



# PLAN PROJET INTERPRÉTEUR DU LANGAGE LIR

---

PROJET PROPOSÉ PAR FRÉDÉRIQUE BARRIOS

Nicolas CAMINADE, Sylvan COURTIOL,  
Pierre DEBAS, Heïa DEXTER,  
Lucàs VABRE

# Sommaire

<b>I</b>	<b>Plan projet</b>	<b>3</b>
<b>1</b>	<b>Présentation du projet</b>	<b>5</b>
1.1	Définition générale du besoin : l'Interpréteur LIR . . . . .	5
1.2	Cahier des charges . . . . .	5
1.3	Définitions et acronymes . . . . .	5
1.4	Charte de projet . . . . .	6
1.4.1	Objectifs du projet . . . . .	6
1.4.2	Périmètre du projet . . . . .	6
1.4.3	Demandes hors périmètre . . . . .	6
1.4.4	Principaux livrables identifiés . . . . .	6
1.4.5	Les acteurs du projet . . . . .	6
1.4.6	Autres moyens et ressources . . . . .	7
1.4.7	Conditions d'acceptation . . . . .	7
1.4.8	Principaux risques identifiés et politique de gestion des risques . .	7
1.5	Étude générale du besoin . . . . .	7
1.5.1	Les acteurs . . . . .	8
1.5.2	Résumés de cas d'utilisation . . . . .	8
1.5.3	Récits d'utilisation (user stories) . . . . .	11
<b>2</b>	<b>Organisation du projet</b>	<b>12</b>
2.1	Présentation du cycle de vie itératif . . . . .	12
2.2	Répartition des rôles . . . . .	12
2.3	Plan communication . . . . .	12
2.3.1	Localisation géographique des intervenants . . . . .	12
2.3.2	Moyens de communication utilisés . . . . .	13
2.3.3	Réunions projets MOE . . . . .	13
2.3.4	Comités de Pilotage . . . . .	13
2.4	Assurance qualité . . . . .	13
2.4.1	Normes et standards de travail à observer (formalisme de modélisation, méthodes de contrôle, méthodes de développement, cycle de vie, conventions de code...) . . . . .	13
2.4.2	Manuel qualité et démarche qualité à observer (suivant la politique qualité de l'organisation), suivi et contrôle qualité (organisation, fréquence, participants). . . . .	13
2.5	Ressources matérielles et logicielles . . . . .	13
<b>3</b>	<b>Pilotage du projet</b>	<b>14</b>
3.1	Cycle de vie itératif . . . . .	14
3.2	Estimation initiale . . . . .	14
3.3	Planification prévisionnelle initiale . . . . .	14

3.4	Durée et ordonnancement des principales tâches et itérations . . . . .	14
3.5	Identification des premiers jalons . . . . .	14
3.6	Calendrier prévisionnel . . . . .	14
3.7	Organisation des réunions projets et comités de pilotage . . . . .	14
3.8	Suivi du projet par période . . . . .	14
3.8.1	Suivi d'avancement et mesure des écarts par rapport au prévisionnel revu lors de la période précédente . . . . .	15
3.8.2	Synthèse par "tableau de bord" . . . . .	15
3.8.3	Résultats des tests et recette de prototype de la période . . . . .	15
3.8.4	Résultats des revues/suivis/contrôles qualité de la période . . . . .	15
3.8.5	Identification des principaux écarts et problèmes constatés, solutions possibles . . . . .	15
3.8.6	Propositions de modification de la planification prévisionnelle pour tenir compte des corrections à apporter . . . . .	15
3.8.7	Comptes-rendus des réunions projets de la période . . . . .	15
3.8.8	Compte-rendu du comité de pilotage de la période . . . . .	15
3.8.9	Planification prévisionnelle révisée pour les périodes suivantes (en fonction des décisions prises) . . . . .	15
<b>II</b>	<b>Annexes</b>	<b>16</b>
<b>A</b>	<b>Sujet Interpréteur LIR</b>	<b>17</b>
<b>B</b>	<b>Gestion de la configuration de l'Interpréteur LIR</b>	<b>1</b>
1	Logiciels de développement . . . . .	1
1.1	Environnement de Développement Intégré . . . . .	1
1.2	Contrôle des versions du code . . . . .	1
1.3	Organisation . . . . .	1
1.4	Modélisation . . . . .	1
2	Logiciels généraux . . . . .	1
2.1	Communication . . . . .	1
2.2	Éditeur de texte . . . . .	2
2.3	Partage distant des fichiers . . . . .	2
3	Sécurité . . . . .	2
<b>C</b>	<b>Récits d'utilisation de l'Interpréteur LIR</b>	<b>1</b>
1	Commande . . . . .	2
2	Commande debut . . . . .	3
3	Commande fin . . . . .	4
4	Commande defs . . . . .	5
5	Commande affiche . . . . .	6
6	Commande affiche avec une expression . . . . .	7
7	Commande var pour une chaîne de caractères . . . . .	8
8	Commande var pour un entier . . . . .	9
9	Expression concaténation sur chaîne de caractères . . . . .	10
10	Expression logique . . . . .	11
11	Expression arithmétique . . . . .	12

Première partie

Plan projet

# Introduction

Dans le cadre des projets tuteuré du semestre 2 de première année de DUT informatique de l'année 2020-2021, le sujet de l'Interpréteur LIR a été proposé par F. Barrios, un des enseignants de l'IUT de Rodez. Ce document a pour but de rassembler les informations fondamentales relatives à la gestion du projet. Ce plan projet est un document de référence du projet qui sera complété tout au long de son avancement.

