Un Environnement de Calcul Domaine-Spécifique

Projet d'Ingénierie Dirigée par les Modèles

L'essor de l'informatique moderne est en grande partie due à la capacité des ordinateurs d'agréger de grandes quantités de données et de réaliser des calculs dessus. Aujourd'hui encore, l'un des logiciels les plus utilisés au monde est Microsoft Excel, et ce depuis les années 80-90.

De manière générale, les logiciels de la famille des *tableurs* dont Excel est le représentant le plus connu, sont largement utilisés et ce dans tous les domaines, de la comptabilité aux sciences expérimentales.

Mais les tableurs sont des outils parfois un peu complexes à manipuler, surtout si l'on n'est pas à l'aise avec l'outil informatique ou que l'on ne maîtrise pas totalement le fonctionnement profond du logiciel, ce qui peut donner naissance à des erreurs dramatiques [1]. Ce sont par ailleurs des outils très *standalone* et centralisés, ce qui rend difficile l'automatisation de certaines tâches (importation/exportation de données, calculs automatiques, etc.), que l'on aimerait héberger sur un serveur, par exemple.

Deux mondes se font face : d'un côté, des gens qui savent quelles données prendre et quels calculs réaliser, et de l'autre les gens qui savent stocker les données et réaliser les calculs. Dans des domaines particulièrement précis, il est parfois difficile d'accorder les deux.

Ce contexte justifie donc l'emploi de techniques issues de l'ingénierie dirigée par les modèles, afin de permettre à un utilisateur, expert de son domaine mais pas d'informatique, de mettre en place assez simplement des outils pour la gestion de données.

Table des matières

1 Proposition **2**

2 Spécification partielle et travail demandé **3**
2.1 Fonctionnalités attendues 3
2.2 Un scénario possible d'utilisation de la suite 5
2.3 Remarques et conseils 7

3 Rapport **8**

4 Autres livrables **9**

5 Oral **10**

6 Critères d'évaluations **10**

1. Voir par exemple : `https://www.science.org/content/article/one-five-genetics-papers-contains-errors-thanks-microsoft-excel`

1 Proposition

Le but de ce projet est de proposer un ensemble d'outils (une « suite ») basés sur la plate-forme Eclipse et les technologies EMF, permettant à un utilisateur de définir des *schémas de donnée* ainsi que des calculs automatisés sur ces schémas. Les spécifications données par l'utilisateur permettent de générer des outils spécifiques à ces schémas pour importer, visualiser et exporter des données qui suivent ledit schéma. On pourrait ensuite livrer ces outils à d'autre utilisateurs finaux (des *target users*).

Le point d'entrée pour l'utilisateur de la suite est un *modèle* qui représente un *schéma de table*, c'est-à-dire un ensemble de colonnes, auxquelles des données doivent se conformer. Le schéma de table représente en quelques sortes *l'en-tête* d'une table, et les données importées peupleront les lignes, en suivant la spécification du schéma.

Certaines colonnes peuvent être calculées à partir d'autres colonnes via un *algorithme* (par exemple, la colonne 4 est la somme des colonnes 1 et 3), ou référencer une valeur en provenance d'une autre table. Dans ce cas là, un outil pourrait compléter lui-même, pour chaque ligne d'une table, les cases des colonnes calculées et extérieures spécifiées (par exemple, calculer le contenu de la colonne 4 pour chaque ligne). Pour laisser le plus de latitude possible à l'utilisateur, un tel algorithme pourrait être associé à un script ou une fonction dans un langage orienté calcul (Python, Matlab, Julia) ou défini *on-the-fly* à l'aide d'un langage spécifique.

On peut également envisager de laisser à l'utilisateur la possibilité d'exprimer des *contraintes*, à vérifier lorsqu'on importe/calcule des données (par exemple, les valeurs dans une colonne doivent se trouver dans un intervalle donné). Ces contraintes pourraient entraîner des messages d'avertissement lors de la manipulation de donnée, ou même des messages d'erreur et un arrêt des outils, en fonction de la criticalité de ces contraintes.

