

Titre: Génération de code automatisée pour les modules d'un ERP libre, application à la création d'un réseau d'entraide et d'amélioration continue
Title:

Auteur: Mathieu Benoit
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Benoit, M. (2023). Génération de code automatisée pour les modules d'un ERP libre, application à la création d'un réseau d'entraide et d'amélioration continue
Citation: [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
<https://publications.polymtl.ca/53421/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/53421/>
PolyPublie URL:

Directeurs de recherche: Samuel Bassetto
Advisors:

Programme: Maîtrise recherche en génie industriel
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Génération de code automatisée pour les modules d'un ERP libre, application à
la création d'un réseau d'entraide et d'amélioration continue**

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

Mai 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Génération de code automatisée pour les modules d'un ERP libre, application à
la création d'un réseau d'entraide et d'amélioration continue**

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 :

Martin TRÉPANIÉ, président

Samuel Jean BASSETTO, membre et directeur de recherche

Giovanni BELTRAME, membre

DÉDICACE

*Aux instigateurs du logiciel libre, ce robot logiciel codeur est pour vous, vive la AGPLv3+ :
une licence importante pour les applications en ligne qui souhaitent utiliser et contribuer
aux logiciels libres tout en préservant les droits de la communauté et en favorisant la
collaboration et l'innovation !*

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche en génie industriel, Samuel Basseto, qui m’a aiguillé vers le projet en lien avec l’amélioration continue en contexte industriel et qui a fait la liaison avec l’Accorderie, ainsi qu’avec des projets similaires. Je veux exprimer aussi ma gratitude envers les membres de son laboratoire de recherche, LABAC, pour tous les commentaires pertinents, les réunions et les moments plus décontractés et, en particulier, Lucas Jacquet qui a choisi de faire un projet connexe au mien.

Merci à l’Accorderie et sa directrice, Nadia Mohamed-Azizi, d’avoir permis que l’organisation devienne un projet d’étude. Dans le même ordre d’idée, merci à Geneviève David et Camille Bolduc-Boyer du Centre d’excellence sur le partenariat avec les patients et le public du CR-CHUM pour leur confiance et leur participation dans ce qui est devenu le deuxième cas du projet d’étude.

Je voudrais souligner l’apport financier de la Fondation Trottier, ainsi que l’investissement de la compagnie TechnoLibre en salaire pour l’avancement et la réalisation du projet ORE.

Un grand merci à mes relecteurs : Alexandre Benoit pour son travail sur la forme, Hassan Kassi pour son mentorat et ses idées et Simon de Montigny pour son soutien, pour le peaufinage de la présentation des résultats et l’amélioration de la cohérence de ces derniers.

Un grand merci à mes parents, Sylvie Lemieux et Sylvain Benoit, pour avoir cru en moi et m’avoir donné leur soutien financier et moral, non seulement pour ma maîtrise, mais tout au long de mes études.

Un merci spécial à ma conjointe, Marie-Michèle Poulin, pour son appui indéfectible tout au long du projet, non seulement dans les trivialités de la vie courante, mais grâce à ses idées, ses questionnements et ses connaissances, ainsi qu’à son apport au document au point de vue de la révision du langage et la traduction.

Enfin, un merci à mon bébé, Elizabeth Benoit, qui me donne la motivation d’avancer et qui me donne espoir qu’elle veuille créer des communautés et qu’elle ait envie de vivre libre et selon ses propres choix.

RÉSUMÉ

Les programmeurs de logiciels de planification des ressources d'entreprise (ERP) développent les mêmes fonctionnalités d'un système à l'autre avec la même technique d'implémentation d'une fonctionnalité à une autre. Les ERP sont complexes et demandent de longues durées de programmation, les taux d'erreurs sont élevés. L'automatisation d'écriture de code est une solution pour la simplification du travail du programmeur. Un robot logiciel développeur, suivant les bases de l'industrialisation, pourrait être orienté vers les besoins de la communauté et permettrait de développer des fonctionnalités à une vitesse accélérée à l'aide de la rétro-ingénierie. Plus la quantité d'information est disponible, plus le robot sera efficace, tirant tous les avantages du logiciel libre i.e. utiliser, copier, étudier et modifier tout en distribuant sans restriction.

Ce mémoire présente et valide un concept d'un auto-reproducteur logiciel utilisant une technique de rétro-ingénierie en Python. La recherche porte sur le développement d'une technologie auto-développeur bonifiée par de l'auto-amélioration avec une technique d'auto-ingénierie et aussi un auto-générateur qui est paramétrable pour démarrer une chaîne de développement sur des modules Odoo. Pour y arriver, nous avons développé plusieurs modules Odoo incluant la génération de code qui permet de générer des modules Odoo à partir de métadonnées, d'appliquer de la rétro-ingénierie pour faire de l'auto-reproduction sur un module Odoo pour extraire les métadonnées, contenant une interface qui nécessite peu ou pas de code et d'autres pratiques logicielles pour augmenter l'accessibilité.

Des liaisons ont été faites entre la gestion d'une communauté autour d'un projet technologique libre et le démarrage pour un gestionnaire d'un réseau d'entraide, assisté par un générateur de code automatisé pour mettre en place de l'amélioration continue sur le développement et les habitudes des participants.

Le robot logiciel libre codeur est en première phase de développement incluant la génération de code, l'interface avec l'utilisateur et la rétro-ingénierie pour appliquer de l'amélioration continue orientée au support d'un réseau d'entraide. La machine est présentement limitée à la génération d'application web sur Odoo version 12.0 en utilisant ERPLibre 1.5.0.

L'automatisation du développement de logiciel pourra concrètement permettre l'accélération de création de fonctionnalités et la réduction des coûts de développement. De plus, on peut penser que le générateur de code offrira la possibilité aux chercheurs d'être plus efficaces dans leurs travaux en facilitant le développement de leurs propres outils pouvant mieux tracer, s'interfacer et avoir le contrôle de leurs données.

ABSTRACT

Enterprise resource planning (ERP) programmers develop the same functionality from one system to another with the same implementation technique from one feature to another. ERPs are complex and require long programming times, and error rates are high. Code writing automation is a solution for simplifying the programmer's work. A software robot developer, following the basics of industrialization, could be oriented towards the needs of the community and would allow to develop functionalities at an accelerated speed with the help of reverse engineering. The more available information is, the more efficient the robot will be, benefiting from all the advantages of free (as freedom) software i.e. use, copy, study and modify while distributing without restriction.

This dissertation presents and validates a concept of a software auto-reproducer using a reverse engineering technique in Python. The research focuses on the development of a self-developing technology enhanced by self-improvement with a self-engineering technique and also an auto-generator that is configurable to start a development chain on Odoo modules. To make this happen, we have developed several Odoo modules including code generation that allows to generate Odoo modules from metadata, apply reverse engineering to auto-reproduce an Odoo module to extract metadata, containing an interface that requires little or no code and other software practices to increase accessibility.

The free (as freedom) software robot coder is in the first phase of development including code generation, user interface and reverse engineering to apply continuous improvement oriented to support a peer support network. The machine is currently limited to web application generation on Odoo version 12.0 using ERPLibre 1.5.0.

TABLE DES MATIÈRES

DÉDICACE	iii
REMERCIEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE DES MATIÈRES	vii
LISTE DES TABLEAUX	xi
LISTE DES FIGURES	xii
LISTE DES SIGLES ET ABRÉVIATIONS	xiv
LISTE DES ANNEXES	xv
CHAPITRE 1 INTRODUCTION	1
1.1 Contexte et problématique	1
1.2 Objectif et but	1
1.2.1 Choix de la plateforme <i>ERP</i>	2
1.2.2 Introduction Accorderie	5
1.2.3 Introduction CEPPP	5
1.3 Organisation générale du document	6
CHAPITRE 2 REVUE DE LITTÉRATURE	7
2.1 Caractéristiques de la plateforme Odoo	7
2.1.1 Architecture MVC	7
2.1.2 Architecture ORM	7
2.1.3 Architecture modulaire par héritage	8
2.1.4 Fonctionnalité du <i>hook</i> lors de l'installation d'un module	8
2.1.5 <i>Website builder</i>	9
2.1.6 Internationalisation et localisation	9
2.1.7 ERPLibre	9
2.2 Robot logiciel développeur	10

2.3	Génération de code	10
2.4	Rétro-ingénierie	11
2.5	Logiciel Libre et <i>Open Source</i>	12
2.6	Sécurité	12
2.7	Les complexités de développer <i>ERP</i>	13
2.8	<i>DevOps</i>	14
2.9	Logiciel <i>no-code</i> / <i>low-code</i>	15
2.10	Création d'une communauté	16
2.10.1	Guide construire une communauté <i>Open Source</i>	18
2.10.2	Critères éthiques de GNU concernant l'hébergement de logiciel	18
2.11	Poïèse	18
2.11.1	Définition de la poïèse	19
2.11.2	Technopoïèse	19
2.11.3	Allopoïèse	20
2.11.4	Autopoïèse	20
2.11.5	Sympoïèse	21
2.12	Définitions et concepts	22
2.12.1	Cadre conceptuel	22
2.12.2	Exemples illustratifs d'auto-reproducteur	23
2.12.3	Exemples illustratifs de générateur de code	24
2.12.4	Tester un générateur de code	28
CHAPITRE 3	OBJECTIFS ET MÉTHODOLOGIE	30
3.1	SO-1 - Générateur	30
3.2	SO-2 - Rétro-ingénierie	30
3.3	SO-3 - Interface	31
3.4	SO-4 - Déploiement	31
3.5	SO-5 - Réseau d'entraide	31
3.6	Environnement informatique	31
3.7	Méthodologie de test	32
3.7.1	Couverture du code	32
CHAPITRE 4	RÉSULTAT EXPÉRIMENTAUX	33
4.1	Implémentation : du modèle conceptuel au modèle opérationnel	33
4.1.1	Développement et amélioration continue	33
4.1.2	Architecture	34
4.1.3	Auto-générateur	36

4.2	Résultats propres à SO-1	37
4.2.1	Génération par gabarit	37
4.2.2	Génération de l'architecture MVC	37
4.2.3	Générer un module à partir d'une base de données externes	38
4.2.4	Génération de code par des données	39
4.2.5	Interprétation des résultats de SO-1	39
4.3	Résultats propres à SO-2	40
4.3.1	Extraction de code et reproduction	40
4.3.2	Amélioration continue sur la génération	41
4.3.3	Test de validation de génération de codes	41
4.3.4	Règles de codage standardisées	42
4.3.5	Interprétation des résultats de SO-2	42
4.4	Résultats propres à SO-3	43
4.4.1	Classification des techniques développées	43
4.4.2	Interface du générateur de code	43
4.4.3	Interprétation des résultats de SO-3	43
4.5	Résultats propres à SO-4	44
4.5.1	Utilisation d'un conteneur Docker	44
4.6	Résultats propres à SO-5	44
4.6.1	Guide : créer une communauté autour d'une technologie pour un réseau d'entraide libre	45
4.6.2	Projet module «auto_backup»	45
4.6.3	Projet module SRS	45
4.6.4	Projet espace Accorderie	46
4.6.5	Projet Portail CEPPP	47
4.6.6	Interprétation des résultats de SO-5	48
4.7	Discussion	49
4.7.1	Limitations	49
4.7.2	Comment les résultats obtenus soutiennent-ils le libre ?	49
4.7.3	Avancement sur le développement en réseau d'entraide	50
4.7.4	Avancement de la technopoïèse	50
4.7.5	Réalisation du robot logiciel générateur de code	51
CHAPITRE 5 CONCLUSION ET RECOMMANDATIONS		54
5.1	Synthèse des travaux	54
5.2	Limitations de la solution proposée	55

5.3	Améliorations futures	55
5.4	Importance de la recherche	57
RÉFÉRENCES		59
ANNEXES		64

LISTE DES TABLEAUX

Tableau 1.1	Tableau des dates de lancement du logiciel Odoo à partir de la version 6.0	3
Tableau 1.2	Nombre de modules, contenant un manifest installable, par version Odoo, à partir du premier janvier, minuit, par année, sur la plateforme ERPLibre 1.5.0.	4
Tableau 4.1	Statistiques sur le code des «wizards» sur la GUI qui gère la génération de MVC	37
Tableau 4.2	Statistiques sur le code de l'extraction de données de modules Odoo .	40
Tableau 4.3	Statistiques sur l'ensemble des modules Odoo pour le projet Accorderie.	47
Tableau 4.4	L'évolution entre la génération et la réingénierie des statistiques sur les langages du portail CEPPP	48

LISTE DES FIGURES

Figure 2.1	Altération du code avec la rétro-ingénierie et la réingénierie. Image modifiée [20]	12
Figure 2.2	Processus d'implantation d'un <i>ERP</i> en cinq étapes. Image adaptée de l'auteur [30]	14
Figure 2.3	Le processus <i>DevOps</i> . Source : inconnue	15
Figure 2.4	Fonctionnalités des plateformes <i>low-code</i> . Image provenant de [33] . .	16
Figure 2.5	Les composantes principales d'une plateforme de développement <i>low-code</i> . Image provenant de [34]	17
Figure 2.6	Mode direct et inverse	23
Figure 2.7	Exemple de code Hello, World!	23
Figure 2.8	Exemple de code Quine	23
Figure 2.9	Exemple de technique de génération de code basique d'un « <i>Hello World</i> »	24
Figure 2.10	Exemple de technique de génération de code basique paramétrable d'un « <i>Hello World</i> »	25
Figure 2.11	Exemple de technique de génération de code par template avec Jinja2 d'un « <i>Hello World</i> »	26
Figure 2.12	Exemple de template Jinja2 avec logique	26
Figure 2.13	Exemple de génération de module MVC Odoo avec la technique du <i>Scaffold</i>	27
Figure 2.14	Exemple de génération de module thème Odoo avec la technique du <i>Scaffold</i>	27
Figure 2.15	Exemple de technique de génération de code avec rétro-ingénierie d'un « <i>Hello World</i> »	29
Figure 4.1	Interaction du développeur avec le générateur de code	34
Figure 4.2	Architecture du générateur de code dans son ensemble, nommé machine. Le développeur peut modifier le code source directement ou utiliser l'interface de la machine qui passe par un ensemble de composantes.	35
Figure 4.3	Exemple support MVC dans GUI du générateur de code	38
Figure 4.4	Démonstration du code pour le MVC dans la section <i>wizards</i> du module du générateur de code	38
Figure 4.5	Démonstration des modifications effectués pour adapter le code de la migration de base de données pour supporter plus de fonctionnalités.	39

Figure 4.6	Int�raction entre les d�veloppeurs et les outils de d�veloppement dans un r�seau d'entraide	50
Figure 4.7	Architecture du g�n�rateur de code	52

LISTE DES SIGLES ET ABRÉVIATIONS

AGPL	<i>GNU Affero General Public License</i>
AGPLv3	<i>GNU Affero General Public License version 3</i>
AMD	<i>Advanced Micro Devices</i>
API	<i>Application Programming Interface</i>
AST	<i>Abstract Syntax Tree</i>
CC0	<i>No copyright reserved</i>
CEPPP	Centre d'excellence sur le partenariat avec les patients et le public
CPU	<i>Central processing unit</i>
CSV	<i>Comma-separated values</i>
CVE	<i>Common vulnerabilities and exposures</i>
DB	<i>Database</i>
ERP	Progiciel de gestion intégré
i18n	Internationalisation et la localisation
IA	Intelligence artificielle
JSON	<i>JavaScript Object Notation</i>
LCNC	<i>Low-Code-No-Code</i>
LGPL	<i>GNU Lesser General Public License</i>
MVC	Modèle-Vue-Contrôleur
NLP	<i>Natural language processing</i>
OCA	<i>Odoo Community Association</i>
ORM	<i>Object-Relational Mapping</i>
OSRM	<i>Open Source Routing Machine</i>
PEP8	<i>Python Enhancement Proposal 8</i>
PHP	<i>PHP : Hypertext Preprocessor</i>
PME	Petite et moyenne entreprise
RAM	<i>Random-access memory</i>
SCSS	<i>Sass Cascading Style Sheet</i>
SRS	Spécification des exigences logicielles
SSH	<i>Secure Shell</i>
SQL	<i>Structured Query Language</i>
USD	<i>United States Dollar</i>
XML	<i>Extensible Markup Language</i>

LISTE DES ANNEXES

Annexe A	Méthodologie revue de littérature et développement Agile	64
Annexe B	GUI générateur de code - les modèles	65
Annexe C	GUI générateur de code - les champs	66
Annexe D	GUI générateur de code - les codes	67
Annexe E	GUI générateur de code - les «hooks»	68
Annexe F	Test couverture technique générateur de code	69
Annexe G	Les 7 étapes pour démarrer un projet dans un réseau d'entraide . . .	71
Annexe H	Guide d'intégration des membres dans la communauté	72
Annexe I	Guide d'aide à la gestion du comportement en communauté	73
Annexe J	Guide de développement public	74
Annexe K	Guide de résolution de problème	75
Annexe L	Guide sur la documentation de projet	76
Annexe M	Guide sur les moyens de sécurité à mettre en place	77
Annexe N	Guide sur les bonnes pratiques du logiciel libre	78
Annexe O	Diagramme modèle de données Espace Membre Accorderie 2019 . . .	79
Annexe P	Diagramme nouveau modèle de données Espace Membre Accorderie 2023	80
Annexe Q	Diagramme processus pour demander, offrir, établir un échange et le valider - Accorderie 2023	81
Annexe R	Diagramme modèle de données du portail CEPPP septembre 2022 . .	82
Annexe S	Vue formulaire administration portail CEPPP	83
Annexe T	Vue formulaire partenaire portail CEPPP	84
Annexe U	Code u_C^0 dans le générateur de code	85

CHAPITRE 1 INTRODUCTION

1.1 Contexte et problématique

La valeur du marché des solutions *ERP*, *Enterprise Ressources Planning* ou progiciel de gestion des ressources d’une entreprise, 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 petite et moyenne entreprise (PME), en 2022 [3]. Le développement de système *ERP* est complexe, nécessite une maintenance exigeante et le risque d’introduire des erreurs est important [4] [5].

La compagnie Odoo a réussi à devenir l’un des principaux fournisseurs d’*ERP* à *Open Source* au monde [6], avec une communauté active de développeurs et d’utilisateurs. Odoo relève plusieurs enjeux et défis dans les *ERP*, notamment l’intégration de toutes les fonctions de l’entreprise, la personnalisation, la flexibilité, l’évolutivité et l’accessibilité.

Alors, comment serait-il possible d’accélérer le développement de fonctionnalités de la plateforme *ERP* Odoo¹ 12 communautaire ?

Pour répondre à ce besoin, la plateforme ERPLibre [7] a été créée dans l’objectif d’accélérer le développement de la plateforme Odoo communautaire. Ce mémoire va se concentrer sur la génération de code par des techniques de rétro-ingénierie et la gestion d’une communauté, dans un contexte d’un projet de logiciel libre, une solution développée, démontrée dans ce mémoire, en complément de la plateforme ERPLibre.

1.2 Objectif et but

Les objectifs des travaux effectués sont de développer un générateur de code pour accélérer le développement de fonctionnalité pour gérer les besoins d’une communauté, ainsi que de réaliser un auto-reproducteur, c’est-à-dire un générateur de code qui est capable de s’auto-générer, pour accélérer le développement de ce dernier.

Le but de ce mémoire est de montrer une solution qui contient des limitations, sur différents niveaux de production de développement logiciel, dans le domaine des *ERP*, pour aider les réseaux d’entraide dans l’adaptation de leurs fonctionnalités tout en accélérant le développement avec des solutions libres.

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

1.2.1 Choix de la plateforme *ERP*

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

Odoo a été créé initialement sous le nom de TinyERP, en février 2005, cette plateforme a évolué au fil du temps, elle a été renommée pour OpenERP autour d'octobre 2009, passant sous licence GNU Affero General Public License (AGPL), qui est une licence libre gauche d'auteur ayant pour but d'obliger les services accessibles par le réseau de publier leur code source.

À partir de 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 *Odoo Community Association* (OCA). Ainsi, la version 16.0 est la version stable actuelle. La recherche de modules commence à partir de la version 9.0, là où débute la divergence entre une version communautaire et entreprise.

Au printemps 2020, Odoo version 12.0 a été choisi par ERPLibre². Une recherche de modules, par version d'Odoo, a été effectuée sur 11 Go de code et de données, sur le projet ERPLibre version 1.5.0, voir le tableau 1.2. En date du premier janvier 2023, la version 12.0 est celle qui est retenue, avec 2 977 modules, puisqu'elle est celle qui a le plus de modules sur les 133 répertoires gérés par ERPLibre. Cette tendance pourrait changer dans le futur, mais la version 12.0 est celle dont fait l'objet ce mémoire.

Pour obtenir les résultats du tableau 1.2, un script a été développé pour trouver la quantité de modules en cherchant dans les 133 répertoires Git³, puis dans toutes les versions d'Odoo.

Parfois, la quantité de modules 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 jour ont évolué depuis.

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

Dans la section «total» du tableau 1.2, la section unique signifie que la somme va ignorer les

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

3. Logiciel de gestion de versions décentralisées

Tableau 1.1 Tableau des dates de lancement du logiciel Odoo à partir de la version 6.0

Légende	Version actuelle	Anciennes versions avec maintenance étendue	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 utilisée dans ERPLibre 1.5.0	
13.0	octobre 2019		
14.0	octobre 2020		
15.0	octobre 2021		
16.0	octobre 2022		

doublons. En date du premier janvier 2023, il y a eu, au total, 17 309 modules, dont 6 063 modules uniques. Cela 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, il peut avoir des bogues ou des fonctionnalités différentes entre elles.