# Chapitre 1

## Présentation du projet

### 1.1 Définition générale du besoin : l'Interpréteur LIR

L'Interpréteur LIR est un interpréteur d'un langage de programmation simple, il sera nommé LIR pour Langage IUT de Rodez. Un interpréteur est un automate enchaînant les tâches suivantes : analyse lexico-syntaxique d'une ligne de commande puis interprétation.

Une ligne entrée par un utilisateur sera donc : soit une commande à exécuter immédiatement, soit une ligne de programme à mémoriser pour une exécution ultérieure. Une ligne de programme se distinguera d'une ligne de commande par le fait qu'elle sera toujours précédée d'un "numéro d'ordre" appelé aussi "étiquette".

### 1.2 Cahier des charges

Le document en annexe fourni par la maîtrise d'ouvrage (MOA) définit l'interpréteur attendu avec les éléments du Langage IUT de Rodez, la syntaxe des instructions de programmation et des commandes générales attendues dans le logiciel final. Le document précise également le comportement attendu de l'interpréteur lors de son utilisation suivi d'un exemple d'une session sous cet interpréteur LIR.

### 1.3 Définitions et acronymes

**Analyse syntaxique :** La vérification de la conformité aux contraintes syntaxiques définies par une grammaire.

**Analyse lexicale :** L'identification des éléments du vocabulaire d'un langage dans une description textuelle (scanning) et la recherche des unités lexicales (lexèmes).

**Grammaire :** Contraintes syntaxiques définissant les constructions correctes (autorisées) d'un langage.

**Interpréteur :** Programme capable d'analyser les instructions d'un langage (évolué) et de les exécuter directement.

**Langage :** Outil de description et d'expression.

**Langage IUT de Rodez (LIR)**

**Sémantique :** Étude du sens des unités linguistiques et de leurs combinaisons. Aspect de la logique qui traite de l'interprétation et de la signification des systèmes formels, par opposition à la syntaxe, entendue comme l'étude des relations formelles entre formules de tels systèmes (d'après le dictionnaire Larousse).

**Syntaxe :** Partie de la grammaire qui décrit les règles par lesquelles les unités linguistiques se combinent en phrases. En logique, étude des relations formelles entre expressions d'un langage (d'après le dictionnaire Larousse). Aussi, la syntaxe est spécifiée par des grammaires et des notations formelles.

**Vocabulaire :** Symboles de base utilisés dans un langage.

## 1.4 Charte de projet

### 1.4.1 Objectifs du projet

Réaliser un interpréteur capable d'exécuter un script ou une série d'instructions dans le langage LIR avec les outils et connaissances et mis à disposition par l'IUT de Rodez.

### 1.4.2 Périmètre du projet

Ce projet doit être mené jusqu'à obtention d'un interpréteur capable d'exécuter toutes les commandes précisées dans le cahier des charges fourni.

### 1.4.3 Demandes hors périmètre

Il n'y a pas de demandes hors périmètre.

### 1.4.4 Principaux livrables identifiés

**Livrables :** plan projet, dossier de projet, CD (de préférence un dossier compressé plutôt qu'un CD) contenant les codes exécutables les fichiers de données, les codes sources et la version numérique du dossier et le manuel utilisateur.

### Définition du cadre

**Coût :** À définir par le chef de projet (P. Debas).

**Délais :** Deux dates butoirs identifiées.

— Remise du projet le vendredi 28 mai 2021.

— Soutenance du projet la semaine du 7 juin 2021.

**Qualité :** Projet codé en Java dans les respects des conventions et bonnes pratiques.

### 1.4.5 Les acteurs du projet

L'équipe MOE : N. CAMINADE, S. COURTIOL,  
P. DEBAS, H. DEXTER,  
L. VABRE

La MOA : F. Barrios

Le contrôle qualité : F. Barrios et J. Accot

#### 1.4.6 Autres moyens et ressources

Pas de moyens ou ressources supplémentaires.

#### 1.4.7 Conditions d'acceptation

Pas d'exigence ou de contraintes supplémentaires.

