

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

mars 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 TRÉPANIER, président

Samuel 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'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 progiciels 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 libbre 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 ERP Libre 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	xiii
LISTE DES ANNEXES	xiv
 CHAPITRE 1 INTRODUCTION	1
1.1 Contexte et problématique	1
1.1.1 Choix de la plateforme ERP	1
1.1.2 Introduction Accorderie	5
1.1.3 Introduction CEPPP	5
1.2 Définitions et concepts	6
1.2.1 Caractéristiques de la plateforme Odoo	6
1.2.2 Cadre conceptuel	9
1.2.3 Exemples illustratifs d'auto-reproducteur	10
1.2.4 Exemples illustratifs de générateur de code	10
1.2.5 Tester un générateur de code	15
 CHAPITRE 2 REVUE DE LITTÉRATURE	17
2.1 Robot logiciel développeur	17
2.2 Génération de code	17
2.3 Logiciel Libre et Open Source	17
2.4 Sécurité	18
2.5 Les complexités de développer ERP	19

2.6	DevOps	20
2.7	Logiciel no-code / low-code	20
2.8	Création d'une communauté	22
2.8.1	Communication non violente	23
2.8.2	Guide construire une communauté Open Source	23
2.8.3	Guide fusée	24
2.8.4	Critères éthiques de GNU concernant l'hébergement de logiciel	24
2.9	Poïèse	25
2.9.1	Définition de la poïèse	25
2.9.2	Technopoïèse	26
2.9.3	Allopoïèse	26
2.9.4	Autopoïèse	27
2.9.5	Sympoïèse	28
CHAPITRE 3 OBJECTIFS ET MÉTHODOLOGIE		29
3.1	SO-1 - Générateur	29
3.2	SO-2 - Rétro-ingénierie	29
3.3	SO-3 - Interface	30
3.4	SO-4 - Déploiement	30
3.5	SO-5 - Réseau d'entraide	30
3.6	Environnement informatique	30
3.7	Méthodologie de test	31
3.7.1	Couverture du code	31
CHAPITRE 4 RÉSULTAT EXPÉRIMENTAUX		32
4.1	Implémentation : du modèle conceptuel au modèle opérationnel	32
4.1.1	Développement et amélioration continue	32
4.1.2	Architecture	33
4.1.3	Auto-générateur	35
4.2	Résultats propres à SO-1	36
4.2.1	Génération par gabarit	36
4.2.2	MVC	36
4.2.3	Générer un module à partir d'une base de données externes	36
4.2.4	Génération de code par des données	37
4.2.5	Interprétation des résultats de SO-1	37
4.3	Résultats propres à SO-2	38
4.3.1	Extraction de code et reproduction	38

4.3.2	Amélioration continue sur la génération	39
4.3.3	Test de validation de génération de codes	39
4.3.4	Règles de codage standardisées	40
4.3.5	Interprétation des résultats de SO-2	40
4.4	Résultats propres à SO-3	41
4.4.1	Classification des techniques développées	41
4.4.2	Interface du générateur de code	42
4.4.3	Interprétation des résultats de SO-3	42
4.5	Résultats propres à SO-4	43
4.5.1	Utilisation d'un conteneur Docker	43
4.5.2	Interprétation des résultats de SO-4	43
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	44
4.6.2	Démarrage d'un projet	44
4.6.3	Intégration d'un membre	44
4.6.4	Comportement en communauté	45
4.6.5	Outils de développement public	45
4.6.6	Résolution de problème	46
4.6.7	Documentation	46
4.6.8	Sécurité	46
4.6.9	Développement libre	47
4.6.10	Communication	47
4.6.11	Interprétation des résultats de SO-5	48
4.7	Projets divers	48
4.7.1	Projet module «auto_backup»	48
4.7.2	Projet module workflow design	49
4.7.3	Projet module STARS	49
4.7.4	Projet module SRS	49
4.8	Projet espace Accorderie	49
4.9	Projet Portail CEPPP	51
4.10	Comment les résultats obtenus soutiennent-ils le libre ?	52
4.11	Avancement sur le développement en réseau d'entraide	52
4.12	Avancement de la technopoïèse	53
4.13	Réalisation du robot logiciel générateur de code	55
4.13.1	Autonomie	55

4.13.2 Adaptable	55
4.13.3 Compétences techniques	55
4.13.4 Compétences sociales	55
4.13.5 Article «A template-based code generator for web applications» . . .	56
CHAPITRE 5 CONCLUSION	57
5.1 Synthèse des travaux	57
5.1.1 Générateur de code	57
5.1.2 Gestion de communauté autour de projet libre	57
5.1.3 Projet Accorderie	57
5.1.4 Projet Portail CEPPP	57
5.2 Limitations de la solution proposée	58
5.2.1 Couverture des tests	58
5.2.2 Personnalisation du développement	58
5.2.3 Auto-reproduction du générateur de code	58
5.2.4 Extraction de code et de modèles de base de données	59
5.3 Améliorations futures	59
5.3.1 Amélioration du générateur de code	59
5.3.2 Projet Accorderie	61
5.3.3 Projet Portail CEPPP	61
5.3.4 NLP	61
5.4 Importance de la recherche	61
RÉFÉRENCES	63
ANNEXES	69

LISTE DES TABLEAUX

Tableau 1.1	Tableau des dates de lancement du logiciel Odoo à partir de la version 6.0	2
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	Les 7 étapes pour démarrer un projet dans un réseau d'entraide . . .	45
Tableau 4.2	L'évolution entre la génération et la ré-ingénierie des statistiques sur les langages du portail CEPPE	51

LISTE DES FIGURES

Figure 1.1	Mode direct et inverse	9
Figure 1.2	Exemple de code Hello, World!	10
Figure 1.3	Exemple de code Quine	11
Figure 1.4	Exemple de technique de génération de code basique d'un «Hello World»	11
Figure 1.5	Exemple de technique de génération de code basique paramétrable d'un «Hello World»	12
Figure 1.6	Exemple de technique de génération de code par template avec Jinja2 d'un «Hello World»	13
Figure 1.7	Exemple de template Jinja2 avec logique	13
Figure 1.8	Exemple de génération de module MVC Odoo avec la technique du Scaffold	14
Figure 1.9	Exemple de génération de module thème Odoo avec la technique du Scaffold	14
Figure 1.10	Altération du code avec la rétro-ingénierie et la ré-ingénierie. Image modifiée [15]	15
Figure 1.11	Exemple de technique de génération de code avec rétro-ingénierie d'un «Hello World»	16
Figure 2.1	Le processus DevOps. Source : Pease, 2017. [36]	20
Figure 2.2	Communication non violente en 4 étapes et 2 modes. Dessin réalisé par Célia Portail [42]	23
Figure 2.3	Guide fusée en 7 étapes pour démarrer un projet pour un gestionnaire de communauté, de CimarLab [44]	24
Figure 4.1	Interaction du développeur avec le générateur de code	33
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.	34
Figure 4.3	Exemple support MVC dans GUI du générateur de code	36
Figure 4.4	Procédure STARS dans l'application Projet vue Diagramme	49
Figure 4.5	Suivi des tâches de projet avec procédure STARS en vue Kanban . .	50
Figure 4.6	Intéraction entre les développeurs et les outils de développement dans un réseau d'entraide	53
Figure 4.7	Architecture du générateur de code	54

LISTE DES SIGLES ET ABRÉVIATIONS

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

LISTE DES ANNEXES

Annexe A	GUI générateur de code - les modèles	69
Annexe B	GUI générateur de code - les champs	70
Annexe C	GUI générateur de code - les codes	71
Annexe D	GUI générateur de code - les «hooks»	72
Annexe E	Test couverture technique générateur de code	73
Annexe F	Diagramme modèle de données Espace Membre Accorderie 2019 . . .	75
Annexe G	Diagramme nouveau modèle de données Espace Membre Accorderie 2023	76
Annexe H	Diagramme processus pour demander, offrir, établir un échange et le valider - Accorderie 2023	77
Annexe I	Diagramme modèle de données du portail CEPPP septembre 2022 . .	78
Annexe J	Vue formulaire administration portail CEPPP	79
Annexe K	Vue formulaire partenaire portail CEPPP	80
Annexe L	Code u_C^0 dans le générateur de code	81

CHAPITRE 1 INTRODUCTION

1.1 Contexte et problématique

La valeur du marché des solutions ERP s'établissait autour de 40 milliards USD mondialement en 2020 [1, 2]. Le coût moyen par utilisateur, sur 5 ans, s'élevait à 9 000\$, pour une PME, en 2022 [3]. Le développement de système ERP est complexe, nécessite une maintenance exigeante et le risque d'introduire des erreurs est important [4] [5].

La plateforme Odoo a réussi à devenir l'un des principaux fournisseurs d'ERP 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.

Comment accélérer le développement de fonctionnalités de la plateforme ERP Odoo¹ 12 communautaire ?

La plateforme ERPLibre [7] a été créée dans l'objectif d'accélérer le développement de la plateforme Odoo communautaire. Ce mémoire va mettre l'accent sur la génération de code par des techniques de rétro-ingénierie et la gestion d'une communauté, dans un contexte d'un projet de logiciel libre, une solution développée, démontrée dans ce mémoire, pour complémenter la plateforme ERPLibre.

1.1.1 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 choisi 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 «Tiny ERP», en février 2005, cette plateforme a évolué au fil du temps, elle a été renommée pour OpenERP autour d'octobre 2009, passant sous licence AGPL.

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

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

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		

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. Ainsi, en date du premier janvier 2023, la version 12.0 est encore le bon choix 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 en 2024, selon l’évolution.

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 pour 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.

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

3. Logiciel de gestion de versions décentralisées

Dans la section «total» du tableau 1.2, la section unique signifie que la somme va ignorer les 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, ils peuvent avoir des bogues ou des fonctionnalités différentes entre eux.

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.

Légende	Total OCA Entreprise	Version actuelle	Anciennes versions avec maintenance étendue	Anciennes versions ou fin de maintenance				
17309/10728/6581 modules à supporter le 1er janvier 2023								
17465/10952/6513 modules le 15 février 2023								
156/132/24 nouveaux modules en 31 jours, durant janvier 2023								
Odoo version	2016	2017	2018	2019	2020	2021	2022	2023
6.1	295 269 26	299 270 29	299 270 29	299 270 36	299 270 36	299 270 36	299 270 36	299 270 36
7.0	637 597 40	633 592 41	634 593 41	635 594 41	669 619 50	669 619 50	669 619 50	671 619 52
8.0	741 597 144	1092 907 185	1215 996 219	1265 1027 238	1290 1036 254	1297 1043 254	1340 1049 291	1341 1050 291
9.0	135 46 89	456 346 110	725 602 123	776 643 133	796 659 137	803 666 137	844 666 178	850 669 181
10.0		523 111 412	954 713 241	1 537 953 584	1 647 1 047 600	1 685 1 085 600	1 754 1 109 645	1 765 1 120 645
11.0			288 77 211	1 398 658 740	1 710 929 781	1 797 1 000 797	1 860 1 023 837	1 869 1 032 864
12.0				784 137 647	1 837 993 844	2 503 1 464 1 039	2 851 1 633 1 218	2 977 1 693 1 284
13.0					617 115 502	1 445 844 601	2 024 1 310 714	2 241 1 506 735
14.0						906 129 777	2 150 1 143 1 007	2 648 1 698 950
15.0							786 96 690	1 669 865 804
16.0								972 206 766
Total								
Somme	1 808 1 509 299	3 003 2 226 777	4 115 3 251 864	6 701 4 282 2 419	8 872 5 668 3 204	11 411 7 120 4 291	14 584 8 918 5 666	17 309 10 728 6 581
Support Odoo	1 808 1 509 299	2 704 1 956 748	3 182 2 388 794	4 495 2 391 2 104	5 811 3 084 2 727	6 651 3 437 3 214	7 811 4 182 3 629	7 530 4 275 3 255
Unique	1 244 1 047 197	1 995 1 435 560	2 260 1 845 415	3 214 2 172 1 042	3 927 2 610 1 317	4 676 3 080 1 596	5 452 3 572 1 880	6 063 3 980 2 083
133/72/61 répertoires de modules dans ERPLibre 1.5.0								

1.1.2 Introduction Accorderie

L'Accorderie, un réseau d'entraide québécois, était à la recherche d'une plateforme améliorée, avec des technologies plus récentes. Les membres ont commencé, depuis quelques années, à utiliser des plateformes alternatives dû à 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 deviendra plus attractive.

