GÉNIE LOGICIEL ORIENTÉ OBJET (GLO-2004)
ANALYSE ET CONCEPTION DES SYSTÈMES ORIENTÉS OBJETS (IFT-2007)

Automne 2016

Module 00

Martin.Savoie@ift.ulaval.ca

B. ing, Chargé de cours, département d'informatique et de génie logiciel

UNIVERSITÉ LAVAL

Plan

- Parler de moi un peu ainsi que de l'équipe du cours
- Présenter le plan de cours et les objectifs
- Expliquer les changements qui sont apportés cette session-ci
- Projet de session, enfin!
- Formation des équipes
- Demande de dépôt git et laboratoire

Équipe du cours

En classe
(GLO-2004)

À distance
(IFT-2007)

Martin.Savoie@ift.ulaval.ca

Martin.Savoie@ift.ulaval.ca

Support projet de session

(questions sur le projet, organisation des présentations en fin de session, …)

Maxime.Charron.4@ulaval.ca

Jean.Bouchard.7@ulaval.ca

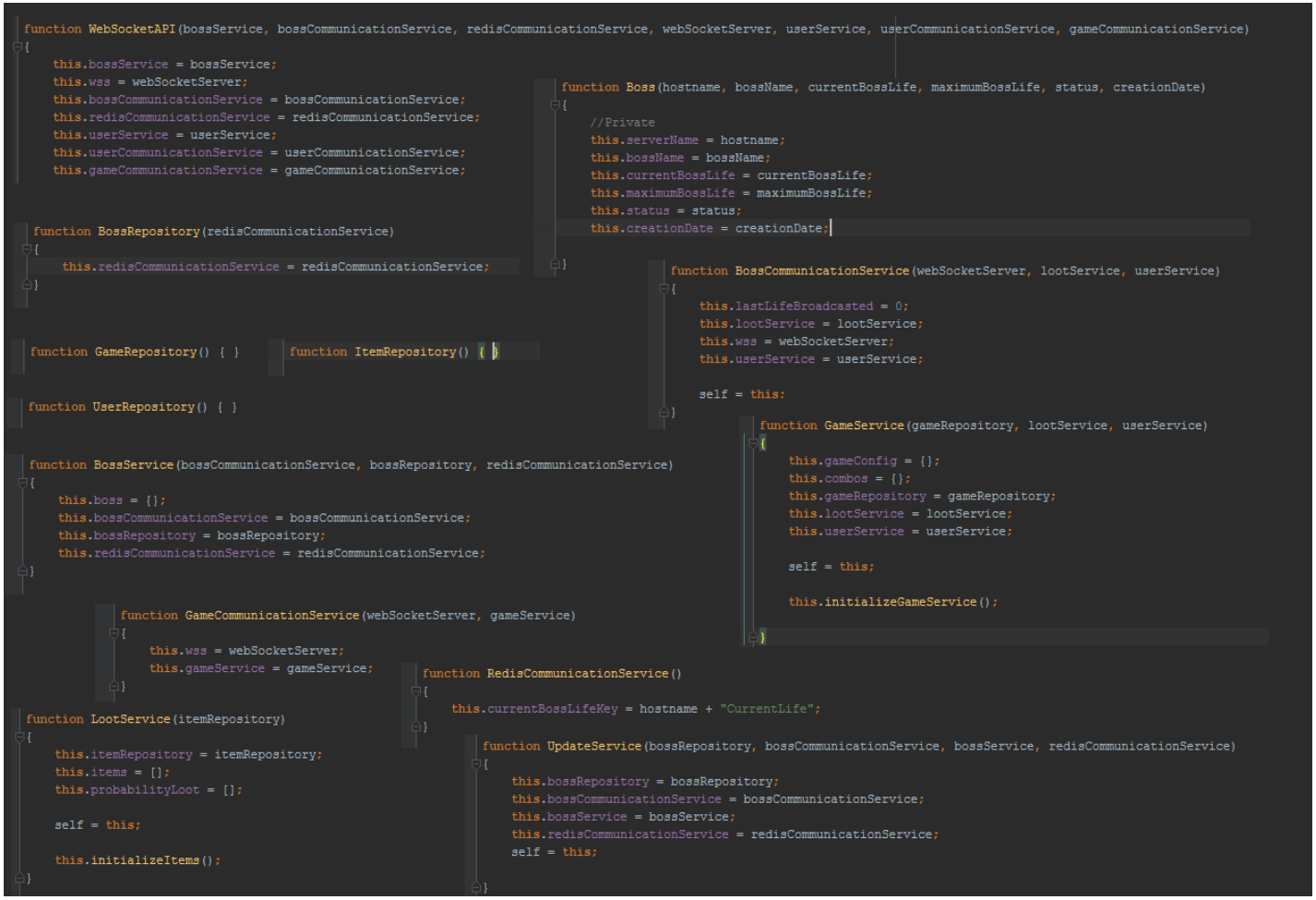
Équipe du cours

- Maxime Charron
  - Étudiant genie logiciel
  - Devops
- Jean Bouchard
  - Étudiant 2e cycle en informatique
  - Travaille sur un système d'aide à la décision pour des décideurs en entreprise
  - Il y a toujours des contraintes qui ne sont pas exprimable mathématiquement
  - L’objectif est de permettre de visualiser les solutions sans refaire une réoptimisation coûteuse

Équipe du cours

- Défauts: Je parle vite et le français
- 20 juillet 2016
- Retour aux études
- LinkedIn
  - https://ca.linkedin.com/in/martinsavoie
- Auxiliaire pour ce cours depuis 2 an et demi
- Fait quelques améliorations pour le cours
- Encore très proche du monde étudiant!
- Si commentaire, suggestion ou amélioration?

