1 Astuces et bonnes pratiques pour l'analyse statique

1.1 Workflow d'une analyse statique

Une analyse statique exploite un modèle d'un artefact d'un projet logiciel (principalement son code source), afin d'en extaire des propriétés. Ces propriétés seront ensuite utilisées dans le cadre d'une extraction de connaissance ou application de traitements à des objectifs précis.

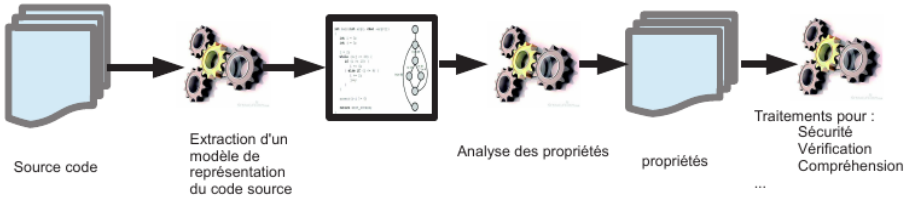

Figure 1: Workflow d'une analyse statique

Chaque entité ainsi figurant dans le workflow devra avoir sa propre description et son propre rôle afin de pouvoir l'exploiter correctement. Premièrement, il faut disposer d'un composant permettant d'exploiter un modèle du code source. On parle d'un **extracteur de modèles**. Par exemple, un **parseur** est un extracteur permettant d'obtenir un **arbre syntaxique abstrait** (*AST* pour *Abstract Syntax Tree*) d'un code source logiciel.

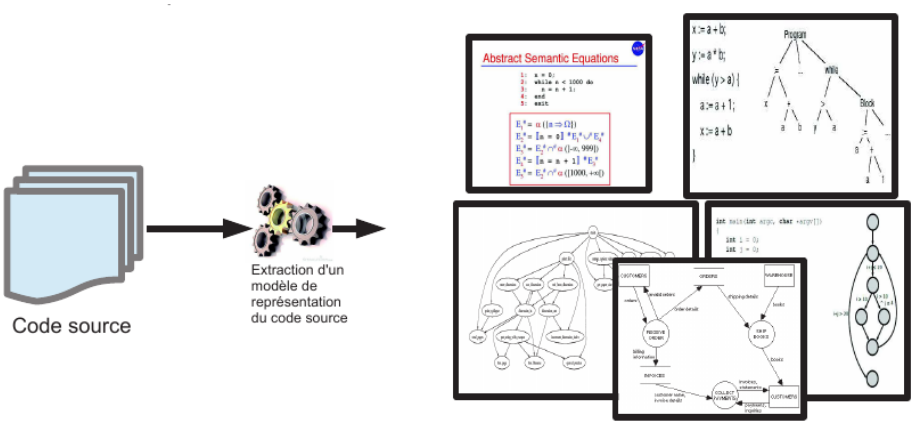

Figure 2: Extraction de modèles d'un code source

Deuxièment, selon le type du modèle extrait, un composant permettant d'en extraire/calculer des propriétés s'avère nécessaire. On parle d'un **extracteur/calculateur de propriétés**. Par exemple, un **visiteur** est un

extracteur de propriétés des noeuds d'un **AST**, alors que les **algorithmes de graphe** sont des extracteurs de propriétés des **graphes de contrôle** (*CFG* pour *Control Flow Graph*) d'une méthode.

Figure 3: Extraction de propriétés d'un modèle du code source

Troisièment, selon les propriétés extraites et les objectifs de l'analyse, un composant **processeur de modèles** peut être utilisé pour jouer le rôle de l'orchestrateur de l'opération. Autrement dit, le processeur (1) référencera un projet logiciel fourni par un client, (2) utilisera un extracteur de modèles pour en extraire le modèle du code source, (3) appliquera des extracteurs de propriétés pour en extraire des propriétés, et enfin (4) réalisera des traitements/calculs plus complexes afin d'en extraire une connaissance ou accomplir un objectif précis.

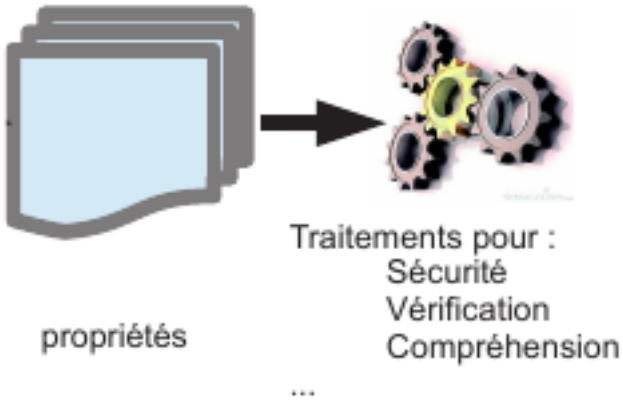

Figure 4: Traitements à partir des propriétés d'un modèle du code source

Finalement, une classe client peut utiliser et interagir avec les composants du

workflow.

