

# Étude pratique : Amélioration de la complétion automatique de L<sup>A</sup>T<sub>E</sub>Xila

Axel Caro

François Boschet

Maximilien Richer

2014-2015

# Contents

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introduction</b>   | <b>2</b> |
| 1.1      | Latexila . . . . .  | 2        |
| <b>2</b> | <b>Étude pratique</b>                                       | <b>3</b> |
| 2.1      | Tâche à réaliser . . . . .                                  | 3        |
| 2.1.1    | La complétion dans LaTeXila 2.2 . . . . .                   | 3        |
| 2.1.2    | Une complétion dynamique . . . . .                          | 3        |
| <b>3</b> | <b>Réalisation</b>  | <b>4</b> |
| 3.1      | Intégration d'une recherche au document courant . . . . .   | 4        |
| 3.1.1    | Invite de complétion . . . . .                              | 4        |
| 3.2      | Intégration d'une recherche à plusieurs documents . . . . . | 5        |
| 3.3      | Intégration des fichiers non-ouverts . . . . .              | 6        |
| 3.4      | Gestion de projet . . . . .                                 | 7        |
| 3.4.1    | Git . . . . .   | 7        |
| <b>4</b> | <b>Conclusion</b>   | <b>8</b> |

# Introduction

Les *études pratiques* sont des projets réalisés chaque année pas les élèves du département Informatique de l'INSA de Rennes qui s'étalent sur 10 mois.

Cette étude pratique en particulier se présente sous la forme d'une contribution à un logiciel dont le code source est libre, c'est à dire qu'il est mis à disposition du public qui peut s'il le souhaite proposer des améliorations.

## 1.1 Latexila

Latexila est un projet d'éditeur LaTeX pour le projet Gnome, dont le développement a commencé en 2009<sup>1</sup> à l'initiative de Sébastien Wilmet, qui est encore à ce jour le mainteneur du projet. Bien que commencé en C, le projet a été porté vers le langage Vala en 2010. Latexila adopte une approche centrée sur le code, ce qui lui permet d'être bien plus léger que nombre d'autres éditeurs, comme TeXMaker<sup>2</sup> par exemple.

De par cette philosophie, il ne possède pas de module WYSIWYG<sup>3</sup>, ce qui le rend moins facile d'accès. Il met cependant à la disposition de l'utilisateur des outils puissants qui en font bien plus qu'un simple éditeur de texte, comme :

- La compilation assistée
- Une gestion de projet simplifiée
- Une aide à la saisie (sous forme de complétion) des commandes  $\text{\LaTeX}$

---

<sup>1</sup>Voir <https://wiki.gnome.org/Apps/LaTeXila/History>

<sup>2</sup>[http://www.xm1math.net/texmaker/index\\_fr.html](http://www.xm1math.net/texmaker/index_fr.html)

<sup>3</sup>What You See Is What You Get - Un mode d'édition où l'utilisateur édite directement une vue correspondant au rendu final

# Étude pratique

## 2.1 Tâche à réaliser

La tâche à réaliser est l'amélioration de la complétion des commandes de référence `\ref{clef}`. Ces commandes permettent de créer un lien cliquable vers un endroit précis du document, qui est matérialisé par un `\label{clef}`.

Lorsque l'utilisateur souhaite réaliser une référence, la complétion doit lui proposer une liste des clefs possible.

### 2.1.1 La complétion dans LaTeXila 2.2

La version 2.2 de LaTeXila fournit une complétion exhaustive des éléments du langage  $\text{\LaTeX}$  en se basant sur un fichier XML descriptif lu au démarrage du programme. Cette complétion est donc dite *statique*, car elle n'adapte pas les propositions en utilisant le contenu entré par l'utilisateur. Il lui est donc impossible de proposer une complétion pour `\ref`.

### 2.1.2 Une complétion dynamique

Afin d'améliorer ce mécanisme de complétion pour qu'il prenne en charge l'argument de la commande `\ref`, il convient de lui adjoindre une composante *dynamique*, qui prendra en compte les éléments `\label` déclarés par l'utilisateur.

# Réalisation

La réalisation s'est faite par petits pas. Sur une base de deux semaines de travail, l'équipe se donnait un objectif à atteindre pour le point suivant avec l'encadrant. Ces points bi-mensuels permettaient de discuter des directions à prendre pour le développement.

## 3.1 Intégration d'une recherche au document courant

Notre premier objectif a été de récupérer l'ensemble des labels déclarés dans le document courant.

Après avoir analysé le code, nous avons décidé d'utiliser tout d'abord un analyseur syntaxique  $\text{\LaTeX}$  fourni par Latexila. Il construit l'arbre représentatif de la structure du document. Ces résultats sont ensuite intégrés au *Completion Provider*, classe utilisée jusqu'alors pour gérer la complétion statique, et mis à jour lors de l'ouverture et de la sauvegarde du document.