Ma vision des objectifs du cours



```
function WebSocketAPI(bossService, bossCommunicationService, redisCommunicationService, webSocketServer, userService, userCommunicationService, gameCommunicationService)
{
    this.bossService = bossService;
    this.wss = webSocketServer;
    this.bossCommunicationService = bossCommunicationService;
    this.redisCommunicationService = redisCommunicationService;
    this.userService = userService;
    this.userCommunicationService = userCommunicationService;
    this.gameCommunicationService = gameCommunicationService;
}

function Boss(hostname, bossName, currentBossLife, maximumBossLife, status, creationDate)
{
    //Private
    this.serverName = hostname;
    this.bossName = bossName;
    this.currentBossLife = currentBossLife;
    this.maximumBossLife = maximumBossLife;
    this.status = status;
    this.creationDate = creationDate;
}

function BossRepository(redisCommunicationService)
{
    this.redisCommunicationService = redisCommunicationService;
}

function BossCommunicationService(webSocketServer, lootService, userService)
{
    this.lastLifeBroadcasted = 0;
    this.lootService = lootService;
    this.wss = webSocketServer;
    this.userService = userService;

    self = this;
}

function GameRepository() { }

function ItemRepository() { }

function UserRepository() { }

function GameService(gameRepository, lootService, userService)
{
    this.gameConfig = {};
    this.combos = {};
    this.gameRepository = gameRepository;
    this.lootService = lootService;
    this.userService = userService;

    self = this;

    this.initializeGameService();
}

function BossService(bossCommunicationService, bossRepository, redisCommunicationService)
{
    this.boss = {};
    this.bossCommunicationService = bossCommunicationService;
    this.bossRepository = bossRepository;
    this.redisCommunicationService = redisCommunicationService;
}

function GameCommunicationService(webSocketServer, gameService)
{
    this.wss = webSocketServer;
    this.gameService = gameService;
}

function RedisCommunicationService()
{
    this.currentBossLifeKey = hostname + "CurrentLife";
}

function LootService(itemRepository)
{
    this.itemRepository = itemRepository;
    this.items = [];
    this.probabilityLoot = [];

    self = this;

    this.initializeItems();
}

function UpdateService(bossRepository, bossCommunicationService, bossService, redisCommunicationService)
{
    this.bossRepository = bossRepository;
    this.bossCommunicationService = bossCommunicationService;
    this.bossService = bossService;
    this.redisCommunicationService = redisCommunicationService;
    self = this;
}
```

Ma vision des objectifs du cours

- Premier gros travail en équipe
  - Apprendre à travailler ensemble
- Se donner les outils pour réaliser un projet
- Apprendre à communiquer et faire des compromis
- Autonomie

Génie logiciel orienté objet

- L'objectif principal de ce cours est de former l'étudiant à la conception d'applications logicielles d'envergure selon les principes fondamentaux du génie logiciel et de la programmation par objets.
- Aux termes de ce cours, l'étudiant devrait pouvoir :
  - Collaborer et communiquer avec les membres d'une équipe pour la réalisation d'une application logicielle complexe comprenant une interface utilisateur. Cela implique la participation à l'élaboration d'un projet logiciel, de son cahier des charges, de son échéancier, de sa réalisation, de sa mise à l'épreuve et de sa documentation.
  - Concevoir une application logicielles modulaire en mettant à profit les patrons de conceptions de base du génie logiciel (notamment, un découpage strict entre l'interface utilisateur et la logique applicative).
  - Apprendre de manière autonome un nouveau langage de programmation
  - Mettre en œuvre la méthodologie « unifiée » (Processus unifié), le langage de modélisation UML et le langage de programmation JAVA qui seront utilisés pour la réalisation du projet.
  - Poser un regard critique sur la formation reçue, dans une perspective d'amélioration continue.