1.2.2 Introduction Accorderie

La première étude de cas du présent mémoire a été effectuée sur l'Accorderie. Ce réseau d'entraide québécois était à la recherche d'une plateforme améliorée, avec des technologies plus récentes. Les membres ont commencé, depuis quelques années, à utiliser des plateformes alternatives, à cause de l'émergence des plateformes de réseaux sociaux, pour communiquer et échanger des services, sans passer par la plateforme Espace Membre. En ajoutant des fonctionnalités, dont l'automatisation⁴ des processus d'échange de temps, la plateforme devient plus attractive.

Nous avons décelé que le présent projet pourrait répondre aux besoins de l'Accorderie, avec le générateur de modules Odoo, qui permet de faciliter, entre autres, la maintenance dans le temps. De plus, la plateforme ERPLibre a le potentiel de leur éviter des coûts sur les licences, 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 droit d'auteur 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 premier janvier 2003. «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» [8]. 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.

1.2.3 Introduction CEPPP

La seconde étude de cas porte sur le CEPPP, Centre d'excellence sur le partenariat avec les patients et le public qui était à la recherche d'une plateforme pour la gestion du partenariat avec les patients et le public. Le Portail des partenaires («Portail») du 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 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. Donc, une solution [9], était déjà présente, mais elle était incomplète. Il manquait, dans cette solution, la gestion de l'anonymisation et l'interface demandait beaucoup de navigation pour atteindre l'information et exigeait un clic de souris pour chaque champs par section.

4. La gestion des feuilles de temps est manuelle avec validation par un membre de la communauté.

Le présent projet venait pallier aux problèmes rencontrés en facilitant et en accélérant le développement, en permettant une personnalisation et en développant l'anonymisation des données et était donc tout indiqué comme second cas pratique d'étude.

1.3 Organisation générale du document

Le chapitre 2 présente une revue de littérature concernant les termes de robot logiciel développeur, la génération de code, la rétro-ingénierie, les logiciels libres et *Open Source*, la sécurité en lien avec le développement logiciel, la complexité du développement des ERP, le *DevOps*, les interfaces logicielles *no-code / low-code*, la création de communauté et la poïèse. Il termine par le cadre conceptuel, qui contient des explications sur un générateur de code accompagné de rétro-ingénierie, ainsi que des exemples de générateur de code avec une méthode de test. Dans le chapitre 3, nous verrons en détail la méthodologie de travail et les objectifs de développement pour réaliser le générateur de code qui est séparé en 5 points : générateur de code, application de la rétro-ingénierie, l'interface du générateur de code, le déploiement du générateur de code et application dans des réseaux d'entraide. Le chapitre 4 comporte les résultats présentés, ainsi qu'une discussion générale. Enfin, le chapitre 5 conclut ce mémoire avec une synthèse critique et des pistes pour des recherches futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous présentons une revue de littérature concernant les termes de robot logiciel développeur, la génération de code, la rétro-ingénierie, les logiciels libres et à *Open Source*, la sécurité en lien avec le développement logiciel, la complexité du développement des *ERP*, le *DevOps*, les interfaces logicielles *no-code / low-code*, la création de communauté et la poïèse.

2.1 Caractéristiques de la plateforme Odoo

2.1.1 Architecture MVC

Pour générer des modules Odoo, le générateur de code a besoin de s'appuyer sur l'architecture Modèle-Vue-Contrôleur(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 par le modèle, la vue et le contrôleur. Le modèle représente les données et les règles de l'application. De plus, le modèle est responsable de la manipulation des données, de leur stockage et de leur récupération. La vue représente la présentation des données. De plus, la vue est responsable de l'affichage, à l'utilisateur final, des données stockées dans le modèle. Le contrôleur gère les interactions entre le modèle et la vue. De plus, 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 la maintenance.

2.1.2 Architecture ORM

L'utilisation de requêtes *Structured Query Language* (SQL) pour communiquer avec des bases de données demande un temps considérable au développeur à implémenter et elles nécessitent la programmation d'interface avec la base de données. L'utilisation d'un *Object-Relational Mapping* (ORM) permet de bonifier la productivité, de faciliter la maintenance et d'augmenter 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 orientée objet¹. 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.

2.1.3 Architecture modulaire par héritage

L'ensemble des fonctionnalités nécessaires à la gestion des procédures et ressources des organisations rend le développement complexe. Pour réduire cette complexité du développement logiciel, choisir une architecture modulaire permet la réutilisation de code et la création de relations fonctionnelles personnalisables au contexte des organisations.

L'héritage dans Odoo 12 peut se faire de deux manières : l'héritage par extension et l'héritage par substitution. L'héritage par extension 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. L'héritage par substitution 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 d'une organisation, sans toucher au code source d'origine et sans compromettre la compatibilité, par exemple, avec les mises à jour futures.

2.1.4 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 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 d'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.

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

2.1.5 Website builder

Le *Website Builder* est un outil qui permet une autonomie aux utilisateurs lambda² (augmenter l'accessibilité) dans la création et mise à jour de contenus de pages web sur le site vitrine de l'organisation. C'est un mécanisme *Low-Code-No-Code* (LCNC)³ dans Odoo 12 pour créer et modifier un site web multi-pages par un mécanisme de *drag and drop*⁴ avec des *snippets*⁵ paramétrables. Il permet de modifier une page web en réduisant le besoin de recourir à un expert technique, abaissant ainsi les coûts de développement.

2.1.6 Internationalisation et localisation

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 «Internationalisation et la localisation» (i18n), qui a été référencé [10], qui permet d'adapter les logiciels informatiques à plusieurs langues sur plusieurs localisations [11]. 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 d'extension «.pot», qui contient ces chaînes de caractères. Pour supporter une nouvelle langue et une localisation, on copie le fichier d'extension «.pot» pour faire un fichier d'extension «.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 d'extension «.po» avec les traductions déjà connues par le système, il suffit de le mettre à jour.

2.1.7 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*⁶. L'adaptation d'un *ERP* est complexe et nécessite l'intervention d'experts pour bien répondre aux besoins de l'utilisateur pour la personnalisation de la plateforme aux réalités de l'organisation. Bien que l'OCA travaille pour rendre accessible librement ces fonctionnalités, cela vient limiter les réseaux d'entraide à pouvoir se débrouiller de manière souveraine.

La plateforme ERPLibre a ainsi été créée, en début 2020, en encapsulant la version Odoo 12.0, sous licence AGPLv3, en offrant une alternative 100% libre, dans le but de faciliter

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

3. Qui nécessite peu ou pas du tout du code

4. Glisser-déposer

5. Fragment de code

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

le déploiement et le développement pour une organisation en permettant la gestion des dépendances avec Poetry, l'automatisation du déploiement avec les Docker, la gestion de tous les répertoires de module 1.2 avec Git Repo⁷, et plusieurs scripts pour le développement et une documentation propre pour son utilisation. Cela permet de rendre accessible, au même endroit, toute l'information nécessaire à la gestion de l'*ERP* d'une organisation.

2.2 Robot logiciel développeur

Dans ce mémoire, le terme robot logiciel revient à quelques reprises, ainsi sa définition est «agent intelligent programmé afin d'imiter les capacités d'un être humain dans un système informatique ou afin d'effectuer un ensemble de tâches prédéterminées de manière automatique.» [12] Les autres termes suggérés sont «robot informatique» et «robot». Ainsi, un robot logiciel codeur est une intelligence artificielle orientée au développement logiciel. En anglais, il pourrait être nommé un *DevBot* [13].

Selon les auteurs Erlenhov et al. en 2019 [13], le robot logiciel développeur idéal est un développeur de logiciels artificiel qui est autonome, adaptable et possède des compétences techniques, ainsi que sociales. La référence à l'aspect social d'un robot logiciel développeur est, par exemple, l'outillage aux développeurs autour d'un éditeur asynchrone en temps réel tels que CodeBuddy [14] ou CodePilot [15] dans leur développement collaboratif.

Dans ce projet, nous tentons d'offrir un outil aux développeurs pour accélérer leur développement avec des concepts d'amélioration continue. La solution n'est pas encore autonome, mais il y a un accent sur l'aspect social en proposant des outils de développement intégré.

2.3 Génération de code

Utiliser un générateur de code dans un contexte d'application web est efficace, comme le démontre les auteurs Uyanik et al. en 2020 [16], ils obtiennent un gain de performance en temps de développement de 98.95% et 0 bogue via le générateur de code. Cependant, le code source n'est pas accessible et, par conséquent, il est difficile de comparer son efficacité du côté pratique. De plus, contrairement à la solution proposé dans ce mémoire, cette solution ne semble pas disposer d'un ORM, puisqu'elle génère elle-même le SQL, elle ne contient pas de rétro-ingénierie et elle ne semble pas dédiée pour une communauté.

Dans un autre projet de les auteurs Pichidienthum et al. en 2021 [17], en ajoutant une interface LCNC sur un générateur de code sur Odoo, obtiennent 20% de réduction de temps

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

pour le développement de module, puis des utilisateurs non développeurs peuvent utiliser cet outil pour faire des modules. Cette recherche est similaire à celle expérimentée dans ce mémoire, en excluant l’aspect de la rétro-ingénierie qui est manquante. De plus, la solution LCNC utilisé est un logiciel propriétaire non compatible avec la licence de ERPLibre, qui fait le choix de développer une solution entièrement libre pour la rendre accessible à la communauté.

2.4 Rétro-ingénierie

La rétro-ingénierie⁸ est de l’ingénierie inversée, c’est-à-dire qu’on étudie un objet, ou une technologie, pour comprendre son fonctionnement interne, sa méthode de fabrication. Une fois que le système existant est compris, passant de l’implémentation à la conception, on peut débiter la réingénierie, au besoin, pour altérer le code source et développement un nouveau système cible, voir Figure 2.1. Il y a plusieurs domaines de rétro-ingénierie, tout dépend de quel type d’information ou manipulation dont on a besoin. Dans ce mémoire, l’objectif est d’extraire des informations pour permettre la reproduction de code via le générateur de code. Les techniques diffèrent selon le type d’information à extraire, que ce soit en Python, en Javascript, en XML pour différent type de vue. Les manipulations effectuées sur l’extraction sont pour trouver des modèles spécifiques et, ainsi réduire, la quantité de paramètres à la génération.

Dans le travail de El Idrissi en 2022 [18], il démontre de la rétro-ingénierie pour comprendre le niveau de variabilité des modules dans Odoo, en examinant les dépendances entre les modules. Ces travaux pourraient être pertinents pour notre recherche avec le générateur de code, mais nous ne sommes pas rendu à inspecter un ensemble de module, nous travaillons sur des modules individuels.

De plus, ce mémoire ne fait pas de vérification sur les algorithmes, le travail est fait de manière macro pour détecter les fonctions et pouvoir les reproduire, sans valider le contenu. Par exemple, dans le travail de Laverdière et al. en 2017 [19], ils utilisent la technique du *Pattern Traversal Flow Analysis* pour repérer des vulnérabilités sur les contrôles d’accès, sur Wordpress en PHP. Pour faire le lien avec Odoo, il y a des gestions de privilèges et le code peut, par exemple, rendre accessible l’accès super administrateur en utilisant la méthode «`sudo()`» sur les outils du ORM. Dans le futur, nous pourrions implémenter ce type de vérification pour avertir le développeur d’éventuels problèmes de sécurité.

8. Terme en anglais : *reverse engineering*

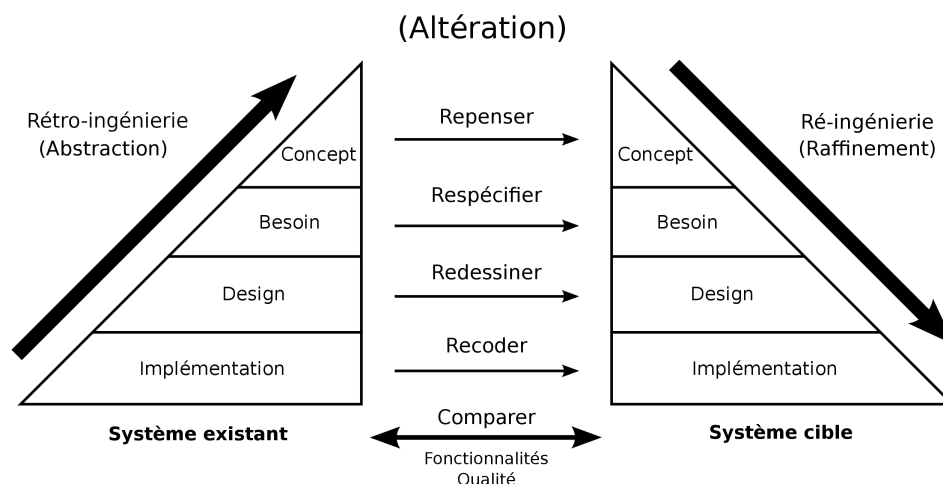


Figure 2.1 Altération du code avec la rétro-ingénierie et la réingénierie. Image modifiée [20]

2.5 Logiciel Libre et *Open Source*

Le partage de bibliothèque *Open Source* est une méthode utilisée pour accélérer le développement, permettant la réutilisation de code. L'*Open Source* permet aussi de supporter l'interopérabilité [21], qui est la capacité de différents logiciels d'interagir et communiquer efficacement entre eux, de manière transparente et harmonieuse, sans entraves ni obstacles, même s'ils ont été développés par différentes organisations. Ainsi, un générateur de code qui produit du code ouvert va pouvoir bénéficier de l'interopérabilité. Cependant, la solution dans ce mémoire est libre d'utilisation, avec licence restrictive AGPLv3, qui oblige de redistribuer les modifications sous la même licence, ce qui réduit l'interopérabilité vers des solutions propriétaires, à l'avantage de la communauté.

D'ailleurs, comme le mentionne les auteurs Hertel et al. en 2003 [22], un des facteurs de motivation important est la perception de l'indispensabilité de ses propres contributions dans une équipe, associée à une évaluation élevée des objectifs de l'équipe et un fort sentiment d'auto-efficacité personnelle. Le générateur de code ne doit pas remplacer l'utilisateur, il doit l'accompagner dans son développement, c'est pourquoi, dans ce mémoire, l'accent est mis sur l'aspect social, en proposant des outils pour accélérer le développement pour les membres de l'équipe.

2.6 Sécurité

Bien qu'il y ait plusieurs aspects de sécurité à considérer, ce qui nous intéresse, ici, est la sécurité logicielle via son développement et ses dépendances.

Thompson démontre en 1984 [23] qu'il est possible d'injecter du code malveillant dans les outils de compilation d'un logiciel pour créer une faille de sécurité dans l'exécutable. Shah explique dans son blog en 2020 [24], que la morale est assez simple : il est impossible de fonctionner⁹ sans faire confiance à quelqu'un. À moins que vous n'ayez écrit tout le code vous-même, vous devrez faire confiance à la sécurité d'une partie du processus de traitement du programme.

Hors, s'il est nécessaire de faire confiance, comment peut-on faire confiance lorsqu'on observe des problèmes de compatibilité de licence libre [25] [26] ? Le développeur choisi des bibliothèques libres qui utilisent d'autres bibliothèques pas nécessairement libres et qui peuvent contenir des failles de sécurité [27], sans en être conscient.

Pour revenir à l'article de Thompson, comme le mentionne Yona dans son blog en 2023, cela suggère que, tant que nous ne sommes pas en mesure de rétroconception¹⁰ et de contrôler pleinement la fonctionnalité des réseaux neuronaux, il existera un risque inhérent. Même si nous parvenons à résoudre le problème d'alignement et à rendre l'intelligence artificielle (IA) conviviale pour l'homme, les attaquants qui obtiennent un accès en écriture aux poids¹¹ pourront implanter leurs portes dérobées. [28]

Ainsi, il serait pertinent de développer un robot codeur qui soit en mesure de faire de la rétro-ingénierie sur une application et ses bibliothèques. Le générateur pourrait comprendre le fonctionnement de cette chaîne de dépendances et générer le code idéal pour la remplacer par une nouvelle application digne de confiance, en validant la propriété intellectuelle libre.

2.7 Les complexités de développer *ERP*

L'auteur Pitetti mentionne, en 2010 : «Le choix d'un *ERP* libre n'implique pas nécessairement une diminution des coûts dans la mesure où le reengineering¹² n'a pas été correctement effectué[...] Un guide des bonnes pratiques permet d'éviter ces erreurs, de limiter les coûts du projet ou de se renseigner sur les différentes phases du projet» [29]. C'est pourquoi il est important de suivre les différentes étapes de réalisation du projet pour réduire les erreurs d'ingénierie sur le développement, voir Figure 2.2 qui démontre un processus d'implantation d'un *ERP* en cinq étapes proposé par l'auteur Braud en 2008 [30]. À ce jour, les travaux du générateur de code viennent n'accélérer que la partie développement de ce processus, mais ils apporteraient un gain considérable si nous pouvions l'appliquer à toutes ces étapes.

9. D'utiliser et d'avoir confiance sur des technologies

10. Faire de la rétro-ingénierie sur un développement.

11. Un poids est la valeur numérique d'un neurone dans un réseau de neurones.

12. réingénierie

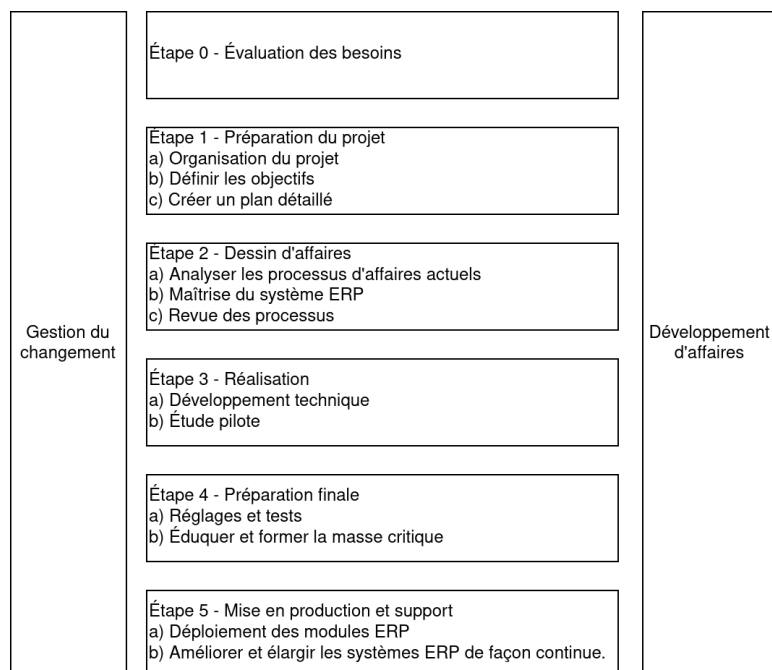


Figure 2.2 Processus d'implantation d'un *ERP* en cinq étapes. Image adaptée de l'auteur [30]

Un autre facteur complexe est l'association des processus d'affaires actuels à la réingénierie vers un nouveau modèle de processus d'affaires adapté aux contraintes technologiques, ainsi que les fonctionnalités déjà existantes sur la plateforme choisie. Comme on peut le constater dans le travail des auteurs Medjek et al. en 2018 [31], un travail a été effectué pour énumérer les fonctionnalités de la plateforme, ainsi que de faire du développement personnalisé, avec tous les diagrammes de conception génie logiciel.

Dans Odoo, le nombre de module augmente avec le temps et diffère entre les versions, une recherche fastidieuse doit être effectuée pour réduire le temps de développement et pour éviter de réinventer la roue. En plus de suivre toutes ces étapes, il faut mettre en place une pérennité pour l'amélioration continue sur le projet pour des adaptations futures aux nouveaux processus de l'entreprise.

2.8 *DevOps*

Comme le mentionne l'auteur Ebert en 2016 [32], le *DevOps* consiste en un développement et une provision de processus commerciaux rapides et flexibles. Il intègre efficacement le développement, la livraison et les opérations, facilitant ainsi une connexion fluide et flexible de ces silos traditionnellement séparés, voir Figure 2.3. Ce mémoire met l'accent sur la section code du *DevOps*, mais il doit éventuellement s'impliquer à toutes les étapes du *DevOps* pour

ainsi automatiser la chaîne de développement au complet.

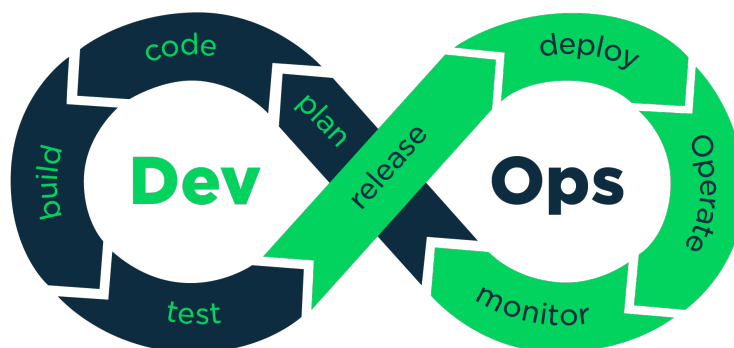


Figure 2.3 Le processus *DevOps*. Source : inconnue

2.9 Logiciel *no-code* / *low-code*

Le LCNC est un concept qui permet à l'utilisateur de développer une plateforme ou des segments d'application en utilisant peu ou pas de code. Selon l'auteur Bock en 2021 [33], la Figure 2.4 définit un ensemble des fonctionnalités nécessaires pour une plateforme LCNC. Les travaux de ce mémoire supportent : la gestion des rôles ; un mécanisme de déploiement et d'exportation ; un mécanisme pour changer le design de l'interface utilisateur ; un mécanisme pour coupler l'interface à un modèle et un contrôleur (MVC) ; un mécanisme pour faire le rendu visuel sur différents types d'appareils ; une gestion des processus du système et de la machine ; un système de gestion des états et des transitions ; un mécanisme de spécification fonctionnel de base ; un générateur de code de composantes ; un mécanisme d'accès à des *Application Programming Interface* (API) externes ; des composantes de conception d'un modèle de données ; des composantes pour spécifier des structures de données ; la gestion de base de données interne et externes par API. Ainsi, la plupart des composantes communes sont supportées dans ces travaux, cependant il manquerait un générateur d'algorithme et un générateur d'interface dédiés au générateur de code.

