SCIENCES SORBONNE UNIVERSITÉ

Auteur : Zhen HOU	30/10/21
Type : STBE	Equipe : Wenzhuo ZHAO, Zhen HOU
Chemin : PISTL/STBE	Statut : Finalisé
Destinataire(s) : Ghjuvan Lacambre	Page(s) 15

Enhancing compiler messages for transfuge programmers

STBE

Version	Date	Modifications	Rédacteur(s)
0.1	08/10/2021	Création	Zhen HOU
0.2	15/20/2021	Ajout Use Case	Zhen HOU
0.2	15/10/2021	Ajout Contraintes, Risques	Wenzhuo ZHAO
0.3	22/10/2021	Ajout Planification	Wenzhuo ZHAO
1.0	22/10/2021	Finalisation	Zhen HOU, Wenzhuo ZHAO
1.1	29/10/2021	Modification	Wenzhuo ZHAO, Zhen HOU

Objectif : *Définition des Spécification Technique des Besoins et des Exigences*

SCIENCES SORBONNE UNIVERSITÉ

Table des matières:

1. Définition de l'application demandée

1.1. Objectifs

L'objectif de ce projet est d'adapter les messages d'erreur [1] du compilateur GNAT, l'interface humaine du compilateur Ada pour GCC, aux erreurs de syntaxe commises par des développeurs nouveaux en Ada, qui disposent des expériences de développement en C/C++, Java ou Python. Ce travail va dans le sens de faire du compilateur d'Ada un "tuteur personnel" en aidant les utilisateurs à apprendre le langage.

1.2. Contexte de l'organisation existante

Les développeurs sont fréquemment confrontés à des barbarismes, c'est-à-dire à des erreurs liées à leur utilisation des conventions d'un langage de programmation lorsqu'ils écrivent dans un autre langage de programmation. Ceci est d'autant plus fréquent lors de l'apprentissage d'un nouveau langage, car les habitudes d'autres langages plus habituels viennent entraver l'apprentissage de la nouvelle syntaxe et de la sémantique.

Pour réaliser l'objectif décrit ci-dessus, nous définissons des procédures et des interactions humaines entre ce logiciel et des utilisateurs:

- Analyse de code saisi par utilisateur et identification des erreurs de syntaxes éventuelles
- Consistant aux différents types d'erreurs, donner un message de correction de code qui est plus compréhensible[2]

1.3. Périmètre du système

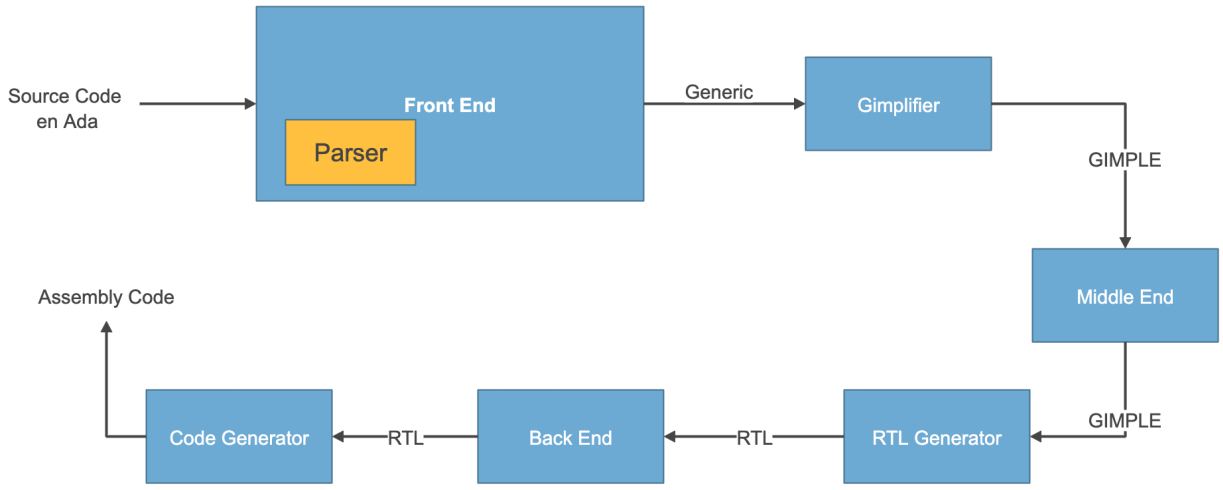
- Analyse des erreurs syntaxique après la compilation d'Ada
- Amélioration des messages des erreurs en les rendant compréhensibles
- On n'est pas chargé d'écrire des documents tutoriels d'Ada

