UNIVERSITÉ DE ROUEN

Compilateur LLVM

Langage jouet Kawa

Spécification Technique de Besoin 0.7

17 avril 2015

Auteur(s): Kheireddine Berkane, Nasser Adjibi
Relecteur(s): Pierre-Luc BLOT, Alexandre PETRE

Version	Date	Changelog
0.6	11/04/2015	— Élimination de l'exigence fonctionnelle EF_3 : Compiler une application en mode partagé — Élimination de l'exigence fonctionnelle EF_4 : Compiler une bibliothèques partagée — Élimination de l'exigence fonctionnelle EF_6 : Indiquer les chemins des dépendances entre le source de l'application et des modules (classe/interface) externes — Élimination de l'exigence de réalisation EXR_4 : Compilation d'application partagée — Élimination de l'exigence de réalisation EXR_6 : Compilation de bibliothèque partagée — Élimination de l'exigence de réalisation EXR_25 : Définition d'attributs ou de variables de type **value** — Élimination de l'exigence de réalisation EXR_26 : Définition méthode value — Élimination de l'exigence de réalisation EXR_41 : Gestion des exceptions — Changement du diagramme de cas d'utilisation après la discussion avec client afin de réduire quelques exigences fonctionnelles et de réalisation
0.7	15/04/2015	— Mettre l'exigence fonctionnelle EF_3 : Afficher la version du compilateur en secondaire — Élimination de l'exigence fonctionnelle EF_4 :Activer l'affichage en couleur — Élimination de l'exigence de réalisation EXR_5 : Garbage collector

Table des matières

1 Objet

LLVM est une infrastructure modulaire permettant la réalisation de chaînes de compilation et conçue pour l'optimisation. Elle met en œuvre une représentation intermédiaire du code qui permet de découpler les langages de l'architecture. Notre objectif est de réaliser, à l'aide de l'infrastructure LLVM, un compilateur pour un langage jouet, que nous appellerons **Kawa**, ce dernier doit supporter :

- Les classes, les classes abstraites, les interfaces
- L'héritage
- Le polymorphisme
- Le système de types sera composé des types primitifs (int, float, etc.), des classes et des interfaces
- Les instructions de contrôle telles ques (if/else, for, while/do, switch, etc).

1.1 Objectifs techniques

L'objectif à travers le choix du langage **kawa** qui est similaire à **java** (un langage de haut niveau d'abstraction) c'est de permettre une facilité ainsi qu'une rapidité de développement par rapport à d'autres langages de programmation tel que le «C». Le code Java est compilé dans un langage intermédiaire Bytecode et est exécuté dans un environnement d'exécution JVM (Java Virtual Machine), dans le cadre de notre projet nous allons montrer que l'on peut compiler un code similaire à java (kawa) en code natif sans passer par une machine virtuelle (MV).

Cette manière de compilation permet d'avoir un code machine en un seul passage à l'encontre de compilateur «javac» qui produit le code binaire en deux passages, un premier passage pour générer de pseudo code (bytecode) et le deuxième passage pour optimiser et traduire localement le bytecode en instructions du processeur de la plate-forme.

Pour réaliser le compilateur du langage **kawa** nous allons utiliser l'infrastructure **LLVM**, à travers cette infrastructure moderne nous allons apprendre la compilation d'un langage évolué dans ces différentes étapes jusqu'à la génération de code machine, car LLVM fournit des bibliothèques facilitant l'écriture de front-end (transformation de code source en une représentation intermédiaire **IR**), ainsi que la partie back-end (transformation de la représentation intermédiaire en code machine).

Nous allons bénéficier dans le cadre de ce projet des différents avantages offerts par **LLVM** et **Clang** (compilateur c++ basé sur llvm) tels que :

- Les messages d'erreurs sont beaucoup plus compréhensibles en matière de détails ainsi que de précision par rapport à d'autres messages d'erreurs renvoyés par un autre compilateur, exemple (GCC).
- La rapidité en temps de la compilation ainsi que l'optimisation de la vitesse d'exécution, clang est trois fois plus rapide que GCC, de plus il consomme moins de ressources (faible utilisation de la mémoire).
- Une architecture en bibliothèque du compilateur qui est suffisamment modulaire ce qui rend facile le développement et l'intégration d'autres bibliothèques au sein du compilateur.
- La portabilité sur d'autres architectures.