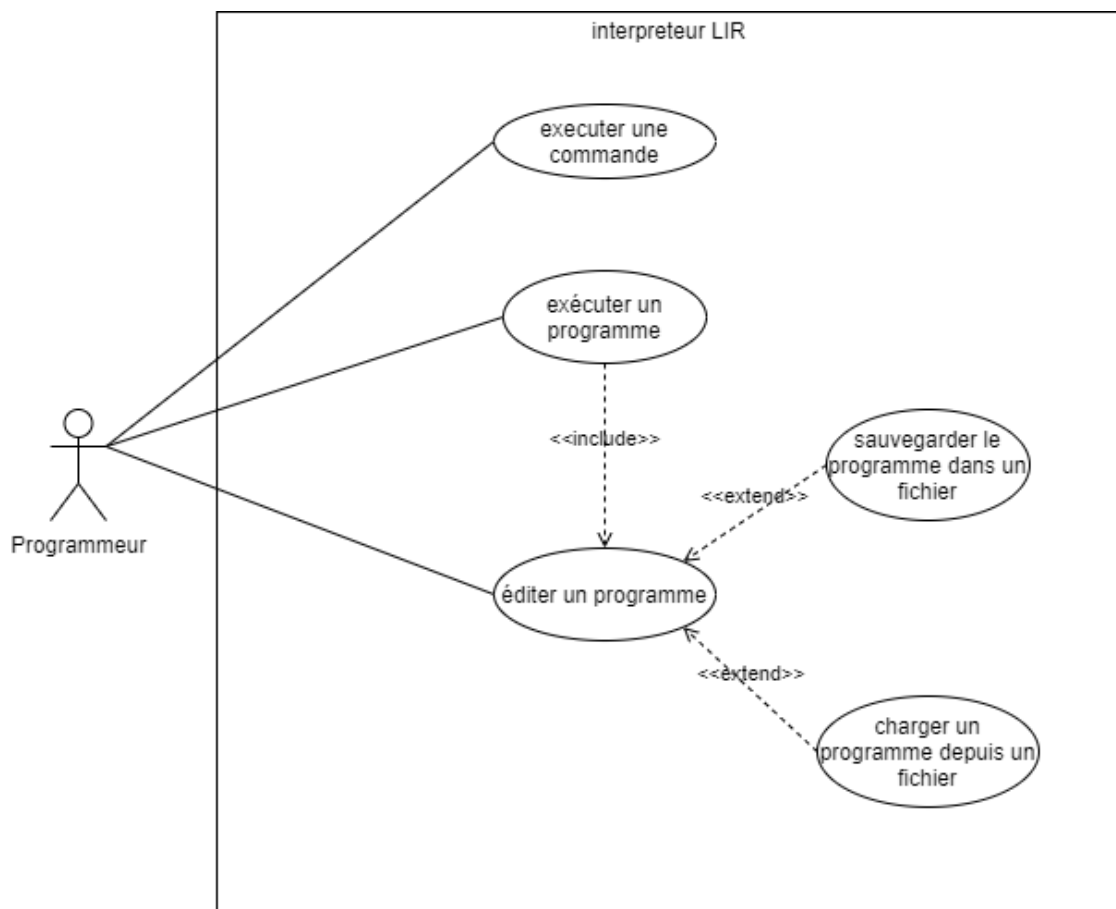
#### 1.4.8 Principaux risques identifiés et politique de gestion des risques

Si possible tous les membres du groupe auront les mêmes droits sur les fichiers communs. En conséquence chaque membre du groupe ne doit pas donner des droits sur ces fichiers à une personne extérieure au projet (autre que MOA). Cf. Gestion de la configuration (produit par S. Courtiol).

Des sauvegardes du dépôt GitHub (contenant toutes les données du projets) seront effectuées régulièrement (fréquence à définir) par le gestionnaire de configuration. Toutes données qui ne sont pas dans le dépôt sont à la responsabilité de chacun. Cf. Gestion de la configuration (produit par S. Courtiol).

### 1.5 Étude générale du besoin

**Diagramme de cas d'utilisation général de l'Interpréteur LIR** comprenant un acteur (le programmeur) et cinq cas d'utilisation identifiés comme suit :





### 1.5.1 Les acteurs

**Programmeur :** Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris nec ultrices velit. Curabitur convallis non ipsum malesuada fringilla.

### 1.5.2 Résumés de cas d'utilisation

#### — Exécuter une commande

**Acteurs** Programmeur : il entre une commande à faire exécuter immédiatement par l'interpréteur.

**Objectifs** Exécuter la commande entrée dans l'interpréteur.

**Pré-Conditions** L'interpréteur LIR est lancé et le curseur est derrière l'invite.

**Post-Conditions** La commande est exécutée et le résultat est affiché.

#### Scénario nominal (grandes étapes)

1. Le programmeur écrit derrière l'invite une ligne de commande.
2. Le programmeur valide cette commande.
3. L'interpréteur effectue une analyse lexico-syntaxique.
4. L'interpréteur interprète la ligne de commande.

#### Scénarios d'échec

**Point 3 du scénario nominal :** la syntaxe de la ligne écrite est incorrecte.

- Un message d'erreur explicite informe le programmeur.
- Retour au point 4 du scénario nominal.

**Point 4 du scénario nominal :** la commande conduit à une erreur d'exécution.

- Un message d'erreur explicite informe le programmeur.
- Retour au point 4 du scénario nominal.