1.4. Interactions entre le système à réaliser et l'organisation existante

- Entrée: un programme Ada, qui peut éventuellement contenir des erreurs syntaxiques
- Sortie:
  - Si le programme ne contient aucune erreur, le compilateur produit le binaire attendu de ce programme
  - Sinon, afficher un(des) message(s) d'erreurs correspondant(s)

2. Définition du système à réaliser

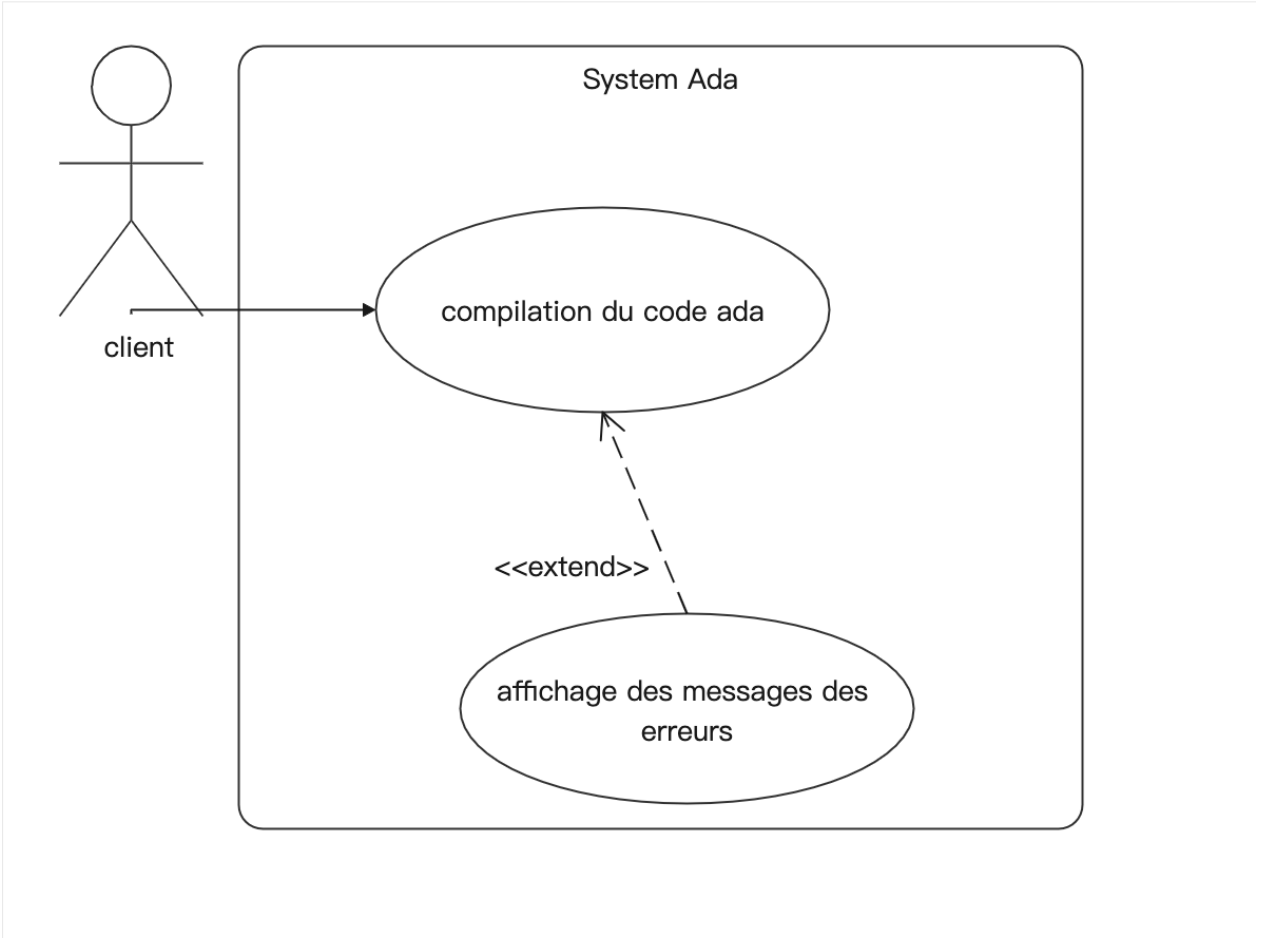
2.1. Architecture fonctionnelle du système