Une fois le schéma de table défini, l'utilisateur peut initier la génération automatique d'outils : librairie (Java ou autre) pour pouvoir charger correctement les données, visualisation, extractions/exportations, scripts de calcul automatique, vérificateur de contraintes, etc.

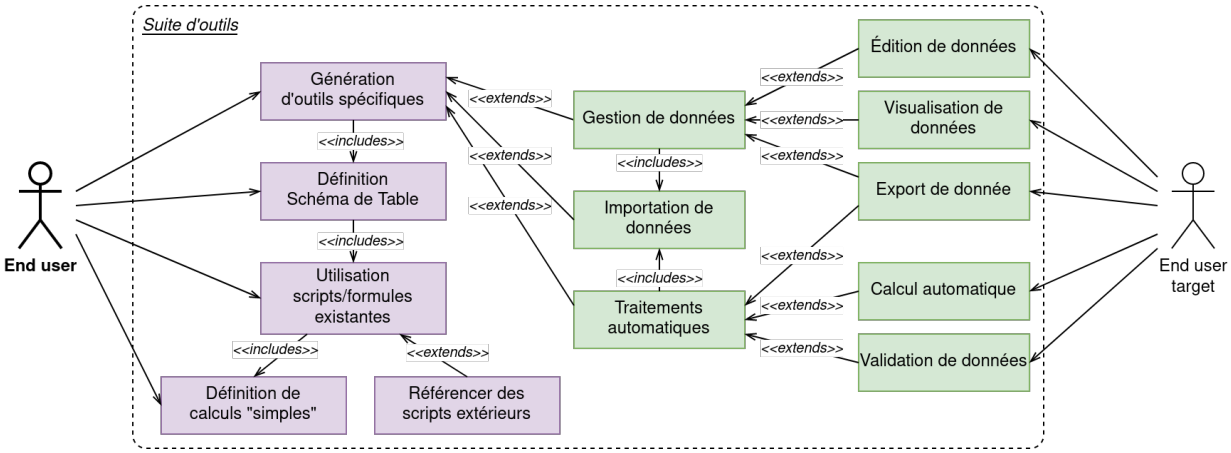

FIGURE 1 – Un diagramme de cas d'utilisation possible

La Figure 1 présente un *diagramme de cas d'utilisation*[2] possible pour schématiser la proposition. On note qu'il y a effectivement deux utilisateurs : le *end user* est l'utilisateur pour lequel on

2. Voir UE Technologies Objet de première année SN

réalise la solution (le *product owner* en quelques sortes), alors que le *end user target* est un autre utilisateur qui pourrait vouloir utiliser les outils générés par le *end user*, via notre suite.

On a donc d'un côté les outils que le end user manipule (en violet) et de l'autre les outils générés à partir de nos outils (en vert).

2 Spécification partielle et travail demandé

Les objectifs pédagogiques principaux du projet sont 1) de vous confronter à des technologies d'ingénierie dirigée par les modèles dans un contexte « réaliste », et 2) d'approfondir et d'expérimenter avec les technologies abordées durant l'UE.

Un des points essentiels du projet est qu'il est réalisé de manière très autonome : c'est à vous d'apporter vos solutions personnelles au problème posé, en toute liberté ; ce projet se contentera de vous donner une *spécification* (partielle), et bien sûr quelques livrables attendus.

2.1 Fonctionnalités attendues

Nous donnons ici la liste *minimale* des fonctionnalités (ou *features*) à réaliser. Libre à vous d'en rajouter si vous en éprouvez le besoin.

F1 L'utilisateur doit pouvoir composer, sauvegarder et consulter des *schémas de table* dans une interface ergonomique et adaptée.

