**HMIN306 – Evolution et Maintenance de Systèmes Logiciels**
**Master 2 Informatique**

# TP 1 : Généralités

## Exercice 1 : Compréhension des concepts liés à l'évolution/maintenance des logiciels

Réalisez un modèle UML montrant les concepts liés au domaine de l'évolution/maintenance de logiciel ainsi que leurs relations.
Exemple de ces concepts : maintenance, évolution, réingénierie, rétro ingénierie, adaptation, analyse statique, analyse dynamique, dette technique, « slicing », extraction d'architecture, extraction de modèles, refactoring, migration, restructuration, correction de bugs, impact des changements, localisation de feature, intégration continue, DevOps, compréhension de logiciel, « code review », « software mining », redocumentation, etc.

## Exercice 2 : Outils de maintenance/évolution

La liste ci-dessous présente un ensemble d'outils utilisés dans le domaine de l'évolution/maintenance des logiciels. Choisissez deux outils parmi cette liste, étudiez ces logiciels, installez-les, utilisez-les et préparez un bilan d'utilisation.

1. **SonarQube :**
   - https://fr.wikipedia.org/wiki/SonarQube
   - https://www.sonarsource.com/
   - https://linsolas.developpez.com/articles/java/qualite/sonar/?page=page_1
   - http://www.methodsandtools.com/tools/tools.php?sonar
   - https://www.baeldung.com/sonar-qube
   - https://www.sourceallies.com/2010/02/sonar-code-quality-analysis-tool/
2. **FindBugs :**
   - http://findbugs.sourceforge.net/
   - https://fr.wikipedia.org/wiki/FindBugs
   - https://www.baeldung.com/intro-to-findbugs
   - https://www.commentcamarche.net/faq/19006-installation-et-utilisation-du-plugin-findbugs-d-elipse
   - https://github.com/findbugsproject/findbugs
3. **Soot :**
   - http://sable.github.io/soot/

- https://www.sable.mcgill.ca/soot/tutorial/pldi03/tutorial.pdf
- https://en.wikipedia.org/wiki/Soot_(software)
- https://github.com/Sable/soot/wiki/Tutorials
- https://courses.cs.washington.edu/courses/cse501/01wi/project/sable-thesis.pdf

4. **Gummtree :**
   - https://github.com/GumTreeDiff/gumtree
   - https://www.labri.fr/perso/falleri/perso/tools/gumtree/
   - https://hal.archives-ouvertes.fr/hal-01054552/document
   - http://courses.cs.vt.edu/cs6704/spring17/slides_by_students/CS6704_gumtree_Kijin_AN_Feb15.pdf
   - https://www.openhub.net/p/gumtree
5. **MicroART:**
   - https://github.com/microart/microART-Tool
   - https://fr.slideshare.net/paolodifrancesco/microart-a-software-architecture-recovery-tool-for-maintaining-microservicebased-systems
   - https://www.researchgate.net/publication/317927398_MicroART_A_Software_Architecture_Recovery_Tool_for_Maintaining_Microservice-Based_Systems
6. **CodeCity and JSCity:**
   - https://wettel.github.io/codecity.html
   - https://wettel.github.io/download/Wettel08b-wasdett.pdf
   - https://www.researchgate.net/publication/221555855_CodeCity_3D_visualization_of_large-scale_software/link/02bfe513998dce533f000000/download
   - https://marketplace.eclipse.org/content/codecity
   - https://fr.slideshare.net/esug/codecity-esug2008
   - https://github.com/ASERG-UFMG/JSCity/wiki/JSCITY
   - https://bjoernkw.com/2016/11/27/jscity-code-complexity-visualization-for-javascript-codebases/
7. **But4Reuse :**
   - https://but4reuse.github.io/
   - https://github.com/but4reuse/but4reuse/wiki
   - https://hal.sorbonne-universite.fr/hal-01531890/document
8. **SonarGraph (SonarJ) :**
   - https://www.hello2morrow.com/products/sonargraph
   - https://www.hello2morrow.com/products/sonargraph/explorer
9. **KDiff3 :**
   - http://kdiff3.sourceforge.net/
   - https://www.fosshub.com/KDiff3.html
10. **JArchitect :**
    - https://www.jarchitect.com/
    - https://en.wikipedia.org/wiki/JArchitect
11. **Checkstyle :**
    - https://checkstyle.sourceforge.io/
    - https://github.com/checkstyle/checkstyle
    - https://www.baeldung.com/checkstyle-java

## Exercice 3 : Analyse d'approches en maintenance et évolution logicielle

La liste ci-dessous présente un ensemble de papiers/articles présentant certaines approches/techniques de maintenance/évolution de logiciels. Choisissez un papier dans cette liste, étudiez-le et préparez une synthèse résumant votre compréhension de son contenu.

1. Do Programmers do Change Impact Analysis?
   - https://www3.nd.edu/~cmc/papers/jiang_emse16.pdf
2. Supporting Microservice Evolution
   - https://www.cs.ubc.ca/~bestchai/papers/microservices-icsme17-nier.pdf
3. The Pricey Bill of Technical Debt: When and by Whom will it be Paid?
   - https://www.researchgate.net/publication/320057934_The_Pricey_Bill_of_Technical_Debt_When_and_by_Whom_will_it_be_Paid
4. Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing
   - http://www.cs.wm.edu/~denys/pubs/ICSME'17-CEL.pdf
5. A Comparative Study of Software Bugs in Clone and Non-Clone Code
   - https://pdfs.semanticscholar.org/ba84/7021cdf1aed7fed4c9bf5bdd3ca2225f009c.pdf
6. Software Practitioner Perspectives on Merge Conflicts and Resolutions
   - http://web.engr.oregonstate.edu/~nelsonni/docs/icsme17-mckee.pdf
7. On the Optimal Order of Reading Source Code Changes for Review
   - https://tobiasbaum.github.io/rp/optimalordering.pdf
8. Bug or Not? Bug Report Classification Using N-Gram IDF
   - https://www.researchgate.net/publication/320883291_Bug_or_Not_Bug_Report_Classification_Using_N-Gram_IDF
9. Refactoring Asynchrony in JavaScript
   - https://www.cs.ubc.ca/~bestchai/papers/promises-icsme17.pdf
10. CCLearner: A Deep Learning-Based CloneDetection Approach
    - https://pdfs.semanticscholar.org/5af1/e0ccb954866899c6bf94c7b60b13c231e0a7.pdf