Ce schéma présente une procédure de compilation de programmes Ada. Le compiler du langage Ada, ainsi que du langage C, C++, Objective-C, Java, qui fait partie de GCC[3], contient 3 composants principaux: Front End, Middle End et Back End. Dans ce projet, nous n'intéressons que le parser d'Ada dans le composant Front End.

Pour compiler un programme Ada, il est parsé par le parser, puis rend le produit de parser à l'ensemble de routines internes de GCC pour la génération de code intermédiaire, l'optimisation de ces codes, l'adaptation et des autres sections pour la compilation.

2.2. Interactions entre cas d'utilisations



2.2.1. acteur

client

Nom	utilisateur

Rôle	peut compiler le code sur Ada peut exécuter le code Ada
Type	utilisateur donne un programme Ada

2.2.2. Description détaillée du cas d'utilisation

2.2.2.1. Fiches détaillées

Titre : UC01 Compiler le code

Acteur : Utilisateur

Description : Permet à l'utilisateur de compiler le code

Pré-condition : Implémenter le support de Ada dans GCC

Scénario Nominal :

SN1. L'utilisateur demande au système de compiler le code.

SN2. Le système compile le code avec succès.

SN3. Le système génère le fichier qui peut être exécuté.

Post-condition : Aucune.

Alternatives :

A1. Compile avec des erreurs

En SN2, le système reçoit le code qu'il y a d' erreurs dedans.

A2.1. Le système reçoit un message d'erreur à l'emplacement problématique.

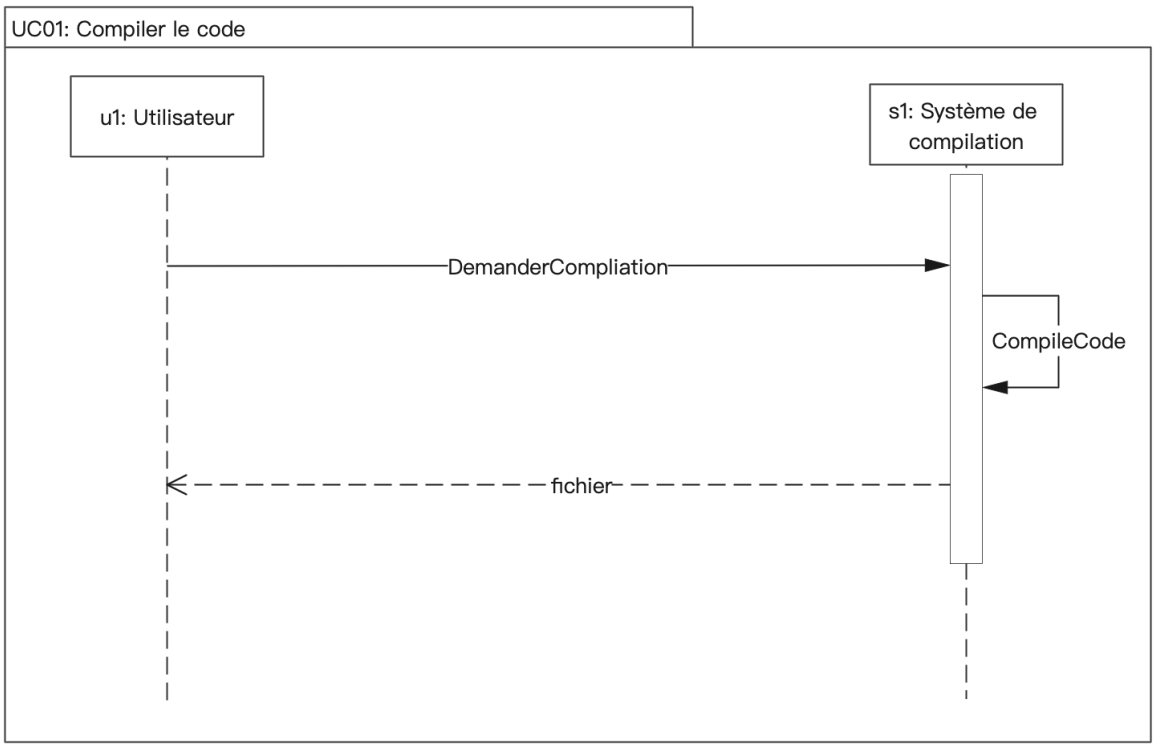
A1.2. Le système affiche le message d'erreur à l'utilisateur.