- **F1.1** L'utilisateur doit pouvoir définir des *colonnes*. Ces colonnes doivent être pourvues d'un *identifiant* unique (pour les références croisées).
- **F1.2** Un schéma de table doit contenir une colonne spéciale pour les identifiants des lignes (pour les références croisées, en particulier **F1.4**).
- **F1.3** L'utilisateur peut déclarer qu'une colonne *dérive d'autres colonnes*, en précisant un *algorithme* et la façon d'utiliser cet algorithme (par ex. quelles colonnes sont utilisées pour l'entrée). La sémantique de cela est que le contenu de la colonne est calculé en utilisant les colonnes indiquées et par l'algorithme indiqué.
- **F1.4** L'utilisateur peut déclarer qu'une colonne *provient d'une autre table*, en indiquant le schéma de la table étrangère et la colonne à extraire de cette table. Cela permet de traiter simultanément plusieurs tables.
- **F1.5** L'utilisateur doit pouvoir définir des *contraintes* sur les colonnes, qui sont des *prédicats* à vérifier lorsque l'on importe des données. Cela permet de détecter des valeurs erronées pour notifier l'utilisateur, par exemple.

F2 L'utilisateur doit pouvoir composer, sauvegarder et consulter des *algorithmes* dans une interface ergonomique et adaptée.

- **F2.1** Un algorithme spécifie une fonction (au sens large) en donnant la *ressource* qui réalise l'algorithme ainsi que le détail de ses entrées et sorties.
- **F2.2** L'utilisateur doit pouvoir définir des entrées et sorties, qui sont reliées à l'interface relative à la ressource (entrées/sorties du calcul, ou arguments/retour de fonction).
- **F2.3** L'utilisateur doit pouvoir fixer une entrée du programme, ce qui permet de proposer un algorithme avec un paramètre particulier.
- **F2.4** L'utilisateur doit pouvoir documenter un algorithme, pour identifier ce que fait chaque entrée, décrire l'algorithme, etc.

F2.5 La ressource associée à l'algorithme est un fichier, un programme ou autre qui est utilisé pour effectuer le calcul. Il peut s'agir d'un script « extérieur » écrit dans un langage quelconque (Python, Matlab ou autre) ou d'une formule définie sur la plate-forme à l'aide d'un langage dédié (voir fonctionnalités **F3**).

F2.6 Les algorithmes sont clairement identifiés et pourraient être regroupés au sein de *catalogues* partageables et réutilisables, rendus disponibles par et pour les utilisateurs.

F3 L'utilisateur doit pouvoir composer, sauvegarder et consulter des scripts de calculs dans un langage dédié propre à la plate-forme.

F3.1 L'utilisateur doit pouvoir spécifier les calculs à l'aide d'une syntaxe concrète graphique.

F3.2 L'utilisateur doit pouvoir déclarer les entrées (les arguments) et les sorties (les résultats) du calcul.

F3.3 L'utilisateur doit pouvoir spécifier les opérations à réaliser à l'aide de *blocs* disposant d'entrées et de sortie. Les opérations disponibles sont *fixes*.

F3.4 L'utilisateur doit pouvoir réaliser (au minimum) des sommes, des produits, prendre l'opposé et l'inverse, la division, le minimum et le maximum de deux valeurs.

F3.5 L'utilisateur doit pouvoir faire appel à des fonctions classiques des mathématiques (par ex. sinus, cosinus, racine carrée, exponentielle, etc.).

F3.6 L'utilisateur doit pouvoir introduire des constantes dans ses calculs, autrement dit des valeurs codées en dur qui ne dépendent pas de l'entrée.

F4 L'utilisateur doit pouvoir générer, à partir d'un schéma de table (et d'éventuels composants satellites), une *librairie* de traitement de données conformes au schéma donné.

F4.1 La librairie sera fournie dans un langage cible, compatible avec les *algorithmes* associés au schéma de table.

F4.2 La librairie doit permettre d'importer des données pour les utiliser ailleurs dans un programme. Les formats d'entrées possible doivent contenir, au minimum, le format CSV [3].

