HEIA-FR

ISC-IL-3

Cahier des charges

Plugin IntelliJ pour GnuProlog

Erwan Sturzenegger

13/10/2022

Professeur : Frédéric Bapst

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Table des matières

1 Introduction

Ce document est le cahier des charges du projet « Plugin IntelliJ pour GnuProlog ». Il contient une description du contexte actuel, des objectifs, des technologies utilisées, du planning de développement et des tâches à réaliser.

2 Contexte du projet

Dans le cadre d'un projet de semestre (printemps 2018), un plugin IntelliJ-IDEA a été réalisé, permettant ainsi d'aider les étudiants lors du développement de programmes Gnu-Prolog. Actuellement, le plugin est employé en 3ème année pour le cours de programmation logique. Bien que fonctionnel, il n'est pas parfait et n'offre pas certaines fonctions qui seraient très utiles. En outre, un bug ne permet pas aux utilisateurs du plugin travaillant sur Windows de lancer l'exécution du programme directement dans le terminal de l'IDE.

3 Objectifs principaux

3.1 Corrections des bugs

A ce jour, il existe différents bugs notamment liés au fait que le plugin doit être porté sur trois différentes plateformes, à savoir : Windows, macOS et Linux.

Sous Windows, il est impossible d'exécuter les scripts Prolog dans le terminal de l'IDE sans provoquer une erreur.

Lors du développement, le plugin génère des exceptions dues à l'utilisation de méthodes dépréciées dans la nouvelle version du SDK.

Il faudra chercher et reproduire les bugs afin de les identifier clairement et de les corriger.

3.2 Créer un pipeline de test

Actuellement, il n'y a aucun test automatique lors de la publication du code sur le Gitlab. Le but est de mettre en place un pipeline qui exécutera des tests unitaires afin de garantir le bon fonctionnement du code. Il est aussi imaginable de créer des « Release » du code directement à l'aide du CI/CD.

3.3 Permettre la navigation

Ajouter une action (par exemple Ctrl+Clic) pour se rendre directement à la première définition d'un prédicat. Cela permet un gain de temps en permettant de naviguer facilement sans devoir chercher la première occurrence en défilant dans tout le fichier.

3.4 Ajouter une possibilité de refactoring

Le refactoring est un outil précieux pour les programmeurs. Il permet de facilement déplacer des fichiers, renommer des variables ou des méthodes/fonctions tout en laissant gérer les mises à jour

dans le code par l'IDE. Ceci permet notamment d'éviter de nombreuses erreurs liées à un oubli lors du renommage.

3.5 Autocomplétion

L'autocomplétion permet un gain de temps en permettant au programmeur de simplement accepter une proposition de l'IDE plutôt que devoir écrire le prédicat en entier. Les erreurs sont aussi réduites en proposant non seulement le nom mais aussi la syntaxe du prédicat visé.

4 Objectifs secondaires

Les objectifs secondaires ne sont pas forcément réalisables dans le temps imparti.

4.1 Mettre en place un « Debugger »

Pour le moment, rien ne permet de suivre le déroulement du programme Prolog « pas à pas ». Il serait judicieux d'y ajouter un « debugger » qui utiliserait les SpyPoint[1] de GnuProlog.

4.2 Afficher les erreurs / warnings

Pour le moment, les erreurs sont affichées dans l'IDE uniquement s'il s'agit d'erreurs de syntaxe. Il serait donc bien d'avoir un retour en temps réel des erreurs de compilation ainsi que des warnings.

Pour ce faire, la compilation en arrière-plan sera utilisée.

4.3 Formatage du code

La mise en page du code étant primordiale à la bonne lecture et à la compréhension de celui-ci, un outil permettant de le formatter instantanément serait très utile.

Afin d'intégrer celui-ci au mieux dans l'univers de Jetbrains, cette fonctionnalité serait accessible depuis le menu « code » et avec un raccourci clavier (Ctrl + Clic).

4.4 Déployer le projet

Le projet est pour le moment interne à l'école d'ingénieur. Le but de cet objectif est de le rendre disponible pour tous les utilisateurs de cet IDE en le déployant sur la marketplace [2]de Jetbrains.

Il serait aussi également intéressant faire une demande pour ajouter le plugin à la liste des contributions[3] de GnuProlog.

[1]Pour plus d'information, voir http://www.gprolog.org/manual/html_node/gprolog013.html#Running-and-stopping-the-debugger
[2] Lien vers la marketplace : https://plugins.jetbrains.com/idea_ce
[3] Contribution à GnuProlog : http://www.gprolog.org/#contribs

5 Technologies

Pour ce projet, l'IDE utilisé sera IntelliJ-IDEA Ultimate Edition qui permet de développer facilement des plugins aussi bien pour lui-même que pour la version Community.

Les plugins seront installés dans une instance d'IntelliJ-IDEA Community séparée permettant de les recharger facilement tout au long du développement.

Pour les langages de programmation, le Java et le Kotlin seront employés dans l'intégralité du projet.

6 Tâches

Voici la liste des tâches détaillées en fonction des objectifs précédemment fixés :

6.1 Analyse

Dans un premier temps, connaître le fonctionnement du plugin existant est primordial pour pouvoir corriger les erreurs de manière fiable et efficace.