#### — Exécuter un programme

**Acteurs** Programmeur : Il lance l'exécution du programme présentement chargé dans l'interpréteur.

**Objectifs** Le but est d'exécuter les instructions du programme chargé.

#### Pré-conditions, Post-conditions

**Pré-conditions** Toutes les instructions chargées sont correctes.

**Post-conditions** Le contexte (variables) de l'interpréteur inclus le contexte final du programme.

**Scénario nominal (grandes étapes)**

1. Le programmeur exécute la commande **lance**.
2. L'interpréteur exécute l'instruction ayant l'étiquette la plus petite.
3. L'interpréteur passe l'instruction suivante (étiquette supérieure la plus proche sauf si changement du compteur ordinal).
4. Tant qu'il reste des instructions avec une étiquette supérieure retour en 3.
5. Le programme a fini de s'exécuter.
6. Le contrôle est rendu au programmeur qui peut à nouveau saisir.

**Scénarios d'échec Point 2 du scénario nominal :** Aucune instruction est chargée dans l'interpréteur

- L'interpréteur affiche un message d'erreur explicite.
- Retour au point 6 du scénario nominal.

## — Éditer un programme

**Acteurs** Programmeur : Il écrit ou modifie un programme à faire exécuter par l'interpréteur.

**Objectifs** Écrire un nouveau programme ou en modifier un existant dans le but de l'exécuter ou de le sauvegarder.

**Pré-conditions** L'interpréteur LIR est en mode édition. Un code vierge est affiché ou un code préexistant est chargé depuis un fichier.

**Post-conditions** Le code source édité est prêt à être exécuté, abandonné ou sauvegardé, selon l'intention du programmeur.

**Scénario nominal (grandes étapes)**

1. Le programmeur écrit une ligne de code par instruction, en la faisant précéder de son étiquette.
2. Le programmeur consulte le code déjà écrit à tout moment avec la commande **liste**. Selon la syntaxe choisie, l'interpréteur affiche la plage demandée ou la totalité des lignes de code du programme dans l'ordre croissant des étiquettes.
3. Le programmeur consulte la liste des identificateurs déclarés et leurs valeurs en entrant la commande **defs**.
4. Au besoin, le programmeur efface une ou plusieurs lignes avec la commande **efface**.
5. Au besoin, le programmeur efface les lignes de code et identificateurs mémorisés et commence un nouveau code avec la commande **debut**.

**Scénarios d'échec**

**Point 2 du scénario nominal :** Aucune ligne de code n'est écrite ou la plage de code à afficher n'est pas correcte.

- L'interpréteur en avise le programmeur au moyen d'un message d'erreur.
- Retour au point 1.

**Point 3 du scénario nominal :** Aucun identificateur n'a encore été déclaré.

- L'interpréteur affiche un message informant le programmeur.
- Retour au point 1.

**Point 4 du scénario nominal :** La plage de ligne à effacer est incorrecte.

- Un message d'erreur en informe le programmeur
- Retour au point 1.

## — Sauvegarder le programme dans un fichier

**Acteurs** Programmeur : il entre la commande de sauvegarde 'sauve' suivit du chemin vers le nom du fichier dans lequel on veut sauvegarder le programme.

**Objectifs** L'objectif est de sauvegarder le programme rédigé dans l'interpréteur, dans un fichier texte

### Pré-Conditions

- Un programme doit être rédigé (au moins une ligne)
- Le chemin vers le fichier ne doit pas contenir de caractères spéciaux (pour éviter les erreurs)

### Post-Conditions

- Le fichier doit être créé (si il n'existe pas déjà)
- Le fichier doit contenir le programme rédigé par le programmeur

### Scénario nominal (grandes étapes)

1. Le programmeur exécute la commande de sauvegarde
2. Le programme empêche la saisie à l'utilisateur
3. Le programme sauvegarde le code saisi (en mettant les étiquettes dans l'ordre croissant) et l'enregistre dans un fichier
4. Le programme affiche un message qui indique la fin de la sauvegarde
5. Le programme permet la saisie à l'utilisateur

### Scénarios d'échec

- Point 2 : Si le chemin du fichier exécuté dans la commande de sauvegarde contient des caractères spéciaux ;
  1. Affiche un message d'erreur spécifiant qu'il ne faut pas de caractères spéciaux dans le chemin
  2. Reprend au point 5
- Point 3 : Si aucun programme n'a été écrit ;
  1. Affiche un message d'erreur spécifiant qu'il faut déjà avoir rédigé le programme avant de le sauvegarder
  2. Reprend au point 5
- Point 3 : Si aucun programme comporte plus de 99999 lignes ;
  1. Affiche un message d'erreur spécifiant que le nombre de lignes dépasse la valeur maximale
  2. Reprend au point 5

## — Charger un programme depuis un fichier

**Acteurs** Programmeur : Il entre la commande "charge" suivie du "chemin"/de l'arborescence du fichier que l'on veut charger.

**Objectifs** Charger un programme en mémoire, en ayant pour seule indication son arborescence

**Pré-conditions, Post-conditions** Il faut que le programme ai été préalablement sauvegardé au stocké en mémoire

**Pré-Conditions** L'interpréteur LIR est en mode édition. Il faut que le programme ai été préalablement sauvegardé au stocké en mémoire, et qu'il soit un fichier texte. Et que sont chemin/arborescence soit accessible à l'interpréteur.