Nous avons décelé que le présent projet pourrait répondre aux besoins de l'Accorderie avec le générateur de module 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 copyright mentionne l'année 2007, par la compagnie GRF Ressource Informatique. De plus, nous avons aussi eu accès à la base de données et, selon les archives, le premier échange tracé est le 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 dont le présent mémoire fait objet comme premier cas pratique d'étude.

1.1.3 Introduction CEPPP

Le CEPPP, Centre d'excellence sur le partenariat avec les patients et le public, était à la recherche d'une plateforme pour la gestion du partenariat avec les patients et le public. Le Portail des partenaires (“Portail”) du 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. Ils avaient donc déjà une solution [9], 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.

Le présent projet venait pallier aux problèmes rencontrés en facilitant et en accélérant le

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

développement, en permettant une personnalisation et en développant l'anonymisation des données et était donc tout indiqué comme second cas pratique d'étude.

1.2 Définitions et concepts

1.2.1 Caractéristiques de la plateforme Odoo

Architecture MVC

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

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

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

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

Architecture ORM

L'utilisation de requêtes SQL pour communiquer avec des bases de données demande un temps considérable au développeur à implémenter et elle nécessite la programmation d'interface avec la base de données. L'utilisation d'un 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 objet⁵. 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.

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

Architecture modulaire par héritage

L'ensemble des fonctionnalités nécessaire 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 :

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

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.

Website builder

Le «Website Builder» est un outil qui permet une autonomie aux utilisateurs lambda⁶ (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 LCNC⁷ dans Odoo 12 pour créer et modifier un site web multi-pages par un mécanisme de «drag and drop»⁸ avec des «snippets»⁹ paramétrables. Il permet de modifier une page web en réduisant le besoin de recourir à un expert technique abaissant ainsi les coûts de développement.

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 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.

ERPLibre

Depuis la version 9.0 d'Odoo, une version communautaire et entreprise ont été créés causant une divergence sur les fonctionnalités, créant le modèle d'affaires «Open Core»¹⁰. 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

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

7. Qui nécessite peu ou pas du tout du code

8. Glisser-déposer

9. Fragment de code

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

le déploiement et le développement pour une organisation en permettant la gestion des dépendances avec Poetry, automatisation du déploiement avec les Docker, la gestion de tous les répertoires de module 1.2 avec Git Repo¹¹, 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.

1.2.2 Cadre conceptuel

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

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

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

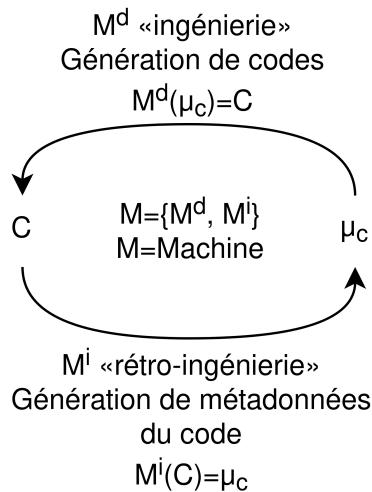


Figure 1.1 Mode direct et inverse

Pour concrétiser le sens de ce modèle formel, nous allons proposer quelques exemples simples.

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

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 1.2.

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, 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.

1.2.3 Exemples illustratifs d’auto-reproducteur

Le Quine

«Un quine [12] (ou programme auto-reproducteur, self-reproducing en anglais) est un programme informatique qui imprime son propre code source» [13] 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 1.3, la sortie textuelle dans la console, lors de l’exécution, est la même que son code.

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

1.2.4 Exemples illustratifs de générateur de code

La génération de code est un des moyens pour soutenir le développeur dans le développement d’un logiciel. C’est la partie «création de code» de la méthodologie DevOps 2.6.

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

Figure 1.2 Exemple de code Hello, World !

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

Figure 1.3 Exemple de code Quine

Technique de génération de code basique

Dans Figure 1.4, 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 1.5, cette technique basique est paramétrable, contrairement au premier exemple Figure 1.4. 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».

Le code de La Figure 1.6 utilise la bibliothèque «Jinja2». C'est un mécanisme similaire à Figure 1.5, cependant la logique est intégrée directement dans le fichier template.

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

Code 1.1 C de Figure 1.4

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

Code 1.2 μ_C de Figure 1.4

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

Code 1.3 M(μ_C) de Figure 1.4

Figure 1.4 Exemple de technique de génération de code basique d'un «Hello World»

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

Code 1.4 C - fichier C.py de Figure 1.5

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

Code 1.5 μ_C - fichier uC.json de Figure 1.5

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

Code 1.6 M(μ_C) de Figure 1.5

Figure 1.5 Exemple de technique de génération de code basique paramétrable d'un «Hello World»

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 1.8, 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 1.9, 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 1.7 C - fichier C.py de Figure 1.6

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

Code 1.8 μ_C - fichier uC.json de Figure 1.6

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

Code 1.9 template - fichier template de Figure 1.6

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

Code 1.10 M(μ_C) de Figure 1.6

Figure 1.6 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 1.7 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 1.8 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 1.9 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 [14] avec un générateur de code, l'objectif est de faire de la ré-ingénierie sur la partie génération, ainsi on altère le code, voir Figure 1.10.

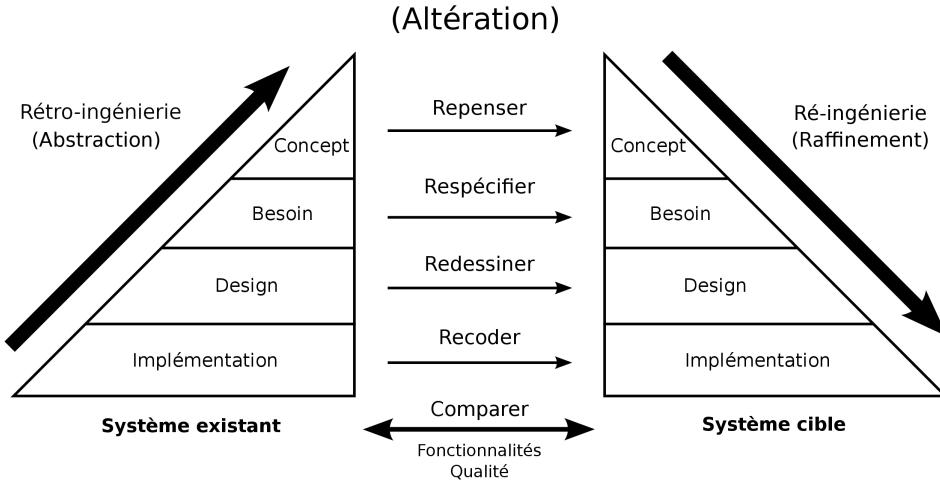


Figure 1.10 Altération du code avec la rétro-ingénierie et la ré-ingénierie. Image modifiée [15]

Dans Figure 1.11, le code 1.12 utilise la bibliothèque 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 1.13 du code source original 1.11 devient formaté en PEP8 [16].

1.2.5 Tester un générateur de code

Il existe le test «Output comparison testing» [17] 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 1.11 C mal formaté - fichier C.py de Figure 1.11

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

Code 1.12 M qui extrait μ_C pour générer C.py de Figure 1.11

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

Code 1.13 C corrigé - fichier C.py de Figure 1.11

Figure 1.11 Exemple de technique de génération de code avec rétro-ingénierie d'un «Hello World»

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 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.» [18] 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» [19].

Selon les auteurs Erlenov et al. en 2019 [19], 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 tel que CodeBuddy [20] ou CodePilot [21] dans leur développement collaboratif.

2.2 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 [22], ils obtiennent un gain de performance en temps de développement de 98.95% et 0 bogue via le générateur de code.

Dans un autre projet de les auteurs Pichidtienthum et al. en 2021 [23], en ajoutant une interface LCNC sur un générateur de code sur Odoo, ils obtiennent 20% de réduction de temps 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.

2.3 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 [24], 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.

D'ailleurs, comme le mentionne les auteurs Hertel et al. en 2003 [25], 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.

2.4 Sécurité

Bien qu'il y a 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 [26] 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 [27], que la morale est assez simple : il est impossible de fonctionner¹ 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 [28] [29] ? Le développeur choisi des bibliothèques libres qui utilisent d'autres bibliothèques pas nécessairement libres et qui peuvent contenir des failles de sécurité [30], 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 rétroconception² 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'IA conviviale pour l'homme, les attaquants qui obtiennent un accès en écriture aux poids³ pourront planter leurs portes dérobées. [31]

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 dépendance et générer le code idéal pour la remplacer par une nouvelle application digne de confiances en validant la propriété intellectuelle libre.

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

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

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

2.5 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 reengineering⁴ 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.» [32] C'est pourquoi il est important de suivre les différentes étapes de réalisation du projet pour réduire les erreurs d'ingénierie sur le développement. Les étapes suivantes ont été définies par l'auteur Braud en 2008 dans son mémoire «Facteurs décisionnels pour l'implantation d'un erp dans les pme : Le rôle de l'évaluation des bénéfices tangibles et intangibles» [33] telles que :

1. Évaluation des besoins ;
2. Préparation du projet ;
 - (a) Organisation du projet ;
 - (b) Définir les objectifs ;
 - (c) Créer un plan détaillé ;
3. Dessin d'affaires ;
 - (a) Analyser les processus d'affaires actuels ;
 - (b) Maîtrise du système ERP ;
 - (c) Revue des processus ;
4. Réalisation ;
 - (a) Développement technique ;
 - (b) Étude pilote ;
5. Préparation finale ;
 - (a) Réglages et tests ;
 - (b) Éduquer et former la masse critique ;
6. Mise en production et support ;
 - (a) Déploiement des modules ERP ;
 - (b) Améliorer et élargir les systèmes ERP de façon continue.

Un autre facteur complexe est l'association des processus d'affaires actuels à la ré-ingénierie 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

4. Ré-ingénierie

dans le travail des auteurs Medjek et al. en 2018 [34], 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 à 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.

2.6 DevOps

Comme le mentionne l'auteur Ebert en 2016 [35], 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 2.1.

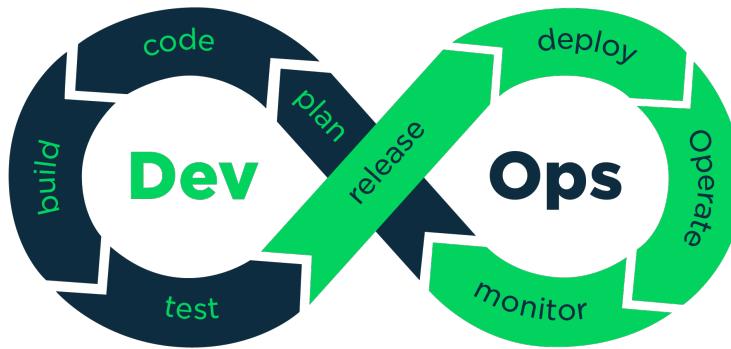


Figure 2.1 Le processus DevOps. Source : Pease, 2017. [36]

2.7 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 pas ou peu de code. Selon l'auteur Bock en 2021 [37], le LCNC a besoin de :

1. Aspect général ;
- (a) Gestion des rôles et permissions par des groupes utilisateurs ou individuelle ;
 - (b) Mécanisme de déploiement et exportation ;

2. Perspective d'interaction ;
 - (a) Mécanisme pour changer le design de l'interface utilisateur ;
 - (b) Mécanisme pour coupler l'interface à un modèle et un contrôleur (MVC) ;
 - (c) Mécanisme pour faire le rendu visuel sur différents types d'appareils ;
3. Perspective dynamique ;
 - (a) Gestion des processus du système et de la machine ;
 - (b) Composantes de modélisation de processus conceptuel ;
 - (c) Système de gestion des états et des transitions ;
4. Perspective fonctionnelle ;
 - (a) Mécanisme de spécification fonctionnel de base ;
 - (b) Générateur d'algorithme ;
 - (c) Générateur de code de composantes ;
 - (d) Mécanisme d'accès à des API externe ;
5. Perspective statique ;
 - (a) Composante de conception d'un modèle de données ;
 - (b) Composante pour spécifier des structures de données ;
 - (c) Gestion de base de données interne ;
 - (d) Gestion de base de données externes par API ;

Selon l'auteur Apurvanand en 2020 [38], il faut les éléments techniques suivants pour supporter une plateforme LCNC :

1. Bases de données ;
2. Services externes ;
3. Gestion des modèles de données ;
4. Plateforme collaborative ;
5. Service infonuage (déploiement, audit de performance, gestion des erreurs/traces/événements, gestion des versions) ;
6. Générateur de code ;
7. Compilateur et optimiseur de code ;
8. Modeleur d'application ;
 - (a) Widget ;

- (b) Connecteur ;
- (c) Processus de logique métier ;
- (d) Capacité de «drag and drop» ;
- (e) Modèle de données ;
- (f) Règles de sécurité ;

L'auteur Apurvanand poursuit en mentionnant les besoins de niveau supérieur d'une plate-forme LCNC :

1. GUI ;
2. Interopérabilité, support entre services externes et base de données ;
3. Support de la sécurité ;
4. Support d'une plateforme collaborative ;
5. Support sur la réusabilité, pouvoir répéter l'utilisation d'une composante dans différents contextes ;
6. Support de la capacité d'un système à maintenir ou à améliorer ses performances ;
7. Mécanisme de spécification de logique du développement des affaires ;
8. Logiciel pour construire des mécanismes ;
9. Support au déploiement ;

Hors, en supportant l'ensemble de ces mécanismes et éléments techniques, nous devrions obtenir un LCNC intégré dans le robot logiciel codeur.