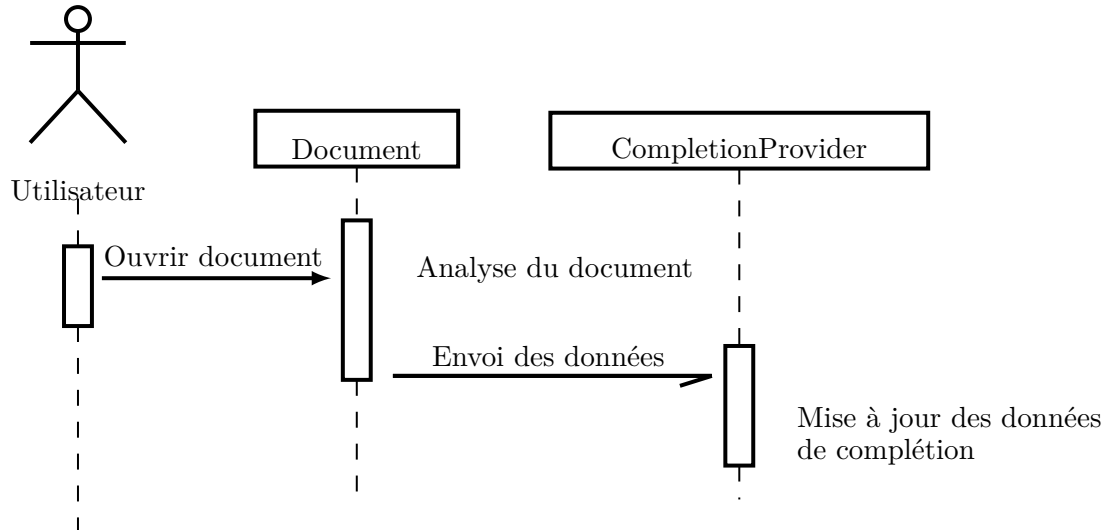


Figure 3.1: Diagramme de séquence d'une analyse de document

### 3.1.1 Invite de complétion

A ce stade, la complétion est alimentée par l'analyseur syntaxique déjà existant qui est utilisé par l'arbre de structure du document courant. Cela implique certaines contraintes comme de ne pas pouvoir l'appeler quand on le souhaite. La mise à jour de la complétion se fait donc en même temps que celle de l'arbre représentant

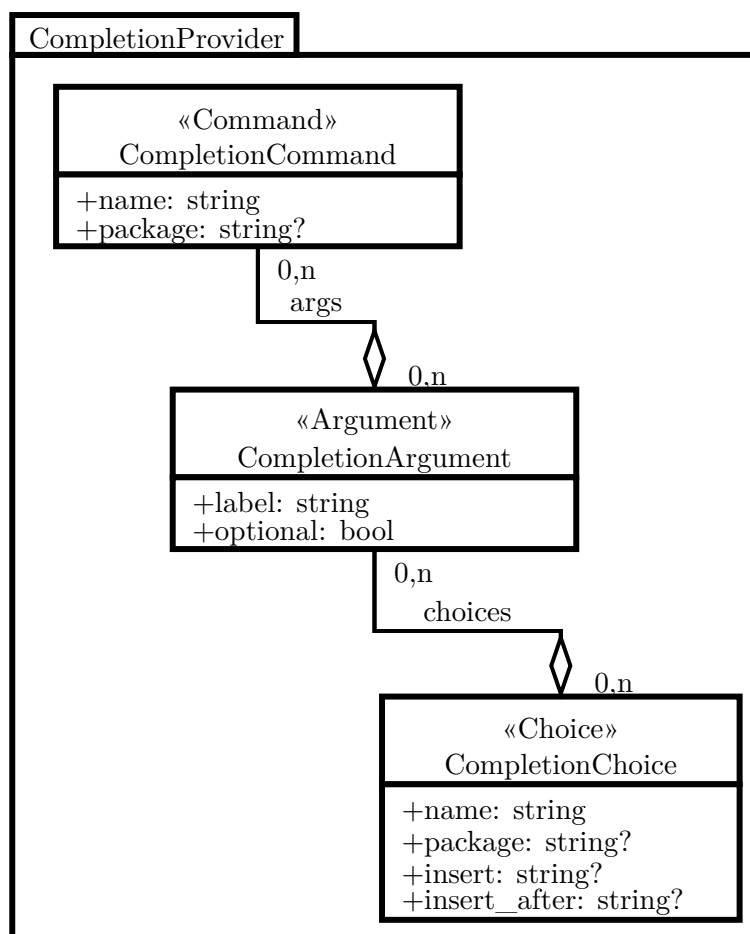


Figure 3.2: Le diagramme UML de la structure de complétion.

la structure du document courant. Ainsi lors du changement de document, par exemple, la complétion n'est pas mise à jour et est donc moins pertinente. D'autre part, on perd les précédentes données de complétion lors de l'analyse d'un autre document. Conserver ces données éviterait des analyses redondantes et permettrait d'économiser du temps de calcul.

