Projets à réaliser dans le cadre de l'UE « Evolution et Restructuration des logiciels » HMIN306.

Informations :

- Un projet peut être réalisé par une équipe constituée de 1 à 3 personnes
- Plusieurs équipes peuvent être affectée à un projet à condition que le nombre maximum de personnes travaillant sur un même projet n'excède pas 05.
- Un lien pour l'inscription à un projet sera disponible sur le site de M. Seriai
- La remise et la présentation des projets sont programmées pour la semaine du 16 au 20 janvier 2016.
- Le livrable de chaque projet doit inclure :
    - Un rapport de réalisation
    - Le code de l'application réalisée
    - Un petit manuel d'utilisation

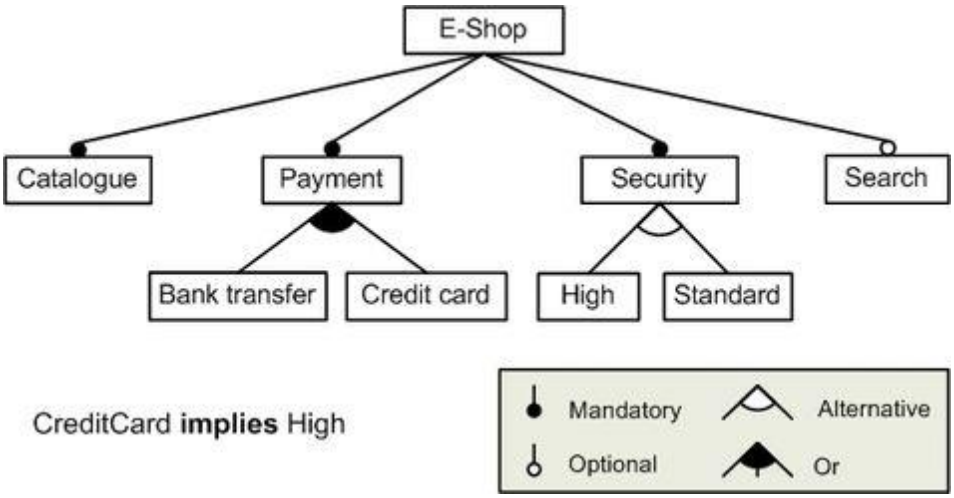
1. Extraction de modèles de caractéristiques par l'analyse de variantes de logiciels orientés objet.

Il s'agit d'analyser le code source de plusieurs variantes d'une application orientée objet en Java pour :

- Identifier tous leurs éléments du code source
- Identifier les éléments communs et les éléments variables entre ces variantes en se basant sur FCA
- Identifier les Features de chaque application en appliquant un algorithme de partition ou un algorithme de clustering

Toutes les informations sont disponibles dans les documents référencés ci-dessous :

a. http://www.lirmm.fr/~seriai/encadrements/theses/rafat/uploads/Divers/iri44.pdf
b. http://www.lirmm.fr/~seriai/encadrements/theses/rafat/uploads/Divers/iriPres.pdf
c. http://www.lirmm.fr/~seriai/encadrements/theses/rafat/uploads/Divers/seke.pdf
d. http://www.lirmm.fr/~seriai/encadrements/theses/rafat/uploads/Divers/lille.pdf
e. http://www.lirmm.fr/~seriai/encadrements/theses/rafat/uploads/Divers/seke2.pdf
f. https://tel.archives-ouvertes.fr/tel-01015102/document
g. MobileMedia : http://homepages.dcc.ufmg.br/~figueiredo/spl/icse08/



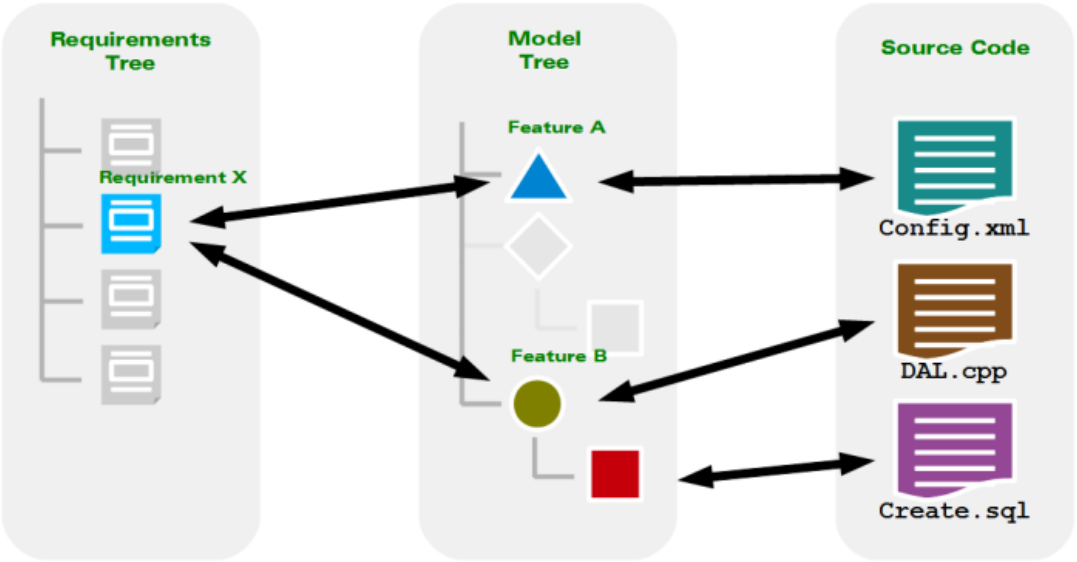
2. Traçabilité entre caractéristiques et codes sources de variantes de logiciels orientés objet.