2.8 Crédit à 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 qui seront rendues disponibles. Un logiciel libre et ouvert évolue proportionnellement au dynamisme de sa communauté[...]» [39]

«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.» [40]

2.8.1 Communication non violente

La mise en place d'une méthode de communication non violente⁵ formalisée par Rosenberg en 2015 [41], voir Figure 2.2, permettrait de réduire les frictions et faciliter l'intégration entre les participants du réseau d'entraide.

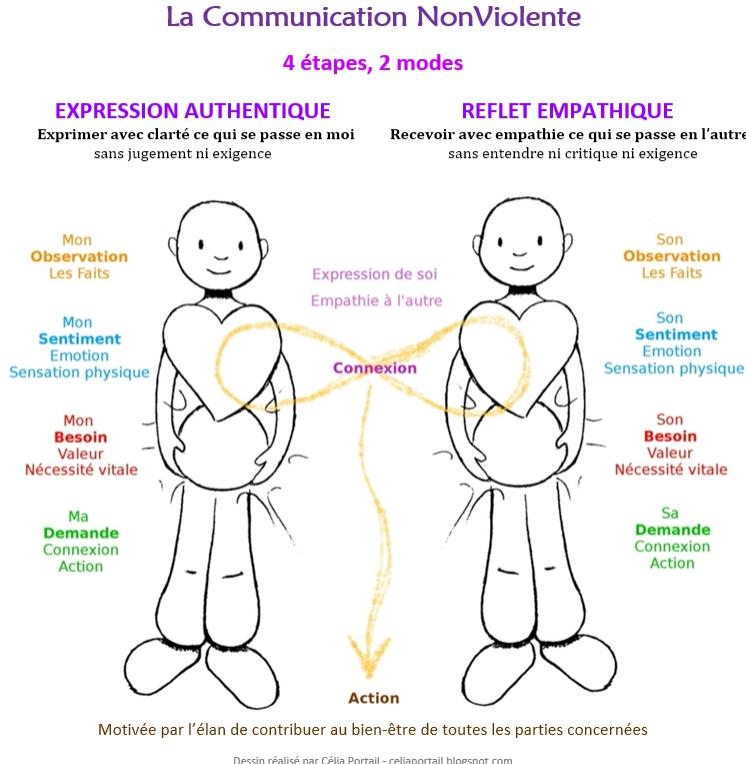


Figure 2.2 Communication non violente en 4 étapes et 2 modes. Dessin réalisé par Célia Portail [42]

2.8.2 Guide construire une communauté Open Source

Un guide [43] est accessible publiquement pour aider les gestionnaires de communautés en 4 sections avec des titres indicateurs d'orientation :

1. Mise en place de votre projet pour le succès ;
2. Cultiver votre communauté ;
3. Résoudre les conflits ;
4. La communauté est le cœur de l'open source.

5. https://fr.wikipedia.org/wiki/Communication_non_violente

Il faut :

1. rédiger un code de conduite ;
2. proposer la contribution directement sur le projet.

Il y a beaucoup d'autres guides, mais en général, ils n'intègrent ni les aspects de génie industriel qui est vulgarisé avec le guide fusée et ni les critères éthiques de GNU concernant l'hébergement de logiciel.

2.8.3 Guide fusée

Un guide en 7 étapes du guide Fusée de CimarLab en 2023 [44], voir Figure 2.3 pour les gestionnaires de projet. Il permet de démarrer un projet rapidement qui nécessite une équipe de personnes pour les rendre efficaces dans la réalisation de leur participation dans le réseau d'entraide.

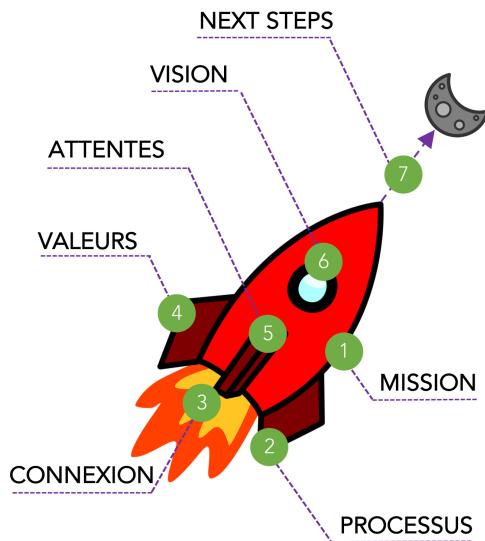


Figure 2.3 Guide fusée en 7 étapes pour démarrer un projet pour un gestionnaire de communauté, de CimarLab [44]

2.8.4 Critères éthiques de GNU concernant l'hébergement de logiciel

La licence AGPLv3 n'est pas toujours bien respectée [45]. Il est important d'assister le développeur dans ses choix de licence et 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 [46] 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 libre⁶, la compatibilité avec les licences à copyleft et leur philosophie, et l'absence de discrimination contre les utilisateurs, quels qu'ils soient. Les questions à répondre :

1. Est-ce que l'hébergeur fournit l'accès au code source des programmes qu'il héberge ?
2. Est-ce que l'hébergeur permet la redistribution des copies des programmes qu'il héberge ?
3. Est-ce que l'hébergeur permet aux utilisateurs d'apporter des modifications aux programmes qu'il héberge et de les partager avec la communauté ?
4. Est-ce que l'hébergeur impose des restrictions sur l'utilisation ou la redistribution des programmes qu'il héberge ?
5. Est-ce que l'hébergeur respecte les licences de logiciels libres et les droits d'auteur associés aux programmes qu'il héberge ?
6. Est-ce que l'hébergeur fournit des informations sur les licences de logiciels libres et les droits d'auteur associés aux programmes qu'il héberge ?
7. Est-ce que l'hébergeur respecte la vie privée et la sécurité des utilisateurs des programmes qu'il héberge ?
8. Est-ce que l'hébergeur fournit un support et une assistance adéquats aux utilisateurs des programmes qu'il héberge ?

2.9 Poïèse

2.9.1 Définition de la poïèse

La poïèse [47] (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.

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

En dehors de l'art, le terme poïèse [48] 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.

2.9.2 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 ontologiquement⁷ le travail de l'un et son résultat dans l'autre.» [49] L'artiste⁸ 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 poïétique 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?» [49]

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

2.9.3 Allopoïèse

L'Allopoïèse peut être décrite comme un système qui développe⁹ 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

7. 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.

8. Voir l'artiste de la «Définition de la poïèse» 2.9.1

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

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.» [50] [51]

2.9.4 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)» [52].

Le maintien de sa propre organisation signifie l'auto-production, voir exemple illustratif auto-reproducteur du code 1.2.3.

Selon l'auteur Nomura en 2000 [53], un système est de l'autopoïèse dans le contexte qu'il est :

1. **Autonome** : il doit être capable d'apporter des changements variés pour maintenir son organisation ;
2. **Individuel** : 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 ;
3. **Connaissant et établis ses limites** : il doit être capable d'établir ses limites dans son processus de reproduction par lui même sans se faire affecter des limites établis par les observateurs externes ;
4. **Absent d'entrant et de sortant** : Les stimuli externes doivent être interprété dans un contexte d'observation pour en retirer de l'amélioration continue, elles ne doivent pas impacter la maintenance de l'organisation directement, mais son évolution doit en prendre compte.

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 autopoïète, de l'article [53], devrait comporter les points suivants :

1. Les composantes du système sont déterminés par les opérations du système ;
2. Les opérations du système sont produites avant les conditions initiales ;
3. Les opérations du système sont seulement exécutées pour leur propre réussite et non pour réaliser la production d'un produit ;

4. 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 [53] 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 automatique 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.

2.9.5 Sympoïèse

La sympoïèse peut être définie comme un système qui se développe en collectivité. C'est un concept utilisée 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 Guillibert 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.» [54]

Ainsi, le développement de technologie en collectivité va permettre de gérer des cas d'urgence humanitaire tel que le «mouvement des villes en transition» [55]¹⁰ en développant des technologies permettant de mettre en place des systèmes d'échange local [56]¹¹.

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

11. https://fr.wikipedia.org/wiki/Système_d'échange_local

CHAPITRE 3 OBJECTIFS ET MÉTHODOLOGIE

L'objectif général de ce projet 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 les cinq sous-objectifs suivants :

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

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

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

SO-4 développer un système de distribution,

SO-5 développer un système de gestion de communauté.

La méthodologie Agile a été adoptée pour le développement de ces composantes d'un générateur de code. 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 [57] a été publié sur Github, cependant il a nécessité des modifications qui seront détaillées dans les résultats.

3.1 SO-1 - Générateur

Cette section sert à la partie de génération de code, génération de module et génération de modèles de données :

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

3.2 SO-2 - Rétro-ingénierie

Cette section sert à la partie rétro-ingénierie, l'amélioration continue et la qualité logicielle :

1. Développer la capacité de comprendre une structure de code et de la reproduire ;
2. Définir ce qui est de l'amélioration continue et son application dans un contexte d'automatisation ;

3. Mise en place de test de validation de code sur des critères de qualité mesurables ;
4. Intégration de règles de codage standardisées pour favoriser le réseau d'entraide.

3.3 SO-3 - Interface

Cette section sert à la partie interface utilisateur :

1. Proposer une classification des techniques que le robot logiciel codeur peut réaliser en programmation ;
2. Développer une interface permettant le contrôle du robot logiciel codeur pour l'orienter dans la programmation de fonctionnalités ;
3. 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 sert à la partie gestion du développeur, ainsi que les projets d'études :

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

3.6 Environnement informatique

La plateforme ERPLibre 1.5.0 contenant Odoo 12 communauté, qui utilise les langages Python 3.7, XML, Javascript et SCSS est l'environnement informatique choisi. 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 avec la technique de parallélisme avec la bibliothèque Asyncio et des nouveaux processus et non des «thread»¹. Seul le test de nouveau projet est exécuté après le parallélisme, 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 fonctionnalité générée.

1. Processus légué

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 1.2.2. 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 [57] 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. L'interface est dé-couplée en deux ensembles de méta-données : μ_C^A et μ_C^B . μ_C^B est l'ensemble qui paramétrise le passage de méta-données au code pour un module spécifique. μ_C^A est l'ensemble qui paramétrise le passage de code aux méta-données tout en préparant un μ_C^B adapté à ce module spécifique. Ce découplage a permis l'adaptation de l'interface au contexte de l'installation de modules sur une instance Odoo via des «hooks». Par la suite, un ensemble supplémentaire de méta-données, noté μ_C^0 , a été dégagé de la programmation manuelle de versions successives de μ_C^A . μ_C^0 sert à initialiser une version de départ de μ_C^A .