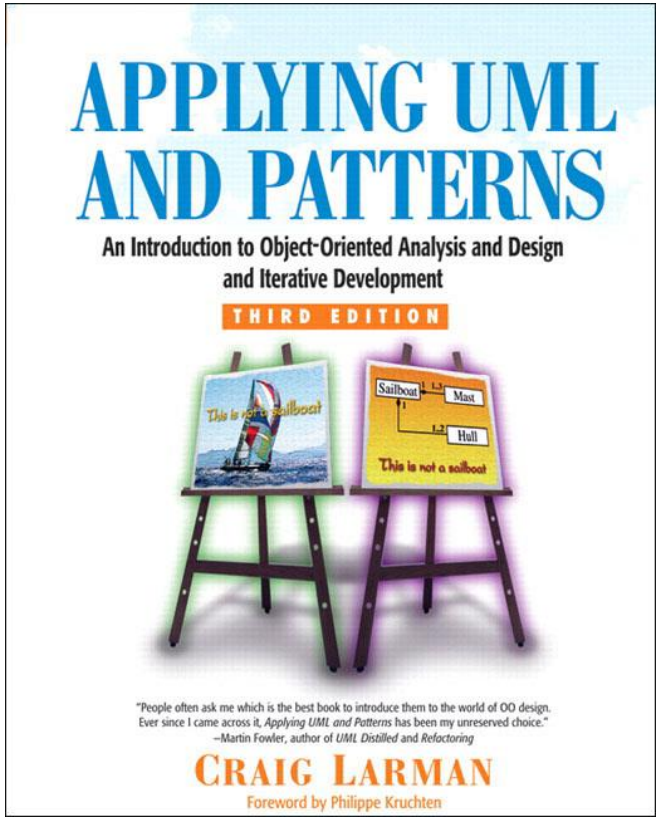
Génie logiciel orienté objet

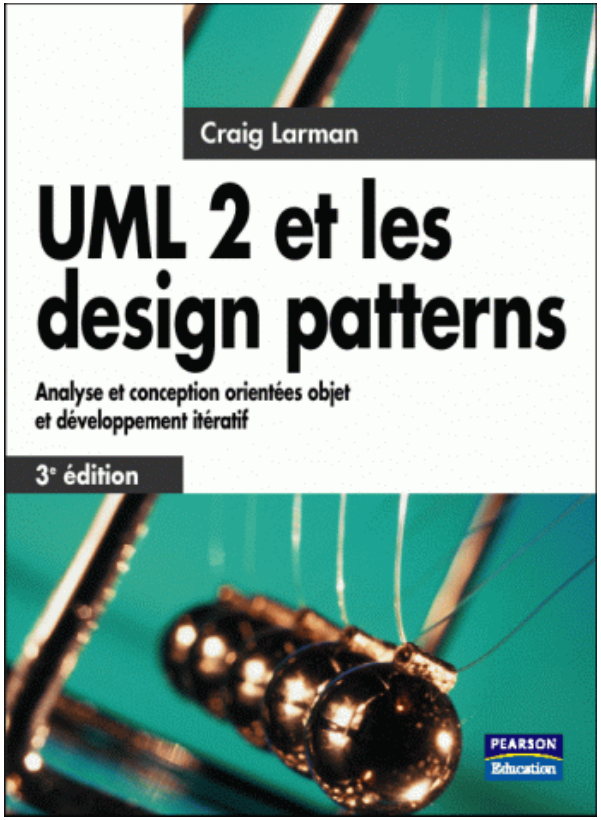
- C'est le premier cours dans lequel l'étudiant conçoit une application logicielle d'envergure en suivant toutes les étapes depuis l'analyse de problèmes jusqu'au développement. Pour les futurs ingénieurs inscrits au cours GLO-2004, les qualités suivantes sont évaluées dans le cadre du projet de session:
  - Analyse de problème (Q2)
  - Conception (Q4)
  - Apprentissage continu (Q12)

- Certaines autres qualités sont développées sans être évaluées dans ce cours, notamment :
  - Utilisation d'outils d'ingénierie (Q5)
  - Travail individuel et en équipe (Q6)

Manuel obligatoire

- Version originale anglaise ou française – au choix





Erratum – Manuel version française

- La partie III devrait se nommer
  - « Itération 1 de la phase d'élaboration »
  - Et non pas « ~~Élaboration de l'itération 1~~ »
- La partie IV devrait se nommer
  - « Itération 2 de la phase d'élaboration »
  - Et non pas « ~~Élaboration de l'itération 2~~ »
- La partie V devrait se nommer
  - « Itération 3 de la phase d'élaboration »
  - Et non pas « ~~Élaboration de l'itération 3~~ »

Déroulement du cours

- Lectures dans le manuel obligatoire
  - Nécessaire à l'atteinte des objectifs (et à la réussite des examens!)
- Deux examens valant 20% chacun = 40%
- Projet de session réalisé en équipes de 4 personnes
  - Divisé en 4 livrables valant respectivement 14%, 15%, 15% et 15% = 59%
  - UML
  - Java
- Charge de travail « normale » d'un cours de 3 crédits: 135 heures (9 heures par semaine)

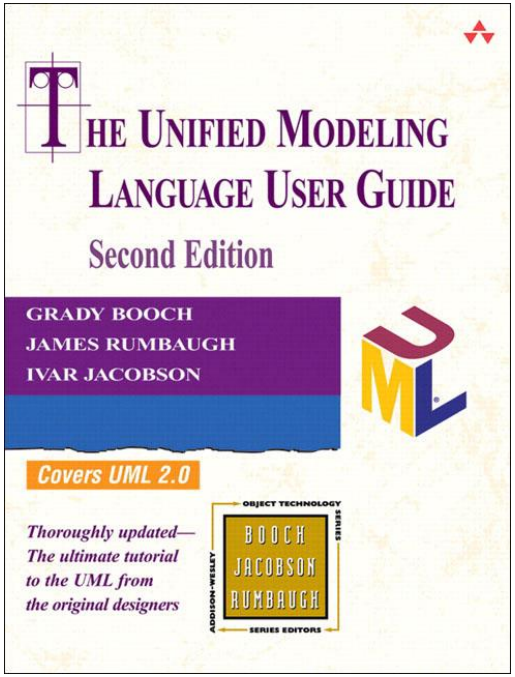
Déroulement du cours

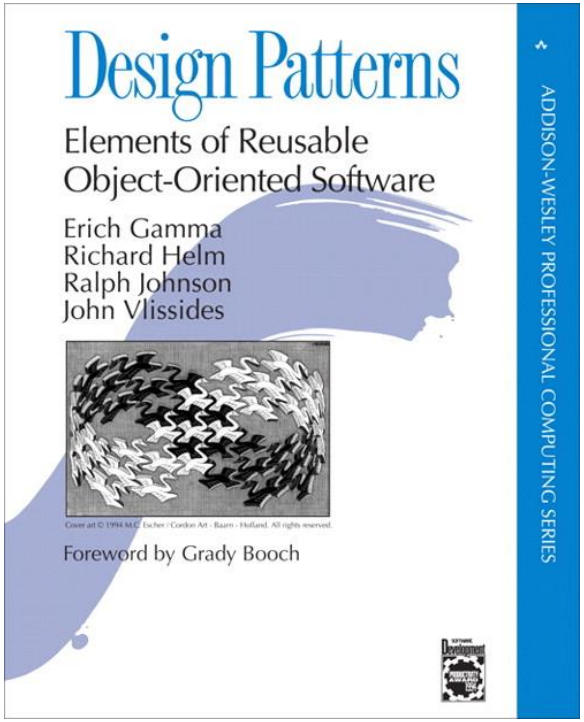
- Lectures dans le manuel obligatoire
  - Nécessaire à l'atteinte des objectifs (et à la réussite des examens!)
- Deux examens valant 20% chacun = 40%
- Projet de session réalisé en équipes de 4 personnes
  - Divisé en 4 livrables valant respectivement 14%, 15%, 15% et 15% = 59%
  - UML
  - Java
- Charge de travail « normale » d'un cours de 3 crédits: 135 heures (9 heures par semaine)

