ISIMA – ZZ2 F2 et F3 - Architectures Logicielles et Qualité
Etude de cas – Réalisation d'un petit logiciel

La mise en œuvre d'un petit logiciel se réalise en binôme ou trinôme il donne lieu à une rédaction en 2 parties. Pour la filière 2, il s'agit de réaliser un code de simulation multi-agents (SMA : intelligence artificielle distribuée sur un domaine de leur choix). Pour la filière 3, même si un sujet de multi-agent est possible en l'orientant sur les négociations et les échanges socio-économiques, il vous sera peut-être plus simple de choisir de développer un petit code en lien avec la Business Intelligence (cela peut-être une fonctionnalité Web pour un tableau de bord, un outil pour du CRM – Customer Relationship Management – un outil pour du reporting, ou tout autre outil de votre choix.

Le choix spécifique du sujet est de votre ressort ainsi que le choix du langage et de l'environnement de développement. C'est un choix qui peut donc être en lien avec vos goûts, votre projet professionnel, vos sujets de stage. Cette autonomie forte fait partie de la difficulté, elle a l'avantage de vous préparer au stage et d'augmenter votre capacité à trouver comment vous dépanner.

Pour cette étude de cas logicielle, il y a 2 comptes rendus attendu.

1) Un petit dossier « analyse métier » - le choix du domaine de votre logiciel, sa description synthétique

Dans votre rédaction vous présenterez votre cahier des charges avec votre analyse des besoins, vos choix et spécifications, les classes et les règles retenues… Vous utiliserez UML, mais vous ne rentrerez pas encore dans la conception (la conception propose une solution le COMMENT – l'analyse n'expose que le contexte du domaine et les objectifs de votre application – le QUOI). Restez modestes sur les objectifs, c'est un TP long mais pas un projet. La réalisation de cette petite étude de cas se fera sur environ 2 mois.

Vous essayerez également de donner des éléments de planification des principales tâches de développement (c'est un planning supposé – il n'est jamais 'exact') ; de même il y aura certainement des variations dans le sujet et les fonctionnalités. Comme sur un développement réel on commence petit, et on avance étape par étape en les validant au fur et à mesure. Si possible lors du 2nd rapport vous fournirez un diagramme de Gant final – celui réellement effectué et vous pourrez le comparer au 1er en montrant les différences avec vos prévisions.

Dans la 1ère phase de rédaction (analyse), vous n'avez pas en entrer dans la conception abstraite ou détaillée, et encore moins dans l'implémentation. Cela fera l'objet de la rédaction du 2nd compte rendu de cette étude.

Vous rédigerez votre analyse pour un volume de 4 à 6 pages avec les diagrammes – elle est à rendre par mail pour le lundi 9 janvier 2023 (le retard est pris en compte dans la notation).

2) Une dossier pour la conception et implémentation

La conception et développement concernent la deuxième partie de la rédaction. Un langage à Objet est vivement conseillé, mais pas obligatoire dans un contexte de développement Web. Les contraintes pour la conception et l'implémentation sont les suivantes :

- Utilisation autant que possible de 3 patrons de conception. Voici 3 exemples de patrons candidats pour ceux qui font un SMA : le patron composite (groupe d'agent), le patron stratégie (comportement d'agents), les patrons MVC/Observer, state et singleton peuvent être aussi être considérés ainsi que d'autres pour des développements Web.

- Un outil de documentation automatique de type Doxygen ou JavaDoc à découvrir et à utiliser.
- Vous utiliserez au moins 2 autres outils du génie logiciel pour ce TP (1 par personne dans le cas d'un binôme), 3 outils dans le cas d'un trinôme (1 outil par personne). Chaque personne choisit un outil qu'elle a envie de découvrir ou d'approfondir, cet outil est mis en œuvre sur l'étude de cas logicielle.
- L'approfondissement de Git / GitLab est fortement conseillée (GitLab est disponible à l'ISIMA).

La rédaction finale comporte un maximum 10 pages de présentation du développement en binôme (ou trinôme). Nous espérons être en présentiel avec des petites démonstrations à mi-parcours. L'évaluation de la qualité du code source, sera également faite. Des guides de styles sont donnés à titre d'exemple.