Il s'agit d'analyser le code source de plusieurs variantes d'une application orientés objet en Java pour :

- Identifier tous leurs éléments
- Identifier les éléments communs et les éléments variables entre ces variantes en se basant sur FCA
- Identifier la traçabilité entre les éléments du code et une liste de Features (caractéristiques) dont une description est donnée pour chaque feature : c-a-d : identifier quel code implémente quel feature. Cette traçabilité est basée essentiellement sur la méthode LSI.

Toutes les informations sont disponibles dans les documents référencés ci-dessous :

a. http://www.lirmm.fr/~dony/postscript/spl-iri13.pdf
b. http://homepages.dcc.ufmg.br/~figueiredo/spl/icse08/

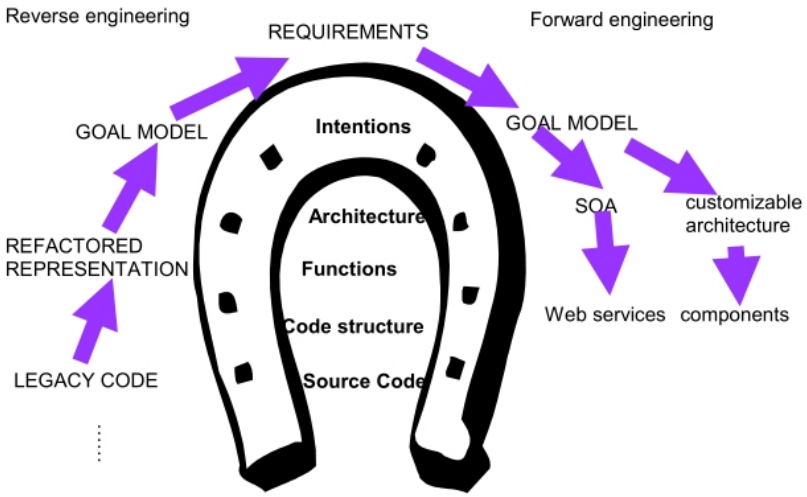


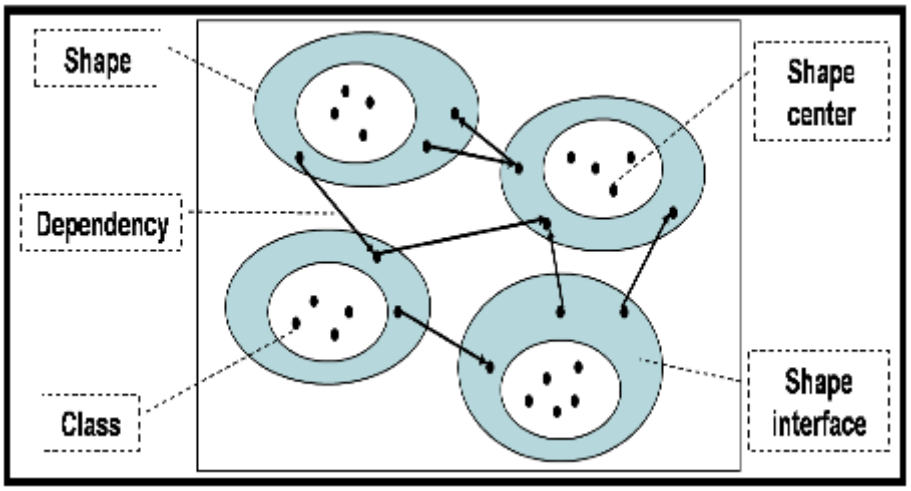
3. Extraction d'architecture de composants/services par l'analyse statique du code source.