**Post-Conditions** Le code source a été entièrement chargé sur LIR alors le chargement s'arrête.

### Scénario nominal (grandes étapes)

1. Le programmeur veut charger un fichier stocké.
2. Le programmeur consulte l'arborescence de son fichier.
3. Le programmeur entre la commande **charge** suivie de l'arborescence de son fichier.
4. L'interpréteur signale au programmeur que le chargement a pu se faire par un "ok".

### Scénarios d'échec

**Point 2 du scénario nominal :** Aucun fichier n'est situé dans l'arborescence signalée

- L'interpréteur en avise le programmeur au moyen d'un message d'erreur.
- Retour au point 1.

**Point 3 du scénario nominal :** Le fichier ne correspond pas au type de fichier accepté par LIR.

- L'interpréteur affiche un message informant le programmeur.
- Retour au point 1.

**Point 4 du scénario nominal :** La ligne de commande est incorrecte.

- Un message d'erreur en informe le programmeur
- Retour au point 1.

**Point 4 du scénario nominal :** Le code source du fichier est corrompu

- Un message d'erreur en informe le programmeur
- Retour au point 1.

### 1.5.3 Récits d'utilisation (user stories)

Les récits d'utilisation

Des récits d'utilisation ont été rédigés pour chaque commande et instruction.

## Chapitre 2

# Organisation du projet

### 2.1 Présentation du cycle de vie itératif

Pour développer l'Interpréteur LIR, le modèle de cycle de vie itératif a été choisi. Ce modèle de développement de logiciel consiste en une succession de cycles de spécification, de conception, de réalisation et de tests, le but est d'enrichir et de « remodeler » des prototypes du logiciel successifs. Par conséquent, une version du logiciel sera un « dernier prototype ».

La gestion du risque va entraîner la mise en place d'un noyau architectural avec des fonctions indispensables du logiciel dès les deux premières itérations. Les itérations suivantes apporteront des corrections et de nouvelles fonctions au logiciel.

Les versions successives des prototypes permettent de matérialiser l'avancement et d'éviter « l'effet tunnel » sur le projet. Ces prototypes (versions 0.x) entretiennent la motivation des différents acteurs du projet : l'équipe MOE, la MOA.

Le principe fondamental à chaque début d'itération est de ne spécifier en détail que les fonctionnalités nécessaires pour cette itération. Ainsi la prise en compte d'évolutions du besoin reste possible jusqu'à la dernière itération. De même le « refactoring » de la conception (largement facilité par les outils) a lieu à chaque étape pour intégrer des évolutions et des ajouts. Le but étant bien sûr de fabriquer le logiciel adapté au besoin en laissant la possibilité de « mûrir » au cours du temps.

Ce type de cycle implique une taille homogène de l'équipe et une polyvalence des équipiers.

### 2.2 Répartition des rôles

Rôles des membres de l'équipe impliqués dans le projet jusqu'au mois de mai 2021 :

Chef de projet MOE	Pierre Debas
Secrétaire de projet	Heïa Dexter
Gestionnaire de configuration	Sylvan Courtiol
Développeur	Nicolas Caminade
Développeur	Lucàs Vabre

### 2.3 Plan communication

#### 2.3.1 Localisation géographique des intervenants

L'équipe MOE, la MOA et les contrôleurs qualités sont basés sur Rodez (12).

La MOA, les contrôleurs qualités, H. Dexter sont basés sur Rodez (12), S. Courtiol sur

Luc-La-Primaube à côté de Rodez (12), P. Debas est basé à la fois sur Rodez et à Albi (81), L. Vabre sur Gages (12) et N. Caminade sur Rodez et Moncaut (47).

### 2.3.2 Moyens de communication utilisés

Les communications formelles sont effectuées via les mails de l'IUT (généralement par le chef de projet) avec les autres membres du projet en CC.  
Serveur Discord spécifique au projet pour communication écrite ou vocale de la MOE.  
Cf. le document Configuration interpréteur du langage LIR produit par le gestionnaire de configuration (S. Courtiol).

### 2.3.3 Réunions projets MOE

Les réunions projet MOE seront hebdomadaires voire bi-hebdomadaires et dans le contexte de la crise sanitaire elles se dérouleront en distanciel via Discord (vocal, visio-conférence). Seront prévue des réunions courtes de 20 minutes et des réunions longues de 1h30.

Ces réunions auront pour objectif de faire le point sur l'avancement du projet, le respect des objectifs fixé sur la période et de fixer les prochains objectifs à remplir d'ici la prochaine réunions. Aussi ces réunions seront l'occasion de faire part de difficultés éventuelles rencontrées par les membres de l'équipe au cours de la semaine et de communiquer les informations sur les prochaines rencontres avec la MOA.

Les comptes-rendus seront rédigés par la secrétaire de projet (H. Dexter) et diffusés sur le serveur Discord de l'équipe sous format texte.

### 2.3.4 Comités de Pilotage

Les comités de pilotage rassembleront la MOA et toute l'équipe de MOE. Les COPIL seront dirigé par le chef de projet éventuellement assisté par le secrétaire.