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

4.1.1 Développement et amélioration continue

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

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

1. <https://erplibre.ca>

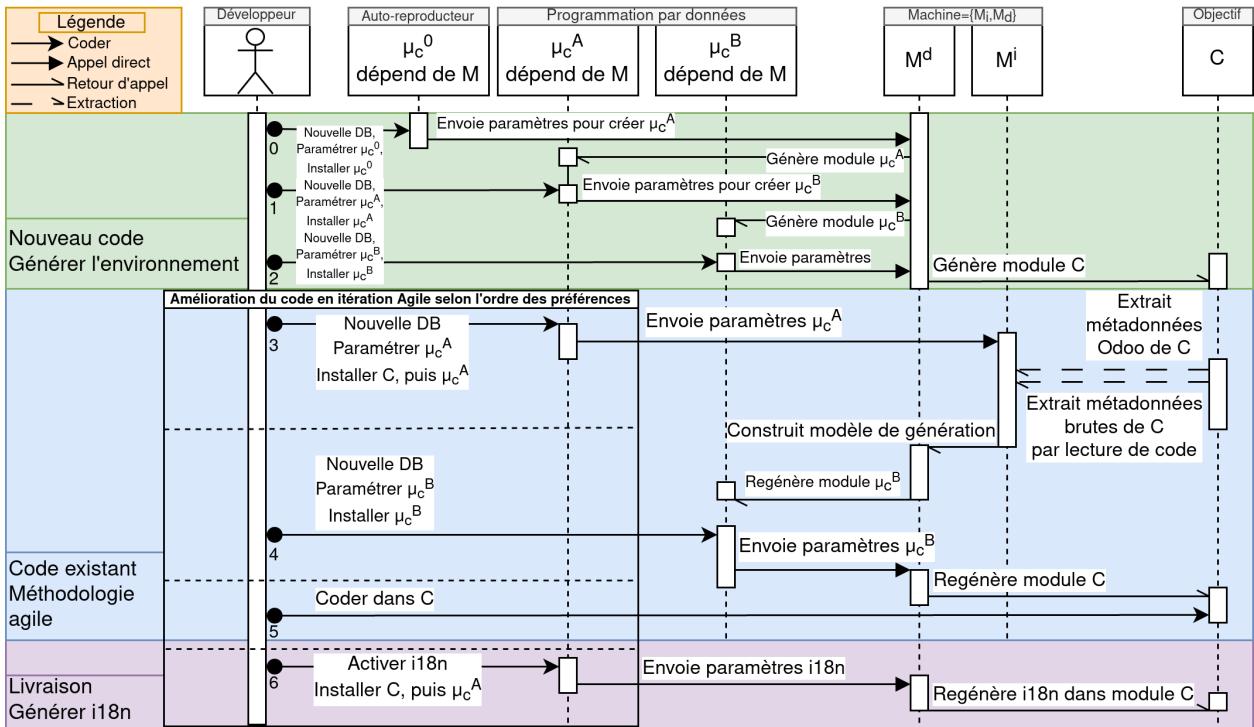


Figure 4.1 Interaction du développeur avec le générateur de code

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

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

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

4.1.2 Architecture

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

Description des différentes sections :

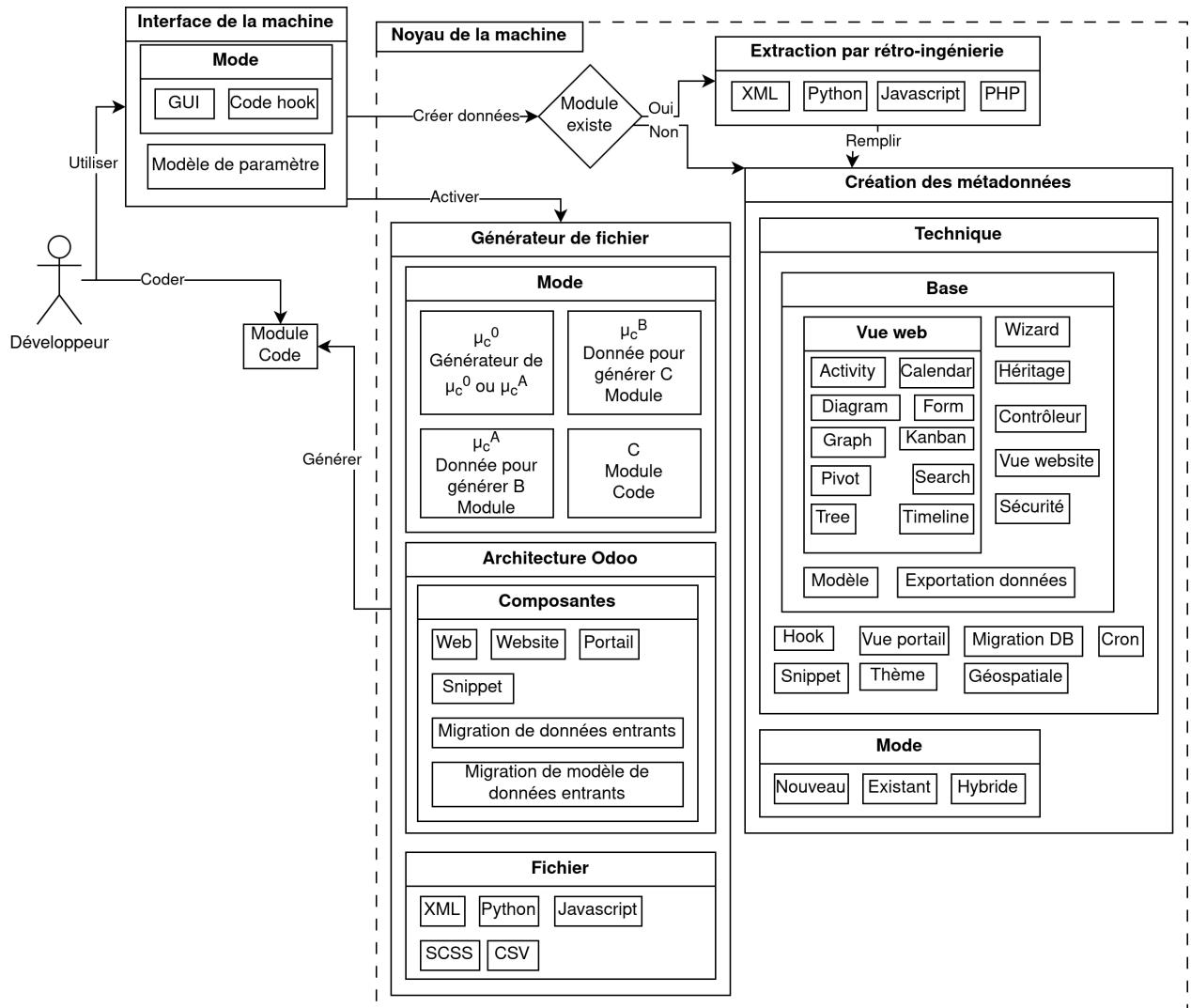


Figure 4.2 Architecture du générateur de code dans son ensemble, nommé machine. Le développeur peut modifier le code source directement ou utiliser l'interface de la machine qui passe par un ensemble de composantes.

1. 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 qu'on vient activer la génération de code.
2. L'extraction par rétro-ingénierie permet de remplir le modèle de méta-données sans passer par la paramétrisation. Il extrait des informations qui ne sont pas accessibles dans le modèle de données d'Odoo sur le module.
3. La création de méta-données se fait soit par l'utilisateur via la GUI ou le «Code Hook», il gère 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».
4. Le générateur de fichier se fait activer par l'interface, mais il prend les méta-données pour faire sa génération selon le mode qu'il doit générer et l'architecture qu'il connaît. Il fait des liaisons entre les modèles et les vues en référence aux noms des champs de chaque modèle de données.

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

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

4.1.3 Auto-générateur

Représenté par μ_C^0 , voir Annexe L, c'est un auto-générateur. Au moment de son installation, il génère le même code que lui-même, au même endroit dans le système de fichier. Une légère modification va créer une autre entité qui sera une déviation dans l'objectif de démarrer une chaîne 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 c'est un succès lorsqu'il n'y a pas de différence. Cependant, sa programmation est actuellement spécifique à sa génération, aucun autre module n'a besoin de cette fonctionnalité unique.

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

4.2 Résultats propres à SO-1

4.2.1 Génération par gabarit

La génération par gabarit était déjà supportée dans la version initiale [57], de plus, il y a eu des améliorations :

1. Utilisation des f-strings au lieu d'utiliser la fonction «format» de String ;
2. Utilisation de la bibliothèque Code-writer en Python² ;
3. Utilisation de la bibliothèque lxml³.

4.2.2 MVC

L'architecture MVC était déjà supportée dans la version initiale [57], de plus, il y a eu des améliorations :

1. Ajout de bouton qui ouvre des «Wizard» pour générer les «Views», les «Models» et les «Controllers», voir Figure 4.3. Ainsi, le développeur peut les configurer et demander de générer les méta-données associées.

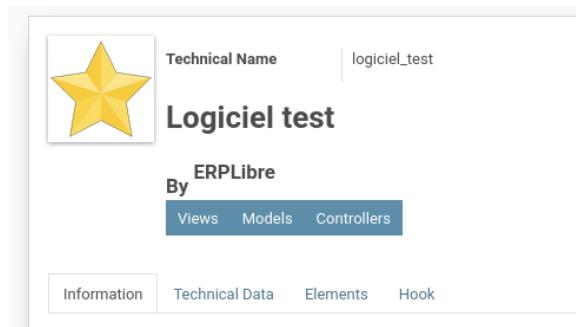


Figure 4.3 Exemple support MVC dans GUI du générateur de code

2. Les règles de sécurité sont ajustées selon les configurations et personnalisables par la suite.

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 [57], cependant il y a eu des améliorations :

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

1. Ajout de type de données, dont celles utilisées par le projet Accorderie ;
2. Ajout d'associations entre les types de données et les différentes personnalisations de l'architecture Odoo ;
3. Interface représentant la base de données avec les contrôleur permettant la configuration de la migration sur le modèle de données désiré ;
4. Gestion des interdépendances (A de B et B de A).

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 métadonnées paramétrables par l'humain. Les deux couches interfèrent entre elles pour permettre l'évolution de la fonctionnalité désirée.

4.2.5 Interprétation des résultats de SO-1

SO-1 Accomplissements

Voici l'ensemble résumé des accomplissements de SO-1 :

1. 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» ;
2. Ajout de «Wizard» pour configurer le MVC selon des paramètres ;
3. Amélioration de l'importation des bases de données externes ;
4. Support de la génération de code par des données ;
5. Augmentation de technique de génération de code, support des «templates» «Qweb», ajout du type de données géospatiales.

SO-1 Feuille de route

Voici la brève feuille de route pour SO-1 :

1. Implémenter les fonctionnalités manquantes dans la GUI pour générer les MVC d'un module ;
2. Augmenter le nombre de technologies à supporter sur l'importation des données ;

3. Ajouter le support de génération sur différentes architectures ;
4. Supporter différentes techniques de sécurités personnalisables, comme l'anonymisation des données ;
5. Générer automatiquement une documentation sur l'utilisation d'une technique ;
6. Supporter la génération sur d'autres systèmes ERP libres tel que Tryton⁴ ;

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 suffit 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 les recopier.

Cependant, il était nécessaire, dans certains contextes, de faire de la micro extraction, comme :

1. L'extraction des noms des constantes, elles sont transformées en valeurs lors de l'exécution ;
2. L'extraction des commentaires, qui n'est pas supportée dans la bibliothèque AST de Python 3.7 ;
3. L'extraction des décorateurs ;
4. L'extraction des paramètres et noms des méthodes/fonctions ;
5. Les méta-information d'un modèle (description, nom, etc.)

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.

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 communauté⁵.

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 génér-

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

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

rateur 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, corrigant automatiquement les problèmes.

L'intérêt de passer par 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 comparaisons des sorties de la génération a été utilisée. Pour procéder, un développeur valide via l'outil Git ce qui est commité⁶. 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.

