Module informatique : approche objet présentation générale

Pré-requis

Les bases de l'algorithmique (affectation, itérations, sous-programmes...)

Objectifs

La modélisation et programmation orientées objet permettent de mieux maîtriser la complexité liée à la création d'un logiciel. Elles permettent également une meilleure capacité d'adaptation et d'évolution d'un logiciel lorsque des fonctionnalités sont modifiées ou ajoutées. Ces avantages ne deviennent évidents que sur des projets de taille conséquente. Mais le temps consacré à ce module, tant en heures encadrées qu'en travail personnel des étudiants est forcément limité. Pour faire face à ces contraintes contradictoires, l'ensemble du module sera centré autour de la réalisation, par les étudiants, d'un logiciel de taille significative.

Cette année, nous voulons réaliser un logiciel permettant de définir un treillis de poutres, et de calculer les efforts au sein des poutres pour vérifier que la structure est cohérente avec les contraintes.

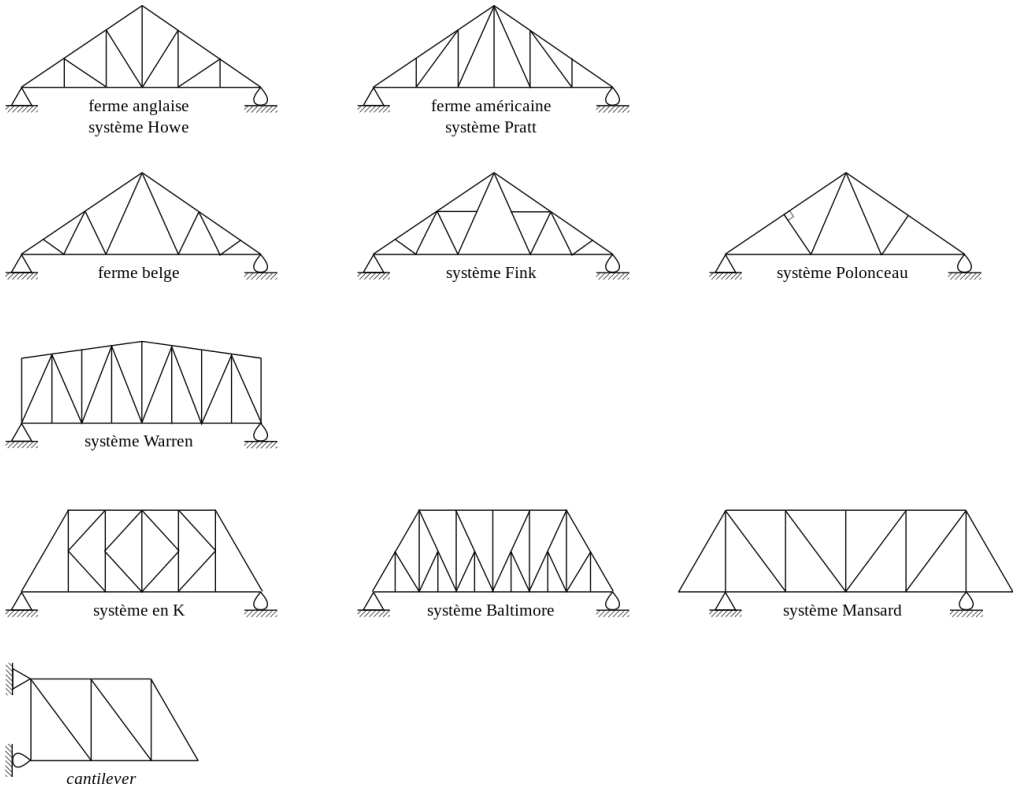

Fig 1 : quelques treillis de poutres isostatiques[1]

Il nous faudra donc une interface permettant à l'utilisateur de créer un treillis et de le visualiser:

Le programme devra ensuite effectuer les calculs et présenter les résultats sous forme textuelle et/ou graphique (par exemple avoir une sortie texte qui précise les efforts en tension/compression pour chacune des poutres, et avoir un code couleur dans la partie graphique pour représenter les poutres