2.2.2.2. Test de Validation

Nom de cas d'utilisation	Compiler le code Ada
Acteur principal	Utilisateur
Objectif	Compiler le code puis l'utilisateur peut exécuter

Pré-condition	Implémenter le support de Ada dans GCC
Hypothèse	Aucune
Scénario	L'utilisateur compile le code. Le système compile avec succès le code que l'utilisateur saisit.
Résultat attendu	Un fichier compilé qui peut être exécuté
Post-condition	Aucune
Moyen de vérification	Tester si le fichier généré peut s'exécuter

2.2.2.3. Diagramme de séquence



3. Gestion des Contraintes

3.1. Les types de contraintes

En respectant les demandes du client et le délai de livraison, nous sommes obligés de travailler sous des contraintes fonctionnelles et matérielles.

3.1.1. Fonctionnelles

- Lors de la compilation le compilateur indique, lorsqu'il ne peut compiler, des messages d'erreur. Le message d'erreur doit être adapté à l'erreur, et donne une correction possible en supposant l'intention réelle de l'utilisateur.
- Il indique parfois des avertissements (warnings) qui ne l'empêche pas de compiler mais qui selon lui peuvent être source d'erreur. Des avertissements peuvent provenir de confusion d'usage de programme Ada. Notre produit doit aussi pouvoir donner des renseignements sur de bon usages du programme Ada.

3.1.2. Performances et Planning

Le programme de compilateur GNAT compilé, distribué sous le package de GCC et exécuté à la machine d'utilisateur, dépend de la puissance de machine d'utilisateur pour sa performance. Aucune contrainte n'existe pour l'utilisation de notre produit.

3.1.3. Matérielles

- Une configuration avant le développement est nécessaire mais difficile sous certains systèmes d'exploitation. La procédure de configuration est différente sous les ordinateurs personnels de notre équipe. Pour faciliter et accélérer le développement, nous décidons de travailler sous un serveur privé virtuel (VPS). Pour travailler sous ce serveur, une bonne connexion internet est demandée.

3.1.4. Procédure de mise en exploitation du nouveau système

Le code source est à rendre sous GitLab. Le client télécharge les patches et les installe dans le package de logiciel final. Nous n'en occupons pas cette partie.

3.2. Analyse des risques

3.2.1. Risques liés aux contraintes fonctionnelles

Contraintes fonctionnelles: message d'erreur bien adapté	
Description	Pouvoir donner des messages d'erreurs bien compréhensibles et utiles
Causes	Le produit final doit pouvoir donner de bons renseignements sur programme
Conséquences	La livraison prend plus de temps que prévu
Moyen de prévention	- Faire référence de solution pour ce problème de compilateur d'autres langages programmation - Demander éventuellement un avis d'expert sur les moyens d'étude
Gravité	Moyen

3.2.2. Risques liés aux contraintes matérielles

Contraintes fonctionnelles: utilisation d'un VPS pour développement	
Description	Pour travailler sous un serveur, une bonne connexion internet est demandée.
Causes	Besoin de travailler sous une machine publique pour équipe
Conséquences	Impossibilité de travailler sans connexion internet
Moyen de prévention	Chercher des espaces de travail fournissant la connexion internet
Gravité	léger

3.3. Prototypage: Grande ligne de GNAT compilateur

3.3.1. Objectifs

Nous vous présentons plus de détails sur les composants du parser de GNAT en donnant une grande ligne des erreurs que les développeurs de C/C++ ou Java peuvent éventuellement commettre.