La fréquence des COPIL est au mieux hebdomadaire et d'une durée d'une demi-heure à trois quarts d'heure selon l'avancement du projet. Les comptes-rendus des COPIL seront rédigés par l'actuelle secrétaire de projet (H. Dexter) et diffusés le lendemain à la MOE du projet.

## 2.4 Assurance qualité

### 2.4.1 Normes et standards de travail à observer (formalisme de modélisation, méthodes de contrôle, méthodes de développement, cycle de vie, conventions de code...)

Pour mener à bien ce projet l'équipe MOE travaille en utilisant le langage UML comme formalisme de modélisation, la méthode de développement dirigé par les tests i.e. la méthode TDD (Test Driven Development) en respectant les Java Code Convention pour un modèle de cycle de vie itératif.

### 2.4.2 Manuel qualité et démarche qualité à observer (suivant la politique qualité de l'organisation), suivi et contrôle qualité (organisation, fréquence, participants).

## 2.5 Ressources matérielles et logicielles

## Chapitre 3

# Pilotage du projet

### 3.1 Cycle de vie itératif

Pour développer l'Interpréteur LIR, le modèle de cycle de vie itératif a été choisi. Ce modèle de développement de logiciel, rappelons-le, consiste en une succession de cycles de spécification, de conception, de réalisation et de tests, le but est d'enrichir et de « remodeler » des prototypes du logiciel successifs. Par conséquent, une version du logiciel sera un « dernier prototype ».

Si le choix de modèle de cycle de vie s'est porté sur le modèle itératif, c'est parce qu'il s'agit d'un modèle "réaliste" et possible à mettre en place dans le cadre des projets tuteurs :

- Une limitation de "l'effet tunnel" pour une meilleure dynamique et motivation des équipes (MOA et MOE).
- Une meilleure acceptation des changements grâce aux prototypes.
- Une meilleure gestion des risques.
- Est adapté pour une équipe de cinq personnes polyvalentes.
- Le principe d'itérations où seules les fonctionnalités nécessaires sont spécifiées en détail en début d'itération ce qui permet une évolution du besoin.

### 3.2 Estimation initiale

### 3.3 Planification prévisionnelle initiale

### 3.4 Durée et ordonnancement des principales tâches et itérations

### 3.5 Identification des premiers jalons

### 3.6 Calendrier prévisionnel

### 3.7 Organisation des réunions projets et comités de pilotage

### 3.8 Suivi du projet par période

Pour chaque période :

- 3.8.1 Suivi d'avancement et mesure des écarts par rapport au prévisionnel revu lors de la période précédente
- 3.8.2 Synthèse par "tableau de bord"
- 3.8.3 Résultats des tests et recette de prototype de la période
- 3.8.4 Résultats des revues/suivis/contrôles qualité de la période
- 3.8.5 Identification des principaux écarts et problèmes constatés, solutions possibles
- 3.8.6 Propositions de modification de la planification prévisionnelle pour tenir compte des corrections à apporter
- 3.8.7 Comptes-rendus des réunions projets de la période
- 3.8.8 Compte-rendu du comité de pilotage de la période
- 3.8.9 Planification prévisionnelle révisée pour les périodes suivantes (en fonction des décisions prises)



## Deuxième partie

### Annexes

Annexe A

Sujet Interpréteur LIR



# GESTION DE LA CONFIGURATION INTERPRÉTEUR DU LANGAGE LIR

---

PROJET PROPOSÉ PAR FRÉDÉRIQUE BARRIOS

**version : 18 mai 2021**

Nicolas CAMINADE, Sylvan COURTIOL,  
Pierre DEBAS, Heïa DEXTER,  
Lucàs VABRE

## Annexe B

# Gestion de la configuration de l'Interpréteur LIR

### Introduction

Ce document a pour but de confirmer par écrit la configuration logicielle choisie pour le projet.

Le contenu de ce document n'est pas fixé et des changements peuvent être apportés. Cependant ce document doit être connu et suivi par les membres du groupe. En cas de modifications, une annonce sur discord sera faite.

Pour toute question ou suggestion se référer au gestionnaire de configuration (présentement Sylvain COURTIOL).

## 1 Logiciels de développement

### 1.1 Environnement de Développement Intégré

Eclipse JEE (version 2020-12)  
JDK 15

### 1.2 Contrôle des versions du code

Git avec dépôt sur GitHub. (Un apprentissage est nécessaire donc pour commencer certaines libertés sont possibles).

### 1.3 Organisation

Via le site Trello (non utilisé pour le moment).

### 1.4 Modélisation

La modélisation UML sera effectuée sur Modelio Open Source (version 4.1).

## 2 Logiciels généraux

### 2.1 Communication

Les communications formelles sont effectuées via les mails de l'IUT (généralement par le chef de projet) avec les autres membres du projet en CC.

Serveur discord spécifique au projet pour la communication écrite ou vocale de la MOE.

Google Meet pour les réunions avec les personnes autres que MOE. Adaptable à ce qui convient le mieux à cette personne.

## 2.2 Éditeur de texte

Le traitement de texte sera fait sous LaTeX notamment avec la distribution MiKTeX et l'IDE TexStudio. Les documents texte sont partagés en PDF ou version papier à la MOA/MOE et en format modifiable .tex seulement à la MOE via la solution de partage distant des fichiers (voir sous-section suivante).