## 3.2 Intégration d'une recherche à plusieurs documents

L'étape suivante a été l'intégration de la complétion pour tous les documents ouverts dans Latexila, permettant ainsi de proposer une complétion propre à chaque document et d'éviter les propositions inappropriées. Problèmes : comment choisir parmi les propositions celles liées au document courant ? Et comment stocker ces différents ensembles de choix de complétion de manière à pouvoir les mettre à jour ?

Ceci nous a conduit à rechercher une structure de donnée adaptée, et à l'implémenter après avoir étudié la documentation de ces structures dans le langage Vala. Nous avons opté pour l'ajout d'une table de hachage au *CompletionProvider*, qui stockera les ensembles de choix de complétion, indexés par le chemin absolu du fichier .tex auquel ils correspondent.

Les choix de complétion proposés à l'utilisateur sont ceux du document courant.

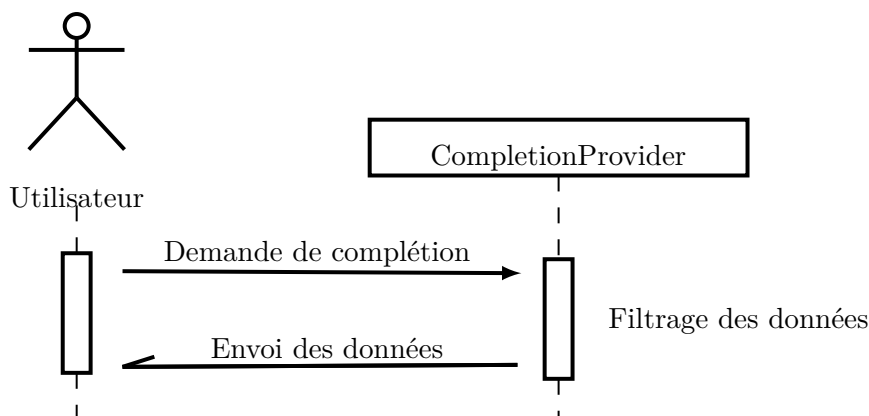


Figure 3.3: Diagramme de séquence d'une demande de complétion.

Ceux-ci sont mis à jour lors de l'appel de l'utilisateur au gestionnaire de complétion. Ainsi, même lors du changement de document, ces choix sont filtrés pour le nouveau document.

À ce stade, la complétion est fonctionnelle pour tous les fichiers ouverts, les données ne sont pas perdues, leur mise à jour est cohérente, et la mise à jour des choix de complétion du gestionnaire de complétion est faite de façon efficace (mise à jour uniquement si nécessaire, donc si l'utilisateur change de document, ou s'il sauvegarde un document). Cependant, il reste un problème : la complétion ne propose que les labels déclarés dans le document courant, alors que  $\text{\LaTeX}$  permet les références entre différents fichiers. Il faut donc revoir le filtrage des choix de complétion.

Cet aspect nous conduit également à proposer des choix de complétion provenant de documents qui n'ont pas été ouverts.

### 3.3 Intégration des fichiers non-ouverts

Après discussion avec notre encadrant, nous avons décidé de limiter les choix de complétion pour un document à l'ensemble des labels déclarés dans les documents .tex de son répertoire. Ce point est sujet à discussion, et nous y reviendrons durant la conclusion.

Pour ce faire, nous avons d'une part modifié notre structure de donnée pour que la clé de la table de hachage qui contient nos données de complétion soit désormais le chemin absolu du répertoire parent du document courant. D'autre part, nous avons décidé de dissocier la complétion des labels du processus qui se charge de la structure du document, séparant ainsi ces deux aspects distincts.

À ce stade, la complétion est entièrement fonctionnelle. Les choix proposés sont les labels déclarés dans les documents .tex du même répertoire que le document courant, l'analyse des documents est faite uniquement au besoin, et le changement de document vers un document d'un autre répertoire modifie correctement les propositions fournies.