1% est attribué pour l'évaluation de l'enseignement

Manuels complémentaires

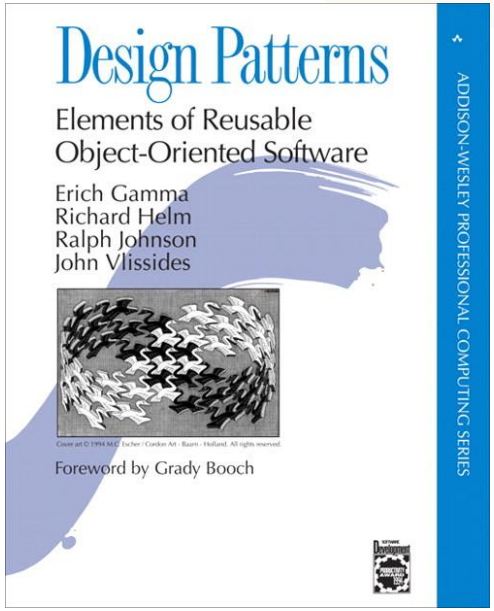
- À la bibliothèque en version papier et électronique





Patrons de conception (design patterns)

- Présente des designs classiques en orienté objet; des solutions éprouvées à des problème fréquents auquel le concepteur fait face.
- Pour un problème donné, suggère une organisation en termes de classes, l'allocation des responsabilités entre les classes et comment les interactions devraient se passer.



Projet de session

- Projet de conception et d'implantation d'une application logicielle de grande envergure.
- Chaque **équipe de 4** doit veiller à ce que le projet rencontre les exigences du cahier des charges à l'intérieur du budget et de l'échéancier prévus.
- Le projet est évalué en étapes (livrables) qui servent à vérifier le niveau de maîtrise du processus de développement et des étapes de conception et d'implémentation d'applications logicielles.
- Développé en Java (avec NetBeans)
- Diagrammes UML produits avec le logiciel Visual Paradigm (informations sur le site du cours)

Java

- On suppose que vous ne connaissez pas Java
- Mais que vous avez de bonnes bases en programmation orientée objet (POO) (C++ ou autre)
- Vous apprendrez Java par vous-même tout au long de la session
- Un Wiki avec des tutoriels a été préparé spécialement pour vous:

  http://www2.ift.ulaval.ca/gaudreault/dokuwiki_a14/doku.php

« Java for programmers » (Deitel & Deitel)

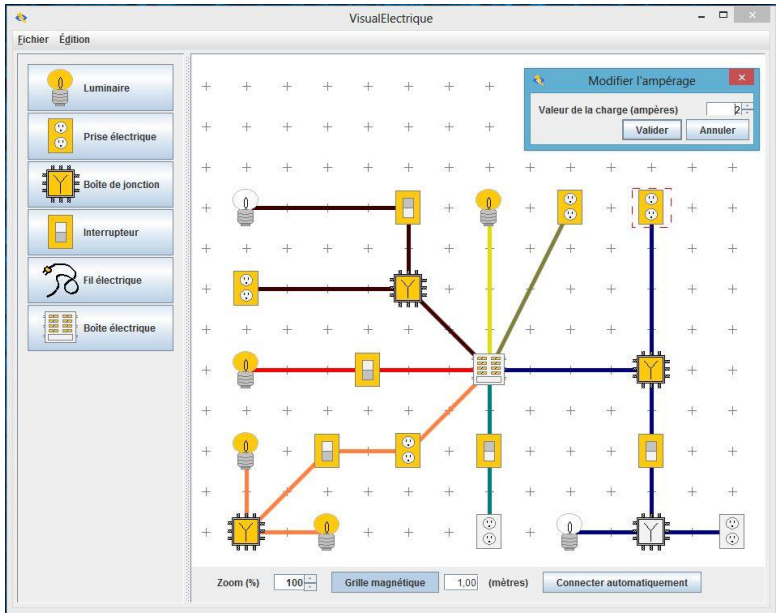
- http://ariane.ulaval.ca/cgi-bin/recherche.cgi?qu=a1995576
  Accessible gratuitement en version électronique en passant par le site de la bibliothèque

Prix Yves-Roy - Meilleur projet en génie logiciel orienté-objet

Ce prix est remis en mémoire de M. Yves-Roy qui a enseigné la conception orientée-objet au département d'informatique et de génie logiciel, de 1994 à 2004. Il a formé des centaines de jeunes informaticiens qui mettent aujourd'hui en œuvre le savoir-faire dont ils ont hérité. M. Roy a également apporté une contribution remarquable à l'Institut National de la Recherche Scientifique du Québec (INRS-Eau) où il a contribué à la réalisation d'une plate-forme logicielle d'analyse hydrologique.