## 2.3 Partage distant des fichiers

Les partages de tous les fichiers généraux et codes sources se feront sur GitHub via le site, le logiciel GitHub desktop ou git. Il y aura également une intégration Discord informant des commits.

## 3 Sécurité

Si possible tous les membres du groupe auront les mêmes droits sur les fichiers communs. En conséquence aucun membre du groupe ne doit donner des droits sur ces fichiers à une personne extérieure au projet (autre que MOA).

Les sauvegardes du dépôt GitHub (contenant toutes les données du projets) seront effectuées régulièrement (tous les 1 ou 2 jours) par le gestionnaire de configuration. Toutes données qui ne sont pas dans le dépôt sont à la responsabilité de chacun. Les sauvegardes sont enregistrées en local par le gestionnaire de configuration ainsi que sur le Google drive partagé du projet.



# ETUDE DU BESOIN RÉCITS D'UTILISATION INTERPRÉTEUR DU LANGAGE LIR

---

PROJET PROPOSÉ PAR FRÉDÉRIQUE BARRIOS

Nicolas CAMINADE, Sylvan COURTIOL,  
Pierre DEBAS, Heïa DEXTER,  
Lucàs VABRE

## Annexe C

# Récits d'utilisation de l'Interpréteur LIR

Texte. Blablabla

# Récits d'utilisation proposés lors de l'itération 1

## 1 Commande

### Récit d'utilisation

**Titre :** Exécution d'une commande

**Récit :** Exécution d'une commande

**En tant que :** programmeur avec l'interpréteur LIR

**Je souhaite :** utiliser une commande directe de l'interpréteur

**Afin de :** obtenir le résultat de cette commande

### Critères d'acceptation

**À partir du fait :** que je suis en train d'utiliser l'interpréteur et que j'ai la possibilité d'entrer une ligne

**Alors :** j'entre une ligne de commande directe et que je la valide

**Enfin :** j'obtiens le résultat de cette commande ou un feedback, si le résultat n'en est pas un, m'informant du bon déroulé de l'exécution de la commande ou de son échec



## 2 Commande debut

### Récit d'utilisation

**Titre :** debut

**Récit :** Réinitialiser un programme vierge

**En tant que :** programmeur

**Je souhaite :** vider l'intégralité du contexte d'exécution

**Afin de :** pouvoir écrire un nouveau programme

### Critères d'acceptation

**À partir de :** un programme chargé en mémoire centrale

**Alors :** j'efface les lignes de code et variables déclarées avec la commande `debut`

**Enfin :** L'interpréteur affiche une page vierge ; je peux écrire un nouveau programme.

### 3 Commande fin

#### Récit d'utilisation

**Titre :** Quitter l'interpréteur (commande fin)

**Récit :** Quitter l'interpréteur (commande fin)

**En tant que :** programmeur avec l'interpréteur LIR

**Je souhaite :** quitter l'interpréteur

**Afin de :** arrêter d'utiliser l'interpréteur LIR pour la session courante

#### Critères d'acceptation

**À partir du fait :** je suis en train d'utiliser l'interpréteur

**Alors :** je souhaite quitter l'interpréteur pour la session courante en exécutant la commande fin

**Enfin :** le processus courant de l'interpréteur LIR s'arrête

## 4 Commande defs

### Récit d'utilisation

**Titre :** Affichages du contexte courant (commande defs)

**Récit :** Affichages du contexte courant (commande defs)

**En tant que :** programmeur avec l'interpréteur LIR

**Je souhaite :** voir toutes les variables définies dans la session courante (identificateur et valeur)

**Afin de :** connaître le contexte actuel de la session courante de l'interpréteur

### Critères d'acceptation

**À partir du fait :** des variables sont définies dans la session courante de l'interpréteur

**Alors :** je souhaite connaître le contexte actuel en exécutant la commande defs

**Enfin :** l'interpréteur affiche chaque variable ligne par ligne avec son identificateur et sa valeur

## 5 Commande affiche

### Récit d'utilisation

**Titre :** Commande affiche

**Récit :** Provoquer le saut de ligne sur la sortie de texte courante

**En tant que :** Programmeur

**Je souhaite :** que l'interpréteur LIR saute une ligne sur la sortie de texte courante

**Afin de :** Provoquer un saut de ligne sur cette sortie

### Critères d'acceptation

**À partir du fait :** que j'ai une sortie de texte courante

**Alors :** je tape la commande affiche

**Enfin :** l'interpréteur saute une ligne sur la sortie de texte courante et nous spécifie si la commande a bien pu s'exécuter sur la console(en tant que feed-back)

## 6 Commande affiche avec une expression

### Récit d'utilisation

**Titre :** Commande affiche (expression)

**Récit :** Afficher le contenu d'une expression sur la console de l'interpréteur

**En tant que :** Programmeur

**Je souhaite :** que l'interpréteur LIR évalue et affiche le contenu de l'expression que l'on lui donne

**Afin de :** pouvoir récupérer/vérifier le/les résultat(s) de son programme

### Critères d'acceptation

**À partir du fait :** que j'ai la possibilité de saisir une ligne de commande

**Alors :** je tape la commande affiche et écrit l'expression dont je veut que la valeur soit affichée à la suite : affiche <expression>