Il y a plusieurs types de tests :

1. Valider l'installation du module généré ;
2. Valider que μ_C^B génère bien le module cible sans différence dans le code ;
3. Valider que μ_C^A génère bien μ_C^B sans différence dans le code ;
4. L'extraction des paramètres et noms des méthodes/fonctions ;
5. Valider que la migration d'une base de données SQL se fait sans différence dans le code.

En exécutant tous les tests, voir Annexe E, 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 comment utiliser le 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.

6. 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 comparé 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 faire 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 modèle⁷.

4.3.5 Interprétation des résultats de SO-2

SO-2 Accomplissements

Voici l'ensemble résumé des accomplissements de SO-2 :

1. Extraction du code via l'utilisation d'un AST et extraction des méta-données dans les fichiers XML ;
2. Amélioration continue sur la génération de code grâce à la reproduction à l'aide de l'extraction du code ;
3. Un outil pour aider à la création de technique de génération à l'aide d'un générateur de générateur de code ;
4. Le générateur de code est accompagné de tests de validation en reproduisant l'ensemble des techniques en démonstration ;
5. La génération de code applique des règles de codage standardisées.

SO-2 Feuille de route

Voici la brève feuille de route pour SO-2 :

1. Finaliser l'implémentation de l'auto-génération sur le générateur de code ;
2. Mettre à jour les tests pour atteindre une couverture de code à 100% ;
3. Ajouter des tests sur les techniques d'extraction de code tel que le PHP.

7. 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» étaient déjà implémentés dans la version initiale [57], mais ils ont reçu des améliorations pour s'agencer aux autres techniques.

Les techniques :

1. Contrôleur ;
 2. Cron ;
 3. Exportation des données ;
 4. Géospatiale (dépend de Modèle) ;
 5. Héritage ;
 6. Hook ;
 7. Migration DB⁸ ;
 8. Modèle ;
 9. Portal ;
 10. Sécurité ;
 11. Snippet ;
 12. Thème ;
 13. Vue web ;
 - (a) Activity ;
 - (b) Calendar ;
 - (c) Diagram ;
 - (d) Form ;
 - (e) Graph ;
 - (f) Kanban ;
 - (g) Pivot ;
 - (h) Search ;
 - (i) Timeline ;
 - (j) Tree ;
 14. «website_leaflet» (dépend de Snippet et Géospatiale) ;
 15. Wizard ;
-
8. importation des données par DB

4.4.2 Interface du générateur de code

L'interface graphique

L'interface graphique existait déjà dans la version initiale [57], 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 .

Ce qui fonctionne :

1. Créer un module ;
2. Renommer un module ;
3. Ajouter des modèles, voir Annexe A, et des champs, voir Annexe B ;
4. Ajouter des menus ;
5. Ajouter la sécurité ;
6. Changer les icônes ;
7. Changer les informations sur les propriétés «manifest» du module ;
8. Ajouter du code, voir Annexe C ;
9. Modification des «hooks», voir Annexe D ;

L'interface «code hook»

L'interface «code hook» 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 test et elle contient la paramétrisation pour des 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

Voici l'ensemble résumé des accomplissements de SO-3 :

1. Ajout de nouvelles techniques et une classification de celles-ci ;
2. Rendre accessible une interface graphique pour paramétriser la génération de code ;
3. Rendre accessible une interface de programmation pour utiliser toutes les fonctionnalités du robot logiciel codeur.

SO-3 Feuille de route

Voici la brève feuille de route pour SO-3 :

1. Ajouter des paramètres pour faire davantage de personnalisation sur les techniques ;
2. Extraire les techniques du générateur et les implémenter une par module ;
3. Supporter les fonctionnalités manquantes sur toutes les techniques pour l'interface graphique ;
4. 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», le générateur de code permet l'utilisation de l'interface graphique pour générer des modules Odoo.

4.5.2 Interprétation des résultats de SO-4

SO-4 Accomplissements

Le générateur de code a été intégré dans un système de distribution via un Docker.

SO-4 Feuille de route

Voici la brève feuille de route pour SO-4 :

1. Développer une synchronisation entre les instances pour permettre de la redondance ;
2. Développer une gestion de son infrastructure via le générateur de code ;
3. Faire participer le robot logiciel codeur à la maintenance de l'infrastructure de déploiement ;
4. Utiliser d'autres systèmes de conteneur en distribution qui sont libres comme «Pod»⁹ ;
5. Développer la capacité du robot logiciel codeur de valider techniquement si le logiciel est AGPLv3 au moment de l'exécution.

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

4.6 Résultats propres à SO-5

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

Un guide hybride a été produit pour comprendre les aspects cités du démarrage d'un projet, de gestion d'une communauté autour d'un projet libre et des règles d'hébergement libres.

4.6.2 Démarrage d'un projet

Le guide du tableau 4.1 permet de démarrer rapidement un projet et s'assurer que les membres impliqués du réseau d'entraide comprennent les mêmes enjeux et s'alignent dans la même direction.

4.6.3 Intégration d'un membre

Nous avons rédigé un guide pour gérer l'intégration d'un membre en 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.

Tableau 4.1 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 figées 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.

4.6.4 Comportement en communauté

Nous avons rédigé un guide pour gérer le 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.

4.6.5 Outils de développement public

Nous avons rédigé un guide pour le 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.

4.6.6 Résolution de problème

Nous avons rédigé un guide de résolution 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 ressentiment ;
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.

4.6.7 Documentation

S'assurer que les documentations suivantes sont accessibles à la racine du 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.

4.6.8 Sécurité

La sécurité est importante pour protéger les données dans la communauté, voici 3 étapes suggérées à intégrer au projet communautaire :

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.

4.6.9 Développement libre

Voici les étapes à suivre en lien avec le développement 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'organisation¹⁰ ;
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 +¹¹ ;
 - (d) CC0¹², ne protège pas l'oeuvre ;
 - (e) CC-BY¹³ ;
 - (f) CC-BY-SA¹⁴, protège l'oeuvre¹⁵ ;
 - (g) LiLiQ-R+¹⁶ ;

4.6.10 Communication

Faire un suivi des émotions/sentiments des membres lors des réunions (hebdomadaires) par rapport au projet et mettre en place des outils favorisant la communication non violente.

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

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

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

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

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

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

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

4.6.11 Interprétation des résultats de SO-5

SO-5 Accomplissements

Voici l'ensemble résumé des accomplissements de SO-5 :

1. Test de la génération sur un module existant de la communauté nommé «auto_backup» ;
2. Des modules de gestion de projet ont été générés pour faire le suivi de la conception fonctionnelle et de l'amélioration continue ;
3. Le projet Accorderie a bénéficié du générateur de code pour la migration de la base de données vers Odoo ;
4. Le projet Portail CEPPEP 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

Voici la brève feuille de route pour SO-5 :

1. Supporter la demande de «Pull Request» sur les projets Git respectif lorsqu'il y a une amélioration. Il doit y avoir un suivi et valider le suivi des règles de contributions de la communauté ;
2. Développer d'autres modules de gestion de projet pour l'accompagnement dans le développement de projet client ;
3. Développer des modules de gestion de communauté sur des projets de logiciels libres ;
4. Développer le suivi du développement des modules communautaires avec une traçabilité sur les résultats avec des métriques de génie logiciel.

4.7 Projets divers

4.7.1 Projet module «auto_backup»

Le module «auto_backup» 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 et μ_C^B a été généré. Le générateur de code a pu modifier le module pour appliquer de la qualité logicielle. Cela a permis de développer la technique de gestion des «Cron», puisqu'il lance des sauvegardes, par SSH ou en local, à des moments spécifiques dans le temps.

4.7.2 Projet module workflow design

C'est un module de gestion de projet qui a été développé entièrement avec le générateur de code qui permet de faire le suivi sur les opportunités, les menaces, les forces, les faiblesses et les objectifs. Il a été utilisé dans le projet Accorderie.

4.7.3 Projet module STARS

Le module STARS dépend de l'application Projet, il permet de configurer une procédure intégrée dans un projet pour suivre les étapes de STARS, voir Figure 4.4.

Ainsi, nous pouvons créer un nouveau projet et suivre cette procédure en ajoutant des tâches d'amélioration continue de son organisation, voir Figure 4.5.

4.7.4 Projet module SRS

C'est un module de gestion de projet qui a été développé entièrement avec le générateur de code. Il permet de faire l'analyse des besoins pour ensuite passer à l'analyse fonctionnelle et, finalement, définir les requis fonctionnels d'un projet. Il a été utilisé, entre autres, pour le projet Accorderie et le projet Portail CEPPP.

4.8 Projet espace Accorderie

Le projet 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 leurs modèles de données basés sur leur base de données en SQL de Mariadb, voir Annexe F.

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

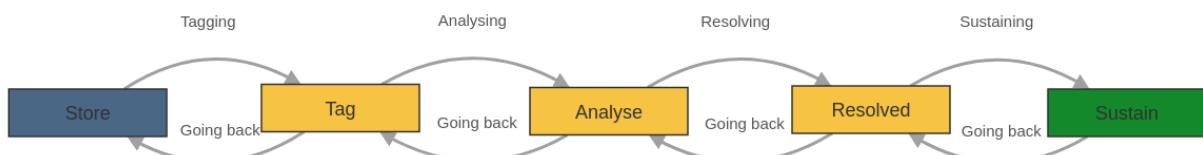


Figure 4.4 Procédure STARS dans l'application Projet vue Diagramme

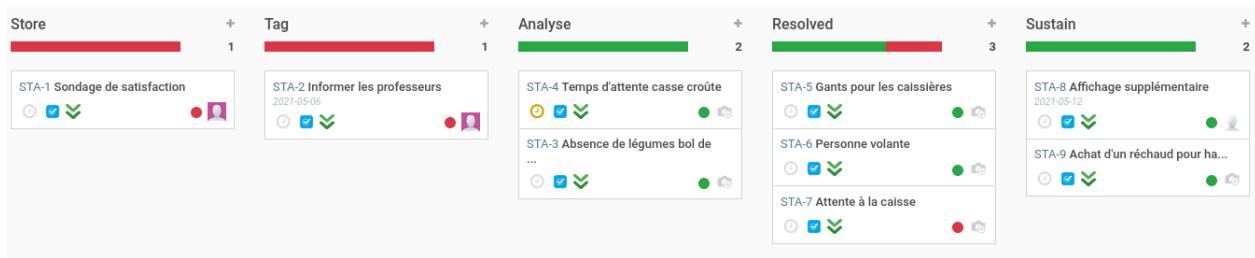


Figure 4.5 Suivi des tâches de projet avec procédure STARS en vue Kanban

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 validées par des membres de la communauté.

Dans l'Annexe G, 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 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.

Le générateur de code a permis d'aider à créer un diagramme pour afficher le processus d'échange de temps, voir Annexe H, 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.

Dû 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.

4.9 Projet Portail CEPPEP

L'objectif est de faire une section portail pour les patients et une section administrative pour les recruteurs, les partenaires et les administrateurs de la plateforme, rendre accessible des formulaires et 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 CSV et un module d'importation des données a été développé.

Extraction :

1. 23 fichiers analysés ;
2. 2851 données extraites ;
3. Capacité de faire la traduction anglais-français sur les données, mais cette fonctionnalité a été annulée puisque les données ont été modifiées dans la ré-ingénierie.

Voici les statistiques du Tableau 4.2 du code après ré-ingénierie et adaptation des fonctionnalités, livraison de la plateforme en début septembre 2023¹⁷.

Tableau 4.2 L'évolution entre la génération et la ré-ingénierie des statistiques sur les langages du portail CEPPEP

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

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.

Le modèle de données du portail CEPPEP dans Odoo 12 contient 24 modèles Annexe I. L'interface administrateur Annexe J contient la fiche du patient dont les partenaires ont accès seulement qu'à la partie anonymisée Annexe K.

17. <https://portailppp.ca>

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 ré-ingénierie, 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.

4.10 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 permettre de pouvoir s'adapter à des situations d'urgence et apporter des solutions rapidement. Les 4 libertés sont :

1. **Étudier** La rétro-ingénierie a permis au générateur de code de comprendre certaines fonctionnalités pour pouvoir recréer les méta-données adéquatement pour la reproduction ;
2. **Copier** L'auto-générateur a été mis en place, il reste à auto-reproduire le robot logiciel codeur par son module principal de générateur de code ;
3. **Modifier** Au moment d'une ré-ingénierie, 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 ;
4. **Utiliser** Le robot logiciel codeur a la capacité d'utiliser ses fonctionnalités générées et d'exécuter des scripts d'automatisation à des périodes de temps adaptables.