Pour ce faire, il faudra lire la documentation du plugin qui détaille très bien les diverses fonctionnalités qui ont été mises en place.

6.2 Implémentation

Ce chapitre concerne les tâches qui seront réalisées lors de l'implémentation.

6.2.1 Corrections des bugs

- Trouver les éventuels bugs existants et non répertoriés
- Analyser la cause des bugs
- Corriger les bugs dans la mesure du possible
- Documenter afin de mieux les identifier s'ils ressurgissent par la suite

6.2.2 Pipeline

- Rechercher de la documentation sur la mise en place des tests pour les plugins dans la documentation des plugins Jetbrains
- Créer quelques tests unitaires pour valider le fonctionnement
- Mettre en place le pipeline avec l'exécution des tests unitaires
- Créer des tests unitaires pour les fonctionnalités existantes

6.2.3 Navigation

- Rechercher de la documentation sur la navigation dans la documentation des plugins Jetbrains
- Analyser et comprendre le fonctionnement de la navigation
- Réaliser l'implémentation de la navigation
- Tester sur Windows / macOS / Linux

6.2.4 Refactoring

- Rechercher de la documentation sur le refactoring (notamment le renommage) dans la documentation des plugins Jetbrains
- Analyser et comprendre le fonctionnement du refactoring et des étapes pour sa mise en place

- Réaliser l'implémentation du refactoring
- Tester sur Windows / macOS / Linux

6.2.5 Autocomplétion

- Rechercher de la documentation sur l'autocomplétion dans la documentation des plugins Jetbrains
- Analyse et comprendre le fonctionnement
- Réaliser l'implémentation de l'autocomplétion
    - Créer une liste des mots supportés
    - Gérer les noms de prédicats de manière dynamique
- Tester sur Windows / macOS / Linux

6.2.6 Formatage du code

- Rechercher de la documentation sur le formatage de code dans la documentation des plugins Jetbrains
- Analyser et comprendre le fonctionnement du formatage de code
- Définir le formatage choisi
- Réaliser l'implémentation du formateur de code
- Tester sur Windows / macOS / Linux

6.2.7 Erreurs et warnings

- Rechercher de la documentation sur le formatage de code dans la documentation des plugins Jetbrains
- Analyser et comprendre le fonctionnement de l'ajout d'erreurs et de warnings
- Étudier une solution pour compiler en arrière-plan et récupérer les erreurs de compilation
- Réaliser l'implémentation
- Tester sur Windows / macOS / Linux