F4.3 La librairie doit permettre de réaliser les vérifications associées aux *contraintes* associées au schéma (**F1**.**5**) sur les données importées.

F4.4 La librairie doit permettre de réaliser les calculs associés au schéma (**F1**.**3**) sur les données importées, et notamment les réaliser *dans le bon ordre* (si une colonne dérivée intervient comme argument d'un calcul, par exemple).

F4.5 La librairie doit gérer correctement les *références croisées* (**F1**.**4**).

F4.6 La librairie doit permettre d'exporter les données dans un format standard, incluant au minimum le format CSV (voir plus haut).

F5 L'utilisateur doit pouvoir générer, à partir d'un schéma de table (et d'éventuels composants satellites) un outil de visualisation spécifique aux données conformes au schéma donné.

F6 L'utilisateur doit pouvoir générer, à partir d'un schéma de table (et d'éventuels composants satellites) un script de calcul automatique, qui prend en entrée des données conformes au(x) schéma(s) requis et met à jour les colonnes dérivées (**F1**.**3**, **F1**.**4**).

3. Voir `https://fr.wikipedia.org/wiki/Comma-separated_values`

2.2 Un scénario possible d'utilisation de la suite

On imagine le scénario suivant : l'utilisateur de notre suite est un ingénieur dans une banque, qui veut réaliser un ensemble d'outils pour gérer des données boursières (variations de prix d'un produit financier ou d'une action en bourse, par exemple).

Pour un produit donné, on peut obtenir, par jour ouvert (lundi à vendredi) le prix du produit à l'ouverture, le prix maximal et minimal atteint, le prix à la fermeture et le volume échangé ce jour [4].

Voici par exemple le cours d'un important conglomérat industriel Allemand sur la période du 18 au 22 septembre 2023 (une semaine boursière), en format CSV.

```
Date,Open,High,Low,Close,Adj Close,Volume
2023-09-18,137.800003,138.500000,136.860001,137.179993,137.179993,806069
2023-09-19,136.759995,136.820007,134.160004,134.539993,134.539993,908349
2023-09-20,134.380005,136.919998,134.380005,136.199997,136.199997,936370
2023-09-21,134.679993,135.440002,133.339996,133.679993,133.679993,1457225
2023-09-22,133.000000,134.380005,132.619995,133.240005,133.240005,1440069
```

L'ingénieur sait que, en plus de ces données, le service finances est intéressé par la *moyenne* du prix (moitié de la somme du prix bas et du prix haut) et la *variation ouverture-fermeture* (différence algébrique entre le prix à la fermeture et le prix à l'ouverture). Il sait par ailleurs que le prix de fermeture ajusté ne lui sert à rien.

Dans un premier temps, il pose un schéma de table qui colle parfaitement au format des données à importer, donc comportant 7 colonnes (`Date` qui sert d'identifiant de la ligne, `Open`, `High`, `Low`, `Close`, `Adj Close`, `Volume`). Il donne, pour chaque colonne, un identifiant unique pour permettre les cross-references (par ex. `banque.finance.raw.date`, `banque.finance.raw.open`, `banque.finance.raw.high`, etc.).

Dans un second temps, il pose un schéma de table qui correspond à ce qu'il veut, en utilisant des colonnes références (**F1**.**4**) et des colonnes dérivées (**F1**.**3**), pour calculer les données additionnelles nécessaires. Ce schéma contient donc 8 colonnes :

- une colonne `Date` qui sert d'identifiant
- une colonne `Ouverture` dont la valeur est issue de la colonne `Open` dans l'autre table
- une colonne `Min` dont la valeur est issue de la colonne `Low` dans l'autre table
- une colonne `Max` dont la valeur est issue de la colonne `High` dans l'autre table [5]
- une colonne `Fermeture` dont la valeur est issue de la colonne `Close` dans l'autre table
- une colonne `Volume` dont la valeur est issue de la colonne `Volume` dans l'autre table
- une colonne `Moyenne` dont la valeur pour chaque ligne est égale à la valeur dans `Min` plus la valeur dans `Max` divisé par 2
- une colonne `Variation` dont la valeur pour chaque ligne est égale à la valeur dans `Fermeture` moins la valeur dans `Ouverture`

Bien sûr, chaque colonne est associée à un identifiant bien choisi (par ex. `banque.finance.date`, `banque.finance.ouverture`, `banque.finance.min`, etc.).