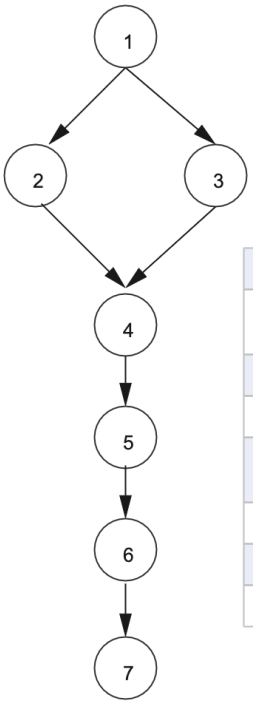
- Déclarations et Control de structure
- Différents types de système
- Fonctions et Procédures
- Paquets
- Objet orienté programme
- Génériques
- Exceptions
- Concurrence
- Programme en bas niveau

4. Planification

4.1. WBS (Work Breakdown Structure)

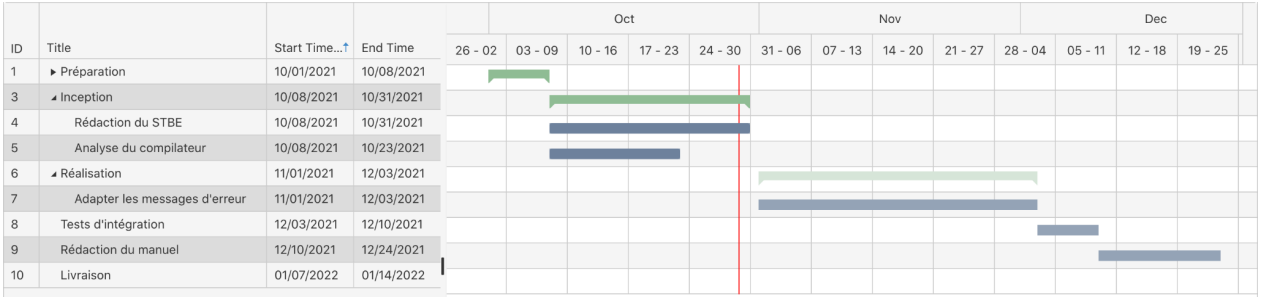

Projet
Inception
Reformulation de la demande
Interview avec le client
Analyse
STBE
Conception
Vue globlae de différentes erreurs
Réalisations
Amélioration de message d'erreurs
Tests d'intégration
Gestion
Tableau de bord
Planning
Livraison continue

4.2. PERT



Numéro de tâche	Nom de tâche	Durée minimum	Durée maximal
1	Location et Configuration de VPS	4 jours	6 jours
2	Rédaction de STBE	13 jours	15 jours
3	Analyse du compilateur	9 jours	11 jours
4	Adapter les messages d'erreur	30 jours	33 jours
5	Integration Tests	5 jours	7 jours
6	Rédaction de manuel	21 jours	22 jours
7	Livraison	1 jours	2 jours

4.3. GANTT



5. Annexes

5.1. Lexique

- GCC: un ensemble de compilateurs créés par le projet GNU. GCC est un logiciel libre capable de compiler divers langages de programmation, dont C, C++, Objective-C, Java, Ada, Fortran et Go.
- GNAT: le compilateur Ada du projet GNU. Il fait partie de GNU Compiler Collection (GCC)

- Parser: L'analyse syntaxique consiste à mettre en évidence la structure d'un texte, généralement une phrase écrite dans une langue naturelle, mais on utilise également cette terminologie pour l'analyse d'un programme informatique. L'analyseur syntaxique (parser, en anglais) est le programme informatique qui réalise cette tâche.
- Compilateur: un compilateur est un programme qui transforme un code source en un code objet

5.2. Bibliographie

1. “Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research.” vol. ITiCSE-WGR ’19, July 15–17, 2019, Aberdeen, Scotland Uk, http://web.eecs.umich.edu/~akamil/papers/iticse19.pdf.
2. “User’s Manual on diagnostics in Clang.” https://gcc.gnu.org/onlinedocs/gccint/Guidelines-for-Diagnostics.html.
3. “An Overview of GCC.” http://bitboom.github.io/an-overview-of-gcc.