2 Documents applicables et de référence

Différents documents de référence :

- Le site LLVM llvm.org.
- Le document de spécification du client Spec1.pdf

3 Terminologie et sigles utilisés

— LLVM : (Low Level Virtual Machine) est une infrastructure de compilateur conçue pour l'optimisation à la compilation.
— Kawa : langage jouet qui reprend quelques fonctionnalités de java.
— Clang : compilateur pour le langage c++, son interface de bas niveau se base sur des bibliothèques llvm pour la compilation. Il est utilisé par Apple.
— GIT : logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus.
— ELF : (Executable and Linkable Format) est un format de fichier binaire utilisé pour l'enregistrement de code compilé (objets, exécutables, bibliothèques de fonctions).
— Ubuntu : Distribution linux sur base Debian.
— C++ : Langage de programmation orienté objet bas niveau.
— POO : Programmation orientée objet.
— Makefile : Fichier regroupant des instructions de compilation avec gestion de dépendances.

4 Exigences fonctionnelles

4.1 Présentation de la mission du produit logiciel

Id	Intitulé	Acteur(s)	Priorité
EF_1	Afficher l'aide	Utilisateur	Secondaire
EF_2	Compiler une application en mode monolithique	Utilisateur	Indispensable
EF_3	Afficher la version du compilateur	Utilisateur	Secondaire
EF_4	Activer l'affichage en couleur	Utilisateur	Secondaire