Hiver 2013 – VisualÉlectrique

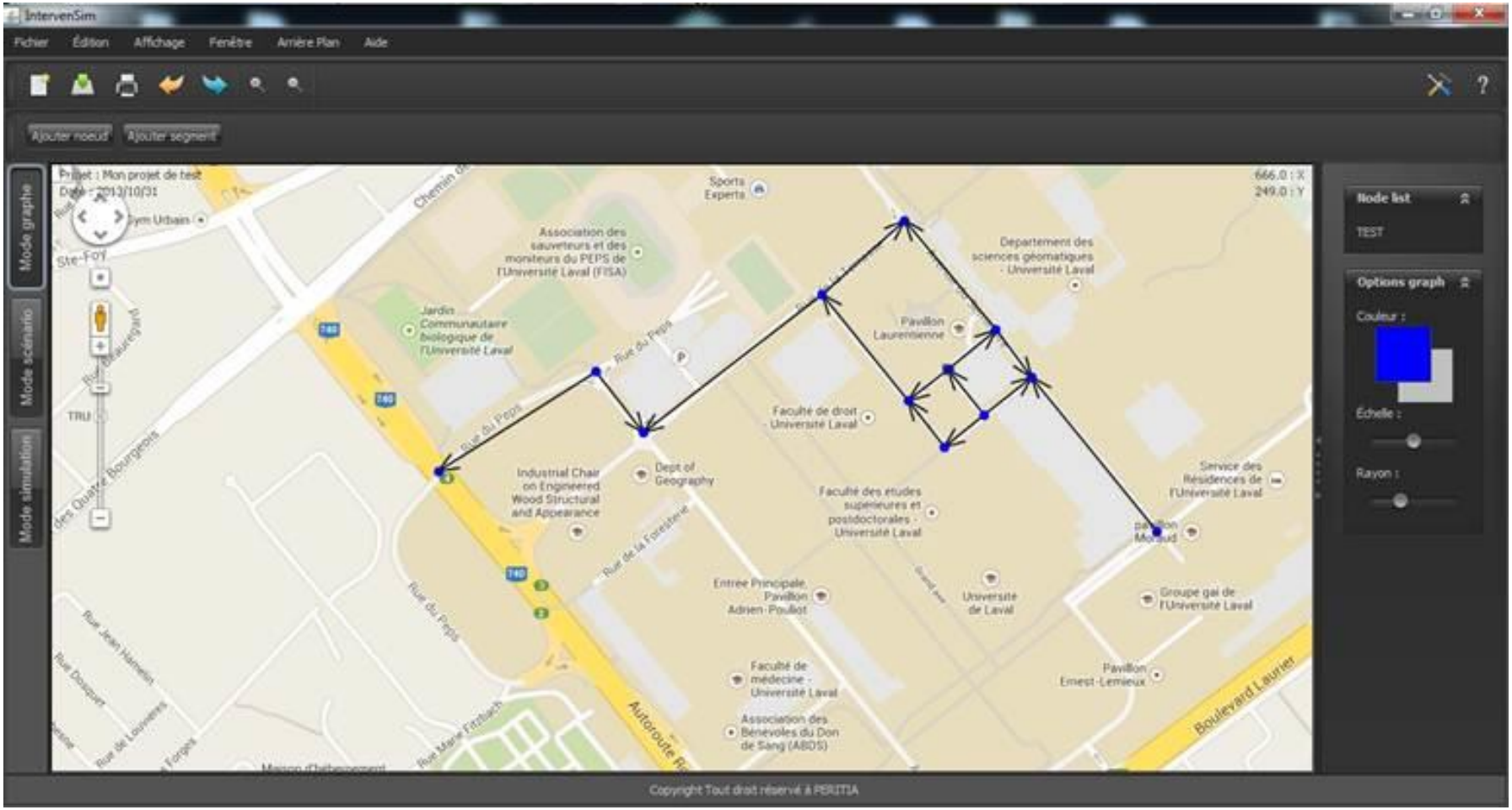
Yassine Attik, Jean-Rock Blanchette, Jean-Christophe Côté, Mathieu Lepage