4. Le site Yahoo Finances permet par exemple de télécharger de telles données pour une période donnée au format CSV (dans l'onglet "Historical Data" du produit).

5. On notera qu'on a ici `Min` *puis* `Max` alors qu'on avait `High` puis `Low`

Pour être sûr de la cohérence des données (dont la source peut varier), il ajoute des contraintes. Notamment, pour les colonnes autres que `Date` et `Variation`, la valeur doit être positive ou nulle. On veut aussi que la valeur dans `Min` soit inférieure ou égale à celle dans `Max` (ce qui pourrait venir d'une potentielle inversion des colonnes, par exemple).

Pour réaliser le calcul de la moyenne, l'ingénieur utilise le langage de calcul dédié (soit il crée la fonction lui-même soit elle est fournie dans un catalogue propre à l'entreprise). La fonction est représentée [6] sur la Figure 2, avec sur la gauche les arguments (`minimum` et `maximum`) et sur la droite le résultat (`resultat`).

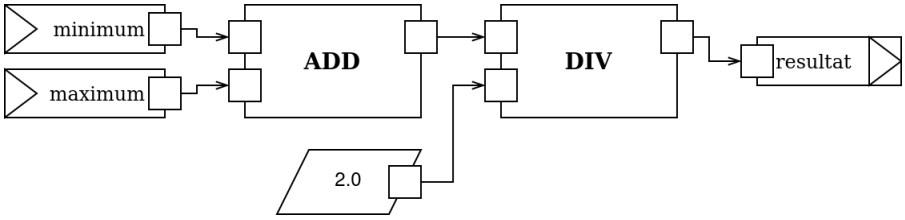

FIGURE 2 – Calcul de moyenne

Pour réaliser le calcul de variation, l'ingénieur a accès à un module Python `finance.py` qui contient, entres autres, la fonction `variation`.

```
...
def variation(debut, fin):
    """Calcul de la variation entre deux valeurs, soit la
        différence algébrique entre ces valeurs"""
    return fin - debut
...
```

FIGURE 3 – Extrait du script `finance.py`

Pour pouvoir utiliser ces deux calculs, on doit les associer à des *algorithmes* (**F2**), qui servent en quelques sortes de *mapping* pour pouvoir utiliser le calcul dans le schéma de table.
On pourrait par exemple avoir (informellement) :

- Algorithme `banque.finance.algo.moyenne`
  - Ressource : `/home/gdthunes/calc/moyenne.calc` de type "petits calculs" [7]
  - Port d'entrée `moyenne.minimum` associé à argument `minimum`
  - Port d'entrée `moyenne.maximum` associé à argument `maximum`
  - Port de sortie `moyenne.sortie` associé à résultat `resultat`
- Algorithme `banque.finance.algo.variation`
  - Ressource : `/home/gdthunes/calc/finance.py` de type Python, sous-programme `variation`
  - Port d'entrée `variation.fin` associé à `argument 1`
  - Port d'entrée `variation.debut` associé à `argument 0`
  - Port de sortie `variation.sortie` associé au retour de la fonction

On configure ensuite le schéma de table :

- la colonne `banque.finance.moyenne` est liée à la sortie `moyenne.sortie`

6. Possibilité de représentation en aucun cas obligatoire
7. On pourrait même envisager les centraliser sur un serveur et mettre une URI ici

— la colonne `banque.finance.min` est liée à l'entrée `moyenne.minimum`

— la colonne `banque.finance.max` est liée à l'entrée `moyenne.maximum`

— la colonne `banque.finance.variation` liée à la sortie `variation.sortie`

— la colonne `banque.finance.ouverture` est liée à l'entrée `variation.debut`

— la colonne `banque.finance.fermeture` est liée à l'entrée `variation.fin`

Une fois tout cela fait, l'ingénieur génère une librairie (Python a priori, puisque le script donné pour le deuxième algorithme est du Python...) pour manipuler les données. Il obtient notamment un script de calcul qui intègre des données selon le premier schéma de table, puis réalise l'import et le calcul pour obtenir des données qui respectent le second schéma de table (**F6**).