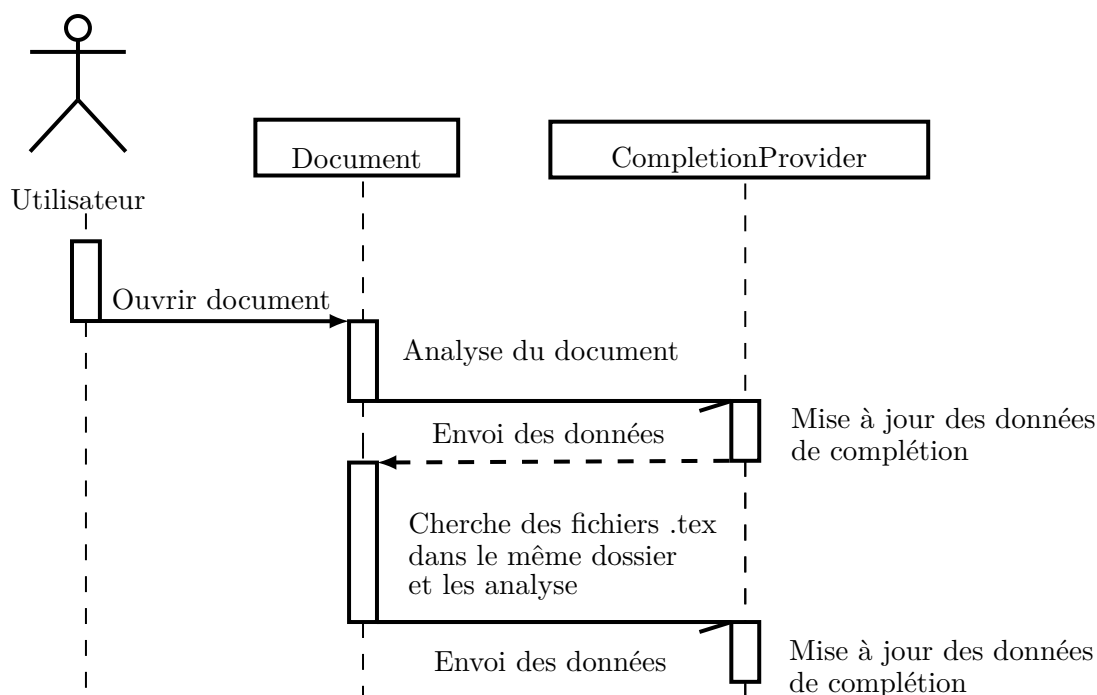


Figure 3.4: Diagramme de séquence d'analyse en arrière plan.

## 3.4 Gestion de projet

### 3.4.1 Git

Git est le VCS<sup>1</sup> utilisé par la très grande majorité des projets open-source. Latexila est hébergé sur le dépôt git du projet Gnome<sup>2</sup> et un miroir est également disponible sur Github<sup>3</sup>. Pour ce projet, nous avons créé un *fork* du dépôt principal sur Github et travaillé sur une branche annexe, régulièrement mise à jour à partir du dépôt principal. Un patch entre la **branche principale** et la **branche de test** est ensuite créé avec `git patch` et posté sur le bugzilla dédié<sup>4</sup>.

<sup>1</sup>Version Control System

<sup>2</sup>[git.gnome.org](https://git.gnome.org)

<sup>3</sup><https://github.com/GNOME/latexila>

<sup>4</sup>Page de ce projet [https://bugzilla.gnome.org/show\\_bug.cgi?id=748069](https://bugzilla.gnome.org/show_bug.cgi?id=748069)

# Conclusion

Le mécanisme de complétion, auparavant *statique* prend désormais en charge une complétion *dynamique* sur les labels déclarés par l'utilisateur. Cependant, notre solution impose que les fichiers sources dont sont issues les labels se trouvent dans le même répertoire que le document ouvert. Or certains utilisateurs sont susceptibles d'utiliser des sous-répertoires pour organiser leurs fichiers, et ne bénéficieraient donc pas de la complétion souhaitée. De plus, il arrive que certains fichiers .tex ne soient que des anciennes versions de rapports, voire des fichiers de tests. Dans ce cas, notre mécanisme de complétion proposerait les labels déclarés dans ces fichiers, alors qu'ils ne sont pas souhaités.

Nous avons tout de même soumis ce patch au mainteneur du projet, monsieur Wilmet, qui nous a fait part de ses commentaires.

Après discussion avec notre encadrant et monsieur Wilmet, nous avons établi que pour palier à ces problèmes, une refonte de la notion de *projet* comme ils sont organisés dans Latexila était nécessaire. On pourrait ainsi proposer une complétion ne prenant en compte que les fichiers faisant partie du même projet, et éviter les problèmes précédemment soulevés.

## Remerciements

Nous remercions Arnaud Blouin, notre encadrant, pour sa disponibilité et ses conseils, ainsi que Sébastien Wilmet, développeur et mainteneur de Latexila, pour l'aide qu'il nous a apporté dans ce premier pas dans l'univers du logiciel libre.