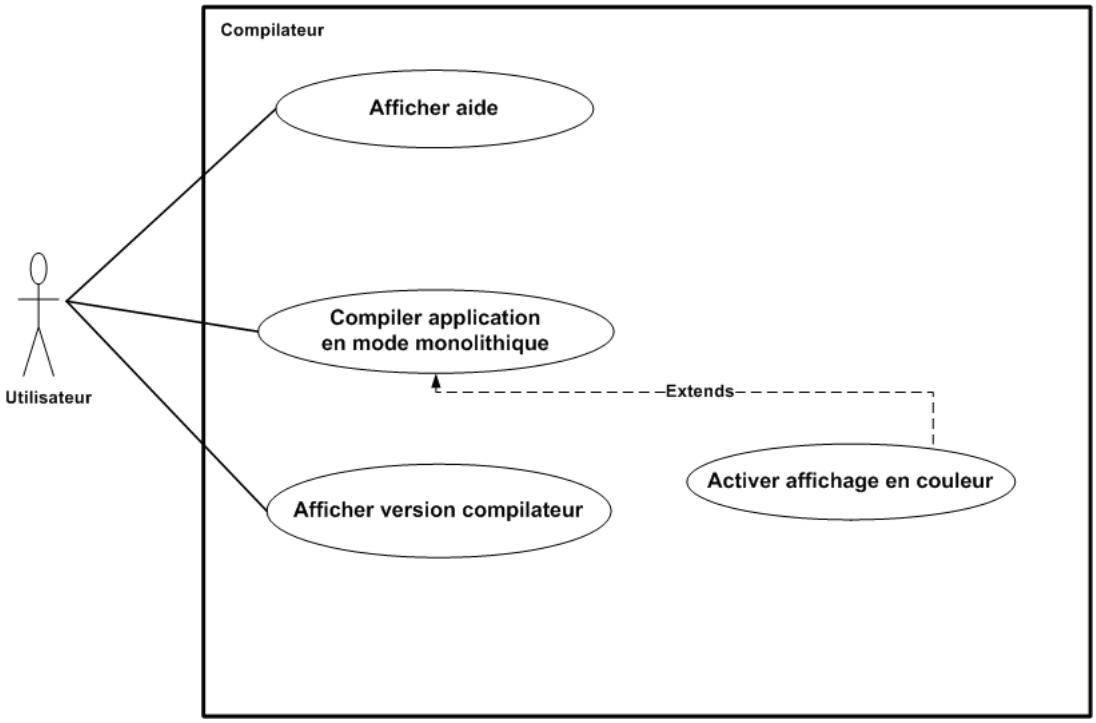

FIGURE 1 – **Cas d'utilisations du compilateur kawa.**

4.2 Cas d'utilisation EF_1

Nom	**Afficher l'aide**
Acteurs concernés	Utilisateur du compilateur
Description	le compilateur affiche la liste des options du compilateur sur la sortie standard à travers une ligne de commande.
Préconditions	L'ordre de priorité entre le commutateur de help (-h ou --help) et le commutateur de version (-v ou --version) est définit par le commutateur en premier paramètre. Le commutateur -h/--help est le premier paramètre obligatoirement.
Evénements déclenchants	Commutateur de ligne de commande :kawac -h ou --help
Conditions d'arrêt	L'aide a été affichée correctement et la main est redonnée à l'utilisateur pour entrer d'autres commutateurs.
Description du flot d'événements principal:	
Acteur(s)	**Système**
Flots d'exceptions:	

4.3 Cas d'utilisation EF_2

Nom	Compiler une application en mode monolithique
Acteurs concernés	Utilisateur du compilateur
Description	L'utilisateur introduit un ensemble de modules (classe/interface) kawa afin de les précompiler et de générer une application sous forme d'un seul exécutable.
Préconditions	Afin de permettre la compilation dans ce mode monolithique il faut : — Indiquer l'emplacement des fichiers sources de l'application. — Introduire le commutateur **-m** en premier paramètre dans la ligne de commande permettant la compilation. — L'absence des deux commutateurs (-h ou --help) et (-v ou --version) en premier paramètre.
Evénements déclenchants	Afin de pouvoir lancer la compilation dans ce mode l'utilisateur doit : — Indiquer au moins un module (une classe avec une méthode main qui constitue le point d'entrée d'une application). — Introduire une ligne de commande avec un commutateur d'action : kawac -m filessources.
Conditions d'arrêt	— La compilation s'est terminée sans erreurs dans ces différentes étapes d'analyses. — La production d'un unique fichier exécutable sous le format d'ELF qui ne dépend d'aucune bibliothèque kawa ainsi que le nom du fichier exécutable correspond au nom du module contenant la méthode main.
Description du flot d'événements principal:	
Acteur(s)	**Système**
Flots d'exceptions:	La compilation peut être interrompue pour divers raisons, on cite : — **EF_2_EXC_1** : Le fichier de sortie n'a pu être créé. — **EF_2_EXC_2** : Le code source d'un des modules de l'application comporte une erreur syntaxique, l'erreur rencontrée pendant cette étape d'analyse est renvoyée sur la sortie standard du compilateur. — **EF_2_EXC_3** : Le code source d'un des modules (classe/interface) dont dépend le programme est introuvable. Un message renvoyé par le compilateur indique le nom du (ou des) module(s) manquants sur la sortie standard. — **EF_2_EXC_4** : Le code source d'un des modules de l'application comporte une erreur sémantique, exemple : l'incompatibilité de type statique lors d'une opération d'affectation. — **EF_2_EXC_5** : Aucun des modules indiqués ne comporte le point d'entrée (main). — **EF_2_EXC_6** : Plusieurs méthodes main ont été trouvées parmi les classes indiquées dans le source, le compilateur renvoie la liste des points d'entrée trouvés sur la sortie standard.

4.4 Cas d'utilisation EF_3

Nom	Afficher la version du compilateur	
Acteurs concernés	Utilisateur du compilateur	
Description	L'utilisateur peut savoir la version du compilateur avec le quel compile ses sources et ses bibliothèques à travers un commutateur de ligne de commande.	
Préconditions	L'odre de priorité entre le commutateur de help (-h ou --help) et le commutateur de version (-v ou --version) est définit par la première occurrence de l'un des deux commutateur i-e si nous avons un **-v** avant un **-h**, le compilateur annule tout le reste et affiche la version	
Evénements déclenchants	Commutateur de ligne de commande :kawac -v ou --version	
Conditions d'arrêt	La version est affichée sur la sortie standard du compilateur.	
Description du flot d'événements principal:		
Acteur(s)		**Système**
Flots d'exceptions:		