Prix Yves-Roy - Meilleur projet en génie logiciel orienté-objet

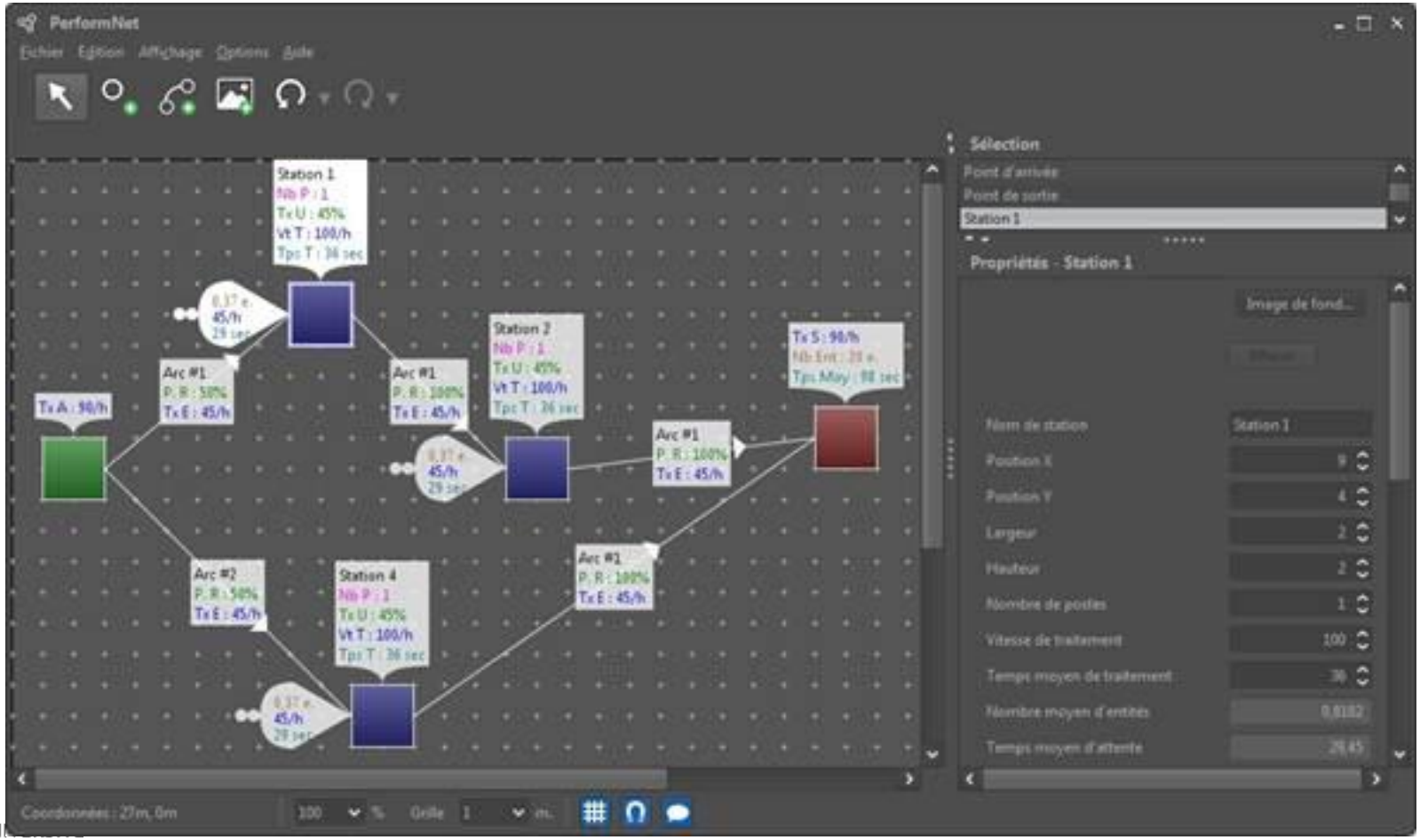
Automne 2013 – IntervenSim (Équipe Perita) http://youtu.be/DOV1-nLjF_s

Pascal Dumoulin, Jean-Philippe Lapierre, Guillaume Lorquet et Martin Savoie



Prix Yves-Roy - Meilleur projet en génie logiciel orienté-objet

Hiver 2014 – PerformNet - logiciel permettant de modéliser et évaluer des réseaux de files d'attente (ex : service de sécurité d'un aéroport)
J.S. Bédard, A. Gariépy, M. Carpentier, D.M. Silva, J.G. Gill-Couture, Y. Caron



Prix Yves-Roy - Meilleur projet en génie logiciel orienté-objet

Cette session – ?????

C’est peut être vous ☺

Logistique (GLO-2004)

- Le laboratoire est un « laboratoire virtuel »; il s'agit de la réalisation du projet de session.
- 4 livrables
- Démo livrables #3 et #4 devant le groupe
- **Pour les étudiants « en classe » capsule prévoit une plage horaire qui sera utilisée pour la démo et les rencontre avec les auxiliaires d'enseignement, de même que pour vos rencontres d'équipe**
- Consignes de sécurité dans les laboratoires
- Support Java

Plagiat

- L'Université Laval est abonnée à un service de détection de plagiat et nous en ferons usage dans ce cours.
- Dans un contexte de développement logiciel, la réutilisation du code est considérée comme une bonne pratique mais les situations suivantes seront considérées comme du plagiat:
  - réutiliser du code source provenant d'un livre ou d'Internet sans en citer la source via un commentaire dans le code source et/ou sans avoir obtenu l'accord du professeur;
  - réutiliser le code d'un autre étudiant (qu'il s'agisse d'un étudiant de la même classe ou d'une autre session);
  - réutiliser du code source provenant d'un travail réalisé par vous-même dans un autre cours, sans le mentionner dans le code source et/ou sans avoir obtenu l'accord du professeur du cours original.

