure 1.2	Altération du code avec la rétro-ingénierie et la ré-ingénierie.	13

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 Odoo¹ 12 communautaire ?

La plateforme ERPLibre² 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 ERPLibre³. 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 Git⁴, puis pour toutes les versions d'Odoo,

1. Anciennement OpenERP, ERP libre web, lien du projet <https://github.com/OCA/OCB>

2. Projet automatisé v1.5.0 contenant Odoo 12.0 <https://erplibre.ca>

3. Première version de ERPLibre : <https://github.com/ERPLibre/ERPLibre/releases/tag/v0.1.0>.

4. 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		

pour tous les modules qui contiennent le fichier «manifest» et que ceux-ci inclus le paramètre 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.

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» [4].

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 solution⁵, 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

5. Lien du projet Git CEPPP CRM https://github.com/lerenardprudent/ceppp_crm/tree/master

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 [5], qui permet d'adapter les logiciels informatiques à plusieurs langues sur plusieurs localisations⁶. 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.

6. Wikipédia https://fr.wikipedia.org/wiki/Internationalisation_et_localisation

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 lambda⁷ (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 objet⁸. 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

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

8. https://fr.wikipedia.org/wiki/Mapping_objet-relationnel

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 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 effectuées, 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 exécutées 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 créées causant une divergence sur les fonctionnalités créant le modèle d'affaires «Open Core»⁹. 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 à la réalité de l'organisation. Bien que l'OCA travaille pour rendre accessible librement ces fonctionnalités, cela vient limiter le réseau d'entraides à pouvoir se débrouiller de manière souveraine.

La plateforme ERPLibre a ainsi été créée 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 Repo¹⁰, et plusieurs scripts pour le développement et

9. https://fr.wikipedia.org/wiki/Open_core

10. <https://gerrit.googlesource.com/git-repo>

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.

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 $M^i(C)=\mu_C$. 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.

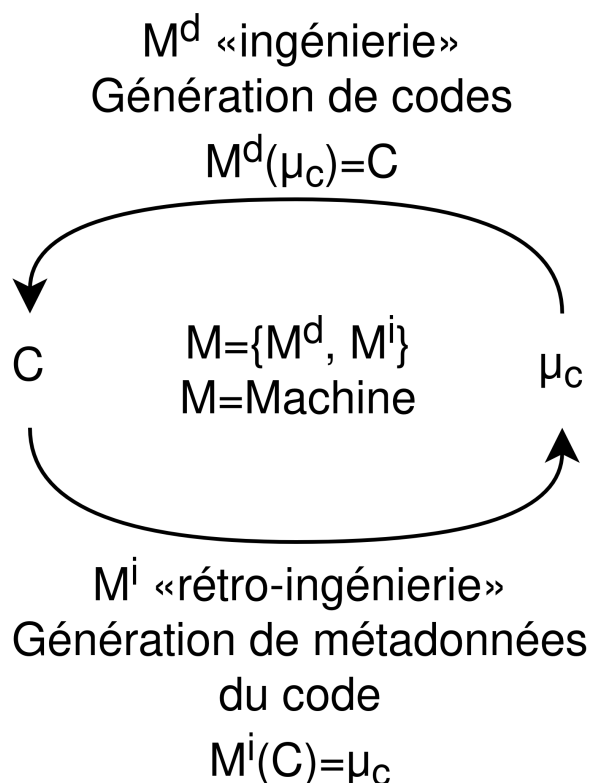


Figure 1.1 Mode direct et inverse

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 [6] (ou programme auto-reproducteur, self-reproducing en anglais) est un programme informatique qui imprime son propre code source»¹¹ 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

11. [https://fr.wikipedia.org/wiki/Quine_\(informatique\)](https://fr.wikipedia.org/wiki/Quine_(informatique))

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.

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

Code 1.3 C

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

Code 1.4 μ_C

```
1 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.

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

Code 1.6 C - fichier C.py

```
1 {
2   "fonction": "print",
3   "argument": "\"Hello, World!\""
4 }
```

Code 1.7 μ_C - fichier uC.json

```
1 import json
2
3 with open("uC.json") as f:
4     metadata = json.load(f)
5
6 result = f"{metadata.get('fonction')}({metadata.get('argument')})\n"
7
8 with open("C.py", "w") as f:
9     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.

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

Code 1.9 C - fichier C.py

```
1 {
2   "fonction": "print",
3   "argument": "\"Hello, World!\""
4 }
```

Code 1.10 μ_C - fichier uC.json

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

Code 1.11 template - fichier template

```
1 import json
2
3 from jinja2 import Template
4
5 with open("template") as f:
6     template = Template(f.read())
7
8 with open("uC.json") as f:
9     metadata = json.load(f)
10
11 result = template.render(
12     fonction=metadata.get("fonction"), argument=metadata.get("argument")
13 )
14
15 with open("C.py", "w") as f:
16     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.