Selon l'auteur Apurvanand en 2020 [34], voir Figure 2.5, pour mettre en place une plateforme de développement avec réduction de code, il est nécessaire d'avoir une base de données, des services externes, un répertoire de modèles réutilisables, une communauté de collaborateur, un compilateur, un optimisateur des requêtes, un générateur de code, un ensemble de service au développement (journalisation, déploiement, audit, versionnage de code), ainsi qu'une application visuelle pour tout produire tous les éléments nécessaires à l'application désirée.

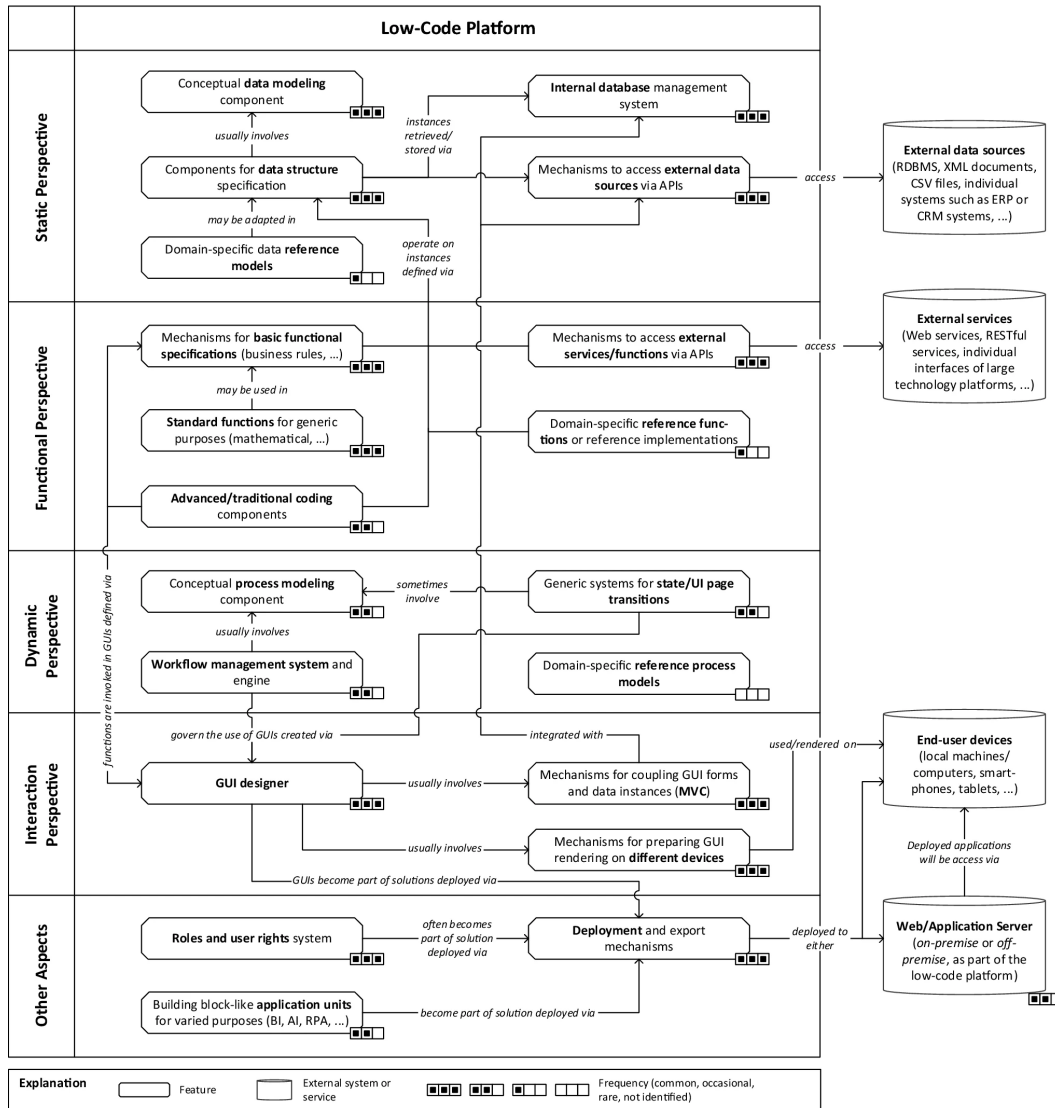


Figure 2.4 Fonctionnalités des plateformes *low-code*. Image provenant de [33]

Dans les travaux de ce mémoire, tous ces éléments sont déjà accessibles, mais en début de maturité.

2.10 Création d'une communauté

«Il est important de comprendre que les logiciels libres et ouverts sont soutenus et entretenus, non pas uniquement par un éditeur logiciel traditionnel mais également par une communauté constituée de ses utilisateurs, qu'il s'agisse d'éditeurs, de sociétés de services spécialisées ou d'utilisateurs. Ce sont ces communautés qui décident de l'orientation technologique, de l'adaptation et de l'évolution du code source ainsi que des versions et des mises à jour

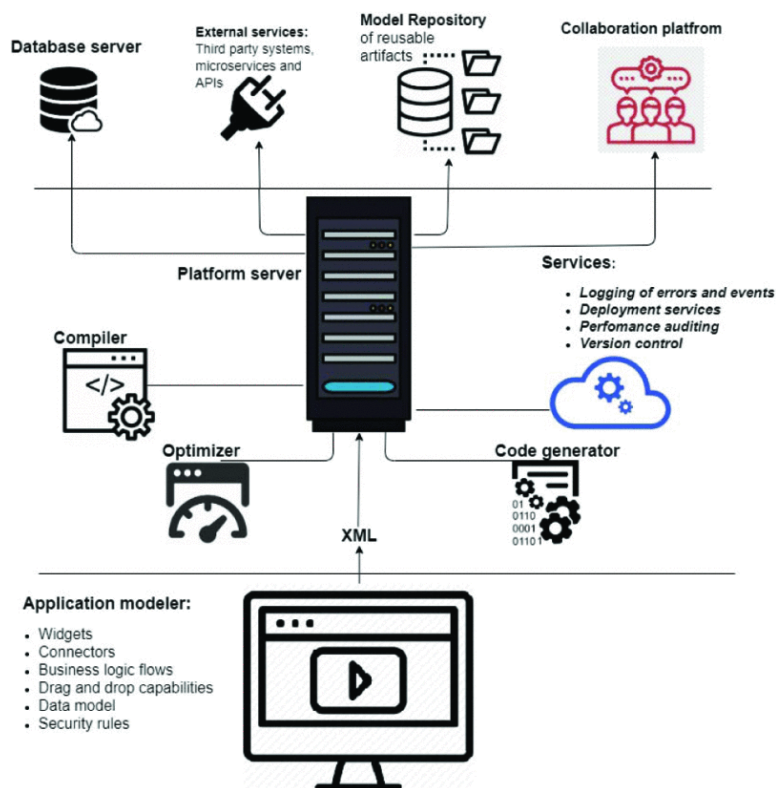


Figure 2.5 Les composantes principales d'une plateforme de développement *low-code*. Image provenant de [34]

qui seront rendues disponibles. Un logiciel libre et ouvert évolue proportionnellement au dynamisme de sa communauté[...]» [35]

Selon l'auteur Meyer et al. en 2007, «La conviction selon laquelle la créativité d'un groupe est à la fois le fruit de l'autonomie et de l'engagement de ses membres commence à se diffuser au sein des sciences sociales. Le logiciel libre en offre un parfait exemple ; c'est un logiciel dont le code source est rendu ouvertement disponible et modifiable par tous : une fois publié, le code source n'appartient plus à son créateur et son évolution dépend de ce que les autres usagers en font.» [36]

Un logiciel libre est en vie tant qu'il y a une communauté active qui le maintient. De plus, il est important de mettre l'humain au centre des opérations et ça serait une erreur de vouloir automatiser, rendre 100% autonome le générateur de code, puisqu'il doit rester au service des communautés.

2.10.1 Guide construire une communauté *Open Source*

Un guide [37] est accessible publiquement pour aider les gestionnaires de communautés en 4 sections avec des titres indicateurs d'orientation : la mise en place de votre projet pour le succès, cultiver votre communauté, résoudre les conflits et la communauté est le coeur de l'*open source*. Il y a beaucoup d'autres guides, mais en général, ils n'intègrent ni les aspects de génie industriel et ni les critères éthiques de GNU concernant l'hébergement de logiciel.

2.10.2 Critères éthiques de GNU concernant l'hébergement de logiciel

La licence AGPLv3 n'est pas toujours bien respectée [38]. Il est important d'assister le développeur dans ses choix de licence et dans la compréhension des licences des bibliothèques tierces. Cependant, une fois que le logiciel est prêt à être mis en production, il y a des critères éthiques concernant l'hébergement de logiciel [39] qui doivent être accessibles sur les projets du réseau d'entraide. Un guide, avec des critères mesurables, pour les services destinés à tous ceux qui veulent utiliser un service pour héberger publiquement du code source libre, ainsi qu'éventuellement des programmes exécutables devra être mis à la disposition des organisations. Ces critères se concentrent sur la protection de la vie privée, le fonctionnement sans JavaScript non libre¹³, la compatibilité avec les licences à la gauche d'auteur¹⁴ et leur philosophie, et l'absence de discrimination contre les utilisateurs, quels qu'ils soient.

Ce mémoire ne réutilise pas ce guide et n'en propose pas une version adaptée, puisque nous trouvons qu'il n'est pas nécessaire de le modifier. En fait, il faudrait mettre une validation automatisée sur les critères d'hébergement à suivre en lien avec la partie déploiement du *DevOps* et au niveau de la surveillance pour protéger les utilisateurs et leurs droits dans les communautés.

2.11 Poïèse

La génération de code est proche des concepts de systèmes qui se copient, reproduisent et s'entretiennent. Ainsi, l'analogie avec le vivant passe par l'emploi de termes des approches systémiques courantes en biologie. Nous allons mieux définir ces termes dans cette section.

13. <https://www.fsf.org/campaigns/freejs>

14. Copyleft

2.11.1 Définition de la poïèse

La poïèse [40] (ou poïesis) est un terme d'origine grecque qui désigne le processus créatif de fabrication, de production ou de création. Il est souvent utilisé dans le contexte de l'art et de la littérature pour décrire le processus de création d'une œuvre, que ce soit un poème, une pièce de théâtre, un roman ou une peinture.

Dans ce contexte, la poïèse est considérée comme un processus actif et dynamique, impliquant l'imagination, l'inspiration, la créativité et la maîtrise technique. Elle implique souvent un certain niveau d'engagement personnel et émotionnel de la part de l'artiste ou du créateur.

En dehors de l'art, le terme poïèse [41] peut également être utilisé pour décrire tout processus de création ou de production, y compris dans des domaines tels que la science, la technologie ou l'industrie.

Les termes «Allopoïèse», «Autopoïèse», «Sympoïèse» ont été inventés pour décrire des phénomènes biologiques, hors dans ce mémoire, ils ont été adaptés pour un contexte technologique.

2.11.2 Technopoïèse

La technopoïèse peut être décrite comme un système qui développe une technologie.

La technologie est là pour assister l'utilisateur et l'accompagner dans l'évolution de celle-ci. «Parce que l'appareil prend place entre la manifestation de l'œuvre et le travail de l'artiste en les découplant, en leur imposant une langue intermédiaire qui code puis décode. L'artiste produit des lignes de code, que la technologie intègre pour fournir à l'œuvre la source de sa manifestation : elle sépare ontologiquement¹⁵ le travail de l'un et son résultat dans l'autre.» [42] L'artiste¹⁶ ici signifie un programmeur informatique dans un processus créatif de fabrication, de production ou de création sur des technologies.

Pour éviter que la machine ne prenne le dessus, il faut l'orienter vers la technopoïèse. «Si la technologie est un médium parasite, alors ne suffit-il pas de compter sur la charge poétique du médium primaire, pour conserver à la poïèse ses caractères nécessaires – et imaginer l'art technologique comme un art d'abord, agrémenté, partiellement seulement, de technologie?» [42]

Ainsi, le terme «technopoïèse» fait référence à la capacité de l'humanité à créer et à façonner la technologie pour répondre à ses besoins et à ses désirs. La relation entre la sympoïèse et la technopoïèse permettrait de concevoir des technologies plus durables et écologiquement

15. Une ontologie est une représentation formelle et explicite de la connaissance d'un domaine, qui spécifie les concepts, les relations et les entités qui existent dans ce domaine et comment ils sont interconnectés.

16. Voir l'artiste de la «Définition de la poïèse» 2.11.1

responsables, qui favorisent la production collective et collaborative dans les écosystèmes.

2.11.3 Allopoïèse

L'Allopoïèse peut être décrite comme un système qui développe¹⁷ quelque chose avec des composantes externes.

Sur Wikipédia, «L'allopoïèse est le processus par lequel un système produit quelque chose qui n'est pas le système lui-même. Ceci est le contraire de l'autopoïèse.[...] La plupart des processus de production industrielle sont allopoïétiques : une chaîne de montage peut produire des voitures mais pas les machines utilisées dans cette forme de production. [...] La reproduction n'est pas une auto-production.» [43] [44]

Ce mémoire démontre la mise en place d'une allopoïèse via la génération de code, ainsi une machine qui génère un produit logiciel.

2.11.4 Autopoïèse

L'autopoïèse peut être décrite comme un système qui se développe par soi-même avec seulement ses composantes internes.

«L'autopoïèse est la propriété d'un système de se produire lui-même, en permanence et en interaction avec son environnement, et ainsi de maintenir son organisation (structure) malgré son changement de composants (matériaux) et d'informations (données).[...]le maintien de sa propre organisation (auto-production)» [45].

Le maintien de sa propre organisation signifie l'auto-production, voir exemple illustratif autoreproducteur du code 2.12.2.

Selon l'auteur Nomura en 2000 [46], un système est de l'autopoïèse dans le contexte qu'il est autonome, c'est-à-dire qu'il doit être capable d'apporter des changements variés pour maintenir son organisation ; il est individuel, c'est-à-dire qu'il doit être indépendant dans sa définition, par sa prise de décision par rapport aux observateurs externes, en reproduisant à répétition et en maintenant son organisation ; il est connaissant et établit ses limites, c'est-à-dire qu'il doit être capable d'établir ses limites dans son processus de reproduction par lui-même, sans se faire affecter des limites établies par les observateurs externes ; il est absent d'entrant et de sortant, c'est-à-dire que les stimulus externes doivent être interprétés dans un contexte d'observation pour en retirer de l'amélioration continue, ils ne doivent pas impacter la maintenance de l'organisation directement, mais son évolution doit en prendre compte.

17. production/fabrication : utiliser sans limitation, Modifier pour adapter, Étudier pour comprendre le fonctionnement et Copier pour reproduire.

Le concept de vue sur les entrants et sortants d'un système est une perception des observateurs externes et ne clarifie pas l'organisation ou les opérations de production du système.

La conception d'un système autopoïèse, de l'article [46], devrait comporter les points suivants : les composantes du système sont déterminées par les opérations du système ; les opérations du système sont produites avant les conditions initiales ; les opérations du système sont seulement exécutées pour leur propre réussite et non pour réaliser la production d'un produit ; dans les opérations du système, ce qui se passe à l'intérieur du système est clairement différent des jugements des observateurs externes.

Appliquer l'autopoïèse sur un système est de forcer un changement de point de vue vers l'intérieur du système, puisque l'extérieur est matière à interprétation par la distinction de son environnement. En sciences naturelles, ce changement de point de vue est difficilement acceptable, puisque le point de vue est fait par des observateurs externes.

Des modèles mathématiques sont expliqués dans l'article [46] tel un système de réparation du métabolisme ((M,R) systems), introduit par Rosen, pour démontrer le *Quasi-Autopoietic Systems*. Puis, il y a des modèles d'apprentissage automatiques qui ont été inspirés de l'autopoïèse pour effectuer des tâches de reconnaissance de formes. Pour pouvoir représenter l'autopoïèse, en mathématique ou en modèle informatique, il est nécessaire de trouver un mécanisme d'un système qui crée son espace avec ses limites et son environnement, par soi-même.

Ainsi, dans ce mémoire, nous tentons de réaliser l'autopoïèse avec des concepts d'auto-reproducteur à travers un *ERP* libre.

2.11.5 Sympoïèse

La sympoïèse peut être définie comme un système qui se développe en collectivité. C'est un concept utilisé en écologie et en théorie des systèmes pour décrire les processus de production collective et collaborative dans les écosystèmes. Elle se distingue de l'autopoïèse, qui est le processus par lequel un système produit et reproduit ses propres composants de manière autonome. Par exemple, les coraux sont des collectifs d'organismes en interaction qui produisent des structures complexes telles que des récifs, qui ont des effets bénéfiques sur l'écosystème dans son ensemble.

Selon Guilibert en 2019, «La nature est une puissance d'engendrement qui surgit et s'auto-produit. Donna Haraway a récemment proposé de remplacer le concept «d'autoproduction» par celui de «sympoïèse» qui désigne la coproduction du milieu par des espèces en interrelations plutôt que l'activité autonome de certains organismes isolés.» [47]

Ainsi, le développement de technologie en collectivité va permettre de gérer des cas d'urgence humanitaire tel que le «mouvement des villes en transition» [48]¹⁸ en développant des technologies permettant de mettre en place des systèmes d'échange local [49]¹⁹.

La liaison avec ce mémoire, nous tentons de réaliser l'autopoïèse en mettant l'accent sur la collectivité, la collaboration de développement à travers un réseau d'entraide, pour ainsi développer un système de sympoïèse. Cependant, la sympoïèse n'est pas réflétée dans les résultats, c'est plutôt une vision.

2.12 Définitions et concepts

2.12.1 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 2.6. 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 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.

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 Figure 2.7.

De plus, dans ce mémoire, nous utilisons le terme «générateur de code» qui est parfois représenté par «machine», cela fait référence à un ensemble de modules. Lorsque nous utilisons le terme robot logiciel codeur, nous faisons référence au «générateur de code» qui y est intégré,

18. https://fr.wikipedia.org/wiki/Ville_en_transition

19. https://fr.wikipedia.org/wiki/Systeme_d'echange_local

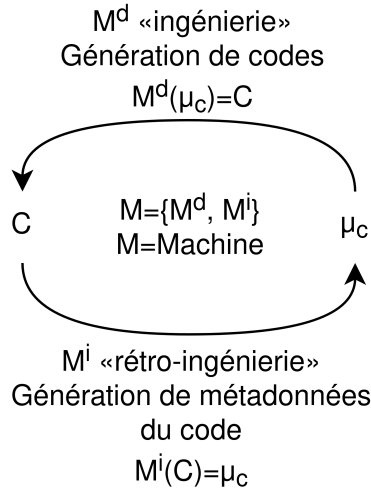


Figure 2.6 Mode direct et inverse

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

Figure 2.7 Exemple de code Hello, World!

mais aussi à un ensemble de scripts dans ERPLibre, puisque le robot doit avoir un contrôle sur son projet en entier, jusqu'au déploiement de celui-ci. Ainsi, la machine est au niveau conceptuel théorique, le générateur de code est au niveau composante logicielle, et le robot logiciel codeur est au niveau de solution logicielle.

2.12.2 Exemples illustratifs d'auto-reproducteur

Le Quine

«Un quine [50] (ou programme auto-reproducteur, self-reproducing en anglais) est un programme informatique qui imprime son propre code source» [51] 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 Figure 2.8, la sortie textuelle dans la console, lors de l'exécution, est la même que son code.

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

Figure 2.8 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.

2.12.3 Exemples illustratifs de générateur de code

La génération de code est un des moyens pour soutenir le développeur dans le développement d'un logiciel. C'est la partie «création de code» de la méthodologie *DevOps* 2.8.

Technique de génération de code basique

Dans Figure 2.9, la fonction «eval» en Python est dynamique, c'est-à-dire qu'elle permet l'exécution à partir d'une chaîne de caractères, 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.

Dans Figure 2.10, cette technique basique est paramétrable, contrairement au premier exemple Figure 2.9. 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.

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 «template».

