UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

PROJET DE LA SESSION

PAR

Raphaël Céré

Guanting He

Alexis Lécuyer

Jérémy Veillette

DÉPARTEMENT DE MATHÉMATIQUES ET D'INFORMATIQUE

TRAVAIL PRÉSENTÉ À MAHMOUD MOHANNA

DANS LE CADRE DU COURS INF 1018

ANALYSE DE PROGRAMMES

DÉCEMBRE 2022

Table des matières

Introduction

Lors du présent rapport, il sera sujet de l’aspect mining ou minage d’aspect, de son utilité et du lien avec le cours que nous avons suivi. Premièrement, il faudrait définir ce qu’est l’aspect mining il tente d'identifier les préoccupations transverses dans le contenu. Il peut également fournir des informations permettant de classer les aspects communs qui se produisent dans différents types de contenu, tels que les actualités et les données sociales. Nous commencerons par expliquer plus en profondeur ce qu’est l’aspect mining en expliquant, par exemple, certains des sujets qui y sont liés tels que les aspects ou encore les préoccupations transversales. Ensuite, il sera sujet de l’utilisation de l’aspect mining ou, en d'autres mots, du pourquoi de son développement. et finalement, nous ferons un lien entre le cours et les concepts liés à l’aspect mining. Le lien entre les concepts d’aspect mining et ce que nous avons vu en classe sera établi au fur et à mesure qu’un concept sera exploré.

Thème spécifique (Recherche)

Qu’est-ce que l’aspect mining et les préoccupations transverses?
L’aspect mining consiste à découvrir des préoccupations transverses dans le code source orienté objet d’une application. Une préoccupation transverse est une problématique qui ne se représente pas dans une seule classe, car en fonction du contexte où elle se trouve, son application peut varier. Un bon exemple est un système de logs. D’une classe à l'autre, on désire peut-être avoir une sortie différente pour les logs. Elles posent un sérieux problème dans plusieurs applications, car elles créent beaucoup de duplication de code, elles sont éparpillées dans l’application et leurs comportements sont plus difficiles à comprendre. Celà nuit à la maintenabilité et la compréhension des programmes. La programmation orientée aspect est une solution à cette problématique et l’aspect mining est une activité essentielle pour la transition de systèmes vers ce modèle.

Quelles sont les motivations d’utiliser l’aspect mining?
L’aspect mining est une activité presque essentielle pour migrer un système vers un modèle orienté aspect. Puisqu’elle permet d’identifier les préoccupations transverses, on peut ensuite faire une identification des aspects à créer. En Java, l’extension de langage AspectJ permet la gestion des aspects.

Quelles approches sont possibles pour l’appliquer?
Il existe deux types d’approches principales pour identifier le code problématique. La première intitulée “Navigateurs dédiés” est une forme de programme avancé qui navigue le code source d’un projet dans le but d’y trouver des préoccupations transverses. Généralement, il faut leur fournir un bout de code en particulier à inspecter. Un exemple d’analyseur qui pourrait aider à faire ces identifications est JavaCC. La deuxième intitulée “Identification automatique d’aspects candidats” est complémentaire à l’approche précédente. Cette approche regroupe plusieurs techniques qui, en manipulant le code ou en l’exécutant permettent d’obtenir différents indices à propos de la présence potentielle de préoccupations transverses.

Quelques techniques d’identification automatique d’aspects candidats
Une de ces techniques consiste à analyser les appels de fonctions à l’exécution, dans le but d’y trouver des patrons récurrents. Cela observe donc le type de fonction appelé, ses paramètres, son ordonnancement et son contexte. Par exemple, on appelle toujours une fonction à l’avant dernière ligne précédant le “return” et ses paramètres changent. Cela peut potentiellement indiquer la présence d’une préoccupation transverse.

L’analyse conceptuelle formelle de traces d’exécution consiste à essayer d’analyser les traces d’exécutions dans le but de regrouper des objets par groupes de fonctionnalités et ressemblances d’attributs. Cette forme d’analyse se base sur le principe des treillis, une structure algébrique qui représente différents liens entre des points. Lorsqu’on utilise un outil tel que Dynamo, du code qui sert à observer l’exécution est ajouté au code à analyser, puis lors de l’exécution, plusieurs traces sont conservées. Une fois complétée, la sortie obtenue est ensuite

analysée par un algorithme d'analyse formelle de concept de sorte à analyser l'information obtenue par le code d'observation. On peut obtenir de ce genre de technique des informations à propos de l'éparpillement des attributs et méthodes et de l'entrecroisement de méthodes d'un même concept. À partir de ces informations, on peut avoir des indices sur les potentielles préoccupations transverses.