Dans la foulée, il génère un outil qui permet d'obtenir une représentation des données sous forme de tableau (un script qui génère un tableau en HTML ou un outil avec une interface graphique...). On pourrait même pousser l'exercice et permettre à l'utilisateur de spécifier une façon de générer un script `gnuplot`[8], pour faire des graphiques comme celui de la figure 4.

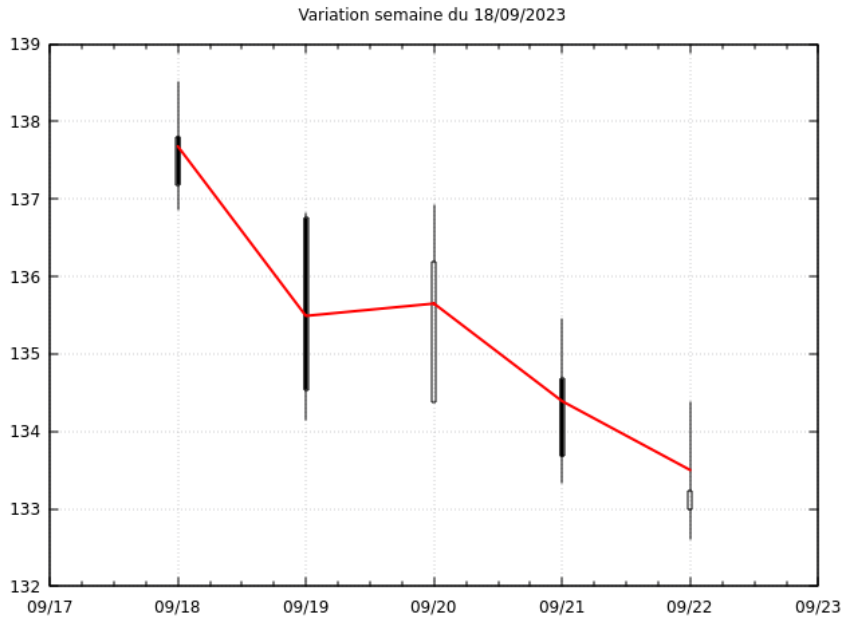

FIGURE 4 – Exemple de sortie des données avec `gnuplot`

2.3 Remarques et conseils

Imprécisions, ambiguïtés et manquements de la spécification Les zones floues dans les fonctionnalités demandées sont **volontaires**. Partout où quelque chose n'est pas précisé ou expliqué en détail, c'est à **vous** de prendre des décisions et de faire des choix. Ces choix doivent être explicités, expliqués et argumentés, bien évidemment.

Au sujet des méta-modèles Le projet aura pour coeur *des* méta-modèles. Il est possible de tout mettre dans un seul méta-modèle mais c'est généralement une mauvaise idée (pour des raisons

8. `http://www.gnuplot.info/`

de séparation des préoccupations, de maintenance, de travail en groupe, etc.).

Soyez rigoureux sur la création et la maintenance des méta-modèles. Pensez notamment à leur donner des noms, URI et namespaces différents.

Au sujet des références croisées EMF permet à un méta-modèle d'importer un autre méta-modèle, et ainsi de faire référence à des objets extérieurs. Simultanément, un modèle peut charger un autre modèle, ce qui permet au éléments du modèle initial de référencer des éléments du modèle chargé. On appelle ça des *références croisées* ou *cross-references*.