```

1 {% for student in students %}
2   <li>
3     {% if student.score > 80 %}:{% else %}:{% endif %}
4     <em>{{ student.name }}:</em> {{ student.score }}/{{ max_score }}
5   </li>
6 {% 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.

```

1 > ./odoo/odoo-bin scaffold module_default ./
2 > tree module_default/
3 controllers
4     controllers.py
5     __init__.py
6 demo
7     demo.xml
8     __init__.py
9     __manifest__.py
10 models
11     __init__.py
12     models.py
13 security
14     ir.model.access.csv
15 views
16     templates.xml
17     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».

```

1 > /odoo/odoo-bin scaffold -t theme theme_module ./
2 > tree theme_module/
3 demo
4     pages.xml

```



```

5 __init__.py
6 __manifest__.py
7 static
8     src
9         scss
10             custom.scss
11 views
12     options.xml
13     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-ingénierie¹² 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.

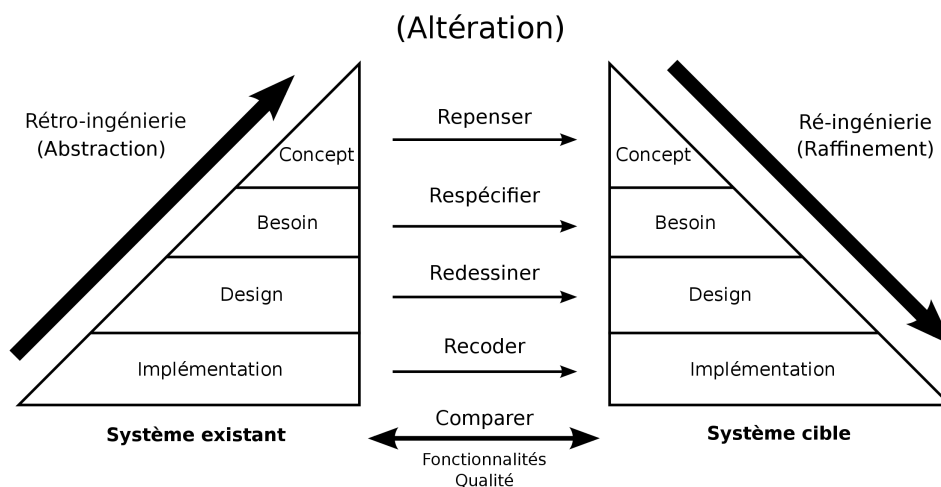


Figure 1.2 Altération du code avec la rétro-ingénierie et la ré-ingénierie.

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

12. https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique

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 PEP8¹³.

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

Code 1.16 C mal formaté - fichier C.py

```
1 import ast
2
3 with open("C.py", "r") as f:
4     code = f.read()
5
6 # Extraction du AST
7 tree = ast.parse(code)
8 node = tree.body[0]
9
10 if (
11     isinstance(node, ast.Expr)
12     and isinstance(node.value, ast.Call)
13     and isinstance(node.value.func, ast.Name)
14 ):
15     # Si une expression executable de type function est trouve
16     fct_arg = ""
17     for arg in node.value.args:
18         if isinstance(arg, ast.Str):
19             # Cherche un parametre
20             fct_arg = arg.s
21             break
22     # Template
23     result = f"""{node.value.func.id}("{fct_arg}")\n""
24
25 with open("C.py", "w") as f:
26     f.write(result)
```

Code 1.17 M qui extrait μ_C pour générer C.py

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

Code 1.18 C corrigé - fichier C.py

13. <https://peps.python.org/pep-0008/>

1.2.5 Tester un générateur de code

Il existe le test «Output comparison testing»¹⁴ 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.

14. https://en.wikipedia.org/wiki/Software_testing

CHAPITRE 2 REVUE DE LITTÉRATURE

CHAPITRE 3 MÉTHODE

CHAPITRE 4 RÉSULTAT

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. 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, n°. 1-2, p. 152–168, 2014. [En ligne]. Disponible : <https://id.erudit.org/iderudit/1041610ar>
- [5] A. Lehtola *et al.*, *Current platform support for internationalization*. United States : Wiley, 1997, p. 229–287.
- [6] A. Sarkar, “Quines are the fittest programs : Nesting algorithmic probability converges to constructors,” 2020. [En ligne]. Disponible : <https://arxiv.org/abs/2010.09646>