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

Code 2.1 C de Figure 2.9

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

Code 2.2 μ_C de Figure 2.9

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

Code 2.3 M(μ_C) de Figure 2.9

Figure 2.9 Exemple de technique de génération de code basique d'un «*Hello World*»


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

Code 2.4 C - fichier C.py de Figure 2.10

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

Code 2.5 μ_C - fichier uC.json de Figure 2.10

```
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 2.6 M(μ_C) de Figure 2.10

Figure 2.10 Exemple de technique de génération de code basique paramétrable d'un «*Hello World*»

Le code de La Figure 2.11 utilise la bibliothèque «Jinja2». C'est un mécanisme similaire à Figure 2.10, cependant la logique est intégrée directement dans le fichier template.

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 Figure 2.13, tout est presque commenté, rien n'est utilisable, mais nous avons la structure MVC. Le gain d'accélération au développement est minime.

Cette fois-ci, dans Figure 2.14, tout est vide, le module «*theme__module*» produit une erreur à l'installation. Il est plus efficace de copier et de modifier un thème existant que d'utiliser le «*scaffold*».

Le gain d'accélération au développement peut être considéré comme négligeable. Copier un module fonctionnel et l'adapter est plus efficace. Admettons 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 ou lire la documentation officielle sur comment développer un module Odoo.

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

Code 2.7 C - fichier C.py de Figure 2.11

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

Code 2.8 μ_C - fichier uC.json de Figure 2.11

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

Code 2.9 template - fichier template de Figure 2.11

```
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 2.10 M(μ_C) de Figure 2.11

Figure 2.11 Exemple de technique de génération de code par template avec Jinja2 d'un «Hello World»

```
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 %}
```

Figure 2.12 Exemple de template Jinja2 avec logique

```

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

```

Figure 2.13 Exemple de génération de module MVC Odoo avec la technique du *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

```

Figure 2.14 Exemple de génération de module thème Odoo avec la technique du *Scaffold*

Technique de génération de code par rétro-ingénierie

Lorsqu'on veut faire de la rétro-ingénierie [52] avec un générateur de code, l'objectif est de faire de la réingénierie sur la partie génération, ainsi on altère le code, voir Figure 2.1.

Dans Figure 2.15, le code 2.12 utilise la bibliothèque *Abstract Syntax Tree (AST)* 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 logicielle entre l'extraction d'information et la génération du code. Dans ce contexte, le résultat 2.13 du code source original 2.11 devient formaté en PEP8 [53].

2.12.4 Tester un générateur de code

Il existe le test «*Output comparison testing*» [54] 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 textuelle.

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 opérées.

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

Code 2.11 C mal formaté - fichier C.py de Figure 2.15

```
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 2.12 M qui extrait μ_C pour générer C.py de Figure 2.15

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

Code 2.13 C corrigé - fichier C.py de Figure 2.15

Figure 2.15 Exemple de technique de génération de code avec rétro-ingénierie d'un «*Hello World*»

CHAPITRE 3 OBJECTIFS ET MÉTHODOLOGIE

Dans ce chapitre, nous examinons l'objectif général de ce projet qui a été de créer et de valider le fonctionnement d'un robot logiciel générateur de code qui sert à l'implantation d'un *ERP* libre. Plus spécifiquement, nous allons nous concentrer sur cinq sous-objectifs (SO) suivants, soit : développer un générateur de code de module sur Odoo ; développer une logique d'amélioration continue sur l'écriture du code ; développer une interface permettant de paramétrer la génération de code ; développer un système de distribution et développer un système de gestion de communauté.

La méthodologie classique en recherche scientifique, voir l'annexe A, est représentée par la section à droite qui débute par «Faire une revue de littérature», elle est bonne en théorie. Cependant, au niveau pratique, nous avons adopté la méthodologie Agile¹ pour le développement des composantes d'un générateur de code, représentées par la section de gauche du schéma en annexe A. Les fonctionnalités de génération de code ont été validées par des tests de comparaison entre les codes générés et les codes révisés par le développeur. Dans les recherches de solution existante, le générateur de code développé par Luis en 2019 [55] a été publié sur Github, cependant il a nécessité des modifications qui seront détaillées dans les résultats.

3.1 SO-1 - Générateur

Cette section concerne la génération de code, la génération de module et génération de modèles de données. Les étapes à accomplir sont de développer une logique d'écriture de module sur une architecture de MVC avec support de plateforme web, la mise en place d'un concept de gabarit de code qui génère du code, la mise en place de la génération de code à partir de données et la génération d'un module à partir d'une base de données externe.

3.2 SO-2 - Rétro-ingénierie

Cette section porte sur la partie de rétro-ingénierie, l'amélioration continue et la qualité logicielle. Les étapes à accomplir sont de développer la capacité de comprendre une structure de code et de la reproduire, de définir ce qui est de l'amélioration continue et son application dans un contexte d'automatisation, de mettre en place des tests de validation de code sur des

1. https://fr.wikipedia.org/wiki/M%C3%A9thode_agile

critères de qualité mesurables et d'intégrer des règles de codage standardisées pour favoriser le réseau d'entraide.

3.3 SO-3 - Interface

Cette section concerne l'interface utilisateur. Les étapes à accomplir sont de proposer une classification des techniques que le robot logiciel codeur peut réaliser en programmation, de développer une interface permettant le contrôle du robot logiciel codeur pour l'orienter dans la programmation de fonctionnalités et de rendre disponible une interface LCNC pour permettre aux utilisateurs de programmer leurs fonctionnalités.

3.4 SO-4 - Déploiement

Pour la section déploiement, il faut développer un système de distribution du robot logiciel codeur.

3.5 SO-5 - Réseau d'entraide

Cette section comporte la gestion du développeur, ainsi que les projets d'études. Les étapes à accomplir sont de documenter les processus de développement pour amener les utilisateurs à contribuer et les faire participer à la maintenance, de mettre des guides pour permettre le sentiment d'appartenance, de mettre en place une politique tolérance zéro avec un système de communication non violente et créer un lieu de discussion publique, d'élaboration du prototype pour les spécifications du réseau de l'Accorderie du Canada et d'élaboration du prototype pour les spécifications de l'organisme CEPPP du Canada.

3.6 Environnement informatique

L'environnement informatique choisi est la plateforme ERPLibre 1.5.0 contenant Odoo 12 communauté, qui utilise les langages Python 3.7, XML, Javascript et SCSS. Nos tests et développements 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.

3.7 Méthodologie de test

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

Le tout a été programmé grâce à la technique de parallélisme avec la bibliothèque *Asyncio* et des nouveaux processus et non des *thread*². Seul le test de nouveau projet est exécuté après le parallélisme, ce qui ajoute plus de temps à l'exécution.

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

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 exécutées dans la machine. Ça devient une métrique de la performance d'utilisation sur la quantité de fonctionnalités générées.

2. Processus légé

CHAPITRE 4 RÉSULTAT EXPÉRIMENTAUX

Nous présentons les résultats en commençant par décrire l'implémentation du modèle conceptuel du robot logiciel codeur présenté dans la sous-section 2.12.1. Ensuite, nous présenterons 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, basée sur une GUI [55] 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 ERPLibre¹.

L'interface permet l'interactivité avec un utilisateur ou le système-cible. Dans le cadre conceptuel 2.12.1, l'interface est découpée 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 , ce sont les macro-méta-données, alors que μ_C^B , sont 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 itérations agiles, les actions

1. <https://erplibre.ca>

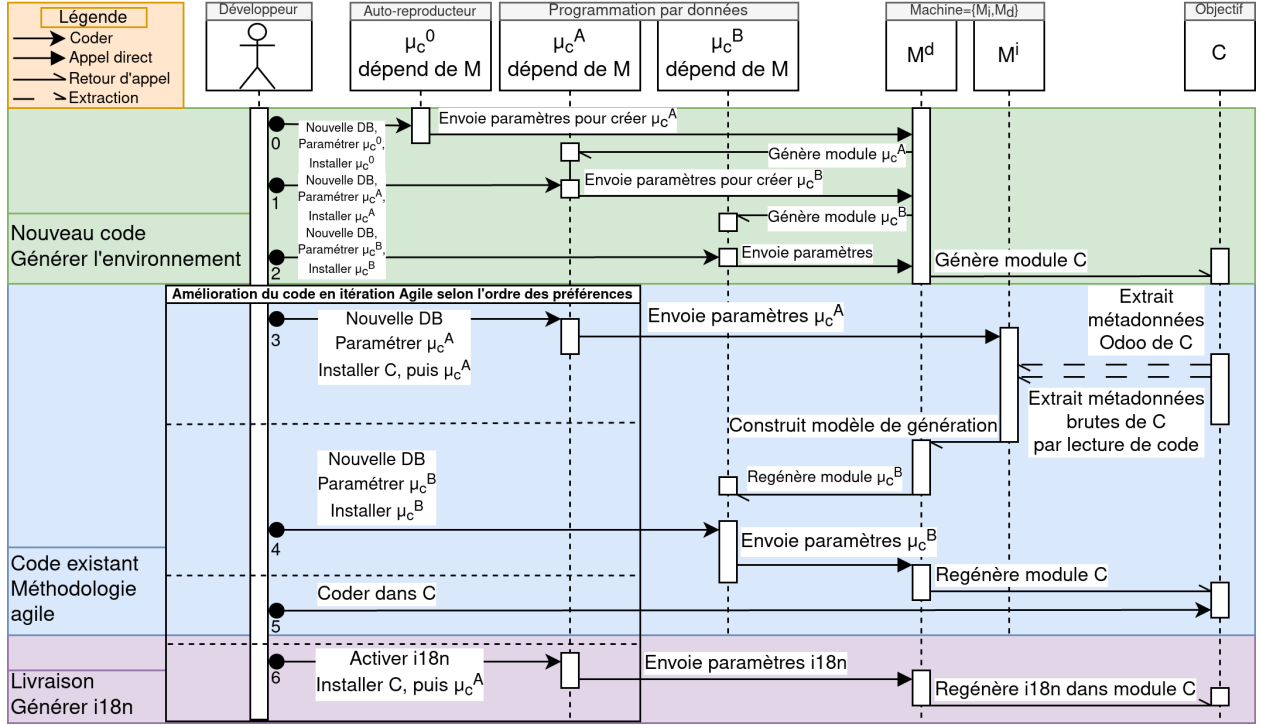


Figure 4.1 Interaction du développeur avec le générateur de code

de 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, grâce à la rétro-ingénierie. 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 alors que 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 chargé de nouveau, pour ensuite générer les langues, sinon elles sont corrompues par les traces de μ_C^B .

4.1.2 Architecture

La présente section décrit et explique l'architecture choisie.

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

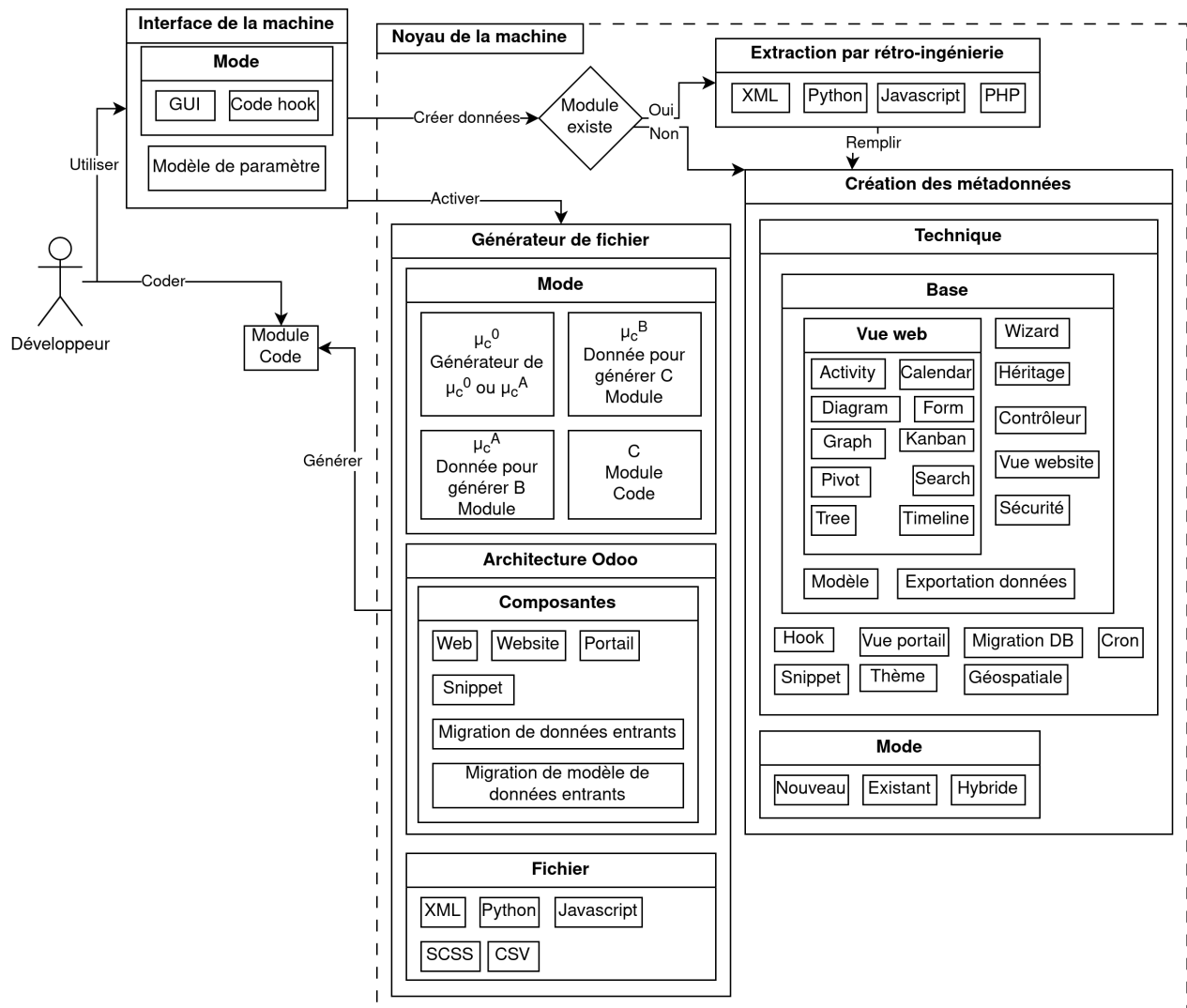


Figure 4.2 Architecture du générateur de code dans son ensemble, nommé machine. Le développeur peut modifier le code source directement ou utiliser l'interface de la machine qui passe par un ensemble de composantes.

La section de l'interface machine permet à l'utilisateur de créer un modèle de données pour indiquer à la machine quelle opération effectuer avec les données associées. Plusieurs combinaisons, héritables, sont possibles, selon les différentes techniques. De plus, c'est ici que l'on vient activer la génération de code.

La section de l'extraction par rétro-ingénierie permet de remplir le modèle de méta-données, sans passer par la paramétrisation. Ça extrait des informations qui ne sont pas accessibles dans le modèle de données d'Odoo sur le module.

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

La section du générateur de fichiers se fait activer par l'interface, mais 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 afin de permettre une génération de code.

La sécurité dépend du modèle, le contrôleur dépend du modèle et la vue dépend du contrôleur et du modèle.

4.1.3 Auto-générateur

On arrive à l'auto-générateur représenté par μ_C^0 , voir Annexe U. Au moment de l'installation de celui-ci, il génère le même code que lui-même, au même endroit dans le système de fichiers. Une légère modification va créer une autre entité qui sera une déviation dans l'objectif de démarrer une chaîne de production logicielle.

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 on obtient 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 proposée pour faire de l'amélioration continue.

4.2 Résultats propres à SO-1

Nous examinerons maintenant les résultats propres à chaque sous-objectif décrit dans le chapitre 3, méthode.

À l'intérieur de cette section, le code généré est fonctionnel et utilisable selon des techniques désirées. Cependant, ce code existe et est fonctionnel dans certaines limitations. En effet, la génération de code est peu paramétrable, elle sert de base au développeur pour qu'il personnalise ce code à ses propres contextes fonctionnels. En pratique, le développeur doit toujours faire de la réingénierie sur le code généré. L'avantage est que le développement est accéléré, puisqu'une base fonctionnelle solide est créée.

4.2.1 Génération par gabarit

La génération par gabarit était déjà supportée dans la version initiale [55], de plus, il y a eu des améliorations telles que l'utilisation des *f-strings* au lieu d'utiliser la fonction *format* de *String*, l'utilisation de la bibliothèque Code-writer en Python² et l'utilisation de la bibliothèque lxml³.

4.2.2 Génération de l'architecture MVC

L'architecture MVC était déjà supportée dans la version initiale [55], de plus, il y a eu des améliorations telles que les règles de sécurité sont ajustées selon les configurations et personnalisables par la suite et l'ajout de bouton qui ouvre des *Wizard*⁴ pour générer les vues, les modèles et les contrôleurs, voir Figure 4.3, Figure 4.4 et le tableau 4.1 sur les statistiques du code. Ainsi, le développeur peut les configurer et demander de générer les méta-données associées.

Tableau 4.1 Statistiques sur le code des «wizards» sur la GUI qui gère la génération de MVC

Langage	Fichiers	%	Lignes de code	%	Commentaire	%
Python	5	55.6	1809	58.5	560	17.3
XML	4	44.4	178	92.7	10	5.2
Total	9	100.0	2068	60.4	570	16.6

2. <https://pypi.org/project/code-writer/>

3. <https://pypi.org/project/lxml/>

4. https://github.com/ERPLibre/odoo-code-generator/tree/9d79f67/code_generator/wizards

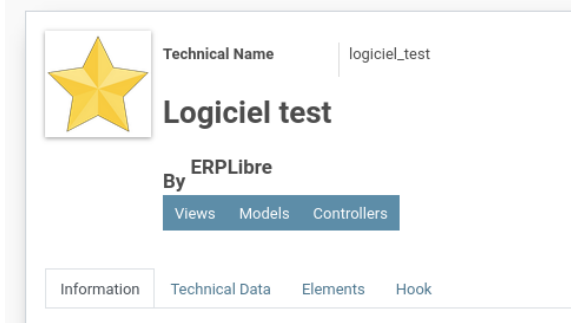


Figure 4.3 Exemple support MVC dans GUI du générateur de code

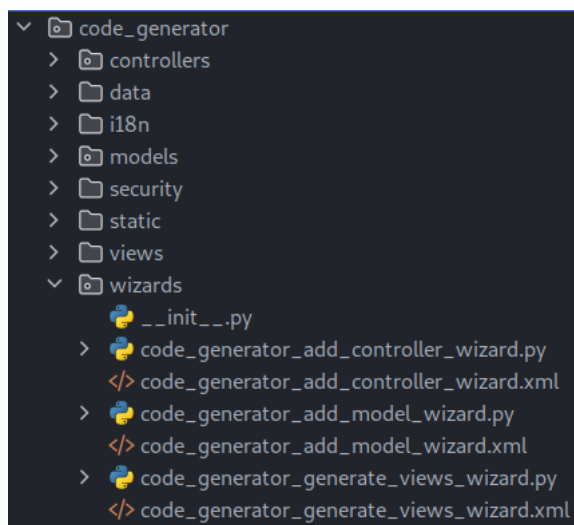


Figure 4.4 Démonstration du code pour le MVC dans la section *wizards* du module du générateur de code

4.2.3 Générer un module à partir d'une base de données externes

La migration de données à partir de SQL était déjà supportée dans la version initiale [55], cependant il y a eu des améliorations⁵, telles que l'ajout de types de données, dont ceux utilisés par le projet Accorderie, l'ajout d'associations entre les types de données et les différentes personnalisations de l'architecture Odoo, la modification de l'interface qui représente la base de données avec les contrôleurs pour permettre la configuration de la migration sur le modèle de données désiré et la gestion des interdépendances entre modèles. Pour gérer le problème d'interdépendances, il a fallu créer une séquence de création des données avec une seule dépendance et, à la toute fin, ajouter l'interdépendance en mettant à jour les données.

5. https://github.com/ERPLibre/odoo-code-generator/tree/9d79f67/code_generator_db_servers

Il y a eu au moins 1639 ajouts de lignes de code, voir Figure 4.5.

```
git diff bcf3ab938c1e133171ac5d76bb5cb8e8a869caa 928be020f50240bb299a6cf1632e9
13f0f121798 --compact-summary ./code_generator_db_servers/
code_generator_db_servers/README.rst | 2 +-
code_generator_db_servers/__manifest__.py | 31 +-
code_generator_db_servers/data/code_generator.xml | 12 +-
code_generator_db_servers/i18n/es.po | 6 +-
code_generator_db_servers/models/__init__.py | 9 +-
.../models/code_generator.py (gone) | 560 -----
.../models/code_generator_db.py (new +x) | 233 ++++
.../models/code_generator_db_column.py (new +x) | 524 +++++++
.../models/code_generator_db_table.py (new +x) | 1327 ++++++++
.../models/ir_model.py (new +x) | 13 +
code_generator_db_servers/requirements.txt (new) | 2 +
.../security/ir.model.access.csv | 3 +-
.../static/description/icon.png | Bin 9633 -> 19030 bytes
.../static/description/index.html | 57 +-
code_generator_db_servers/views/code_generator.xml | 176 ++
15 files changed, 2297 insertions(+), 658 deletions(-)
```

Figure 4.5 Démonstration des modifications effectués pour adapter le code de la migration de base de données pour supporter plus de fonctionnalités.

4.2.4 Génération de code par des données

La génération de code par des données a la capacité de prendre les données via les interfaces utilisateurs telles que la GUI et le «Code *hook*». Cela permet la personnalisation pour obtenir un logiciel adapté à ce que l'utilisateur est capable d'exprimer.

Le robot logiciel codeur est une machine qui, grâce à son interface «code *hook*», se fait commander par deux couches de méta-données paramétrables par l'humain. Les deux couches interfèrent entre elles pour permettre l'évolution de la fonctionnalité désirée. Pour voir un exemple de génération de code par données, voir Annexe U.

4.2.5 Interprétation des résultats de SO-1

SO-1 Accomplissements

Pour résumer l'ensemble des accomplissements du premier sous-objectif (SO-1), nous avons fait une simplification de l'écriture de technique dans le générateur de code avec l'utilisation de *f-string*, de la bibliothèque *Code-writer* et de la bibliothèque *lxml*, nous avons fait un ajout de *Wizard* pour configurer le MVC selon des paramètres, nous avons amélioré l'importation des bases de données externes, supporté la génération de code par des données et augmenté le nombre de techniques de génération de code, le support des *templates Qweb* et fait l'ajout du type de données géospatiales.