On a donc deux cas de figure : 1) méta-modèle A référence méta-modèle B, modèle MA conforme à A référence modèles MB_1, MB_2, ... conformes à méta-modèle B, ce qui permet d'utiliser des concepts de B dans des modèles de A, 2) méta-modèle A, modèle MA conforme à A référence modèles MA_1, MA_2, ... conformes à méta-modèle A, ce qui permet d'avoir un modèle qui utilise un autre modèle similaire ou découper son modèle en plusieurs parties/modules réutilisables.

À noter que si le méta-modèle A utilise le méta-modèle B, il faut générer les sources du méta-modèle B *d'abord*, puis référencer le genmodel de B lors de la création du genmodel de A, afin que les sources de A contiennent les références correctes aux sources de B.

Au sujet des syntaxes concrètes On rappelle qu'il vaut mieux avoir une syntaxe abstraite claire et facile à utiliser que adaptée à une syntaxe concrète précise. En général, il est plus avantageux de faire un méta-modèle par syntaxe concrète et une transformation de modèle à modèle pour réduire les modèles issus de la syntaxe concrète à des modèles conformes à la syntaxe abstraite.

Excepté pour le langage de calcul en interne (**F3**), nous vous laissons volontairement libres sur les syntaxes concrètes à utiliser. Il peut s'agir d'éditeurs arborescents si vous estimez que c'est pertinent, mais nous vous encourageons vivement à réaliser des syntaxes diverses et à essayer de vous porter sur des technologies adaptées pour chacun de vos méta-modèles. Dans tous les cas, vos choix devront être justifiés.

3 Rapport

Pour des raisons évidentes de diversité des rendus, le livrable principal du projet est un *rapport*. Ce rapport doit expliquer en détail le travail réalisé, et en particulier la *couverture fonctionnelle* (quelles fonctionnalités sont présentes et à quel point elles répondent à la spécification), les choix de conception et les compléments apportés à la spécification, le cas échéant. Il doit également présenter un regard critique sur les travaux réalisés : ergonomie, passage à l'échelle, manquements, améliorations possibles, etc.

Le rapport devra être rédigé dans une police d'écriture avec empattement (appelé aussi *serif*, par ex. Times, Garamond, etc.) et en taille 11 points. Il devra être rédigé sur du papier A4 avec des marges verticales de 3 centimètres et des marges horizontales de 2,5 centimètres.
Il devra comporter *impérativement* les éléments suivants :

— Une page de garde avec le numéro et les membres du groupe ainsi qu'un titre

— Une table des matières

— Une liste des figures

— Une introduction

— Une conclusion, qui doit inclure un bilan sur le projet (soit personnel soit de groupe), expliquant notamment les points de difficulté, et si possible une critique sur le sujet proposé

— Une description détaillée de ce que contient le rendu (cf Section 4) : description succincte de chaque projet et des fichiers importants (méta-modèles, modèles exemple, fichiers de description, scripts, etc.).

Chaque méta-modèle doit être donné sous forme d'un schéma (diagramme de classe Ecore) lisible et commenté.

Il est très vivement recommandé de faire apparaître des *figures* des outils réalisés, ainsi que des schémas pour certains aspects (les transformations ou chaînes de transformations, typiquement). Il est déconseillé de faire apparaître du code, excepté si c'est une entrée à un outil (syntaxe concrète textuelle par exemple).

L'usage d'un modèle de langage et assimilés (type GPT) pour générer tout ou partie du rapport est strictement interdit dans le cadre de ce projet.

Conformément à l'article L122-5 n° 3 du code la propriété intellectuelle, il est possible d'utiliser une oeuvre rendue publique pour contribuer à son discours à condition d'en indiquer clairement et sans ambiguïté la provenance et l'auteur, et de clairement identifier la citation en tant que tel.