Lors d'une analyse formelle de concept de nom et de méthode, on effectue une analyse des identifiants à l'aide d'un algorithme dit FCA (Formal Concept Analysis). L'idée derrière cette approche réside dans le fait que les préoccupations intéressantes dans le code source sont reflétées par une convention de nommage dans les classes et les méthodes du système. Lors de l'utilisation de cet algorithme, il va devoir y être envoyé en tant qu'entré objet les classe et les méthode du programme et en tant qu'entrée de type attribut, il prendra des sous-chaîne (substring) générées par des entités du programme qui y sont utilisées en tant qu'objet. Par exemple, une classe nommée QuotedCodeConstant est divisée en chaînes 'Quoted', 'Code' et 'Constant'. Les sous-chaînes avec peu de sens, comme "a", "avec", etc. sont écartées des résultats. Les concepts résultants consistent de groupes maximaux d'entités de programme qui partagent un nombre maximal de sous-chaînes. Après avoir filtré les concepts sans importance, il reste un grand nombre de concepts qui doivent être inspectés manuellement. En plus d'être capable de détecter un certain nombre d'idiomes de programmation, de modèles de conception et certaines opportunités de refactoring en limitant les concepts à ceux qui sont transversaux (c'est-à-dire que les méthodes et classes impliquées appartiennent à au moins deux hiérarchies de classes différentes), la même approche peut également être utilisée pour l'extraction d'aspects.

Quelles sont les limites de l'aspect mining?

Bien que l'aspect mining puisse être utilisé dans de nombreuse situation, il n'en reste pas moins que c'est un outil et qu'il est muni de limitations comme n'importe quel autre. Par exemple, l'une de ses grandes limitations provient du fait qu'il n'existe pas de définition claire et exacte pour les préoccupations transverses. Certains diraient qu'il s'agit d'un comportement systématique dont la mise en œuvre est éparpillée dans le reste d'une autre mise en œuvre et d'autres diraient qu'il s'agit de propriétés qui ne peuvent pas être proprement encapsulées dans une procédure généralisée. Une autre limitation de l'aspect mining provient du fait que les techniques actuelles d'aspect mining sont insuffisantes pour distinguer le code dispersé résultant de l'absence d'objets du code dispersé qui apparaît en raison de l'absence d'aspects. Cette limitation des techniques d'aspect mining se produit parce qu'elles lient le code dispersé à des aspects, indépendamment du fait que le code dispersé apparaît dû à l'absence de certaines abstractions ou dû à des limitations inhérentes des mécanismes orientés objet pour encapsuler les préoccupations transversales. Certaines limitations ont toutefois des solutions, par exemple, les préoccupations transverses doivent être bien définies dans le cadre d'une utilisation de l'aspect mining.

Conclusion

En conclusion, l'aspect mining est un outil efficace pour améliorer les programmes initialement programmés en orienté objet, car il permet de les moderniser et de les transformer en orienté aspect. Bien que la technique ait quelques petites limitations, ces limitations n'en restent pas moins que mineures et elles n'ont pas un impact majeur ou marqué sur son utilisation. Nous avons vu, lors de cet article, la pertinence de l'utilisation de l'aspect mining pour extraire les préoccupations transversales qui causent une répétition dans les programmes. Il a aussi été vu les approches utilisées lors de l'utilisation de l'aspect mining, par exemple, le navigateur dédié qui navigue le code pour trouver les préoccupations transversales ou encore l'identification automatique d'aspects candidats qui, elle, est complémentaire du navigateur dédié. Les techniques d'identification des préoccupations ont aussi été touchées lors de ce rapport. Tous ces points sur lesquels nous avons porté notre attention amène une question, est-ce que l'aspect mining se montrera être un outil marquant et aura-t-il un impact important dans le futur de l'industrie?

Bibliographie

NER & Aspect Mining - PaloServices
https://d-nb.info/992172780/34
Object Identification and Aspect Mining in Procedural Object-Oriented Code (hal.science)
(PDF) A Survey of Aspect Mining Tools and Techniques | Kim Mens - Academia.edu