4.11 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. L'outil STARS est rendu accessible pour faire de l'amélioration continue au sein des organisations. 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.

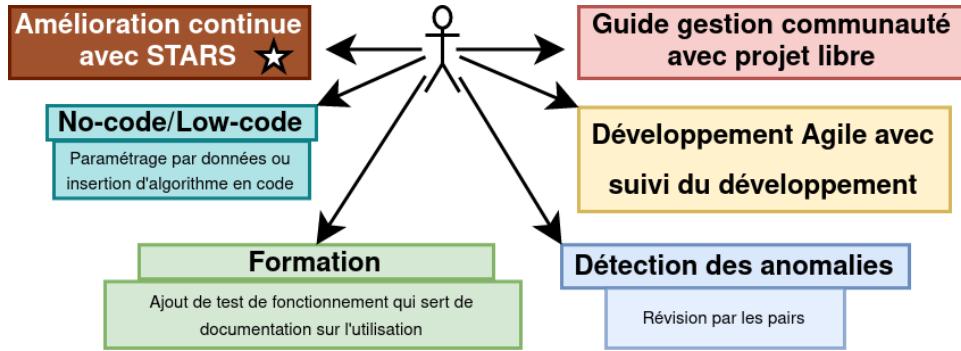


Figure 4.6 Intéraction entre les développeurs et les outils de développement dans un réseau d'entraide

4.12 Avancement de la technopoïèse

Voir Figure 4.7, la ré-ingénierie 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 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-écraser¹⁸ et perdre son travail.

18. 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.

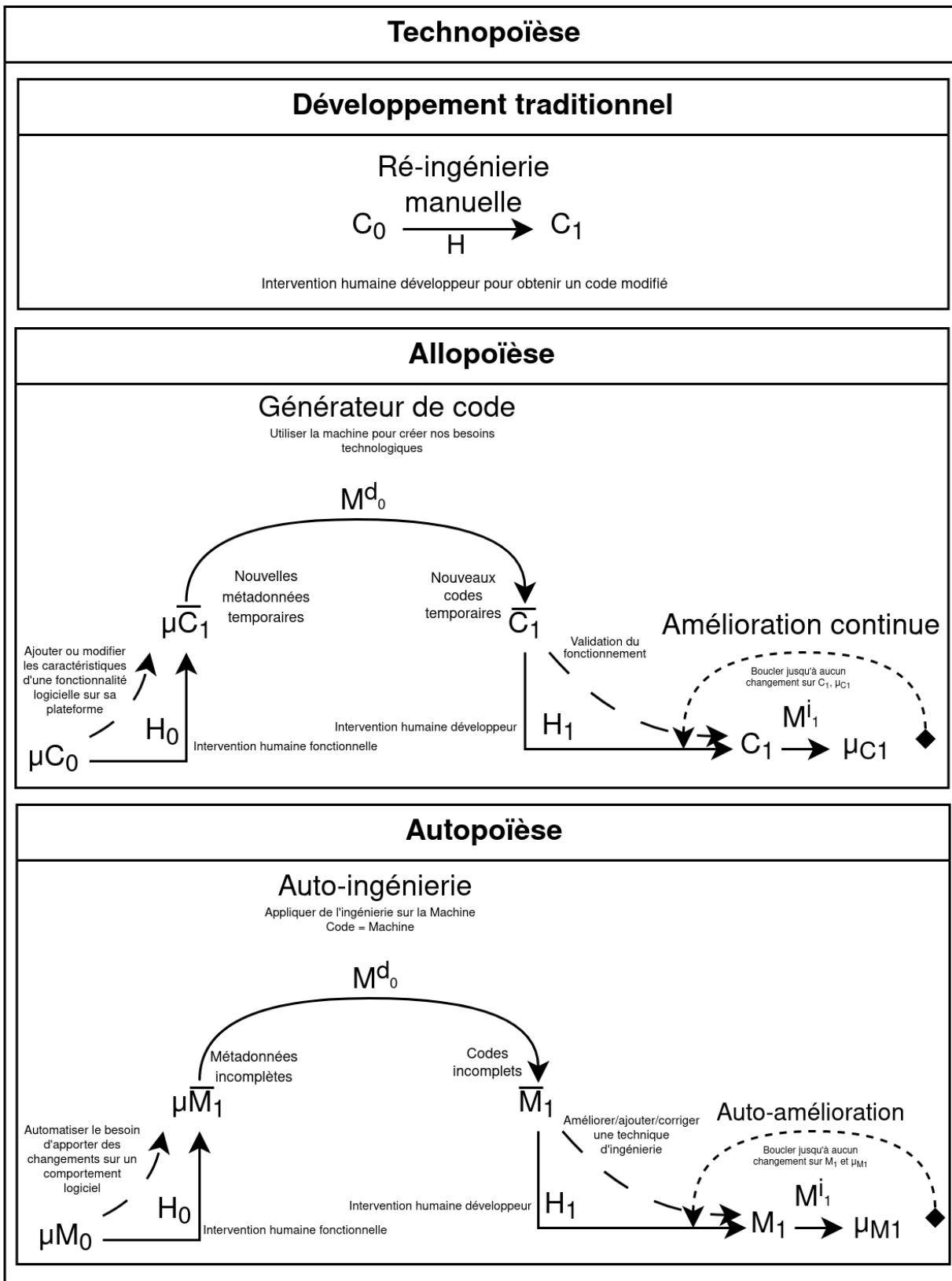


Figure 4.7 Architecture du générateur de code

4.13 Réalisation du robot logiciel générateur de code

Dans la revue littérature section 2.1, nous avons déterminé 4 critères pour définir un robot logiciel codeur :

4.13.1 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 4.12.

4.13.2 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.

4.13.3 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.

4.13.4 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 paramétriser 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.

4.13.5 Article «A template-based code generator for web applications»

Le code source de cette application «A template-based code generator for web applications» n'est pas accessible facilement au public, par conséquence, il sera difficile de comparer son efficacité du coté pratique. En effet, elle ne dispose pas de ORM, elle ne fait pas de rétro-ingénierie et elle n'a pas été dédiée pour une communauté. Cependant, elle démontre bien l'efficacité d'un générateur de code sur une solution ERP web.

CHAPITRE 5 CONCLUSION

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.

5.1.1 Générateur de code

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.

5.1.2 Gestion de communauté autour de projet libre

Pour la gestion de communauté autour de projet libre, 3 guides ont été rédigés pour permettre d’outiller le gestionnaire de communauté à intégrer des participants dans leur développement technologique. Un guide pour démarrer un projet, un guide pour gérer les participants et un guide pour les règles éthiques d’hébergement. De plus, l’outil STARS est accessible directement dans la plateforme ERP pour l’amélioration continue.

5.1.3 Projet 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.

5.1.4 Projet Portail 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 administrations et portails. Cependant, nous avons cessé d'utiliser le générateur de code suite à la ré-ingénierie, 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 sont causées, en général, par le manque de temps de terminer les développements et seulement la ligne critique a été suivie pour réaliser la liste des tâches nécessaires à la réponse des demandes-client.

5.2.1 Couverture des tests

Les tests devraient couvrir 100% du code, cependant la couverture est de 84% pour 3 raisons :

1. Il y a du code fonctionnel non testé, il manque des tests ;
2. Il y a du code désuet qu'il faut nettoyer ou refactoriser ;
3. 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.

5.2.2 Personnalisation du développement

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

5.2.3 Auto-reproduction du générateur de code

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.

5.2.4 Extraction de code et de modèles de base de données

L'extraction de code PHP, Javascript ou de modèles de base de données a été appliqué sur des contextes particuliers tels que le projet Accorderie et CEPPEP. 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 NLP et le projet de support de développement de module dans la communauté Odoo.

5.3.1 Amélioration du générateur de code

Amélioration des instances clients

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 leurs retours d'expérience et leurs habitudes d'utilisation.

Amélioration de l'auto-génération

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. 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.

Amélioration de l'architecture

Voici un survol des améliorations de l'architecture possibles :

1. Parallélisation de tout le code, en tout temps, lorsque cela est possible ;

2. Automatisation de la configuration pour le déverminage ;
3. Automatiser la détection des anomalies ;
4. Améliorer l'interface LCNC pour pouvoir accomplir les mêmes étapes que le mode de paramétrisation «Code hook» ;
5. Supporter de nouvelles architectures dans la génération de code comme :
 - (a) des applications Cordova pour le support mobile natif ;
 - (b) 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 ;
 - (c) d'autres modules sur différents systèmes ERP tels que Tryton ou NextERP ;
 - (d) générer des scripts de développement dans le projet ERPLibre qui ne dépendent pas d'Odoo ;
 - (e) supporter des applications embarquées pour contrôler des automates ou des systèmes de production automatisée ;
6. Implémenter de l'auto-ingénierie sur le développement des techniques du générateur de code ;
7. Génération de code par requis fonctionnel.

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

Amélioration de la gestion de projet et statistiques

Le générateur de code doit offrir des outils de gestionnaire de projet pour suivre le développement, faire la liaison entre les demandes clients et les avancements des développeurs, le suivi de la ré-ingénierie des processus d'affaires de l'entreprise, etc. 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 métadonnées 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.

Suite du développement du générateur de code

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 sa 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 par différentes version de la plateforme.

5.3.2 Projet Accorderie

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.

5.3.3 Projet Portail CEPPP

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 est 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 développeur.

5.3.4 NLP

La technologie NLP 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 interfaçer 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.ummtto.dz/dspace/handle/ummtto/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] A. Sarkar, “Quines are the fittest programs : Nesting algorithmic probability converges to constructors,” 2020. [En ligne]. Disponible : <https://arxiv.org/abs/2010.09646>

- [13] Wikipédia. (2022, 4 septembre) Quine (informatique). [En ligne]. Disponible : [https://fr.wikipedia.org/wiki/Quine_\(informatique\)](https://fr.wikipedia.org/wiki/Quine_(informatique))
- [14] ——. (2022, 10 septembre) Rétro-ingénierie en informatique. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique
- [15] 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
- [16] G. van Rossum, B. Warsaw et N. Coglan. (2023, 25 février) Guide de style pour le code python. [En ligne]. Disponible : <https://peps.python.org/pep-0008/>
- [17] Wikipédia. (2023, 13 mars) Liste de tests informatiques. [En ligne]. Disponible : https://en.wikipedia.org/wiki/Software_testing
- [18] O. québécois de la langue française. (2018) robot logiciel. [En ligne]. Disponible : <https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/8350665/robot-logiciel>
- [19] L. Erlenkov *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>
- [20] 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>
- [21] 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.
- [22] 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>
- [23] S. Pichidtienthum, P. Pugsee et N. Cooharojananone, “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>
- [24] 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>
 - [25] 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>
 - [26] 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
 - [27] 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/>
 - [28] R.-H. Pfeiffer, “License incompatibilities in software ecosystems,” 2022. [En ligne]. Disponible : <https://arxiv.org/abs/2203.01634>
 - [29] 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>
 - [30] 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>
 - [31] 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>
 - [32] 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>

- [33] O. BRAUD, “Facteurs decisionnels pour l’implantation d’un erp dans les pme : Le role de l’evaluation 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>
- [34] 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.
- [35] 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>
- [36] J. Pease. (2017, 9 juin) Devops part 1 : It’s more than teams. Contegix. [En ligne]. Disponible : <https://www.contegix.com/devops-part1-its-more-than-teams/>
- [37] 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>
- [38] 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>
- [39] 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
- [40] 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>
- [41] M. Rosenberg et D. Chopra, *Nonviolent Communication : A Language of Life : Life-Changing Tools for Healthy Relationships*, ser. Nonviolent Communication Guides. PuddleDancer Press, 2015. [En ligne]. Disponible : <https://books.google.ca/books?id=A3qACgAAQBAJ>
- [42] C. Portail. (2020, 20 avril) Osbd en cnv. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:OSBD_en_CNV.jpg
- [43] (2023, 24 mars) Building community. [En ligne]. Disponible : <https://opensource.guide/fr/building-community/>
- [44] “Faire une super Équipe engagée,” CimarLab à la Polytechnique Montréal, 24 mars 2023. [En ligne]. Disponible : <https://www.polymtl.ca/labac/>