4 Autres livrables

Outre les éléments demandés dans ce sujet, vous développerez un ou plusieurs exemples originaux du type de celui expliqué à la section 2.2. Ils permettront de montrer que vous avez bien compris les objectifs de ce projet. Ils seront utilisés dans la démonstration qui sera faite lors de l'oral du projet.

Le rendu se fera par **Git** obligatoirement, et se composera, en plus du rapport, de tous les workspaces Eclipse constitués dans le cadre du projet. Cela doit normalement inclure un ou plusieurs workspaces « principaux » contenant les méta-modèles, ainsi que des workspaces d'Eclipse de déploiement contenant (par exemple) des spécifications Sirius et Xtext, des exemples de modèles, etc. On rappelle que vous devez donner une description du contenu important de ces workspaces (divers projets et fichiers que vous avez pu créer ou éditer) dans le rapport.

Vous devez accorder un soin tout particulier à la rigueur et à la propreté de votre environnement de travail. Adoptez des règles de nomenclature en accord avec les conventions Java, rangez vos workspaces de manière rationnelle, utilisez la *qualification* (`aaa.bbb.ccc....`) pour rapprocher ou séparer les éléments qui doivent l'être.

Un dépôt Git sur l'instance Gitlab de l'école vous sera fourni, mais c'est à vous de gérer ce dépôt en autonomie. Libre à vous de l'utiliser pour travailler en groupe ou de passer par un autre dépôt/autre gestionnaire de version. Notez que vous aurez les droits pour créer et fusionner des branches sur le dépôt de l'école.

Dans tous les cas, votre dépôt doit comporter une branche `main`. C'est cette branche qui servira à l'évaluation, et qui contiendra donc les workspaces, ainsi que le rapport, à la racine du dépôt.

5 Oral

L'oral a pour but de montrer ce qui a été traité du projet. Il s'agit donc de faire une démonstration des différents outils pour montrer qu'ils fonctionnent. Les détails de comment ils ont été réalisés doit être expliqué dans le rapport et n'ont pas à être abordés pendant la démonstration.

Nous vous conseillons vivement de préparer un ou plusieurs *exemples* qui serviront à illustrer les diverses fonctionnalités réalisées. Il est certainement utile de préparer un scénario à répéter, afin de ne pas hésiter pendant l'oral.

L'oral dure 20 minutes, incluant 5 minutes de questions. Un temps de préparation vous est laissé, vous permettant de mettre en place vos démonstrations. L'ordre de passage est tiré au sort, et vous sera indiqué ultérieurement.

6 Critères d'évaluations

À titre indicatif, nous donnons ici les principaux critères d'évaluation. Le poids respectif de chacun de ces aspects n'est volontairement pas explicité.

- Rapport :
  - Respect des consignes (= tous les éléments demandés sont présents, etc.)
  - Structure cohérente et pertinente (= les éléments sont organisés proprement et de manière lisible et intelligente)
  - Complétion et pertinence (= vous parlez de tout ce qui est important)
  - Figures (= vous proposez des diagrammes/listings/... lisibles et pertinentes, que vous commentez dans le discours)
  - Orthographe, grammaire, normes typographiques
- Rendu :
  - Propreté et organisation du rendu
  - Noms pertinents (fichiers, projets)
  - Respect des conventions (noms de projet, de paquets, de classes, d'attributs...)
  - Commentaires et documentation partout où c'est nécessaire
  - Description détaillée du contenu du rendu dans le rapport
- Oral :
  - Couverture exhaustive (= vous montrez tous les outils développés)
  - Pertinence (= vous ne montrez que ce qui est important)
  - Préparation (= tout est déjà en place et vous ne cherchez pas les fichiers)
  - Fluidité et maîtrise du sujet (= vous savez ce que vous faites et où vous allez)
  - Répartition de la parole (= tout le monde parle)
  - Réponses aux questions

La date de l'oral et la date butoir pour le rendu vous seront communiquées ultérieurement. Nous vous conseillons de consulter régulièrement la page moodle de l'UE pour vous tenir à jour sur les détails relatifs au projet.