4.5 Cas d'utilisation EF_4

Nom	Activer l'affichage en couleur	
Acteurs concernés	Utilisateur du compilateur	
Description	L'utilisateur peut activer l'option de l'affichage en couleur, afin de décorer les messages renvoyés par le compilateur la sortie standard dans les différents modes de compilation.	
Préconditions		
Evénements déclenchants	Commutateur de ligne de commande :kawac filessource --color	
Conditions d'arrêt	Le code présent sur la sortie standard a été colorié.	
Description du flot d'événements principal:		
Acteur(s)		**Système**
Flots d'exceptions:		

5 Exigences opérationnelles

Pas de spécifications concernant cette section.

Id	Intitulé	Priorité

6 Exigences opérationnelles d'interface

Pas de spécifications concernant cette section.

Id	Intitulé	Priorité

7 Exigences de qualité

Id	Intitulé	Priorité
EQ_1	les messages d'erreurs renvoyés par le compilateur doivent être explicites, les plus fines possibles et surtout par rapport à ce qu'il s'est passé.	Important

8 Exigences de réalisation

Id	Intitulé	Acteur(s)	Priorité
EXR_1	Nommage des fichiers de sortie	Utilisateur KAWAC	Secondaire
EXR_2	Reconnaissance de la grammaire KAWA	KAWAC	Indispensable
EXR_3	Compilation d'application en monolithique	KAWAC	Indispensable
EXR_4	Gestion des mots clés	KAWAC	Indispensable
EXR_5	Fichier de sortie au format ELF	KAWAC	Indispensable
EXR_6	Reconnaissance et compilation de classes	KAWAC	Indispensable
EXR_7	Reconnaissance et compilation de classes abstraites	KAWAC	Indispensable
EXR_8	Reconnaissance et compilation d'interfaces	KAWAC	Indispensable
EXR_9	Prise en charge du Polymorphisme	KAWAC	Indispensable
EXR_10	Prise en charge de l'héritage de classe	KAWAC	Indispensable
EXR_11	Prise en charge de l'implémentation d'interfaces	KAWAC	Indispensable
EXR_12	Mécanisme de constructeur	KAWAC & Utilisateur KAWAC	Indispensable
EXR_13	Mécanisme de finalisation	KAWAC & Utilisateur KAWAC	Secondaire
EXR_14	Reconnaissance et définition de méthode	KAWAC & Utilisateur KAWAC	Indispensable
EXR_15	Reconnaissance et définition d'attribut	KAWAC & Utilisateur KAWAC	Indispensable
EXR_16	Reconnaissance et définition des variables locales	KAWAC & Utilisateur KAWAC	Indispensable
EXR_17	Définition d'attributs statiques	Utilisateur KAWAC	Important
EXR_18	Définition d'attributs de constantes	Utilisateur KAWAC	Secondaire
EXR_19	Définition méthode á référence	Utilisateur KAWAC	Important

EXR_20	Définition méthode sans référence	Utilisateur KAWAC	Important
EXR_21	Définition méthode finale	Utilisateur KAWAC	Important
EXR_22	Définition méthode statique	Utilisateur KAWAC	Important
EXR_23	Héritage de méthode	KAWAC	Indispensable
EXR_24	Héritage d'attribut	KAWAC	Indispensable
EXR_25	Redéfinition de méthode	Utilisateur KAWAC	Important
EXR_26	Surcharge des méthode	Utilisateur KAWAC	Important
EXR_27	Mutabilité	KAWAC	Indispensable
EXR_28	Concept de visibilité	Utilisateur KAWAC	Indispensable
EXR_29	Portée de variable	KAWAC	Important
EXR_30	Paramétrage de méthode	KAWAC	Important
EXR_31	Mécanisme de contrôle	Utilisateur KAWAC	Important
EXR_32	Mécanisme de bouclage	Utilisateur KAWAC	Important
EXR_33	Bloc de classe	Utilisateur KAWAC	Indispensable
EXR_34	Bloc d'instructions	Utilisateur KAWAC	Indispensable
EXR_35	Bloc de boucle	Utilisateur KAWAC	Important