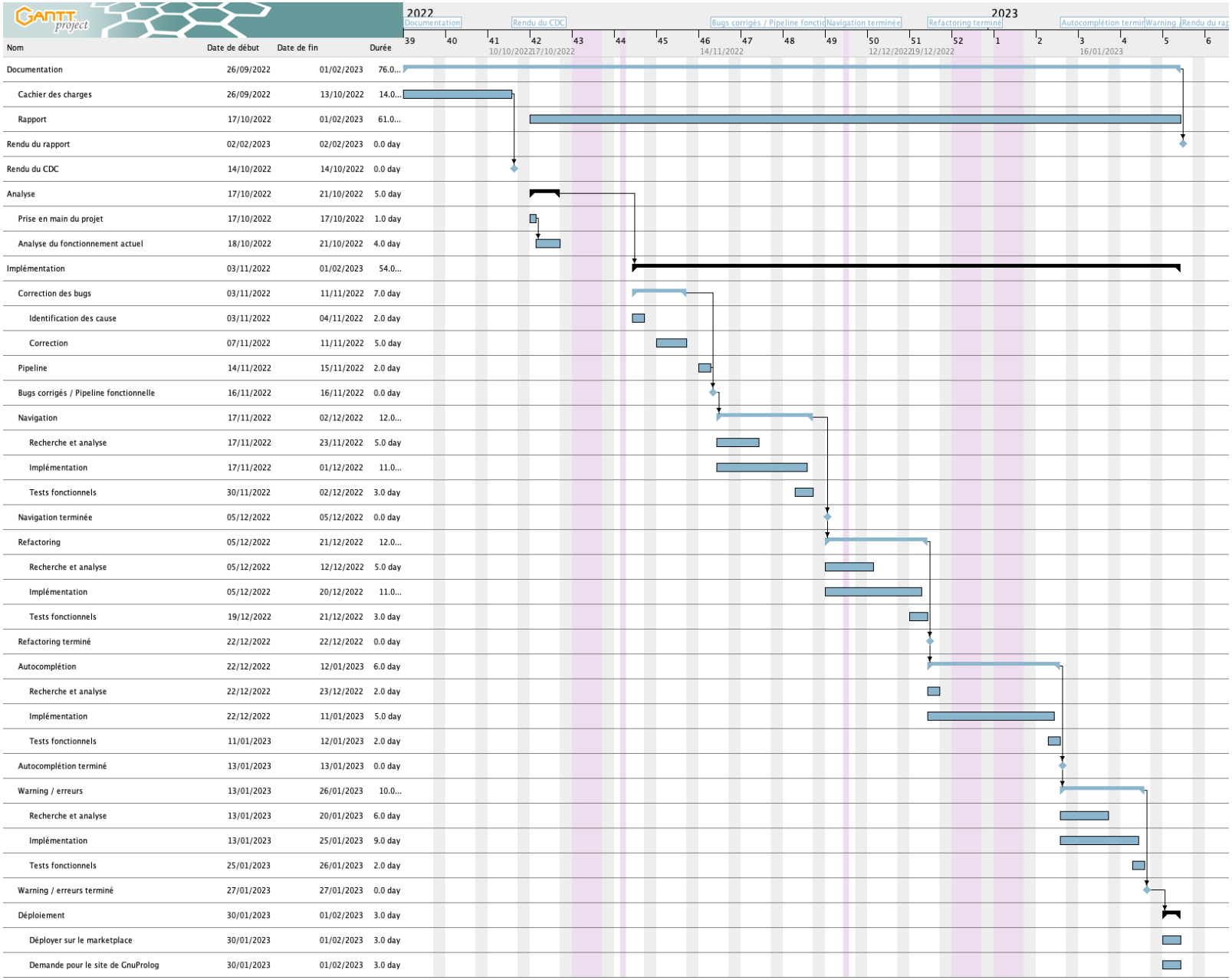
6.2.8 Déploiement sur les plateformes

- Se renseigner sur la marche à suivre pour déployer sur la marketplace
- Faire une demande pour ajouter le plugin sur le site de GnuProlog
    - Éventuellement faire un repo Git public sur Github

6.3 Documentation

- Tenir à jour le rapport tout au long du développement
- Créer une documentation utilisateur

7 Planification – Diagramme de Gantt



Nom	Date de début	Date de fin	Durée
Documentation	26/09/2022	01/02/2023	76.0...
Cachier des charges	26/09/2022	13/10/2022	14.0...
Rapport	17/10/2022	01/02/2023	61.0...
Rendu du rapport	02/02/2023	02/02/2023	0.0 day
Rendu du CDC	14/10/2022	14/10/2022	0.0 day
Analyse	17/10/2022	21/10/2022	5.0 day
Prise en main du projet	17/10/2022	17/10/2022	1.0 day
Analyse du fonctionnement actuel	18/10/2022	21/10/2022	4.0 day
Implémentation	03/11/2022	01/02/2023	54.0...
Correction des bugs	03/11/2022	11/11/2022	7.0 day
Identification des cause	03/11/2022	04/11/2022	2.0 day
Correction	07/11/2022	11/11/2022	5.0 day
Pipeline	14/11/2022	15/11/2022	2.0 day
Bugs corrigés / Pipeline fonctionnelle	16/11/2022	16/11/2022	0.0 day
Navigation	17/11/2022	02/12/2022	12.0...
Recherche et analyse	17/11/2022	23/11/2022	5.0 day
Implémentation	17/11/2022	01/12/2022	11.0...
Tests fonctionnels	30/11/2022	02/12/2022	3.0 day
Navigation terminée	05/12/2022	05/12/2022	0.0 day
Refactoring	05/12/2022	21/12/2022	12.0...
Recherche et analyse	05/12/2022	12/12/2022	5.0 day
Implémentation	05/12/2022	20/12/2022	11.0...
Tests fonctionnels	19/12/2022	21/12/2022	3.0 day
Refactoring terminé	22/12/2022	22/12/2022	0.0 day
Autocomplétion	22/12/2022	12/01/2023	6.0 day
Recherche et analyse	22/12/2022	23/12/2022	2.0 day
Implémentation	22/12/2022	11/01/2023	5.0 day
Tests fonctionnels	11/01/2023	12/01/2023	2.0 day
Autocomplétion terminé	13/01/2023	13/01/2023	0.0 day
Warning / erreurs	13/01/2023	26/01/2023	10.0...
Recherche et analyse	13/01/2023	20/01/2023	6.0 day
Implémentation	13/01/2023	25/01/2023	9.0 day
Tests fonctionnels	25/01/2023	26/01/2023	2.0 day
Warning / erreurs terminé	27/01/2023	27/01/2023	0.0 day
Déploiement	30/01/2023	01/02/2023	3.0 day
Déployer sur le marketplace	30/01/2023	01/02/2023	3.0 day
Demande pour le site de GnuProlog	30/01/2023	01/02/2023	3.0 day

8 Glossaire

GnuProlog : Compilateur pour le langage de programmation logique « Prolog »

IDE : Environnement de développement

IntelliJ-IDEA : environnement de développement intégré destiné au développement de logiciels

SDK : ensemble d'outils logiciels destinés aux développeurs, facilitant le développement d'un logiciel sur une plateforme donnée.

Jetbrains : Éditeur de logiciels pour développeur dont IntelliJ-IDEA.

9 Références

Documentation de Jetbrains :

- https://plugins.jetbrains.com/docs/intellij/custom-language-support.html, consulté le 10.10.2022
- https://plugins.jetbrains.com/docs/intellij/testing-plugins.html, consulté le 10.10.2022