SO-1 Feuille de route

La brève feuille de route du premier sous-objectif se résume à implémenter les fonctionnalités manquantes dans la GUI pour générer les MVC d'un module, à augmenter le nombre de technologies à supporter sur l'importation des données, à ajouter le support de génération sur différentes architectures, à supporter différentes techniques de sécurités personnalisables, comme l'anonymisation des données, à générer automatiquement une documentation sur l'utilisation d'une technique et à supporter la génération sur d'autres systèmes *ERP* libres tel que Tryton⁶

4.3 Résultats propres à SO-2

4.3.1 Extraction de code et reproduction

De la macro et micro extraction ont été réalisées avec plusieurs techniques combinées. Pour pouvoir faire de la reproduction, il a suffi de faire de la macro extraction, c'est-à-dire faire une recherche dans toutes les classes pour copier le contenu de chaque méthode pour le transformer en méta-données et pouvoir faire l'opération directe de générer le code qui a été copié. L'utilisation de l'*AST* a servi à déterminer quelle ligne de code était à découper pour la recopier.

Cependant, il était nécessaire, dans certains contextes, de faire de la micro extraction, telles que l'extraction des noms des constantes, qui sont transformées en valeurs, lors de l'exécution, l'extraction des commentaires qui n'est pas supportée dans la bibliothèque *AST* de Python 3.7, l'extraction des décorateurs et l'extraction des paramètres sur les méthodes.

De plus, les vues ont été extraites dans le but d'obtenir des méta-données spécifiques qui caractérisent la reconstruction de la vue, ces données n'étaient pas accessibles dans les données d'Odoo.

Tout le code en lien avec l'extraction de données se retrouve dans les fichiers⁷ `extrator_controller.py`, `extrator_module.py`, `extrator_module_file.py` et `extrator_view.py`, voir le tableau 4.2 pour comprendre les statistiques sur le code.

Tableau 4.2 Statistiques sur le code de l'extraction de données de modules Odoo

Langage	Fichiers	%	Lignes de code	%	Commentaire	%
Python	4	100.0	1415	72.6	127	6.5

6. <https://www.tryton.org>

7. https://github.com/ERPLibre/odoo-code-generator/tree/9d79f67/code_generator

Certaines informations ont été extraites dans le *Javascript* à l'aide de la bibliothèque *pyjs-parser*. Pour l'extraction d'un projet externe d'une autre technologie, un extracteur de PHP a été développé via un *parser* de la communauté⁸.

C'est en développant les techniques de génération de code que nous réalisons la reproduction. Un script a été développé pour accélérer l'écriture du générateur de code, ainsi qu'un générateur de générateur de code. Ensuite, le développeur peut le transformer légèrement pour prendre les paramètres des méta-données.

4.3.2 Amélioration continue sur la génération

Grâce à l'extraction des méta-données, dès que la technique de génération est bien développée avec les méta-données, le code est automatiquement généré avec de bonnes pratiques logicielles, corrigeant automatiquement les problèmes.

L'intérêt d'utiliser Odoo pour lire le module est qu'il valide déjà certains fonctionnements. Par exemple, il n'y pas d'erreur de syntaxe dans le Python où les XML étaient bien construits.

Ainsi, pour un module désiré, nous utilisons les outils pour générer μ_C^A et μ_C^B , puis en avançant dans le développement, nous bouclons entre le 3 et 4 sur Figure 4.1.

4.3.3 Test de validation de génération de codes

Pour tester ce générateur de code, la technique du test de comparaison des sorties de la génération a été utilisée. Pour procéder, un développeur valide via l'outil Git ce qui est *commité*⁹. Ainsi, un script a été développé pour lancer en parallèle les tests et valider les différences de génération avec ce qui a été *commité* précédemment. Un succès est lorsqu'il y a aucune différence dans le code entre la version générée et la précédente. Ainsi, les tests implémentés permettent de valider l'installation du module généré, de valider que μ_C^B génère bien le module cible sans différence dans le code, de valider que μ_C^A génère μ_C^B sans différence dans le code, ainsi que valider que la migration d'une base de données SQL se fait sans différence dans le code.

En exécutant tous les tests, voir Annexe F, une couverture de 84% est obtenue, tous les tests présents sont un succès, sauf ceux sur l'auto-générateur.

Les tests deviennent une documentation sur l'utilisation du générateur de code. Puisque la génération de code se fait par données, il suffit de changer les paramètres pour générer un autre module et lancer un nouveau projet sur le module généré pour obtenir toute la chaîne.

8. <https://github.com/JameelNabbo/PHP-Parsers>

9. Un terme dans l'outil Git pour valider le code en créant un état dans l'historique.

4.3.4 Règles de codage standardisées

Au moment de générer les fichiers, toutes les sorties textes sont traitées par des outils de mise en forme, en suivant des règles de codage standardisées.

Pour le Python, l'outil *black* est utilisé pour la mise en forme en suivant le standard PEP8 avec *isort* pour réordonner les importations. *Black* donne une mise en forme non naturelle comparée à l'écriture de code pour un humain. Cependant, son résultat facilite la lecture et le suivi des différences pour les futurs ajouts. Le Javascript, le HTML et le XML sont mis en forme avec l'outil Prettier.

De plus, le générateur force le déplacement des classes dans leur fichier respectif pour créer une classe par fichier. Les champs, pour chaque modèle, sont déplacés en ordre alphabétique, mais le premier est celui qui est utilisé pour représenter le modèle¹⁰.

4.3.5 Interprétation des résultats de SO-2

SO-2 Accomplissements

Pour résumer l'ensemble des accomplissements du deuxième sous-objectif, nous avons fait de l'extraction du code via l'utilisation d'un AST et extraction des méta-données dans les fichiers XML, de l'amélioration continue sur la génération de code, grâce à la reproduction à l'aide de l'extraction du code, développé un outil pour aider à la création de technique de génération à l'aide d'un générateur de générateur de code, intégré dans le générateur de code des tests de validation en reproduisant l'ensemble des techniques en démonstration, ainsi qu'appliquer des règles de codage standardisées sur la génération de code.

SO-2 Feuille de route

La brève feuille de route du second sous-objectif se résume à finaliser l'implémentation de l'auto-génération sur le générateur de code, de mettre à jour les tests pour atteindre une couverture de code à 100% et d'ajouter des tests sur les techniques d'extractions de code tel que le PHP.

10. Référence à l'attribut «`__rec_name`»

4.4 Résultats propres à SO-3

4.4.1 Classification des techniques développées

En référence à la Figure 4.2, les techniques «Modèle», «*Form*», «*Tree*», «Contrôleur» et «Migration DB^{11} », étaient déjà implémentées dans la version initiale [55], mais elles ont reçu des améliorations pour s'agencer aux autres techniques. Les autres techniques ont été développées pour permettre une modularité aux modules à générer, puis supporter de nouvelles fonctionnalités telles que la gestion des cartes interactives avec le géospatial.

4.4.2 Interface du générateur de code

L'interface graphique

L'interface graphique existait déjà dans la version initiale [55], elle a été améliorée pour afficher plus d'informations par rapport au développement. Elle sert à faciliter la paramétrisation du générateur de code. Elle n'a pas été priorisée et elle manque de fonctionnalités si nous la comparons à ce qui peut être supporté via la technique *code hooks* avec μ_C^A et μ_C^B . L'interface permet de créer un module, de renommer un module, d'ajouter des modèles, voir Annexe B, et des champs, voir Annexe C, d'ajouter des menus, d'ajouter de la sécurité, de changer les icônes, de changer les informations sur les propriétés *manifest* du module, d'ajouter du code, voir Annexe D et de modification des *hooks*, voir Annexe E ;

L'interface *code hook*

L'interface *code hook*, voir exemple à l'Annexe U, permet d'accéder à la totalité des fonctionnalités du générateur de code via μ_C^A et μ_C^B . Elle a été utilisée pour toutes les démonstrations qui servent de tests et elle contient la paramétrisation pour les modules désirés. De plus, l'avantage de cette interface est qu'elle nous permet d'ajouter du code pour rendre dynamique la paramétrisation.

4.4.3 Interprétation des résultats de SO-3

SO-3 Accomplissements

Pour résumer l'ensemble des accomplissements du troisième sous-objectif, nous avons fait l'ajout de nouvelles techniques et une classification de celles-ci, nous avons rendu accessible

11. Module de migration de base de données

une interface graphique pour paramétrer la génération de code et avons rendu accessible une interface de programmation pour utiliser toutes les fonctionnalités du robot logiciel codeur.

SO-3 Feuille de route

La brève feuille de route du troisième sous-objectif se résume à ajouter des paramètres pour faire davantage de personnalisation sur les techniques, à extraire les techniques du générateur et les implémenter une par module, à supporter les fonctionnalités manquantes sur toutes les techniques pour l'interface graphique et à supporter l'accès à la création de méta-données par la rétro-ingénierie via l'interface graphique.

4.5 Résultats propres à SO-4

4.5.1 Utilisation d'un conteneur Docker

Puisque le générateur de code fait partie de ERPLibre, la version 1.5.0 contient les modules de génération de code. Le déploiement se fait rapidement en utilisant le logiciel Docker et le générateur de code permet l'utilisation de l'interface graphique pour générer des modules Odoo.

SO-4 Feuille de route

La brève feuille de route du quatrième sous-objectif se résume à développer une synchronisation entre les instances pour permettre la redondance, à développer une gestion de son infrastructure via le générateur de code, à faire participer le robot logiciel codeur à la maintenance de l'infrastructure de déploiement, à utiliser d'autres systèmes de conteneur en distribution qui sont libres comme «*Pod*»¹² et à développer la capacité du robot logiciel codeur de valider techniquement si le logiciel est AGPLv3 au moment de l'exécution.

4.6 Résultats propres à SO-5

Le cinquième sous-objectif met en relation le générateur de code avec la communauté. Le but du générateur de code n'est pas de remplacer les humains, mais de mieux les outiller et soutenir la communauté. De plus, le cinquième sous-objectif présente les deux cas d'étude propres à ce mémoire.

12. <https://podman.io/>

4.6.1 Guide : créer une communauté autour d'une technologie pour un réseau d'entraide libre

Avant de démarrer un projet, nous suggérons au gestionnaire de projet de suivre le guide en 7 étapes, voir Annexe G, qui permet de démarrer rapidement un projet et de s'assurer que les membres impliqués du réseau d'entraide comprennent les mêmes enjeux et s'alignent dans la même direction. Par la suite, nous proposons le guide suivant pour gérer l'intégration d'un membre en communauté, voir Annexe H. De plus, nous encourageons les bonnes habitudes en communauté avec le guide suivant pour la gestion du comportement en communauté, voir Annexe I. Ensuite, pour se préparer au développement public, nous conseillons le guide suivant, voir Annexe J. Les problèmes peuvent survenir entre les membres de la communauté, c'est pourquoi nous suggérons le guide suivant pour faciliter la résolution de problème, voir Annexe K. La communauté a besoin ensuite de formation technique et fonctionnelle, voir Annexe L. Puis la sécurité est importante pour protéger les données dans la communauté, ce pourquoi nous suggérons les étapes suivantes à intégrer au projet communautaire, voir Annexe M. Finalement, en lien avec le développement libre, nous suggérons les étapes suivantes, voir Annexe N.

4.6.2 Projet module «auto_backup»

Le module «auto_backup»¹³ est le premier module de la communauté de l'organisation OCA répertoire *server-tools* à avoir été testé dans ce projet, un μ_C^A ¹⁴ et μ_C^B ¹⁵ ont été générés. Le générateur de code a pu modifier le module pour appliquer de la qualité logicielle qui a permis de développer la technique de gestion des *cron*, en lançant des sauvegardes, par SSH ou en local, à des moments spécifiques dans le temps.

4.6.3 Projet module SRS

Le module de spécification des exigences logiciels (SRS), c'est un module de gestion de projet qui a été développé entièrement avec le générateur de code^{16 17 18}. Il permet de faire l'analyse des besoins pour ensuite passer à l'analyse fonctionnelle et, finalement, définir les requis

13. https://github.com/ERPLibre/server-tools/tree/f6054fc/auto_backup

14. https://github.com/ERPLibre/odoo-code-generator-template/tree/b5ae8e/code_generator_template_demo_sysadmin_cron

15. https://github.com/ERPLibre/odoo-code-generator-template/tree/b5ae8e/code_generator_auto_backup

16. https://github.com/ERPLibre/scrummer/tree/45ef25/code_generator_project_srs

17. https://github.com/ERPLibre/scrummer/tree/45ef25/code_generator_template_project_srs

18. https://github.com/ERPLibre/scrummer/tree/45ef25/project_srs

fonctionnels d'un projet. Il a été utilisé, entre autres, pour le projet Accorderie et le projet Portail CEPPP.

4.6.4 Projet espace Accorderie

Le projet¹⁹ a débuté par l'élaboration d'une analyse des besoins fonctionnels, puis un ensemble de requis logiciels ont été rédigés avec un membre du Réseau de l'Accorderie.

Le générateur de code a permis de créer un module Odoo 12.0 avec les modèles de données de l'Accorderie calqués sur leur base de données en SQL de MariadB, voir Annexe O.

Plusieurs corrections ont été effectuées avant la migration : correction des noms des champs pour les uniformiser ; correction des types de champs (exemple le «*True*» était exprimé par la valeur «-1» dans un type «*int*», ainsi ce type a été transformé en booléen) ; enlever les doubles dépendances par changement de l'architecture ; correction des données erronées (un champs est requis, mais il manque des données pour certaines entrées). De plus, le modèle de données n'a pas été conçu pour de l'automatisation, mais plutôt pour que les échanges de services soient validés par des membres de la communauté.

Dans l'Annexe P, on peut observer les adaptations des champs. Par exemple, avec la table «*accorderie_echange_service*», il y a l'ajout des champs : «*nb_heure_estime*» pour avoir une prévision des heures à effectuer, «*nb_heure_dure_trajet*» pour reconnaître le temps de déplacement, «*distance_trajet*» pour connaître la distance qui sera calculée avec le projet libre *Open Source Routing Machine* (OSRM).

La migration du modèle de données a été faite dans un module qui dépend de la technique *Migration DB* qui permet d'importer le modèle. Certaines données nécessitent la création d'un fichier de données XML. Pour les autres données, un autre module a été créé pour ajouter les données directement dans une base de données qui sera migrée vers une mise en production.

Un portail a été généré, pour remplacer les formulaires utilisés par l'ancienne plateforme PHP pour visualiser les entrées. Cependant, cette fonctionnalité a été abandonnée, puisque cette technologie ne plaisait pas.

Une maquette a été conçue pour un nouvel espace membre. Ainsi, nous avons utilisé la technique *website_snippet* pour afficher des données sur le site web et créer des formulaires. Cette base a permis d'accélérer la création de code de communication entre le client et le serveur. À force de faire l'intégration et la personnalisation de cette maquette, il n'y a plus vraiment de code qui provient du générateur de code.

19. https://github.com/TechnoLibre/odoo_accorderie/tree/ae5b4c

Le générateur de code a permis d'aider à créer un diagramme pour afficher le processus d'échange de temps, voir Annexe Q, puis une application en Javascript avec AngularJS a été développée pour afficher ce processus à l'utilisateur, une machine à état, qui permet de revenir, selon des paramètres, à un état du processus.

Dues à des limitations humaines et de temps, tout le reste du projet a dû être fait manuellement, puisque la technologie a été changée pour faciliter le développement de l'interface.

Les statistiques du travail sont démontrées dans le tableau 4.3.

Tableau 4.3 Statistiques sur l'ensemble des modules Odoo pour le projet Accorderie.

Langage	Fichiers	Lignes de code
XML	79	34711
Python	91	19434
Javascript	14	3755
Css	37	2488
Autre	3	57
Total	224	60445

4.6.5 Projet Portail CEPPP

Dans le second cas d'étude pour le Projet Portail CEPPP, l'objectif était de faire une section portail pour les patients et une section administrative pour les recruteurs, les partenaires et les administrateurs de la plateforme, de rendre accessible des formulaires et d'anonymiser les données. Le mandat était de migrer les fonctionnalités de la plateforme qui a été développée sur SuiteCRM en PHP.

Un module d'extraction de PHP a été développé, mais il n'est pas accessible dans les techniques du générateur de code. Le modèle de données était directement dans le code et, puisqu'il est dynamique, il n'est pas dans la base de données. La base de données n'a pas été extraite, les données ont été exportées en *Comma-separated values* (CSV) et un module d'importation des données a été développé. Au total, il y a eu 23 fichiers analysés et 2851 données extraites.

Voici les statistiques du Tableau 4.4 du code après réingénierie et adaptation des fonctionnalités à livraison de la plateforme en début septembre 2023²⁰.

L'anonymisation n'est pas supportée par le générateur de code, puis la personnalisation enlève beaucoup de champs mis de manière générique dans les fichiers XML.

20. <https://portailppp.ca>

Tableau 4.4 L'évolution entre la génération et la réingénierie des statistiques sur les langages du portail CEPPP

Langage	# Ligne extrait	# Ligne personnalisée	# Diff
XML	6 861	3 856	- 3 005
Python	567	1 564	+ 997
Javascript	0	68	+ 68
CSV	25	51	+ 26

Le modèle de données du portail CEPPP dans Odoo 12 contient 24 modèles Annexe R. L'interface administrateur Annexe S contient la fiche du patient dont les partenaires ont accès seulement qu'à la partie anonymisée Annexe T.

Le nombre de lignes de XML a diminué car le générateur de code génère, de base, toutes les vues de tous les champs. Au moment de la réingénierie, il y a eu beaucoup de nettoyage et de données XML effacées. Cependant, le développeur va mettre plus de code Python pour développer des logiques qui ne sont pas supportées par le robot logiciel codeur. Le Javascript ajouté sert à supporter les dates dans le portail. L'ajout de CSV permet l'ajout de permissions et rôles pour l'anonymisation.

4.6.6 Interprétation des résultats de SO-5

SO-5 Accomplissements

Pour résumer l'ensemble des accomplissements du cinquième sous-objectif, nous avons fait un test de la génération sur un module existant de la communauté nommé «auto_backup», des modules de gestion de projet ont été générés pour faire le suivi de la conception fonctionnelle et de l'amélioration continue, le projet Accorderie a bénéficié du générateur de code pour la migration de la base de données vers Odoo et le projet Portail CEPPP a bénéficié du générateur de code pour la migration du code PHP vers Odoo, ainsi que de l'aide au développement de la section Portail.

SO-5 Feuille de route

La brève feuille de route du cinquième sous-objectif se résume à supporter la demande de *Pull Request* sur les projets Git respectifs, lorsqu'il y a une amélioration. Il doit y avoir un suivi et valider les règles de contribution de la communauté, développer d'autres modules de gestion de projet pour l'accompagnement dans le développement de projet client, développer des modules de gestion de communauté sur des projets de logiciels libres et créer le suivi du

développement des modules communautaires avec une traçabilité sur les résultats avec des métriques de génie logiciel.

4.7 Discussion

4.7.1 Limitations

D’abord, il faut noter que le générateur de code comporte certaines limitations : il est peu paramétrable, il manque de tests sur son fonctionnement, il n’y a pas de gestion des combinaisons entre les techniques. De plus, le migrateur de bases de données ne supporte pas tous les types, certains contextes ne sont pas supportés. Il manque aussi un migrateur de données à la volée, afin de transférer les données directement dans la base de données. Pour ce faire, il faut programmer un script manuellement. Enfin, l’interface graphique a été peu testée, l’accent ayant été mis sur le *code hook*, ce dernier ne supporte pas tous les cas.

En ce qui concerne la rétro-ingénierie, le générateur ne permet pas de tout comprendre, il manque de développement, il ne gère pas tous les cas. Il est efficace seulement avec la partie nommée Web et la lecture de sécurité n’est pas supportée, ni les groupes d’utilisateurs.

Enfin, l’autopoïèse n’est pas complète, il n’est pas possible de profiter des fonctionnalités du générateur à son plein potentiel, mais que partiellement.

4.7.2 Comment les résultats obtenus soutiennent-ils le libre ?

Le réseau d’entraide a besoin d’un support technologique libre, puisque permettre aux participants de suivre leurs 4 libertés vont faciliter l’adaptation à des situations d’urgence et apporter des solutions rapidement. Les 4 libertés sont : étudier, copier, modifier et utiliser. La liberté d’étudier est représentée par la rétro-ingénierie qui a permis au générateur de code de comprendre certaines fonctionnalités pour pouvoir recréer les méta-données adéquatement pour la reproduction. La liberté de copier est représentée par l’auto-générateur qui a été mis en place, il reste à auto-reproduire le robot logiciel codeur, par son module principal de générateur de code. La liberté de modifier est représentée au moment d’une réingénierie, la rétro-ingénierie est accessible pour permettre une génération de code automatique en appliquant une mise en forme de code et une validation de la qualité logicielle. Puis la liberté d’utiliser est représentée par le robot logiciel codeur qui a la capacité d’utiliser ses fonctionnalités générées et d’exécuter des scripts d’automatisation à des périodes de temps adaptables.

4.7.3 Avancement sur le développement en réseau d'entraide

Voir Figure 4.6, un ensemble d'outils est rendu accessible aux gestionnaires de communauté, aux développeurs de projet. L'outil *No-code/Low-code* est l'interface du générateur de code pour les assister dans leur développement. La section formation sur le développement comprend les exemples de code via les tests qui sont faits pour être facilement transformables pour gérer des nouveaux contextes. La détection des anomalies est mise en place par la révision par les pairs sur l'état du code. Des guides de gestion de communauté sont accessibles pour faciliter l'ajout de participants. Enfin, la mise en place de méthodologie Agile avec suivi du développement pour l'adaptation aux changements est mise à disposition.