1.1.1 Workflow d'une analyse statique employant un AST

Dans le cadre d'une analyse statique employant un AST, le workflow général précédent peut être spécialisé et adopté.

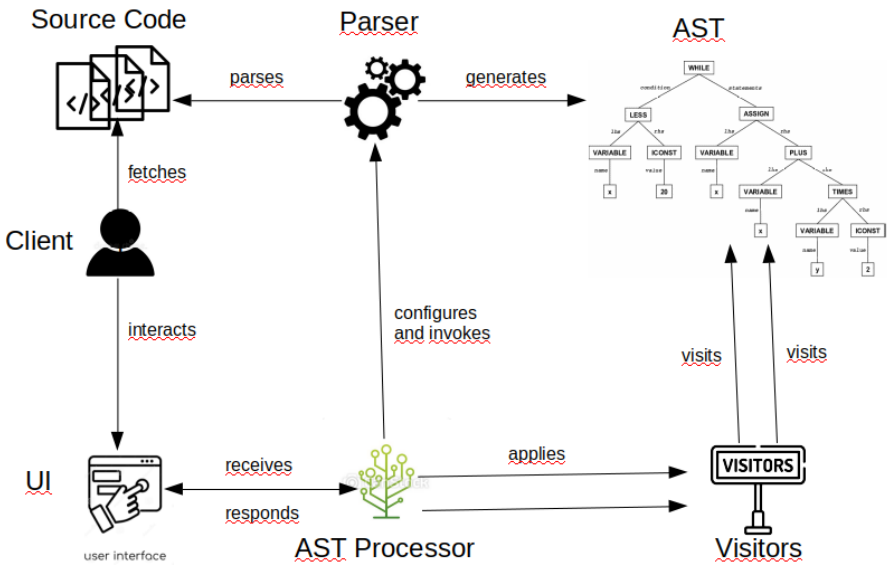

Figure 5: Workflow d'une analyse statique employant des ASTs

Premièrement, l'extraction de l'AST est réalisé par un **parseur d'ASTs**. Des bibliothèques d'analyse statique offrent en général des parseurs de code source. Par exemple `Spoon` et `Eclipse JDT`. Souvent les parseurs fournis ont une API riche et assez complète.

Cepedant, l'utilisation de cette API peut devenir facilement encombrante, et au delà de nos besoins. Dans ce genre de situations, il faut penser au patron de conception `Facade` qui permet de fournir une interface d'abstraction à un/plusieurs systèmes complexes.

L'idée est d'introduire un objet wrapper du système ayant une API complexe. L'objet wrapper permet après d'introduire uniquement les méthodes dont on aura besoin et d'une manière plus déclarative pour faciliter la compréhension du code. D'autres méthodes supplémentaires peuvent être introduites pour combiner l'effet de plusieurs méthodes du système complexe. Par exemple, on peut définir une configuration par défaut du système au sein d'une méthode de l'objet Facade et qui invoque plusieurs opérations de configuration du système complexe.

Deuxièmement, l'extraction/calcul des propriétés d'un AST est réalisé par des **visiteurs**. Les visiteurs, notion introduite par le patron de conception `Visitor`, sont utilisés fréquement dans le cadre d'une analyse statique exploitant des ASTs. En effet, vu qu'un AST est une structure de noeuds de types différents, un visiteur permet d'y associer des traitements dépendant de leurs types. Une hiérarchie de visiteurs peut être utilisée ainsi pour extraire/calculer des propriétés différentes.

Troisièment, la liaison entre le projet, le parseur, et les visiteurs est réifiée au sein d'un **processeur d'ASTs**. Il s'agit d'un processeur utilisant un parseur pour parser l'AST du code source d'un projet, et y appliquant des visiteurs afin d'en extraire des propriétés qu'il utilisera pour extraire une connaissance ou accomplir un objectif précis. Par exemple, un processeur peut être utilisé pour récupérer des informations sur une classe, ou pour réaliser des analyses statistiques sur un projet, etc.

Quatrièment, un client d'une analyse statique utilisant un processeur d'ASTs interagit avec une interface utilisateur (CLI ou GUI) permettant de fournir l'endroit du projet et le type d'analyse à effectuer pour configurer et exécuter le processeur.

1.1.2 Astuces

Le but du TP n'est pas de réutiliser directement le code fourni tel qu'il est. Rappelez-vous que ces bouts de code sont fournis à titre indicatif et démonstratif pur.

Il faut imaginer votre système d'une manière orientée objet. Pensez bien aux patrons de conception, aux principes de conception SOLID, et aux relations entre vos objets. Réifier quand c'est possible et nécessaire. Imaginer une interface utilisateur (CLI ou GUI) qui nous permettra de tester votre travail d'une manière ergonomique.

N'oubliez pas de commiter votre code sur un dépôt distant après avoir implémenté chaque feature. De cette manière vous diviserez vos tâches convenablement et vous laisserez des traces pour observer votre performance.