1 https://fr.wikipedia.org/wiki/Treillis_(assemblage)#/media/Fichier:Treillis_isostatiques.svg

en sur-tension)

L'ensemble des Tds et Tps de ce module seront donc structurés autour de la réalisation de ce logiciel. La réalisation se fera en plusieurs phases :

- une définition du modèle sous-jacent (comment conserver les caractéristiques du treuillis : ses noeuds, ses poutres...)
- programmation de « briques » de base permettant par exemple de résoudre un système linéaire.
- Une gestion du modèle sous forme textuelle (ajout de noeuds/poutres au clavier).
- De modifier le modèle dans un environnement graphique : des interactions sous forme de menus et d'interactions à la souris permettent la création et la modification des objets.

La réalisation de ce projet nous permettra :

- de voir le rôle central de la définition concomitante des fonctionnalités du logiciel et de la structure de données correspondante. Dans ce module, les fonctionnalités ne seront pas formalisées au travers d'un cahier des charges complet, mais plutôt de façon informelle au travers des « users stories »[2] chères aux méthodologies agiles, puis précisées au travers des diagrammes de cas d'utilisation d'UML. La modélisation des données se fera sous la forme de diagrammes de classes UML.
- De voir comment la structuration des données telle que définie par un diagramme de classes UML peut être implémentée dans un langage de programmation orienté objet. Nous programmerons en Java, mais la méthodologie serait la même pour de nombreux langages orientés objet (C#, python objet...)
- de voir l'utilisation de bibliothèques orientées objet au travers de la création d'interface graphique. De nouveaux, les principes seraient les mêmes que nous utilisions d'autres langages, ou des bibliothèques multi-langage telle Qt[3] : s'appuyer sur des classes existantes, et les spécialiser pour réaliser les fonctionnalités propres au projet.
- d'ébaucher une étude de la programmation événementielle : le déroulement des actions n'est pas fixé par le logiciel, mais répond aux actions de l'utilisateur. Dans ce contexte, la définition des diagrammes d'états UML, et leur implémentation simple dans un langage de programmation sera abordé.

Organisation des groupes.

- Le projet se fera normalement par groupe de trois (avec éventuellement un voire deux groupes de deux si l'effectif du TP n'est pas divisible par trois) étudiants appartenant **au même groupe de TP**. Je veux ici préciser les modalités de composition des groupes :
  1. Ce que l'on souhaite :
     a) un groupe fonctionnera d'autant mieux que le « niveau » en informatique des étudiants le composant est homogène : vous avez eu un premier module d'informatique, et vous savez que vous avez plus ou moins de « facilité » en informatique. La note finale du projet prendra en compte la progression du groupe : un groupe avec des difficultés initiales en informatique, mais qui parvient, en coordonnant les efforts de ses membres, à un résultat satisfaisant sera mieux considéré qu'un projet où l'un des membres a presque tout fait pendant que les autres

2 Voir par exemple http://www.agilemodeling.com/artifacts/userStory.htm
3 https://www.qt.io/

ont plus ou moins tentés de comprendre.

b) Il est préférable que les participants à un groupe « s'entendent bien ». Mais ce n'est pas requis : en situation professionnelle, vous devrez collaborer efficacement avec des collègues avec qui vous n'avez que plus ou moins « d'atomes crochus ».