4.7.4 Avancement de la technopoïèse

Voir Figure 4.7, la réingénierie manuelle est le processus habituel d'un développeur. Avec ce projet, nous avons une autopoïèse fonctionnelle semi-automatique, avec intervention humaine, puis une allopoïèse complète avec intervention humaine, lorsque cette dernière est en dehors des techniques maîtrisées par le robot logiciel codeur.

Pour l'Allopoïèse, le robot logiciel codeur vient assister l'intervention humaine à H_0 et H_1 , qui permet de passer des méta-données, sur le module désiré, à une nouvelle version de ce module. Après une modification, il faut boucler avec le mode direct et le mode indirect du générateur de code pour avoir une version stable.

Pour l'Autopoïèse, la génération de code se fait directement sur le module de génération de code, c'est-à-dire que la machine est mise à jour avec l'assistance de la machine. L'humain intervient dans l'adaptation de la fonctionnalité et la correction de technique d'ingénierie au

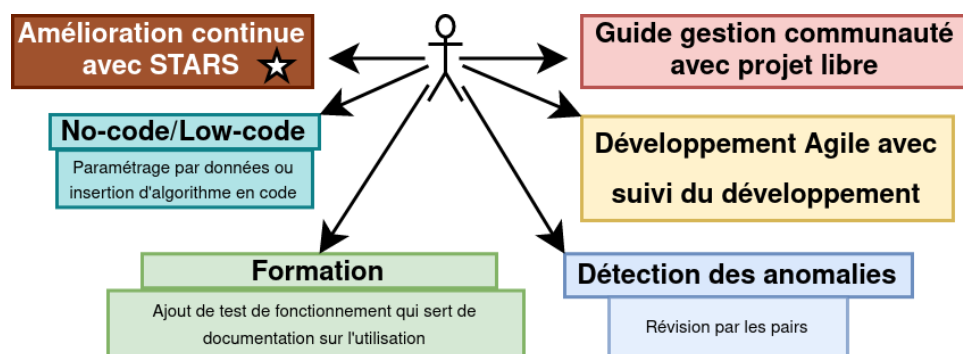


Figure 4.6 Interaction entre les développeurs et les outils de développement dans un réseau d'entraide

moment de la génération, aux endroits H_0 et H_1 . À la fin de la modification, il faut boucler avec le mode direct et le mode indirect du générateur de code pour avoir une version stable de soi-même. Puisque le générateur de code génère le générateur de code, il y a des techniques et guides d'utilisation qui sont mis à la disposition dans le projet pour éviter de s'auto-écraser²¹ et perdre son travail.

4.7.5 Réalisation du robot logiciel générateur de code

Dans la revue littérature section 2.2, nous avons déterminé 4 critères pour définir un robot logiciel codeur :

Autonomie

Nous avons des résultats qui démontrent 100% d'autonomie dans quelques contextes, voir les résultats sur les tests de validation Section 4.3.3. Cependant, le robot logiciel codeur n'est pas 100% autonome pour tous les contextes, il a besoin d'intervention humaine pour des personnalisations ou des techniques non supportées, voir Section 4.7.4.

Adaptable

Le robot logiciel codeur est adaptable, il génère 6 composantes, voir résultat d'architecture section 4.1.2 : web, *website*, portail, *snippet*, migration de données entrantes et migration de modèle de données entrantes. De plus, il est capable d'interagir avec des technologies en dehors d'Odoo comme extraire du code externe en PHP du logiciel SuiteCRM et des bases de données externes (MySQL/SQL Server/PostgreSQL) pour importer des modèles de données et migrer des données.

Compétences techniques

Le robot logiciel codeur contient plusieurs compétences techniques qui sont décrites dans les résultats section 4.4.1. De plus, il permet la personnalisation pour chacune des composantes du nombre de champs, des types de champs et du type d'affichage associé aux champs.

21. C'est un problème qui peut arriver fréquemment dans ce système, de perdre des modifications en bouclant dû à une mauvaise manipulation ou une mauvaise configuration. Il faut toujours *comiter* le travail avant de démarrer la machine.

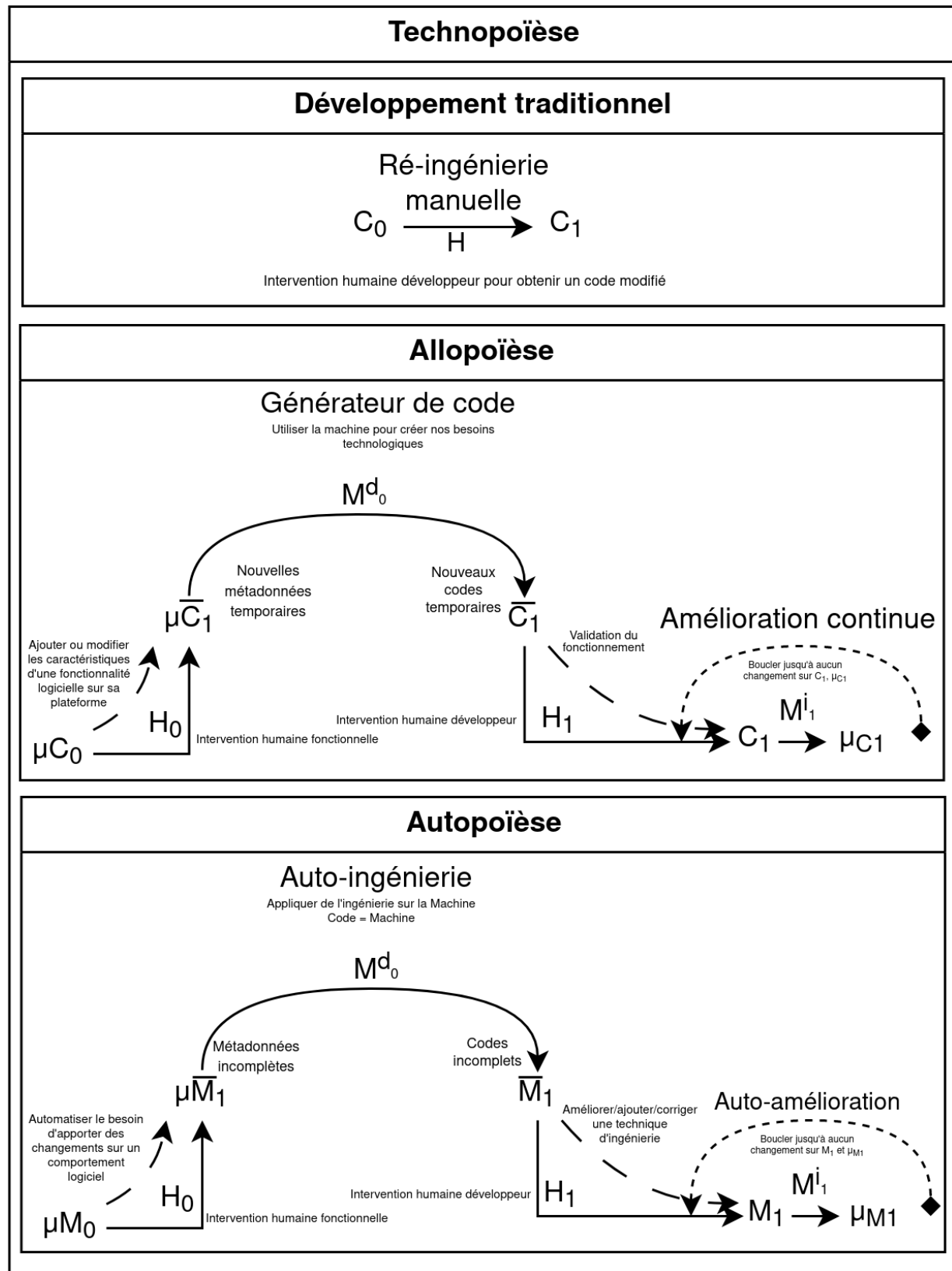


Figure 4.7 Architecture du générateur de code

Compétences sociales

Le robot logiciel codeur donne accès à des outils pour le développeur. Il y a une interface graphique LCNC qui permet la paramétrisation pour la génération de code. Il y a aussi une interface de code pour paramétrer le fonctionnement de la génération accompagnée de la rétro-ingénierie et il permet d'accompagner le développeur dans l'évolution de son module. Il contient aussi une fonctionnalité pour afficher les différences de code entre la version précédente et la version générée, il permet de créer des statistiques sur les lignes de codes et il permet de montrer la couverture de code.

CHAPITRE 5 CONCLUSION ET RECOMMANDATIONS

5.1 Synthèse des travaux

Les résultats obtenus ont permis d'atteindre en tout ou en partie l'ensemble des sous-objectifs énoncés dans le chapitre 3.

En partant de modules existants, le développement du générateur de code a été orienté pour avoir une traçabilité sur les données de génération des modules. Puis, des techniques de rétro-ingénierie ont été implémentées pour permettre l'extraction d'information et intégrer des boucles d'amélioration continue sur le développement, permettant la génération de code automatisée. Un mécanisme de test de génération de code a été implémenté pour éviter la régression sur le développement. Le développement de module dans Odoo 12 est maintenant accéléré pour plusieurs contextes.

Pour la gestion de communauté autour de projet libre, trois guides ont été rédigés pour permettre d'outiller le gestionnaire de communauté à intégrer des participants dans leur développement technologique soit un guide pour démarrer un projet, un guide pour gérer les participants et un guide pour les règles éthiques d'hébergement.

En ce qui concerne l'Accorderie, une analyse des besoins fonctionnels a été effectuée pour comprendre les développements à réaliser. La migration de la base de données d'une plateforme en PHP a été réussie en utilisant le générateur de code, mais elle est encore, à ce jour, en adaptation vers un modèle Odoo plus intégré au ERP. Les efforts ont été mis pour la création d'une interface utilisateur avec des technologies qui n'étaient pas, à la base, supportées dans Odoo.

Pour le second cas d'étude, le CEPPP, une analyse des besoins fonctionnels a été effectuée pour comprendre le développement à réaliser. Le générateur de code a été utilisé en début de projet pour faire la migration des fonctionnalités personnalisées par un ancien développeur sur la plateforme SuiteCRM. Cela a permis d'économiser du temps de développement et de réduire les erreurs possibles de retranscription du langage PHP au langage Python, ainsi que la génération des vues administration et portail. Cependant, nous avons cessé d'utiliser le générateur de code suite à la réingénierie, dès que le projet a commencé à diverger vers des fonctionnalités personnalisées qu'il ne pouvait plus supporter, telle que l'anonymisation des données. La nouvelle solution est utilisée en production.

5.2 Limitations de la solution proposée

Les limitations à la solution proposée sont causées, en général, par le manque de temps afin de terminer les développements. Ainsi, seulement la ligne critique a été suivie pour réaliser la liste des tâches nécessaires à la réponse des demandes des clients.

Par conséquent, les tests devraient couvrir 100% du code, cependant la couverture est de 84% pour 3 raisons : premièrement, il y a du code fonctionnel non testé, car il manque des tests ; ensuite, il y a du code désuet qu'il faut nettoyer ou refactoriser ; enfin, la gestion des erreurs n'est pas couverte, il faudrait les ignorer dans le test de couverture et faire des tests unitaires qui valident la gestion des erreurs.

De plus, La personnalisation de module ERP passe par le nombre de techniques supportées, mais aussi par la paramétrisation de ces techniques. L'accent du projet a été mis sur le support des techniques et sur l'amélioration continue des modules. Toutes les combinaisons n'ont pas été testées et il manque plusieurs possibilités de paramétrisation.

Bien qu'il y ait du progrès dans l'auto-reproduction du générateur de code, cet objectif n'est pas complété. Le générateur de code ne supporte pas encore toutes les techniques et combinaisons que celui-ci nécessite pour fonctionner. Les travaux doivent continuer sur cette ligne critique en améliorant le générateur de code jusqu'à ce qu'il soit apte à s'auto-reproduire.

Enfin, l'extraction de code PHP, Javascript ou de modèles de base de données a été appliqué sur des contextes particuliers tels que les projets Accorderie et CEPPP. Ainsi, il sera nécessaire de faire des ajustements pour supporter de nouveaux projets et s'assurer de mettre ces techniques d'extraction dans les tests de régression.

5.3 Améliorations futures

Ce projet pourrait avoir une incidence beaucoup plus importante en prenant en considération les apports futurs sur les 5 thématiques suivantes soit l'amélioration du générateur de code, le projet Accorderie, le projet Ceppp, le projet *Natural Language Processing* (NLP) et le projet de support de développement de module dans la communauté Odoo.

Le projet pourrait être amélioré au niveau du générateur de code. D'abord, l'amélioration peut se faire au niveau des instances clients. En effet, il faut intégrer la génération de code à l'intérieur des instances clients dans l'objectif de la rendre accessible au gestionnaire de déploiement pour y ajouter les nouvelles fonctionnalités suivantes : démarrer la mise à jour, les tests, les améliorations, la migration, l'importation. Les instances clients devraient proposer aux clients, via les interfaces, l'ajout ou le retrait de fonctionnalités, selon les retours

d'expérience et les habitudes d'utilisation.

Ensuite, il peut y avoir amélioration au niveau de l'auto-génération. En effet, une fois que le générateur de code aura atteint 100% d'auto-génération, il restera limité à ne produire que les fonctionnalités qu'il utilise. Il faut faire des tests pour les fonctionnalités qu'il n'utilise pas (ou les combinaisons non utilisées) pour se reproduire. Il reste à auto-générer toutes ses techniques dans des modules qui font de l'héritage sur le générateur de code. Par ailleurs, l'auto-génération complète va ouvrir la possibilité à une restructuration de l'architecture par le développeur, assistée du générateur, pour faciliter l'ajout de techniques d'ingénierie dans la génération de code.

Il peut aussi y avoir des améliorations au point de vue de l'architecture comme la parallélisation de tout le code, en tout temps, lorsque cela est possible; l'automatisation de la configuration pour le déverminage; l'automatisation de la détection des anomalies; l'amélioration de l'interface LCNC pour pouvoir accomplir les mêmes étapes que le mode de paramétrisation *Code hook*; l'implémentation de l'auto-ingénierie sur le développement des techniques du générateur de code et génération de code par requis fonctionnel.

De plus, il faudrait supporter de nouvelles architectures dans la génération de code comme : des applications Cordova pour le support mobile natif; des extensions Javascript dans Gnome Shell, pour étendre les fonctionnalités du ERP directement sur le bureau d'un ordinateur sans passer par un navigateur web; d'autres modules sur différents systèmes ERP tels que Tryton ou NextERP; générer des scripts de développement dans le projet ERPLibre qui ne dépendent pas d'Odoo; supporter des applications embarquées pour contrôler des automates ou des systèmes de production automatisée.

Ces travaux d'amélioration devront être effectués après l'auto-génération complète des modules Odoo.

Enfin, le générateur de code doit offrir des outils aux gestionnaires de projet pour suivre le développement, faire la liaison entre les demandes des clients et les avancements des développeurs et effectuer le suivi de la réingénierie des processus d'affaires de l'entreprise. Bref, il doit permettre de faire le suivi des étapes des facteurs décisionnels pour l'implantation d'un *ERP*.

De plus, puisque l'état des méta-données évolue, il devient difficile de faire le suivi des performances du générateur de code, puisqu'il vient aider dans les boucles d'itérations. Ainsi, il faudrait dégager des statistiques sur ces itérations pour évaluer la contribution du générateur versus l'implication du développeur.

Autre que la modification de l'architecture, le générateur de code doit générer des tests

fonctionnels sur les modules Odoo, générer de la documentation fonctionnelle et technique et développer de la migration de module et de données.

Une fois l'auto-reproduction complétée, la prochaine étape est de tester la mise à niveau de tous les modules dans la communauté et détecter les techniques manquantes, par supervision du développeur, pour les implémenter. Une fois que tous les modules de la communauté seront gérés, nous pourrons implémenter la migration vers des mises à jour de la plateforme, c'est-à-dire passer d'odoo 12 vers Odoo 14, puis vers Odoo 16. La migration va nécessiter l'adaptation des techniques pour les différentes versions de la plateforme.

Finalement, en ce qui concerne les projets d'étude, une plateforme de l'espace membre est en finalisation de développement, il faudrait poursuivre la mise à jour du générateur de code pour supporter cette nouvelle technologie pour des projets futurs similaires.

De plus, le générateur de code doit être en mesure de produire les personnalisations nécessaires au projet Portail CEPPP, telles que l'anonymisation des données. Lorsque ces fonctionnalités seront supportées, la prochaine étape sera d'amener le client à utiliser, par lui-même, le générateur de code pour la maintenance future de la plateforme. Ainsi, nous pourrons évaluer la facilité d'utilisation pour des utilisateurs non développeurs.

Enfin, il serait intéressant de se pencher sur la technologie NLP qui va permettre de comprendre des textes rédigés par l'utilisateur et les associer à des techniques de programmation. Le NLP est une solution alternative pour interfacer avec l'utilisateur et communiquer avec ce dernier, pour développer des logiciels. De plus, cela va améliorer la rédaction de documentation adaptée aux contextes de l'utilisateur.

Il existe la communauté Hugging Face qui base ses technologies d'apprentissage automatique sur des solutions ouvertes. Il existe des solutions NLP accessibles et compatibles avec le générateur de code.

5.4 Importance de la recherche

En conclusion, le robot logiciel libre codeur est en première phase de développement, incluant la génération de code, l'interface avec l'utilisateur et la rétro-ingénierie pour appliquer de l'amélioration continue orientée au support d'un réseau d'entraide.

L'automatisation du développement de logiciel va permettre l'accélération de création de fonctionnalités et la réduction des coûts de développement.

Le robot va permettre aux chercheurs d'être plus efficaces dans leurs travaux, en facilitant le développement de leurs propres outils, pouvant mieux tracer, s'interfacer et avoir le contrôle

de leurs données.

Les communautés auront accès à des fonctionnalités plus aisément, permettant ainsi l'adaptation à des états d'urgence, grâce, par exemple, au partage de banques de temps de service en temps de crise. Avec les crises mondiales, les états d'urgence seront de plus en plus présents, l'accélération du développement technologique aura un impact sur l'appropriation de la technologie des communautés pour leur permettre de mettre en place des solutions rapidement.