**Enfin :** l'interpréteur évalue dans l'expression spécifiée la valeur de celle-ci et renvoie cette valeur sur la console et affiche un résultat sur LIR (en tant que feed-back) pour nous spécifier si la commande a bien pu s'exécuter

## 7 Commande var pour une chaîne de caractères

### Récit d'utilisation

**Titre :** Commande var (Chaine de caractères)

**Récit :** Initialiser une chaine de caractère dans variable / Changer sa valeur

**En tant que :** Programmeur

**Je souhaite :** que l'interpréteur LIR stock une chaine dans une variable

**Afin de :** pouvoir récupérer/manipuler cette chaine plus tard dans le programme

### Critères d'acceptation

**À partir du fait :** que j'ai la possibilité de saisir une ligne de commande

**Alors :** je tape la commande var et met une chaine de caractère entre double guillemets comme valeur : var <nomVariable>="<chaine>"

**Enfin :** l'interpréteur enregistre dans la variable spécifié la chaine de caractère voulue et renvoie la variable suivie de sa valeur (en tant que feed-back)

## 8 Commande var pour un entier

### Récit d'utilisation

**Titre :** Commande var (Entier)

**Récit :** Initialiser un entier dans variable / Changer sa valeur

**En tant que :** Programmeur

**Je souhaite :** que l'interpréteur LIR stock un entier dans une variable

**Afin de :** pouvoir récupérer/manipuler cet entier plus tard dans le programme

### Critères d'acceptation

**À partir du fait :** que j'ai la possibilité de saisir une ligne de commande

**Alors :** je tape la commande var et met un entier comme valeur :  
| var<nomVariable>=<entier> |

**Enfin :** l'interpréteur enregistre dans la variable spécifié l'entier voulu et renvoie la variable suivie de sa valeur (en tant que feed-back)

## 9 Expression concaténation sur chaîne de caractères

### Récit d'utilisation

**Titre :** Opérateur `+` sur les chaînes de caractères

**Récit :** Concaténation de chaînes

**En tant que :** Programmeur

**Je souhaite :** accoler deux chaînes l'une à la suite de l'autre

**Afin de :** créer des messages dépendant du contexte d'exécution sur la sortie standard. Représenter une valeur entière par son écriture chiffrée en base 10.

### Critères d'acceptation

**À partir de :** deux chaînes de caractères ou une chaîne et un entier, en tant qu'identificateurs déclarés ou expressions littérales.

**Alors :** En utilisant une expression de type `var nouvelleChaine = opeGauche + o` j'obtiens la concaténation de deux chaînes.

**Enfin :** L'identificateur `nouvelleChaine` contient la chaîne constituée des deux primordiales concaténées. L'interpréteur confirme en affichant la nouvelle valeur ou m'informe d'une erreur. L'opération peut être récursive mais n'est pas commutative. Une concaténation s'effectue toujours par la droite.



## 10 Expression logique

### Récit d'utilisation

**Titre :** Expression logique dans un branchement conditionnel

**Récit :** Opérations relationnelles sur deux entiers

**En tant que :** Programmeur

**Je souhaite :** que l'Interpréteur LIR compare deux entiers avec une relation d'ordre ou d'équivalence

**Afin que :** d'exécuter ou non une branche du code avec l'instruction si

### Critères d'acceptation

**À partir de :** d'une ligne de programme à mémoriser et d'identificateurs auxquels une valeur aura été affectée préalablement ou de constantes littérales de type entier signé.

**Alors :** j'entre une expression composée de deux opérandes de type entier signé et d'un opérateur et l'interpréteur évalue l'expression.

Les opérandes peuvent être :

- deux constantes littérales
- deux identificateurs
- une constante littérale et un identificateur

**Enfin :** si l'expression (condition dans l'instruction) est vraie alors l'exécution continuera à partir du numéro de ligne spécifié par l'étiquette, sinon l'exécution continuera en séquence.

## 11 Expression arithmétique

### Récit d'utilisation

**Titre :** Expression arithmétique

**Récit :** Calcul à l'aide d'expression arithmétique

**En tant que :** Programmeur

**Je souhaite :** que l'Interpréteur LIR effectue une opération arithmétique courante (addition, soustraction, multiplication, quotient ou reste d'une division entière)

**Afin que :** j'en exploite ou vois le résultat

### Critères d'acceptation

**À partir de :** d'une ligne de l'interpréteur ou d'une ligne de programme à mémoriser et d'identificateurs auxquels une valeur aura été affectée préalablement ou de constantes littérales numériques.

**Alors :** j'entre une expression composée de deux opérandes de type entier signé et d'un opérateur.

Les opérandes peuvent être :

- deux constantes littérales
- deux identificateurs
- une constante littérale et un identificateur

**Enfin :** j'obtiens le résultat de l'opération ou un message d'erreur m'informant que l'opération est impossible pour les identificateurs ou constantes littérales saisies.

## Récits d'utilisation proposés lors de l'itération 2