Vous pourrez ensuite à la fin de ce 2nd compte rendu, dans la finale du rapport, tout ce qui a été changé par rapport à vos 1ère options d'analyse et/ou de conception. Ces changements (qui sont normaux) ont induit des délais de développement, un décalage des différentes tâches que vous commenterez.

Ce petit logiciel vous initie de façon modeste à une gestion de projet de développement. Pour les binômes : une proposition vous est faite : mise en œuvre simplifiée de la méthode eXtreme Programming. Documentez-vous, il d'agit d'un cycle testeur/développeur. On parle aussi de « Test Driven Development ». Dans ce cas vous pourrez donner un pourcentage de couverture de test du logiciel. Tout comme la présentation de l'analyse, la présentation de la conception se fera avec UML.

Pour le rendu final de cette 2ème partie, si nous sommes en présentiel, imprimer un dossier papier comprenant la partie conception et d'implémentation du logiciel ; l'analyse de vos résultats. Imprimer également vos présentations individuelles d'un outil du développement. Attention : éviter les copier/coller pour les présentations d'outils et citer toutes vos sources (URL). L'université dispose de l'outil « compilatio » pour l'anti-plagiat, il est très efficace.

Un complément individuel est attendu pour présenter sur 4/5 pages l'outil que vous avez retenu. Comme exprimé précédemment, il s'agit de choisir un outil du développement logiciel que vous souhaitez découvrir de façon plus spécifique (3 exemples sont donnés). Les présentations synthétiques de ces descriptions d'outils comporteront, les points suivants:

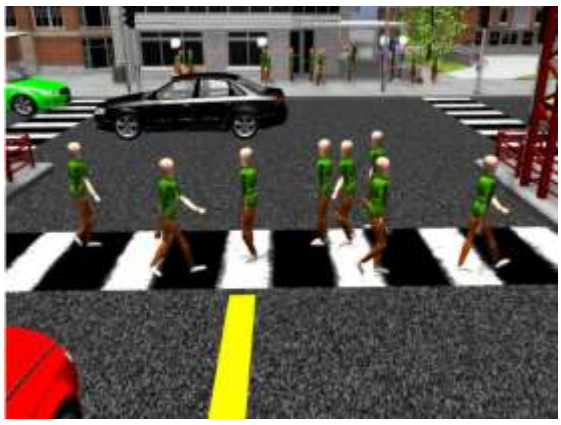
- Résumé.
- Procédure d'installation (url, etc…).
- Présentation des fonctionnalités.
- Mini-guide d'utilisation sur les principales fonctionnalités dans le cadre de cette étude de cas.
- Une synthèse des points clés du logiciel.

Pour les outils à décrire, voici une liste non exhaustive (à rafraîchir…) – un outil à décrire par personne.

Ateliers UML	Objecteering/UML Personal Edition
	Poseidon for UML Community Edition
	Visual Paradigm for UML Community Edition
	Rational Rose
	Poseidon (Argo)
	Bouml
	Dia
	Fujaba tool suite
	O mondo
	…

Gestion de projets	Etude approfondie de make Etude de cmake Etude de Ant Etude de Maven – (Apache Software Foundation) Utilise un paradigme connu sous le nom de Project Object Model (POM) pour décrire un projet logiciel et ses dépendances avec des modules / bibliothèque externes …
Gestion des versions	Git, GitLab, CVS (ancêtre), Tortoise CVS SVN, Subversion Mercury
Etude des performances	Prof / Gprof Jprofiler AQtime Sonar Cube …
Etude des tests	CppUnit, Junit, Valgrind, Kcachegring, …
Logger de traces d'exécution	Log4J…
Production de documentation	Doxygen, Javadoc …
Etude d'un débuggeur	gdb – ddd, dbx – dbxtool Eclipse débuggeur, .Net débugger …
Intégration continue :	Jenkins, Bamboo, Cruse Control, CodeShip, TeamCity Travis CI, GitLab CI, Circle CI

SMA



BI