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] N. Ahituv, S. Neumann et M. Zviran, “A system development methodology for erp systems,” *Journal of Computer Information Systems*, vol. 42, n°. 3, p. 56–67, mars 2002.
- [5] J. Wu et Y. Wang, “Measuring erp success : the ultimate users’ view,” *International Journal of Operations & Production Management*, vol. 26, n°. 8, p. 882–903, Jan 2006. [En ligne]. Disponible : <https://doi.org/10.1108/01443570610678657>
- [6] A. Djouahra et T. Hallalel, “Développement d’une solution erp pour la gestion hôtelière sous la plateforme odoo,” Theses, Université Mouloud Mammeri, 22 mars 2021. [En ligne]. Disponible : <https://www.ummo.dz/dspace/handle/ummo/13078>
- [7] 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>
- [8] 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>
- [9] D. Margulius. (2019, 27 décembre) Projet portail ceppp - suitecrm 7.10.9. [En ligne]. Disponible : https://github.com/lerenardprudent/ceppp_crm
- [10] A. Lehtola *et al.*, *Current platform support for internationalization*. United States : Wiley, 1997, p. 229–287.
- [11] Wikipédia. (2023, 17 février) Internationalisation et localisation - i18n. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Internationalisation_et_localisation
- [12] O. québécois de la langue française. (2018) robot logiciel. [En ligne]. Disponible : <https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/8350665/robot-logiciel>

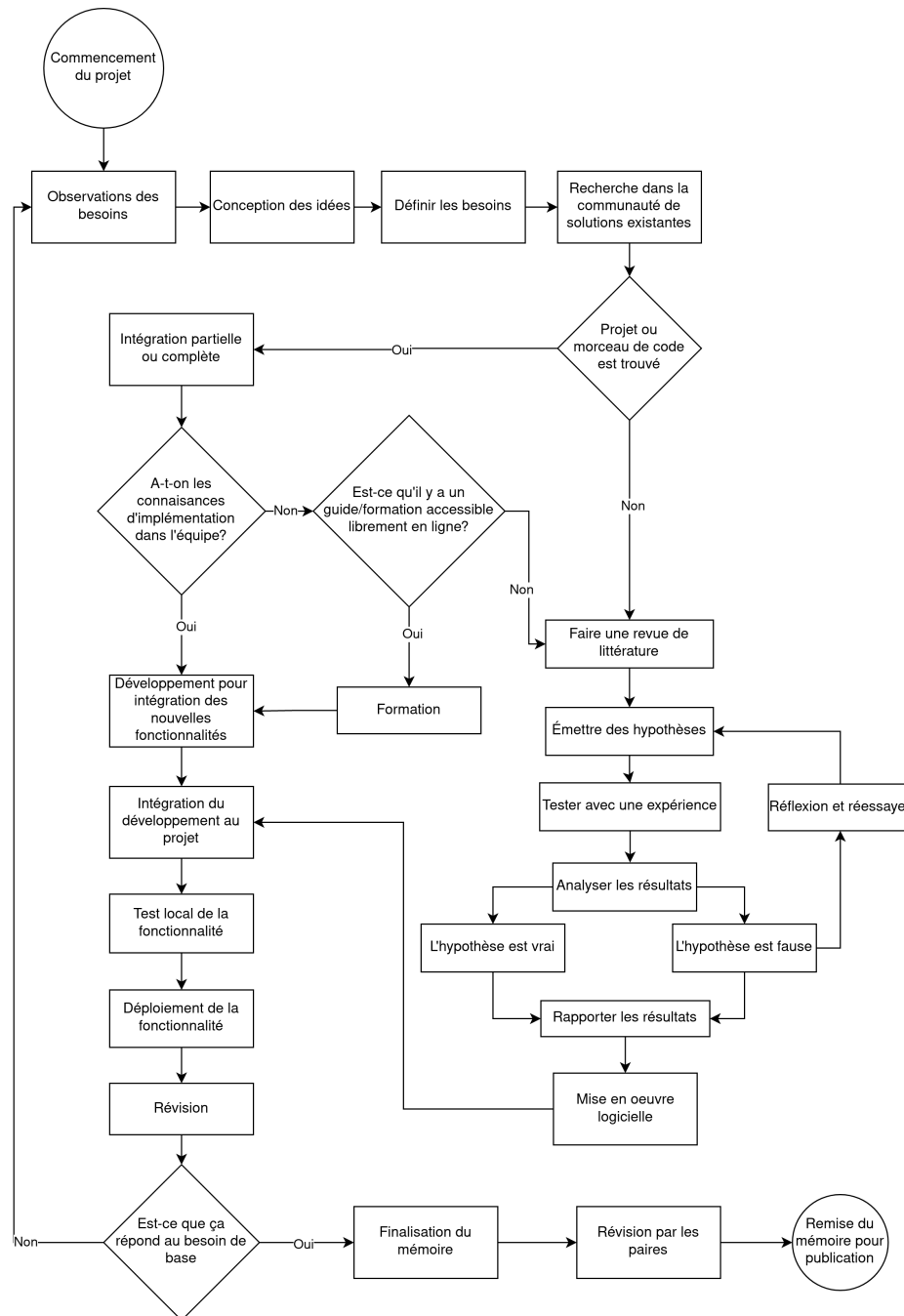
- [13] L. Erlenhov *et al.*, “Current and future bots in software development,” dans *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, 2019, p. 7–11. [En ligne]. Disponible : <https://ieeexplore.ieee.org/document/8823643>
- [14] T. Leelanupab et T. Meephruet, “Codebuddy (collaborative software development environment) : In- and out-class practice for remote pair-programming with monitoring coding students’ progress,” dans *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’19. New York, NY, USA : Association for Computing Machinery, 2019, p. 1290. [En ligne]. Disponible : <https://doi.org/10.1145/3287324.3293750>
- [15] T. Wiedemann, “Open source initiatives for simulation software : Next generation simulation environments founded on open source software and xml-based standard interfaces,” dans *Proceedings of the 34th Conference on Winter Simulation : Exploring New Frontiers*, ser. WSC ’02. Winter Simulation Conference, 2002, p. 623–628.
- [16] B. UYANIK et V. H. . ŞAHİN, “A template-based code generator for web applications,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, n°. 3, p. 1747–1762, article 37, 1 janvier 2020. [En ligne]. Disponible : <https://journals.tubitak.gov.tr/elektrik/vol28/iss3/37>
- [17] S. Pichidtienthum, P. Pugsee et N. Cooharajanane, “Developing module generation for odoo using concept of low-code development platform and automation systems,” dans *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, 2021, p. 529–533. [En ligne]. Disponible : <https://ieeexplore.ieee.org/document/9436754>
- [18] Z. EL IDRISSI, “Reverse engineering variability for configurable systems using formal concept analysis : The odoo case study,” mémoire de maîtrise, University of Namur, 21 june 2022. [En ligne]. Disponible : <https://researchportal.unamur.be/en/studentTheses/1066fd86-fb32-4bfe-8f6e-e0400278ea68>
- [19] M.-A. Laverdière et E. Merlo, “Computing counter-examples for privilege protection losses using security models,” dans *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017, p. 240–249.
- [20] M. V. Emmerik. (2021, 18 juin) Diagramme représentant la forme la plus traditionnelle de rétroingénierie, l’abstraction d’architecture, telle que présentée par e. j. byrne (a conceptual foundation for software re-engineering, icsm 1992, pp. 226-235). [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique#/media/Fichier:Retroingenierie_-_Byrne.svg

- [21] F. Almeida, J. Oliveira et J. Cruz, “Open standards and open source : Enabling interoperability,” *International Journal of Software Engineering & Applications (IJSEA)*, vol. 2, n°. 1, 1 janvier 2011. [En ligne]. Disponible : <https://airccse.org/journal/ijsea/papers/0111ijsea01.pdf>
- [22] G. Hertel, S. Niedner et S. Herrmann, “Motivation of software developers in open source projects : an internet-based survey of contributors to the linux kernel,” *Research Policy*, vol. 32, n°. 7, p. 1159–1177, 2003, open Source Software Development. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S0048733303000477>
- [23] K. Thompson, “Reflections on trusting trust.” *Communications of the ACM*, vol. 27(8), p. 761–763, 1984. [En ligne]. Disponible : https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
- [24] M. Shah. (2020, 1 juillet) A discussion of ken thompson’s "reflections on trusting trust". [En ligne]. Disponible : <https://mananshah99.github.io/blog/2020/07/01/trusting-trust/>
- [25] R.-H. Pfeiffer, “License incompatibilities in software ecosystems,” 2022. [En ligne]. Disponible : <https://arxiv.org/abs/2203.01634>
- [26] M. Feng *et al.*, “Open-source license violations of binary software at large scale,” dans *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, p. 564–568. [En ligne]. Disponible : <https://ieeexplore.ieee.org/document/8667977>
- [27] R. Duan *et al.*, “Identifying open-source license violation and 1-day security risk at large scale,” dans *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA : Association for Computing Machinery, 2017, p. 2169–2185. [En ligne]. Disponible : <https://doi.org/10.1145/3133956.3134048>
- [28] I. Yona. (2023, 16 janvier) Reflections on trusting trust & ai. [En ligne]. Disponible : <https://www.lesswrong.com/posts/BMnhDjJrix5BXE7yr/reflections-on-trusting-trust-and-ai>
- [29] F. Pitetti, “L’implémentation d’un erp libre implique-t-elle nécessairement une réduction des coûts?” *Haute École de Gestion de Genève (HEG-GE) Filière Informatique de Gestion*, 2010. [En ligne]. Disponible : <https://folia.unifr.ch/global/documents/314234>
- [30] O. BRAUD, “Facteurs decisionnels pour l’implantation d’un erp dans les pme : Le role de l’évaluation des benefices tangibles et intangibles,” mémoire de maîtrise, Université du Québec à Montréal, Montréal, QC, avril 2008. [En ligne]. Disponible : <http://archipel.uqam.ca/id/eprint/1229>

- [31] M. Kenza et Y. M. Idir, “Conception et réalisation d’un module erp pour le suivi des patients sur le plan médical et financier au niveau de la clinique el djouher,” Thèse de doctorat, Université Mouloud Mammeri, 2018.
- [32] C. Ebert *et al.*, “Devops,” *Ieee Software*, vol. 33, n°. 3, p. 94–100, 2016. [En ligne]. Disponible : <https://ieeexplore.ieee.org/abstract/document/7458761>
- [33] A. Bock et U. Frank, “Low-code platform,” *Bus Inf Syst Eng*, vol. 63, p. 733–740, 15 novembre 2021. [En ligne]. Disponible : <https://doi.org/10.1007/s12599-021-00726-8>
- [34] A. Sahay *et al.*, “Supporting the understanding and comparison of low-code development platforms,” dans *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, p. 171–178. [En ligne]. Disponible : <https://ieeexplore.ieee.org/abstract/document/9226356>
- [35] G. du Québec. (2013) Logiciels libres et ouverts, guide de référence. [En ligne]. Disponible : https://www.tresor.gouv.qc.ca/fileadmin/PDF/ressources_informationnelles/logiciels_libres/ll.pdf
- [36] M. Meyer et F. Montagne, “Le logiciel libre et la communauté autorégulée,” *Revue d’économie politique*, vol. 117, n°. 3, p. 387–405, 2007. [En ligne]. Disponible : <https://doi.org/10.3917/redp.173.0387>
- [37] (2023, 24 mars) Building community. [En ligne]. Disponible : <https://opensource.guide/fr/building-community/>
- [38] R. Duan *et al.*, “Identifying open-source license violation and 1-day security risk at large scale,” dans *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17. New York, NY, USA : Association for Computing Machinery, 2017, p. 2169–2185. [En ligne]. Disponible : <https://doi.org/10.1145/3133956.3134048>
- [39] I. Free Software Foundation. (2022, 6 mai) Critères éthiques de gnu concernant l’hébergement de logiciel. [En ligne]. Disponible : <https://www.gnu.org/software/repo-criteria.fr.html>
- [40] Wiktionary. (2021, 1 août) Poïèse. [En ligne]. Disponible : <https://fr.wiktionary.org/wiki/-po%C3%AF%C3%A8se>
- [41] Wikipédia. (2022, 28 novembre) Poïétique. [En ligne]. Disponible : <https://fr.wikipedia.org/wiki/Po%C3%AF%C3%A9tique>
- [42] P. Krajewski, “Qu’est-ce qu’un artiste technologique ?” Conférence Conférence Beaux-Arts Lisbonne, mai 2012. [En ligne]. Disponible : https://pkaccueil.files.wordpress.com/2012/06/conf_bxarts.pdf

- [43] Wikipedia. (2018, 28 décembre) Allopoiesis. [En ligne]. Disponible : <https://en.wikipedia.org/wiki/Allopoiesis>
- [44] P. C. WEB. (2007) Allopoiesis. [En ligne]. Disponible : <http://pespmc1.vub.ac.be/ASC/ALLOPOIESIS.html>
- [45] Wikipedia. (2022, 30 octobre) Autopoïèse. [En ligne]. Disponible : <https://fr.wikipedia.org/wiki/Autopoiese>
- [46] T. Nomura, “A computational aspect of autopoiesis,” *System Technology Development Center - Sharp Corporation*, janvier 2000. [En ligne]. Disponible : https://www.researchgate.net/publication/228784157_A_Computational_Aspect_of_Autopoiesis
- [47] P. Guilibert, “Terre et capital : penser la destruction de la nature à l’âge de catastrophes globales,” Theses, Université de Nanterre - Paris X, oct. 2019. [En ligne]. Disponible : <https://theses.hal.science/tel-02929676>
- [48] L. Semal et M. Szuba, “Villes en transition : imaginer des relocalisations en urgence,” *Mouvements*, vol. 63, n°. 3, p. 130–136, 2010. [En ligne]. Disponible : <https://www.cairn.info/revue-mouvements-2010-3-page-130.htm>
- [49] P. Cibois, “Compte-rendu de : David mandin, les systèmes d’échanges locaux (sel). circulations affectives et économie monétaire. paris, l’harmattan, 2009, coll." logiques sociales",” *Socio-logos. Revue de l’association française de sociologie*, n°. 5, 2010. [En ligne]. Disponible : <https://doi.org/10.4000/socio-logos.2449>
- [50] A. Sarkar, “Quines are the fittest programs : Nesting algorithmic probability converges to constructors,” 2020. [En ligne]. Disponible : <https://arxiv.org/abs/2010.09646>
- [51] Wikipédia. (2022, 4 septembre) Quine (informatique). [En ligne]. Disponible : [https://fr.wikipedia.org/wiki/Quine_\(informatique\)](https://fr.wikipedia.org/wiki/Quine_(informatique))
- [52] Wikipedia. (2022, 10 septembre) Rétro-ingénierie en informatique. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique
- [53] 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/>
- [54] Wikipédia. (2023, 13 mars) Liste de tests informatiques. [En ligne]. Disponible : https://en.wikipedia.org/wiki/Software_testing
- [55] 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

ANNEXE A MÉTHODOLOGIE REVUE DE LITTÉRATURE ET DÉVELOPPEMENT AGILE



ANNEXE C GUI GÉNÉRATEUR DE CODE - LES CHAMPS

Code Generator

Modules Advance Settings

Administrator (code_generator)

Ouvrir : O2M Models

Description du Modèle

Modèle

Modèle transitoire

Fil de discussion

Code Generator Writer

code_generator.writer

☐

☐

Type

Dans Applications

Objet de base

code_generator, code_generator_hook

Code Generator Module

Rec Name

Python Class

Inherit ir Model

Dependency

Ir model

Name

Server Constrains

Code generator

Constrained Model Code

Nomenclator?

☐

Champs

Droits d'accès

Règles sur les enregistrements

Notes

Vues

Nom de Champ	Étiquette de Champ	Type de Champ	Requis	Lecture seule	Indexé	Type
_last_update	Last Modified on	date/heure	☐	☒	☐	Champ de base
basename	Base name	caractère	☐	☐	☐	Champ de base
code_generator_ids	Code Generator	many2many	☐	☐	☐	Champ de base
create_date	Created on	date/heure	☐	☒	☐	Champ de base
create_uid	Created by	many2one	☐	☒	☐	Champ de base
display_name	Display Name	caractère	☐	☒	☐	Champ de base
id	ID	integer	☐	☒	☐	Champ de base
list_path_file	List path file	caractère	☐	☐	☐	Champ de base
rootdir	Root dir	caractère	☐	☐	☐	Champ de base
write_date	Last Updated on	date/heure	☐	☒	☐	Champ de base
write_uid	Last Updated by	many2one	☐	☒	☐	Champ de base

Créer un menu

Fermer

ANNEXE D GUI GÉNÉRATEUR DE CODE - LES CODES

Code Generator

Modules Advance Settings

Recherche...

Filtres

Regrouper par

Favoris

1-80 / 147

+ Créer

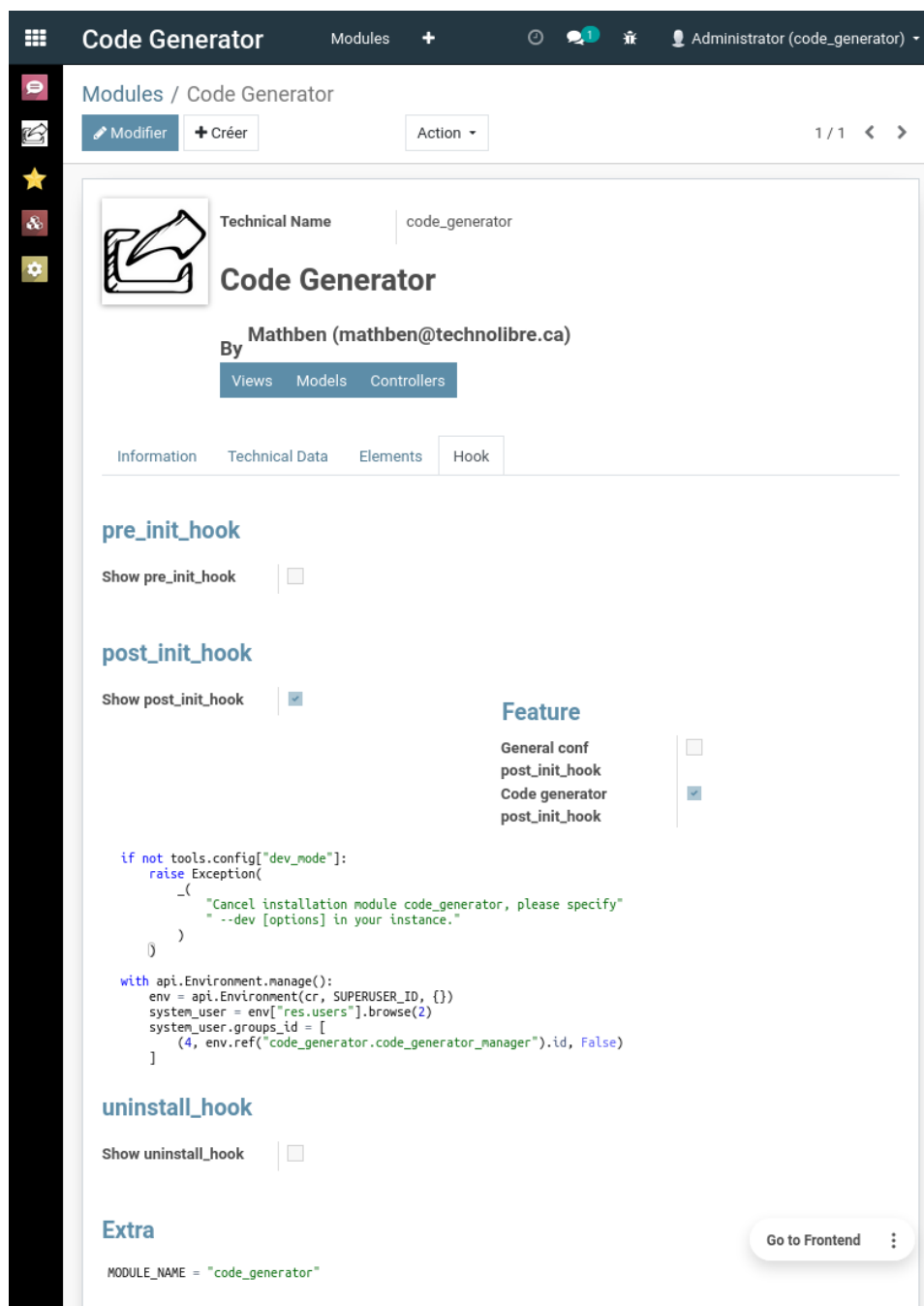
Importer

	Code of pre_init_hook	Decorator	Templated	Work in progress	Model	Module	Method name	Param
☐	<pre>field_ids = [field_id_id for model_id_id in self.model_ids for field_id_id in model_id_id.field_id_id] self.field_ids = [(6, 0, field_ids)]</pre>	@api.onchange("model_ids")	☐	☐	Code Generator Add Controller Wizard	Code Generator	_onchange_model_ids	self
☐	pass	@api.multi	☐	☐	Code Generator Add Controller Wizard	Code Generator	button_generate_add_controller	self
☐	<pre>field_ids = [field_id_id for model_id_id in self.model_ids for field_id_id in model_id_id.field_id_id] self.field_ids = [(6, 0, field_ids)]</pre>	@api.onchange("model_ids")	☐	☐	Code Generator Model Wizard	Code Generator	_onchange_model_ids	self
☐	<pre>if self.clear_fields_blacklist: field_ids = self.env["code.generator.ir.model.fields"].search([("m2o_module", "=", self.code_generator_id_id)]) field_ids.unlink()</pre>	@api.multi	☐	☐	Code Generator Model Wizard	Code Generator	button_generate_add_model	self

Go to Frontend

is_nomenclator = self.option_adding == "nomenclator"

ANNEXE E GUI GÉNÉRATEUR DE CODE - LES «HOOKS»



ANNEXE F TEST COUVERTURE TECHNIQUE GÉNÉRATEUR DE CODE

Technique	base		# instruction	5 085
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B modèle simple	Succès	27	2 983	41
Génération μ_C^B modèle simple avec héritage	Succès	26	3 452	32
Exportation μ_C^B données «helpdesk»	Succès	28	3 900	23
Technique	base + hook		# instruction	5 985
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Auto-génération μ_C^0	Succès	17	4 683	22
Nouveau projet μ_C^0 μ_C^A μ_C^B «Hello World»	Succès	39	4 610	23
Génération μ_C^A portail	Succès	44	3 638	39
Génération μ_C^A modèle simple	Succès	33	3 982	33
Génération μ_C^A modèle simple avec héritage	Succès	32	4 027	33
Exportation μ_C^B données «website»	Succès	28	3 900	23
Génération μ_C^B du générateur de code	Échec	20	3 529	41
Génération μ_C^A du générateur de code	Échec	29	3 690	38
Technique	base + cron		# instruction	6 153
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^A «auto_backup»	Succès	36	3 422	44
Technique	base + portal		# instruction	5 799
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B exemple MariaDB SQL	Succès	80	3 104	46
Technique	base + «theme_website»		# instruction	5 336
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B thème «website»	Succès	27	3 938	26
Technique	base + «website_snippet»		# instruction	5 615
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B individuel «website_snippet»	Succès	26	4 284	24

Technique	base + portal + «website_snippet»	# instruction		6 329
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B multiple «website_snippet»	Succès	36	2 898	54
Génération μ_C^B portail	Succès	34	3 173	50
Technique	base + portal + «Migration DB»	# instruction		6 559
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération migration MariaDB SQL	Succès	121	3 267	50
Technique	base + hook + portal	# instruction		6 699
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^A MariaDB SQL	Succès	78	4 315	36
Technique	base + hook + cron	# instruction		6 153
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^A «auto_backup»	Succès	36	3 422	44
Technique	base + geoengine + «website_leaflet»	# instruction		5 423
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B «website_snippet_leaflet»	Succès	33	3 259	40
Technique	base + cron + hook + «Migration DB» + geoengine + portal + «theme_website» + «website_snippet» + «website_leaflet»	# instruction		8 746
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Tous les tests à succès	Succès	194	1 371	84

ANNEXE G LES 7 ÉTAPES POUR DÉMARRER UN PROJET DANS UN RÉSEAU D'ENTRAIDE

Étape	Description
Mission	Trouver votre mission, vos indicateurs et les objectifs associés.
Processus	Déterminer les étapes pour du développement informatique, de l'assemblage des travaux, des méthodes pour faire des services et de l'amélioration continue. Avoir conscience des gaspillages
Connexion	Mécanisme d'animation du suivi des tâches, des services.
Valeurs	Trouver les valeurs qui vont guider la façon de gérer l'équipe, la communauté, sans être exclusifs ou figés dans le temps. Ils sont un point de repère et aide pour prendre les grandes orientations.
Vision	Détailler un plan stratégique pour la communauté. C'est une projection dans le futur pour permettre de comprendre la direction sur la longue durée.
Prochaines étapes	Passer à l'action en mode itératif avec des méthodologies agiles.