Il s'agit de réaliser une analyse statique du code source d'une application orientée objet en Java et d'identifier son architecture à base de composants/service/modules sachant qu'un type de composant est un cluster de classes. Le regroupement des classes en se basant sur un algorithme de regroupement (regroupement hiérarchique, génétique, recuit simulé, tabou, etc.). L'algorithme de regroupement se base sur certaines les valeurs de certaines métriques (exemple couplage et cohésion entre les classes) pour grouper certaines classes ensemble dans un même type de composant.

Toutes les informations sont disponibles dans les documents référencés ci-dessous :

a. https://pdfs.semanticscholar.org/d74b/21d4c3625a8611d4d6d56336a37b28cee31a.pdf
b. https://tel.archives-ouvertes.fr/tel-00456367/document





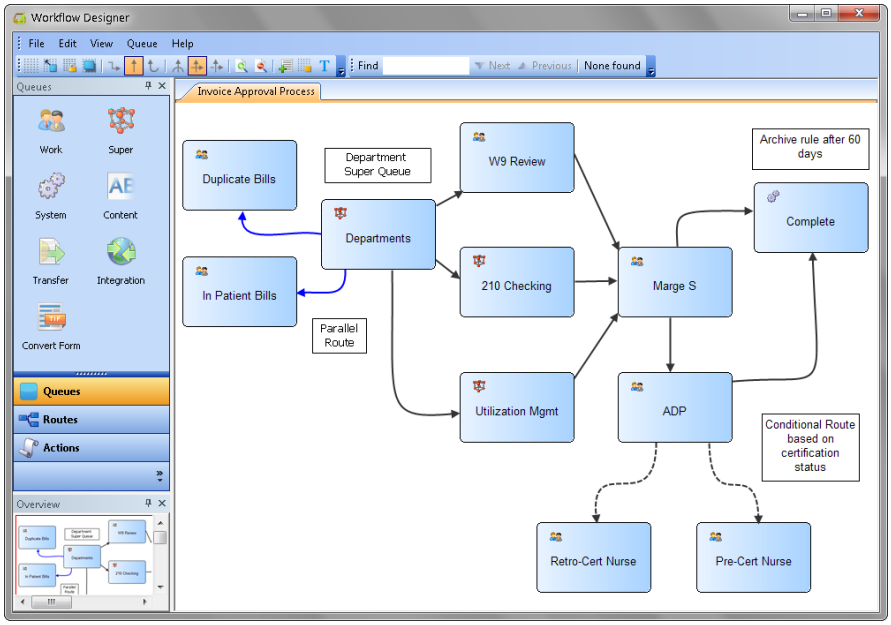
4. Extraction de workflow (diagramme d'activités) par l'analyse statique du code source orienté objet.

L'objectif est d'analyser le code source d'une application orientée objet en JAVA et d'extraire le workflow (diagramme d'activités) de cette application. Pour information ce diagramme peut être utile pour la compréhension du comportement de l'application ou pour l'optimisation de son dépliement sur le cloud.

La méthode d'extraction du workflow est disponible sur le lien : http://www.lirmm.fr/~seriai/uploads/Enseignement/workflow1.pdf

Liens pour comprendre la notion du workflow :

- https://fr.wikipedia.org/wiki/Workflow
- http://www.pnmsoft.com/resources/bpm-tutorial/workflow-tutorial/
- http://docs.alfresco.com/4.0/concepts/wf-whatis-workflow.html
- http://www.commentcamarche.net/contents/332-workflow-gestion-des-processus-metiers
- https://en.wikipedia.org/wiki/Scientific_workflow_system
- http://www-sop.inria.fr/members/Patrick.Valduriez/pmwiki/Patrick/uploads/Publications/jogc2015