2. les contraintes liées à l'établissement de groupes dans un contexte d'enseignement
   a) Il n'est pas question que la détermination des groupes soit l'occasion de pressions ou d'exclusion entre les étudiants.
   b) les enseignants responsables de chacun des groupes de TP doivent avoir la possibilité d'intervenir s'ils estiment que le point (a) n'est pas respecté.
3. Le protocole pour essayer de trouver un compromis entre les point (1) et (2) :
   a) Les groupes sont initialement déterminés aléatoirement : un lien vers un fichier définissant cette répartition initiale vous sera fourni lors du premier cours.
   b) Les étudiants ont la possibilité de proposer des modifications au début de la première séance de TP. Ces modifications peuvent prendre deux formes :
      i. Le cas que nous souhaitons : une nouvelle proposition globale de composition des groupes (qui respecte simplement les contraintes : priorité aux trinômes ; binômes seulement dans le cas ou l'effectif du groupe de TP n'est pas divisible par 3). Cela suppose que tous les étudiants du TP adhèrent à cette proposition.
      ii. Un cas plus simple : un échange d'étudiants entre deux groupes. Cela suppose que tous les étudiants des deux groupes adhèrent à cette proposition
   c) Les étudiants seront consultés durant le TP1 pour s'assurer qu'ils sont d'accord avec les modifications de groupe proposées.
   d) En complément, tout étudiant qui n'est pas d'accord avec la nouvelle répartition peut en informer de façon privative (par mail ou en personne) son enseignant de TP entre le TP1 et le TP2.
   e) L'enseignant responsable du groupe de TP fixe les groupes définitif au début du TP2 : soit il valide les groupes modifiés, soit il annule **toutes les modifications de groupe** -aussi bien (bi) que (bii)- pour revenir aux groupes déterminés aléatoirement en (a). L'enseignant responsable du groupe n'aura pas à motiver sa décision pour ne pas risquer d'aggraver le problème (2a). Pour le dire plus clairement : la possibilité laissée aux étudiants de modifier les groupes est subordonnée au vrai consentement de tous. Si ce consentement n'est pas évident, on revient aux groupes aléatoires. Pour le dire encore autrement : la répartition normale est celle déterminée aléatoirement en (3a). La modification de cette répartition n'est accordée qu'à titre dérogatoire par l'enseignant responsable du groupe de TP s'il estime que toutes les conditions sont réunies.

Evaluation.

L'évaluation portera sur :

- eval1 - devoir sur table ; commun à tous les groupes en TD5 ; pour vérifier l'acquisition des connaissances de base (permettant de démarrer effectivement le projet) en modélisation et programmation objet – Ce devoir comptera pour 20% de la note finale du module.
- eval2 - devoir sur table ; fin de semestre - 40% de la note finale.
- eval3 - projet (rapport+présentation+démo) – 40% de la note finale

Evaluation des projets

- L'évaluation des projets se fera sous la forme d'une mini-soutenance en fin de semestre. Cette présentation se fera devant, au minimum :
  a) l'enseignant responsable du groupe de TP
  b) un autre enseignant responsable d'un autre groupe de TP
- Cette évaluation prendra en compte :
  a) la qualité du logiciel présenté : c'est évident, mais tous les points suivants montrent que ce n'est pas le seul critère d'évaluation.
  b) la cardinalité du groupe (à quantité de travail égal pour chaque étudiant, un binôme aura sans doute un projet moins abouti qu'un trinôme).
  c) La progression du groupe.
- Après une présentation (comportant une démonstration du logiciel) faite par les étudiants, les examinateurs se réservent le droit de poser des questions précises sur le programme permettant de juger de l'implication de chacun des étudiants. Deux stratégies de travail pour la réalisation d'un logiciel sont possibles :
  a) les étudiants travaillent et programment systématiquement ensembles ou au moins par deux. C'est une stratégie efficace[4], et même recommandée par certaines méthodologies de programmation comme XP[5]. Dans ce cas, l'un « tape le code » tandis que l'autre regarde, vérifie qu'il n'y a pas d'erreur, propose des alternatives... Si vous optez pour cette méthode, n'oubliez pas de permuter régulièrement les rôles.
  b) Les étudiants se partagent les tâches, chacun programmant une partie du logiciel.

  Il est bien sûr possible de mixer ces stratégies : réalisation en commun du cœur du logiciel, et réalisation seul de certaines sous-parties annexes.

  Les groupes sont libres de choisir la stratégie qui leur correspond le mieux, mais quelque soit la stratégie retenue, chacun des étudiants doit avoir pris part activement à la programmation. Ce point sera vérifié lors de la soutenance, et des notes différentes pourront être attribuées si les écarts d'investissement au sein du groupe sont jugés trop importants. Un programme insuffisamment maîtrisé (copié) se verra affecter la note de zéro.

4 https://fr.wikipedia.org/wiki/Programmation_en_bin%C3%B4me
5 http://www.extremeprogramming.org/