ANNEXE H GUIDE D'INTÉGRATION DES MEMBRES DANS LA COMMUNAUTÉ

1. Amener les utilisateurs à faire des contributions en participant à la maintenance et en facilitant chaque étape d'implication ;
2. Rendre disponible des tâches pour les nouveaux membres ;
 - (a) Permettre d'utiliser des étiquettes de classement adaptées sur des initiatives proposées par des nouveaux, tels que «suggestion», «problème» ou «question».
3. Remercier la personne pour son intérêt qui veut participer au projet ;
4. Répondre en moins de 24 heures pour accueillir le membre ;
5. Définir les types de contributions nécessaires et la manière qu'on examine une contribution ;
6. Mettre en place un sentiment d'appartenance :
 - (a) Lorsqu'un problème est reporté, demander gentiment s'il peut avoir une contribution ;
 - (b) Mettre la liste des contributeurs dans un fichier du projet ;
 - (c) Remercier les contributeurs dans une infolettre.
7. Émettre des dates de rencontres officielles pour parler du projet par vidéo-conférence pour des communautés locales, qui ont la même langue.

ANNEXE I GUIDE D'AIDE À LA GESTION DU COMPORTEMENT EN COMMUNAUTÉ

1. Proposer un guide sur les comportements désirés ;
2. Réagir publiquement pour chaque message sur la plateforme ;
3. Encourager de publier les notes de réunions pour promouvoir la transparence ;
4. Développer une culture de développement ouvert.

ANNEXE J GUIDE DE DÉVELOPPEMENT PUBLIC

1. Mettre en place un site web de développement en lien avec l'organisation créée ;
2. Documenter publiquement le processus de développement ;
3. Permettre de voir l'avancement des tâches en lien avec les processus ;
 - (a) Permettre de proposer des changements avec un système d'acceptation par les pairs.
4. Montrer la feuille de route du projet, les livrables prévue ;
5. Encourager la publication du travail brouillon avec un état de travail en progression ;
6. Déployer un moyen de discussion public et éviter de répondre en privé.

ANNEXE K GUIDE DE RÉOLUTION DE PROBLÈME

1. Mettre en place un arbre décisionnel avec description des décisions sur un type de problème ;
2. Documenter la résolution d'un problème de développement logiciel ;
3. Permettre aux développeurs de prendre des décisions sur des choix impopulaires basés sur leur ressenti ;
4. Éviter les débats réguliers sur des aspects triviaux ;
5. Concentrer les discussions vers la résolution d'un problème qui mène vers une action.
 - (a) Quel serait la prochaine étape à prendre ?
 - (b) Suggérer des conditions pour de nouveaux progrès, offrir un itinéraire, un chemin à suivre pour obtenir les résultats désirés.

ANNEXE L GUIDE SUR LA DOCUMENTATION DE PROJET

1. Documentation fonctionnelle pour les utilisateurs ;
2. Documentation technique pour le développement ;
3. Documentation de l'hébergement pour le déploiement ;
4. Fichier «*README*» pour l'utilisation rapide du logiciel ;
5. Fichier «*CONTRIBUTE*» pour comment faire de la contribution ;
6. Fichier «*GOVERNANCE*» pour le départage décisionnel.

ANNEXE M GUIDE SUR LES MOYENS DE SÉCURITÉ À METTRE EN PLACE

1. Montrer le niveau de sécurité de l'application et de ses dépendances (faire un suivi des CVE) ;
2. Mettre en place un système de communication des mises à jour nécessaires ;
3. Informer comment sécuriser les clés d'authentification, les données et les configurations personnelles.

ANNEXE N GUIDE SUR LES BONNES PRATIQUES DU LOGICIEL LIBRE

1. Rendre accessible le code source d'une manière facile à accéder, tel qu'un lien visible à partir du site web vitrine du projet ;
2. Suivre les règles d'hébergement de logiciel libre pour permettre l'inclusion ;
3. Expliquer l'importance du choix de la licence libre, ainsi que les différences. Expliquer pourquoi d'autres choix ne sont pas proposés et restreindre l'utilisation du logiciel libre si la licence n'est pas acceptée ;
4. Rendre accessible la documentation sur le logiciel libre en lien avec la localité administrative de l'organisation ¹ ;
5. Toujours proposer des licences libres avec un guide explicatif, ne pas permettre d'utiliser des licences qui ne seraient pas compatibles aux licences libres :
 - (a) LGPL3 et + - elle est acceptable, mais non désirée ;
 - (b) GPL3 et + - pour toutes machines sans communication ;
 - (c) AGPL3 et + ² ;
 - (d) CC0 ³, ne protège pas l'oeuvre ;
 - (e) CC-BY ⁴ ;
 - (f) CC-BY-SA ⁵, protège l'oeuvre ⁶ ;
 - (g) LiLiQ-R+ ⁷ ;

1. Exemple, un document du Québec : https://www.tresor.gouv.qc.ca/fileadmin/PDF/ressources_informationnelles/logiciels_libres/11.pdf

2. <https://www.gnu.org/licenses/agpl-3.0.en.html>

3. <https://creativecommons.org/share-your-work/public-domain/cc0/>

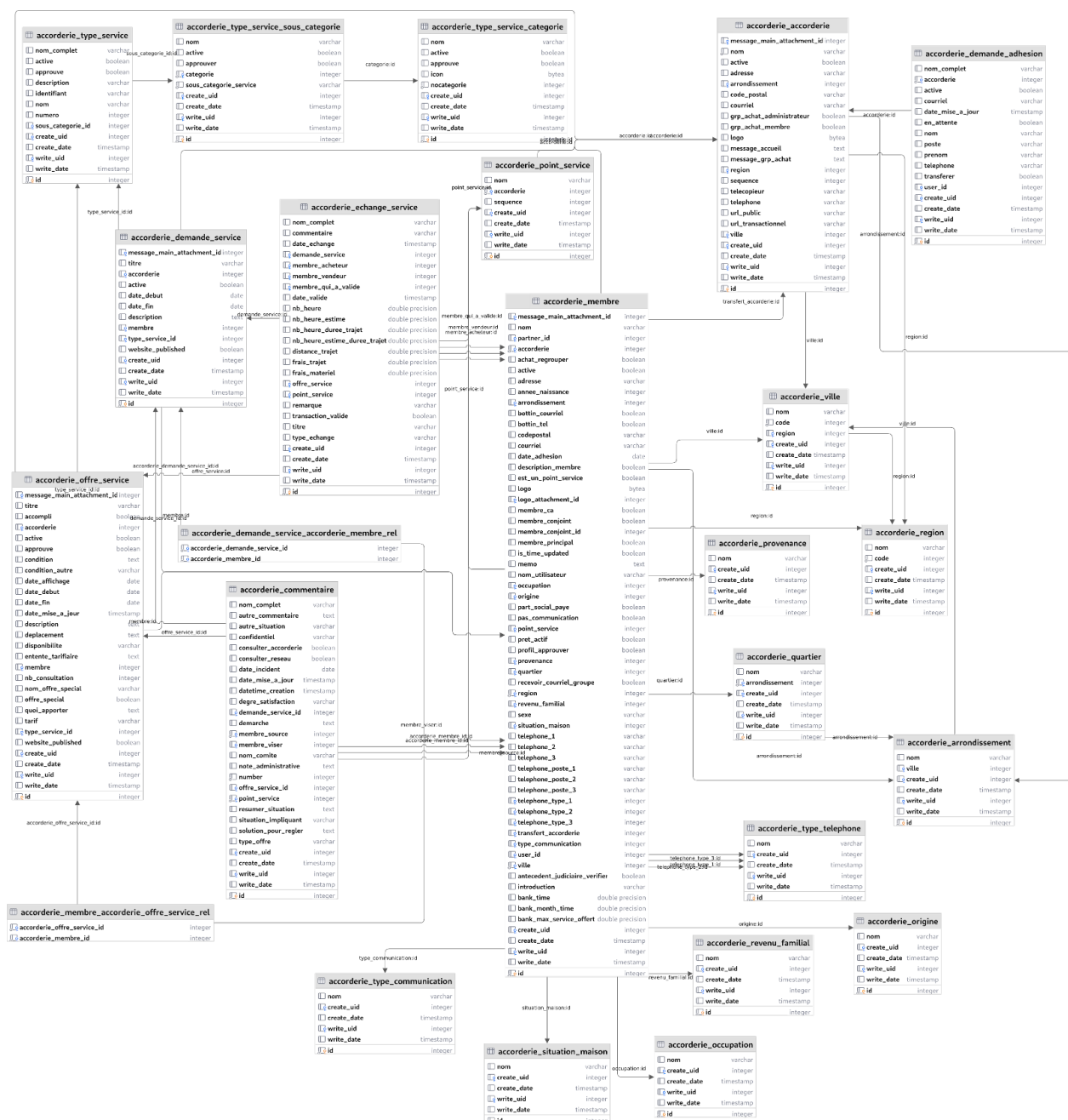
4. <https://creativecommons.org/licenses/by/4.0/>

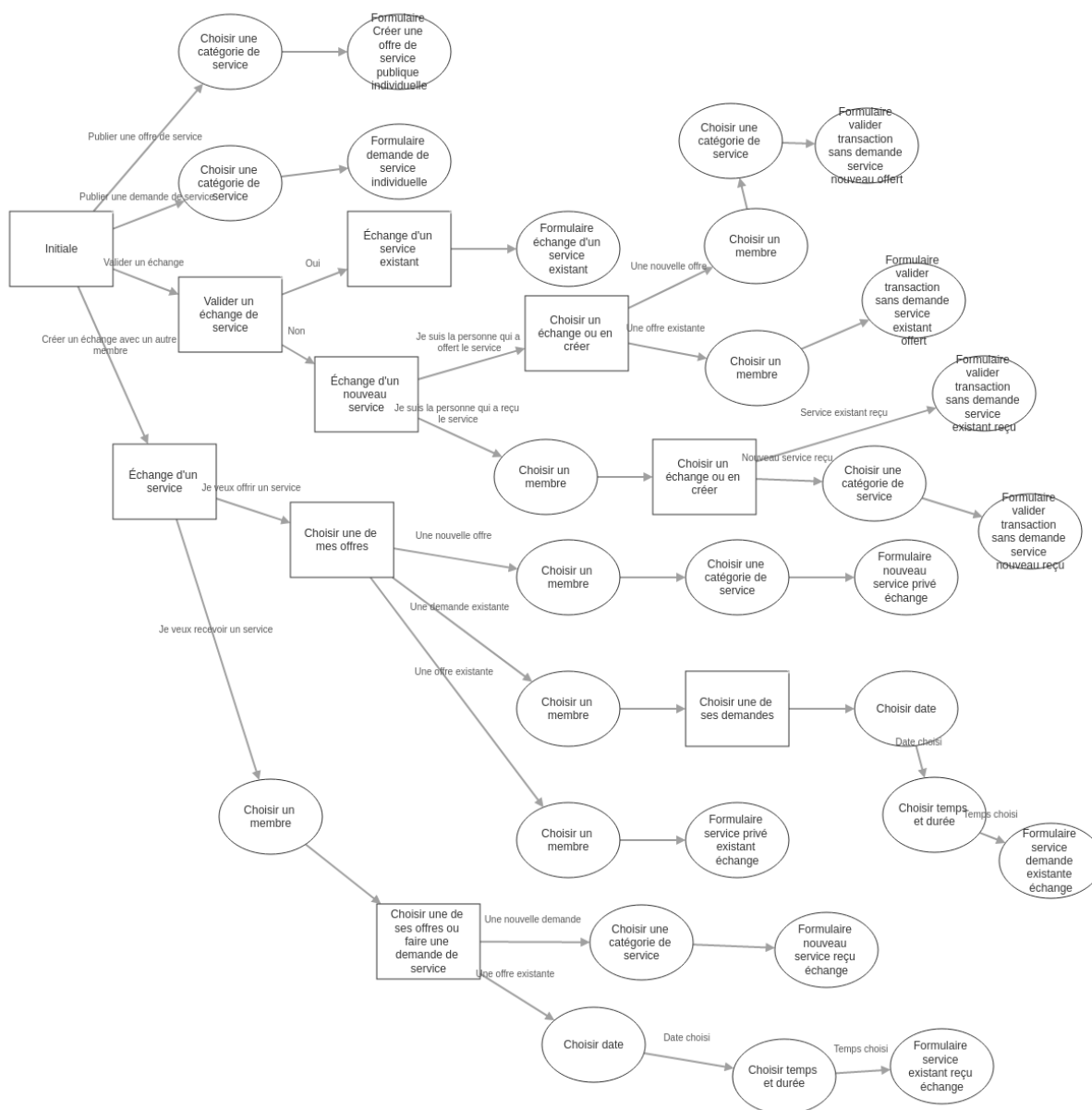
5. <https://creativecommons.org/licenses/by-sa/4.0/>

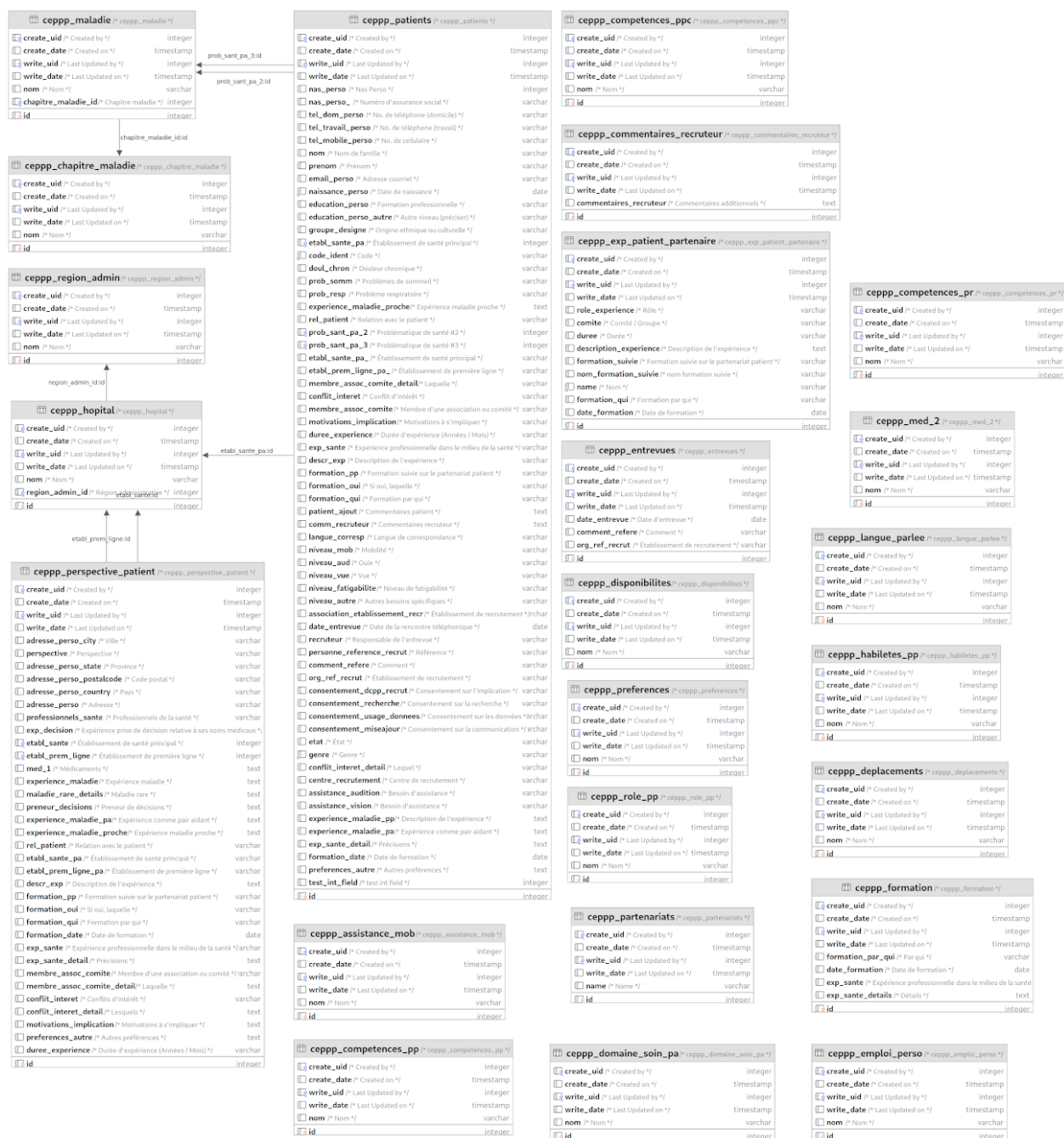
6. Une oeuvre est protégée lorsque les modifications d'autrui reste sous une licence libre pour la redistribution.

7. Licence libre restrictive au Québec : <https://forge.gouv.qc.ca/licence/>











ANNEXE T VUE FORMULAIRE PARTENAIRE PORTAIL CEPPP

The screenshot displays the 'Ceppp Patient Partenaire' interface. The top navigation bar includes a menu icon, the title 'Ceppp Patient Partenaire', and links for 'Patient' and 'Configuration'. On the right, there are icons for a clock, a chat bubble with a '1' notification, and a user profile for 'Mathieu Benoit'. Below the navigation bar, a breadcrumb trail reads 'Les patients de tous les centres de recrutement / Monique Beauvoir / 8796efd1-824d-4a1f-8133-ac10bf5913f9', followed by a pagination indicator '1 / 1' and navigation arrows. A vertical sidebar on the left contains several icons. The main content area features a form for a partner profile. At the top right of the form is a button labeled 'Fiche recruteur'. The form fields are as follows:

Code	8796efd1-824d-4a1f-8133-ac10bf5913f9
Recruteur	Stéphanie Latendresse
Centre de recrutement	CEPPP

Below these fields are four tabs: 'Disponibilité' (selected), 'Savoirs expérimentiels', 'Formations', and 'Implications'. Under the 'Disponibilité' tab, there are three sections:

- Moments de disponibilité (préférence)**: A list of preferred availability times.
- Moments d'indisponibilité**: A list of times when the partner is unavailable.
- Notification**: A checkbox that is checked.
- Recrutement**: A checkbox that is checked.
- Recherche**: A checkbox that is checked.

ANNEXE U CODE U_C⁰ DANS LE GÉNÉRATEUR DE CODE

```

1 import os
2
3 from odoo import SUPERUSER_ID, _, api, fields, models
4
5 # TODO HUMAN: change my module_name to create a specific demo
6 # functionality
7 MODULE_NAME = "code_generator_demo"
8
9 def post_init_hook(cr, e):
10     with api.Environment.manage():
11         env = api.Environment(cr, SUPERUSER_ID, {})
12
13         # The path of the actual file
14         # path_module_generate = os.path.normpath(os.path.join(os.path.
15         # dirname(__file__), '..'))
16
17         short_name = MODULE_NAME.replace("_", " ").title()
18
19         # Add code generator
20         value = {
21             "shortdesc": short_name,
22             "name": MODULE_NAME,
23             "license": "AGPL-3",
24             "author": "TechnoLibre",
25             "website": "https://technolibre.ca",
26             "application": True,
27             "enable_sync_code": True,
28             # "path_sync_code": path_module_generate,
29         }
30
31         # TODO HUMAN: enable your functionality to generate
32         value["enable_template_code_generator_demo"] = True
33         value["template_model_name"] = ""
34         value["template_inherit_model_name"] = ""
35         # value["template_module_path_generated_extension"] = "."
36         value["enable_template_wizard_view"] = False
37         value["force_generic_template_wizard_view"] = False
38         value["disable_generate_access"] = False
39         value["enable_template_website_snippet_view"] = False

```

```

39     value["enable_sync_template"] = False
40     value["ignore_fields"] = ""
41     value["post_init_hook_show"] = True
42     value["uninstall_hook_show"] = True
43     value["post_init_hook_feature_code_generator"] = True
44     value["uninstall_hook_feature_code_generator"] = True
45
46     new_module_name = MODULE_NAME
47     if (
48         MODULE_NAME != "code_generator_demo"
49         and "code_generator_" in MODULE_NAME
50     ):
51         if "code_generator_template" in MODULE_NAME:
52             if value["enable_template_code_generator_demo"]:
53                 new_module_name = f"code_generator_{MODULE_NAME[len('code_generator_template_'):]}"
54             else:
55                 new_module_name = MODULE_NAME[
56                     len("code_generator_template_") :
57                 ]
58         else:
59             new_module_name = MODULE_NAME[len("code_generator_") :]
60         value["template_module_name"] = new_module_name
61         value["hook_constant_code"] = f'MODULE_NAME = "{new_module_name}"'
62
63         code_generator_id = env["code.generator.module"].create(value)
64
65         # Add dependencies
66         lst_depend_module = ["code_generator", "code_generator_hook"]
67         code_generator_id.add_module_dependency(lst_depend_module)
68         code_generator_id.add_module_dependency_template(lst_depend_module)
69         # Generate module
70         value = {"code_generator_ids": code_generator_id.ids}
71         env["code.generator.writer"].create(value)
72
73
74 def uninstall_hook(cr, e):
75     with api.Environment.manage():
76         env = api.Environment(cr, SUPERUSER_ID, {})
77         code_generator_id = env["code.generator.module"].search(
78             [("name", "=", MODULE_NAME)]
79         )
80         if code_generator_id:
81             code_generator_id.unlink()

```