- [45] 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>
- [46] 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>
- [47] Wiktionary. (2021, 1 août) Poïèse. [En ligne]. Disponible : <https://fr.wiktionary.org/wiki/-poïèse>
- [48] Wikipédia. (2022, 28 novembre) Poïétique. [En ligne]. Disponible : <https://fr.wikipedia.org/wiki/Poïétique>
- [49] 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
- [50] Wikipedia. (2018, 28 décembre) Allopoiesis. [En ligne]. Disponible : <https://en.wikipedia.org/wiki/Allopoiesis>
- [51] P. C. WEB. (2007) Allopoiesis. [En ligne]. Disponible : <http://pespmc1.vub.ac.be/ASC/ALLOPOIESIS.html>
- [52] Wikipedia. (2022, 30 octobre) Autopoïèse. [En ligne]. Disponible : <https://fr.wikipedia.org/wiki/Autopoiese>
- [53] 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
- [54] P. Guillibert, “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>
- [55] 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>
- [56] 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>

- [57] 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 GUI GÉNÉRATEUR DE CODE - LES MODÈLES

Code Generator Modules Advance Settings Administrator (code_generator) ▾

Modules / Code Generator

Modifier + Créer Action ▾ 1 / 1 ⌘ ⌘ ⌘

 Technical Name code_generator

Code Generator

By Mathben (mathben@technolibre.ca)

Views Models Controllers

Information Technical Data Elements Hook

Groups Models ACLs Rules SQL Constrains Server Constrains Views Action Windows Action Servers Menus Reports

Modèle	Description du Modèle	Type	Modèle transitoire
ir.actions.server	Action du serveur	Objet de base	<input type="checkbox"/>
ir.actions.todo	Assistants de configuration	Objet de base	<input type="checkbox"/>
ir.model.fields	Champs	Objet de base	<input type="checkbox"/>
code.generator.act_window	Code Generator Act Window	Objet de base	<input type="checkbox"/>
code.generator.add.controller.wizard	Code Generator Add Controller Wizard	Objet de base	<input checked="" type="checkbox"/>
code.generator.ir.model.fields	Code Generator Fields	Objet de base	<input type="checkbox"/>
code.generator.generate.views.wizard	Code Generator Generate Views Wizard	Objet de base	<input checked="" type="checkbox"/>
code.generator.menu	Code Generator Menu	Objet de base	<input type="checkbox"/>
ir.model.server_constraint	Code Generator Model Server Constrains	Objet de base	<input type="checkbox"/>
code.generator.add.model.wizard	Code Generator Model Wizard	Objet de base	<input checked="" type="checkbox"/>
code.generator.module	Code Generator Module	Objet de base	<input type="checkbox"/>
code.generator.module.dependency	Code Generator Module Dependency	Objet de base	<input type="checkbox"/>
code.generator.module.external.dependency	Code Generator Module External Dependency	Objet de base	<input type="checkbox"/>

Go to Frontend ⋮

ANNEXE B GUI GÉNÉRATEUR DE CODE - LES CHAMPS

Code Generator Modules Advance Settings Administrator (code_generator) -

Ouvrir : O2M Models

Description du Modèle	Code Generator Writer	Type	Objet de base
Modèle	code.generator.writer	Dans Applications	code_generator, code_generator_hook
Modèle transitoire	<input type="checkbox"/>		
Fil de discussion	<input type="checkbox"/>		

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

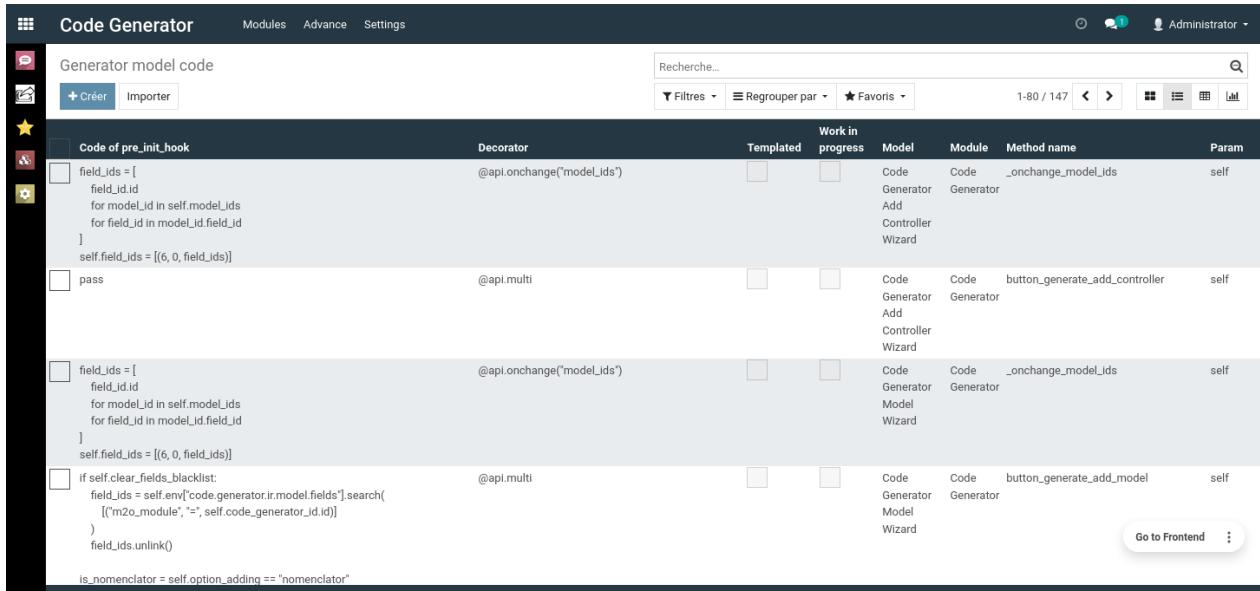
Nom de Champ	Étiquette de Champ	Type de Champ	Requis	Lecture seule	Indexé	Type
_last_update	Last Modified on	date/heure	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Champ de base
basename	Base name	caractère	<input type="checkbox"/>	<input type="checkbox"/>	<input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

Créer un menu

Fermer

ir.module.module Module Objet de base

ANNEXE C GUI GÉNÉRATEUR DE CODE - LES CODES



The screenshot shows the Odoo Code Generator interface. The left sidebar has icons for Home, Model, Advanced, and Settings. The main area is titled "Generator model code" with buttons for "+Créer" and "Importer". A search bar at the top right contains "Recherche...". Below it are filters for "Filtres", "Regrouper par", and "Favoris", along with a page number "1-80 / 147" and navigation arrows. The main content is a table with the following columns: "Code of pre_init_hook", "Decorator", "Templated", "Work in progress", "Model", "Module", "Method name", and "Param". There are four entries in the table:

Code of pre_init_hook	Decorator	Templated	Work in progress	Model	Module	Method name	Param
<pre>field_ids = [field_id.id for model_id in self.model_ids for field_id in model_id.field_ids] self.field_ids = [(6, 0, field_ids)]</pre>	@api.onchange("model_ids")	<input type="checkbox"/>	<input type="checkbox"/>	Code Generator	Code Generator	_onchange_model_ids	self
<pre>pass</pre>	@api.multi	<input type="checkbox"/>	<input type="checkbox"/>	Code Generator	Code Generator	button_generate_add_controller	self
<pre>field_ids = [field_id.id for model_id in self.model_ids for field_id in model_id.field_ids] self.field_ids = [(6, 0, field_ids)]</pre>	@api.onchange("model_ids")	<input type="checkbox"/>	<input type="checkbox"/>	Code Generator	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	<input type="checkbox"/>	<input type="checkbox"/>	Code Generator	Code Generator	button_generate_add_model	self
<pre>is_nomenclator = self.option_adding == "nomenclator"</pre>							

A "Go to Frontend" button is located at the bottom right of the table.

ANNEXE D GUI GÉNÉRATEUR DE CODE - LES «HOOKS»

The screenshot shows the Odoo Code Generator module interface. The top navigation bar includes 'Modules', a message center with 1 notification, and the user 'Administrator (code_generator)'. The main area displays the 'Code Generator' module details:

- Technical Name:** code_generator
- By:** Mathben (mathben@technolibre.ca)
- Views Models Controllers:** Views (selected), Models, Controllers

The 'Hook' tab is selected, showing the following code snippets:

pre_init_hook

```
Show pre_init_hook | 
```

```
if not tools.config["dev_mode"]:
    raise Exception(
        _(
            "Cancel installation module code_generator, please specify"
            " - -dev [options] in your instance."
        )
)
```

post_init_hook

```
Show post_init_hook | 
```

<pre>if not tools.config["dev_mode"]: raise Exception(_("Cancel installation module code_generator, please specify" " - -dev [options] in your instance."))</pre>	Feature <input type="checkbox"/> General conf <input checked="" type="checkbox"/> post_init_hook <input type="checkbox"/> Code generator <input checked="" type="checkbox"/> post_init_hook
---	--

```
with api.Environment.manage():
    env = api.Environment(cr, SUPERUSER_ID, {})
    system_user = env["res.users"].browse(2)
    system_user.groups_id = [
        (4, env.ref("code_generator.code_generator_manager").id, False)
    ]
```

uninstall_hook

```
Show uninstall_hook | 
```

Extra

```
MODULE_NAME = "code_generator"
```

Go to Frontend :

ANNEXE E TEST COUVERTURE TECHNIQUE GÉNÉRATEUR DE CODE

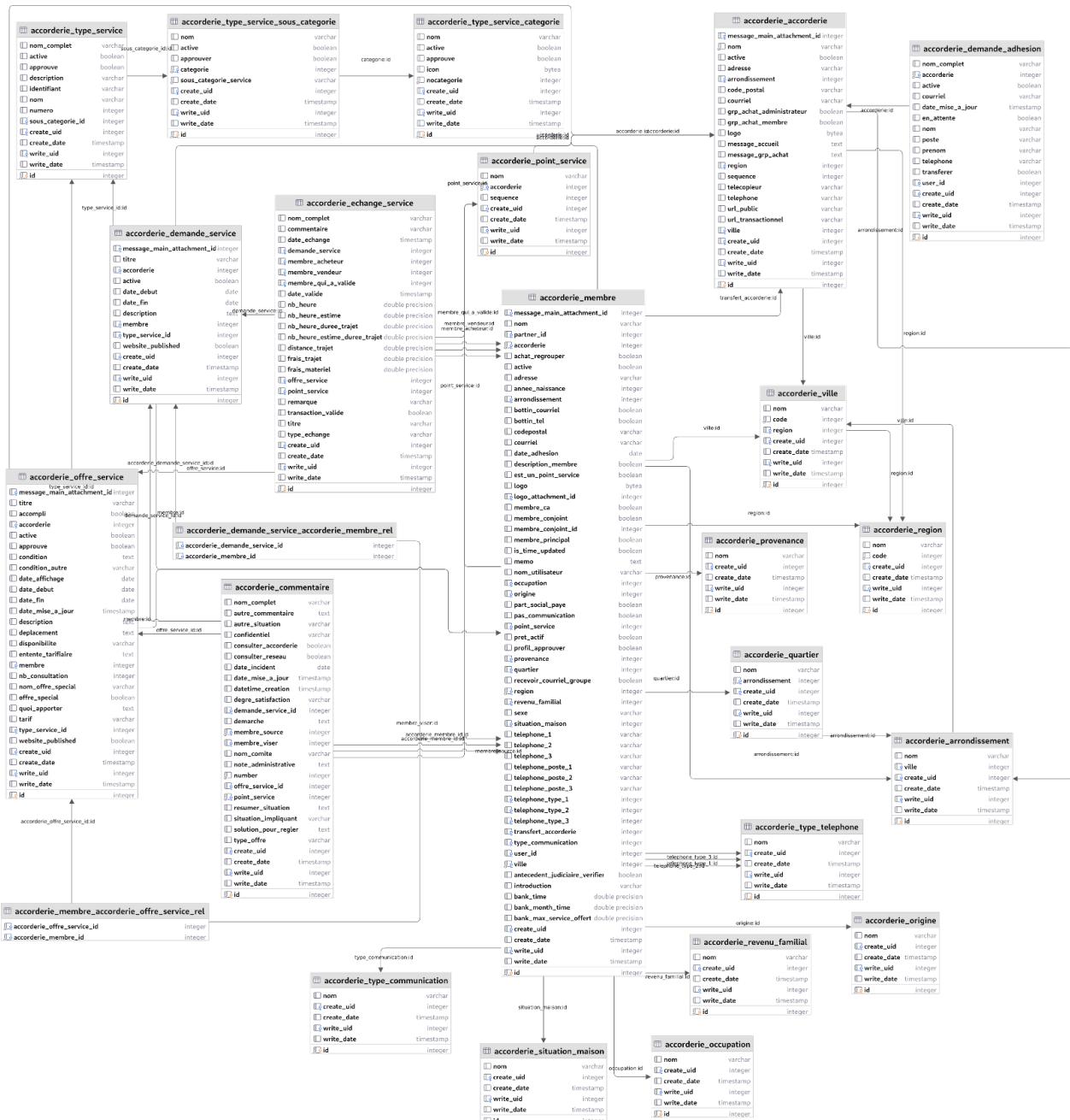
Technique	base	# instruction		5 085
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Génération μ_C^B modèle simple	Succès	27	2 983	41
Génération μ_C^B modèle simple avec héritage	Succès	26	3 452	32
Exportation μ_C^B données «helpdesk»	Succès	28	3 900	23
Technique	base + hook	# instruction		5 985
Description du test	Status	Durée (s)	# de Miss	Cover (%)
Auto-génération μ_C^0	Succès	17	4 683	22
Nouveau projet $\mu_C^0 \mu_C^A \mu_C^B$ «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