5. Implémentation d'une partie de l'approche GumTree en Java

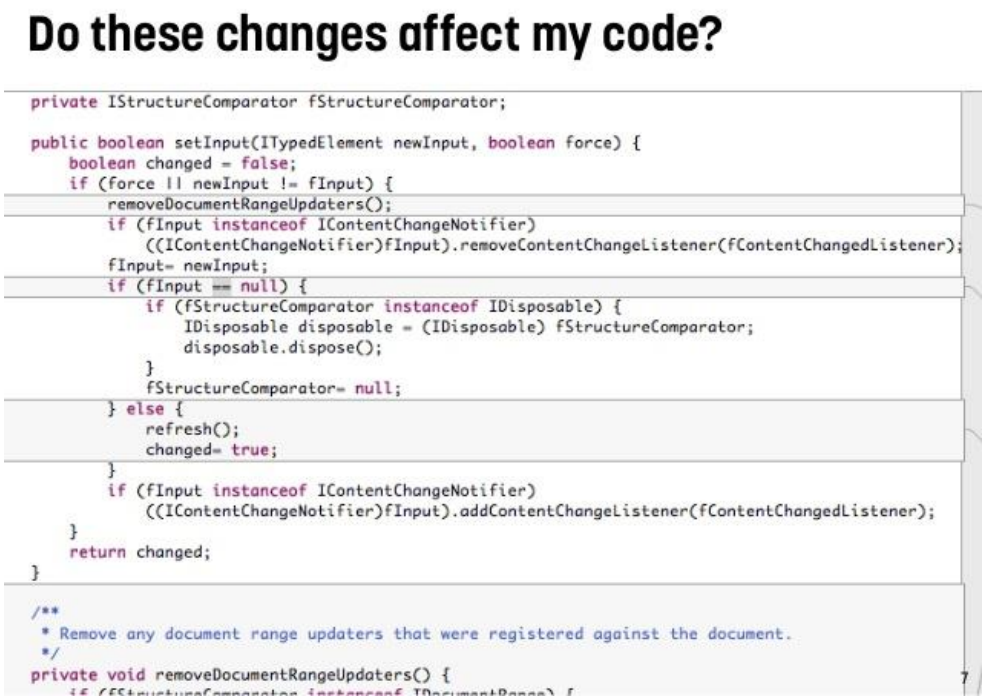
GumTree is a complete framework to deal with source code as trees and compute differences between them. It includes possibilities such as:

- converting a source file into a language-agnostic tree format
- export the produced trees in various formats
- compute the differences between the trees
- export these differences in various formats
- visualize these differences graphically

Compared to classical code differencing tools, it has two important particularities:

- it works on a tree structure rather than a text structure,
- it can detect moved or renamed elements in addition of deleted and inserted elements.

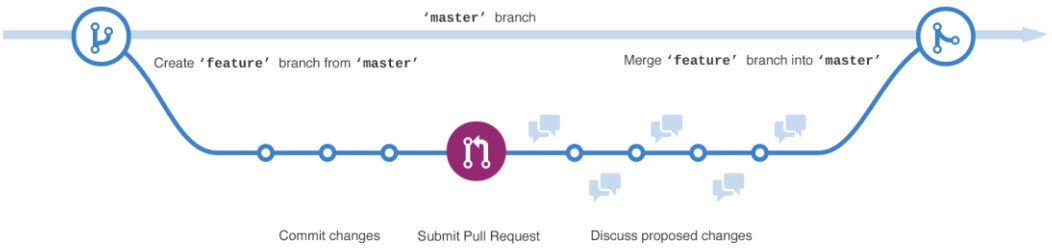
a. https://hal.archives-ouvertes.fr/hal-01054552/document
b. https://github.com/GumTreeDiff/gumtree
c. https://github.com/GumTreeDiff/gumtree/wiki



6. Fouille des données des GITs

Il s'agit de définir une application permettant de consulter des projets déposés sur un GIT

- Pour extraire un ensemble d'informations de base, utiles par rapport au déroulement des projets tels que les commits, types de commit, acteurs des commits, code concerné par le commit, etc.
    - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
    - http://www.lirmm.fr/~seriai/uploads/Enseignement/git1.pdf
- En plus de ces informations, les étudiants travaillant sur ce projet doivent s'inspirer de la lecture d'un ou de plusieurs articles pour proposer d'autres informations déduites des informations de base. Des exemples de ces articles sont donnés ci-dessous :
    - http://www.lirmm.fr/~seriai/uploads/Enseignement/git2.pdf
    - http://www.lirmm.fr/~seriai/uploads/Enseignement/git3.pdf
    - http://www.lirmm.fr/~seriai/uploads/Enseignement/git4.pdf



7. Timeline sémantique de projet avec Diggit

Dans ce projet il sera demandé de réaliser une analyse Diggit qui permet de construire une timeline sémantique de l'activité de projets open-source. Le principe est de se baser sur des projets qui publient régulièrement sur la branche master des commits associés à des tags de versions (1.2, 1.3, 1.3.1 comme sur https://github.com/doublespeakgames/adarkroom) .
L'objectif est d'analyser automatiquement les commits qui ont eu lieu entre deux tags pour déduire ce qu'il s'est passé en terme de développement sur le projet entre les deux versions. Les actions que nous considèrerons entre deux versions sont : ajout de fonctionnalités, amélioration des tests, amélioration de la documentation, bugfixes… Pour cela, il faut analyser plusieurs informations contenues dans les commits: les fichiers impactés, le contenu du changement ainsi que le message de commit. En résumé, au cours de ce projet, il faudra :

- Trouver un petit corpus de projet qui contient des tags sur la branche master
- Réaliser une analyse diggit qui
    - Extrait la liste des tags
    - Ordonne la liste des tags
    - Extrait la liste des commits entre chaque couple de tags adjacents
    - Extrait les actions de développement
    - Sauvegarde le résultat dans un fichier texte
- Faire tourner l'analyse sur les projets du corpus