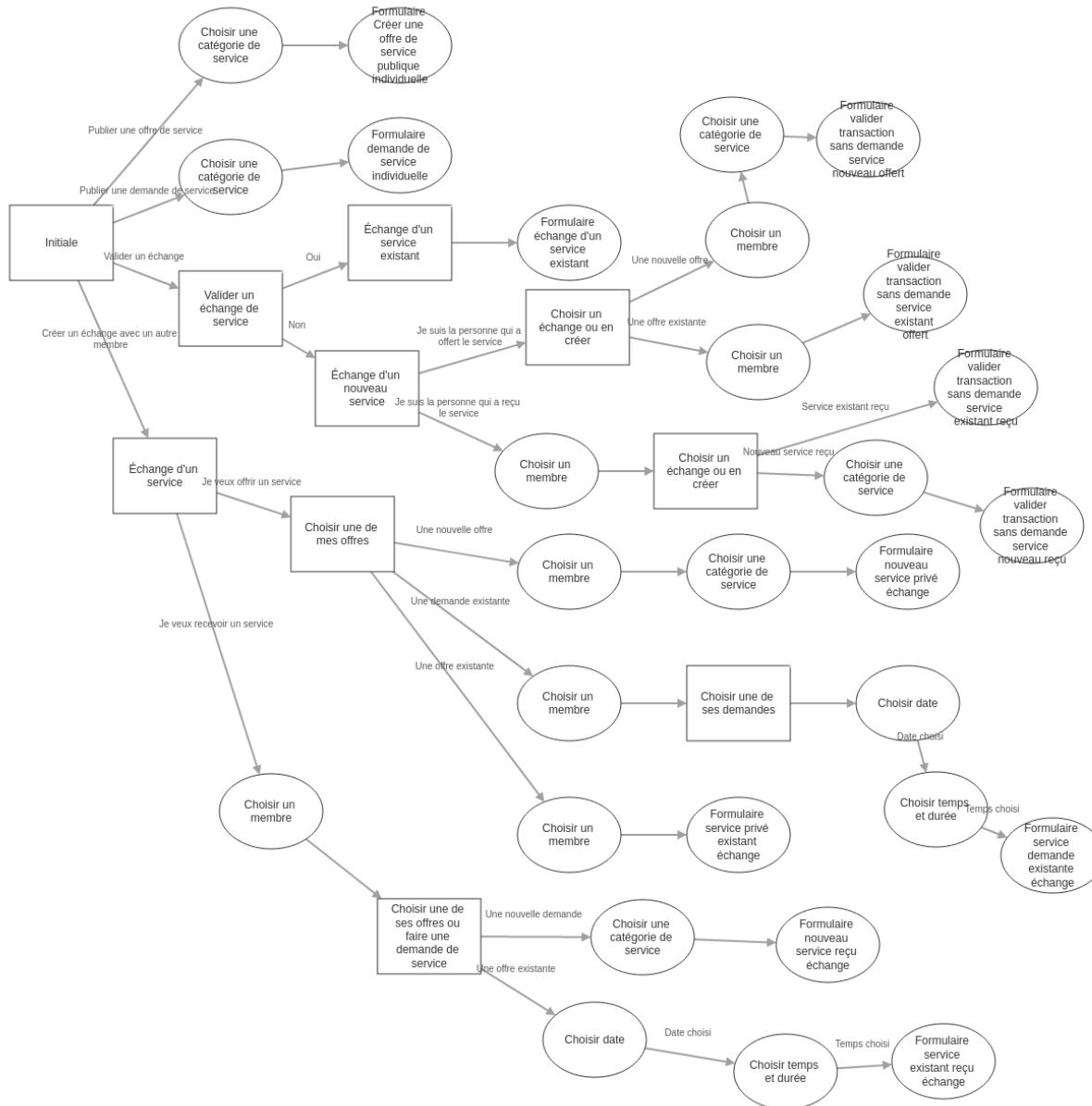
ANNEXE F DIAGRAMME MODÈLE DE DONNÉES ESPACE MEMBRE ACCORDERIE 2019



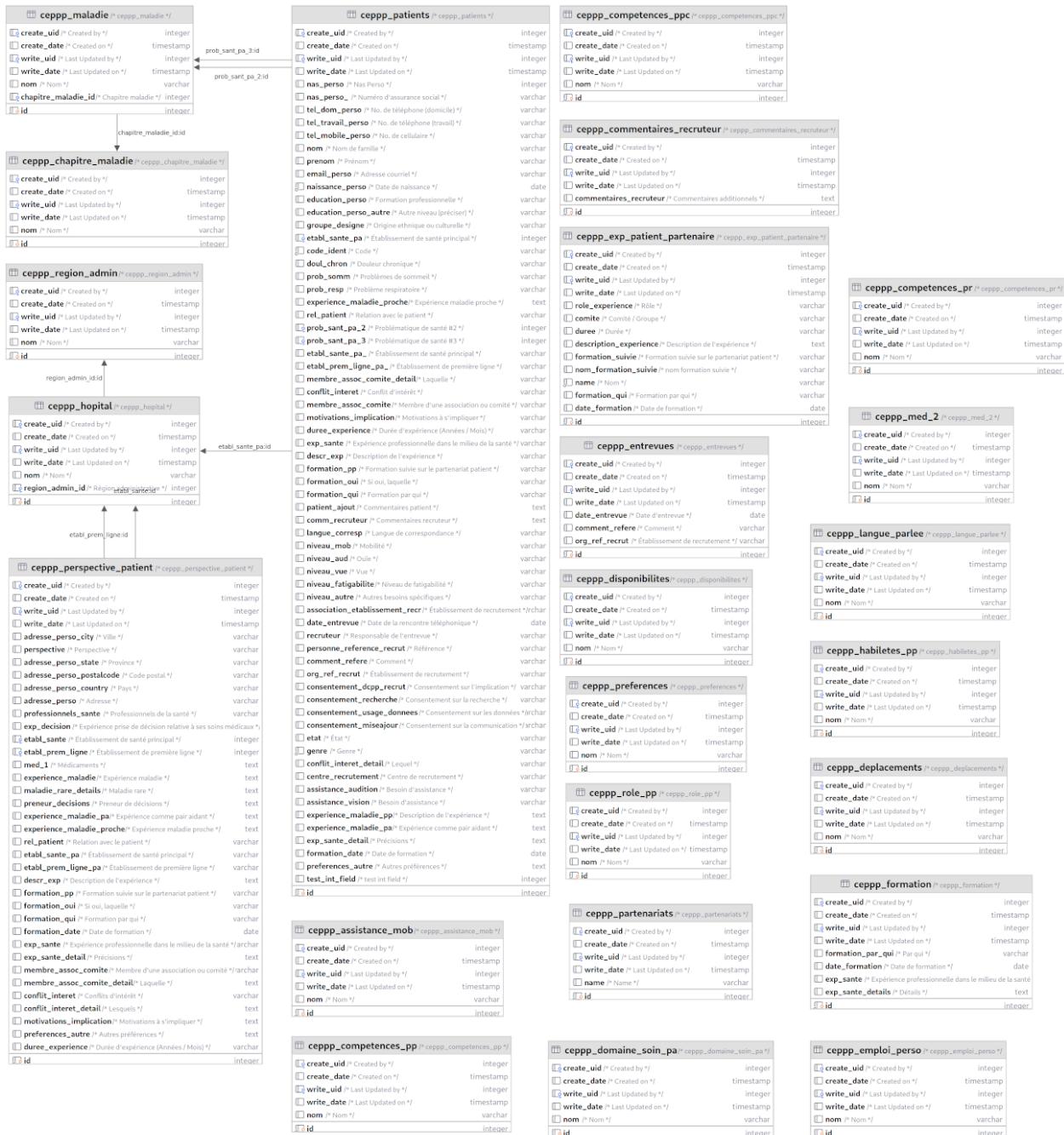
ANNEXE G DIAGRAMME NOUVEAU MODÈLE DE DONNÉES ESPACE MEMBRE ACCORDERIE 2023



ANNEXE H DIAGRAMME PROCESSUS POUR DEMANDER, OFFRIR, ÉTABLIR UN ÉCHANGE ET LE VALIDER - ACCORDERIE 2023



ANNEXE I DIAGRAMME MODÈLE DE DONNÉES DU PORTAIL CEPPP SEPTEMBRE 2022



ANNEXE J VUE FORMULAIRE ADMINISTRATION PORTAIL CEPPP

Ceppp Patient Partenaire

Patient Configuration

Mathieu Benoit ▾

Les patients de tous les centres de recrutement / Monique Beauvoir
[Modifier](#) [+ Crée](#)
Action ▾
1 / 3



Monique Beauvoir

Activé e37f67dc Contact

Patient	Monique Beauvoir
Recruteur	Stéphanie Latendresse
Centre de recrutement	CEPPP

[Infos](#) [Consentements](#) [Compétences](#) [Commentaires](#) [Disponibilité](#) [Savoirs expérientiels](#) [Formations](#) [Implications](#)

Description

Date de naissance	1980-08-07
Âge	42
Sexe	Femme
Genre	
Héritage culturel	
Langues parlées/écrites	(Anglais) Autre
Autre langues parlées/écrites	

Préférence

Mode de communication privilégié
Patient actif-passif

Coordonnée

Téléphone	(514) 555-5555
Cellulaire	
Courriel	monique_bauvoir@exemple.ca
Adresse	4545, Avenue Pierre-De Coubertin Montréal Quebec (CA) H1V 3N7 Canada

Autre

Occupation

Envoyer un message
Enregistrer une note
Planifier une activité
0
Suivre
1 ▾

Aujourd'hui

 Note écrite par Bot - il y a une minute

- Nom: Monique Beauvoir
- Actif: true
- Recruteur: Stéphanie Latendresse
- Consentement aux notifications/communications: true
- Consentement au recrutement: true

ANNEXE K VUE FORMULAIRE PARTENAIRE PORTAIL CEPPP

The screenshot shows a web-based application interface for 'Cepp Patient Partenaire'. The top navigation bar includes links for 'Patient' and 'Configuration', and a user profile for 'Mathieu Benoit'. The main content area displays a patient record for 'Monique Beauvoir' with the ID '8796efd1-824d-4a1f-8133-ac10bf5913f9'. The record details include:

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

Below the details, there are four tabs: 'Disponibilité' (selected), 'Savoirs expérientiels', 'Formations', and 'Implications'. A link 'Fiche recruteur' is visible in the top right corner of the main content area. On the left side, there is a vertical sidebar with several icons.

ANNEXE L 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()

```