Contenu des examens

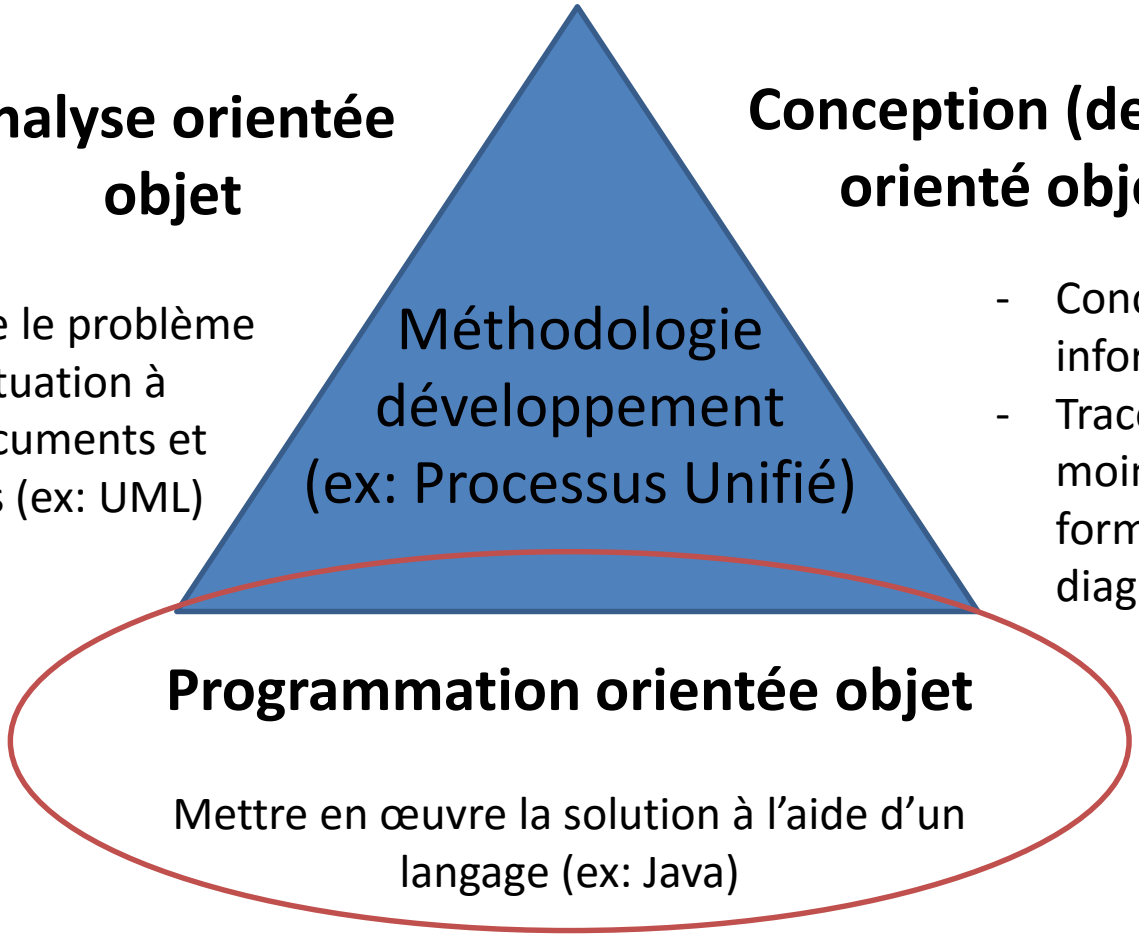
- Méthodologie de développement appelée « Processus unifié »
  - Analyse et conception
  - UML

Focus examen 1

- Étude de patrons de conception (« design patterns ») classiques favorisant de bonnes constructions orientées objet
- Éléments récapitulatifs

Focus examen 2

Génie logiciel orienté objet



- Comprendre le problème
- Décrire la situation à l'aide de documents et diagrammes (ex: UML)

- Concevoir une solution informatique
- Tracer des plans (plus ou moins détaillés) sous la forme de documents et diagrammes (ex: UML)

Programmation orientée objet – Concepts

- Encapsulation des données (« **attributs** ») et des traitements (« **méthodes** ») au sein d'entités appelées « **objets** »
- Les objets communiquent entre eux par échange de **messages** (invocation des méthodes). Les attributs d'un objet ne sont pas accessibles par un autre objet (principe d'**encapsulation**).
- **Classe** vs objet
- **Héritage** (redéfinir des méthodes, ajouter des méthodes)
- **Polymorphisme**

Si tout ceci vous semble un peu loin, il est impératif de réviser vos cours précédents!

Orienté objet / Principaux bénéfices

- Du point de vue du « code »
  - Maintenance
  - Réutilisabilité
  - …
- De point de la modélisation
  - Modélisation plus près de la réalité (plus naturelle)
    - (exemple de l'aéroport)
    - (exemple de Robo sapiens)

Robo sapiens

OCTAS

ROBO SAPIENS Version 1.0

* A.ROB *

Sortie 1 Sortie 2

Positif Arrêt Négatif

Positif Arrêt Négatif

Lier les sorties

Mémoriser

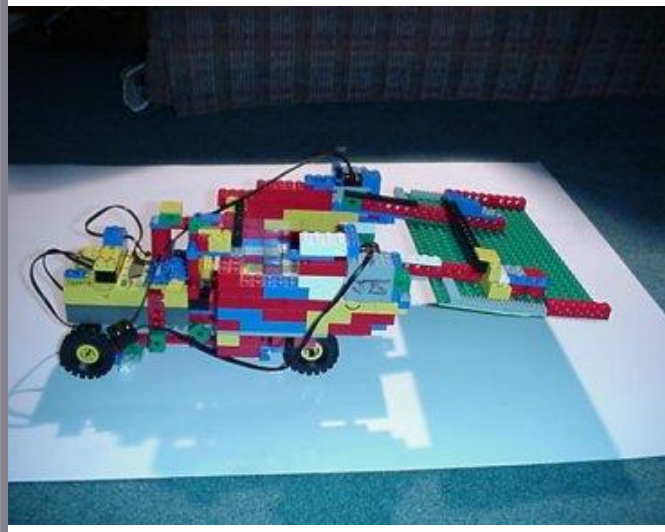
Interrupteur 1:

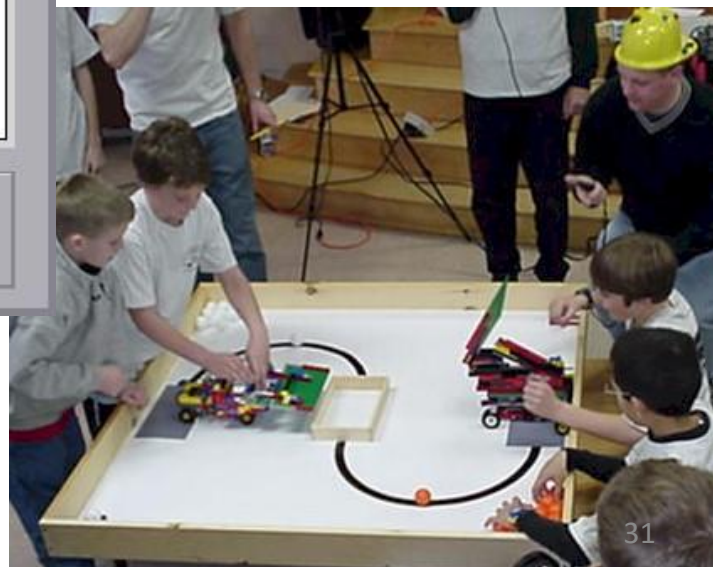
Interrupteur 2:

Exécuter une procédure

```
; La procédure « Longe_un_mur » fait ceci
; ▪ le robot avance jusqu'à rencontrer un
;   obstacle (interrupteur 1)
; ▪ il se tasse d'une largeur et recommence
;   jusqu'a ce qu'il se retrouve a l'extrémité
;   de l'obsacle

; Ce fichier contient d'autres procédure
; intéressantes (utilisées par Longe_un_mur)
; Avance_Un_Peu, Recule_Un_Peu
; FonceDansObstacle, Avance_D'une_Longueur
; Tasse_D'une_Largeur

[Longe_Un_mur]
FonceDansObstacle
Tant que il actif Fais Tasse_d'une_largeur

[TasseD'uneLargeur]
ReculeUnPeu
Droite90
attends 0,5
AvanceD'uneLongueur
Gauche90
AvanceUnPeu

[ReculeUnPeu]
recule
attends 0,3 sec
arretetout

[AvanceUnPeu]
Avance
attends 0,4 sec
ArreteTout
```

Ouvrir Nouveau Enregistrer Quitter Aide

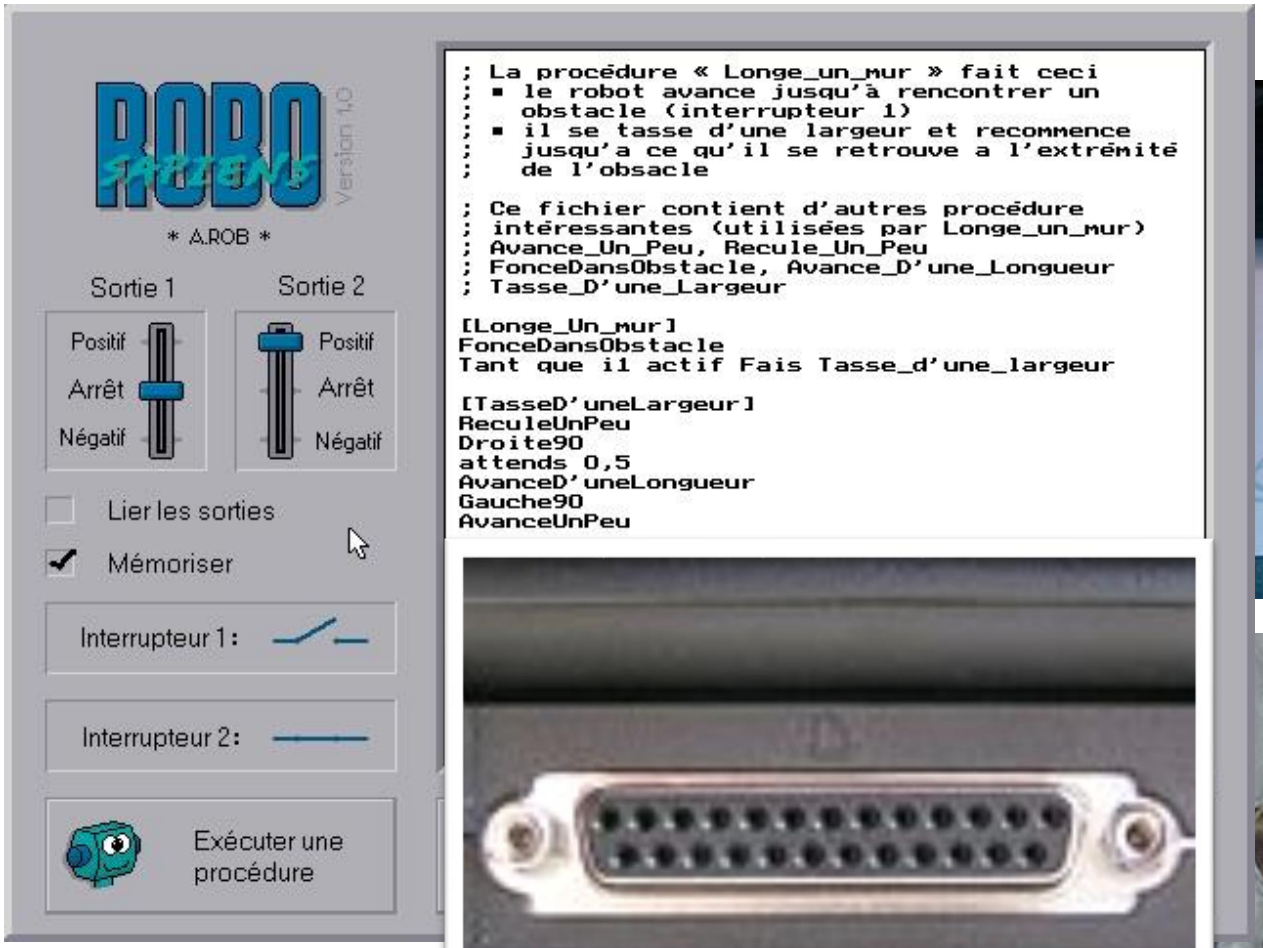




UNIVERSITÉ LAVAL

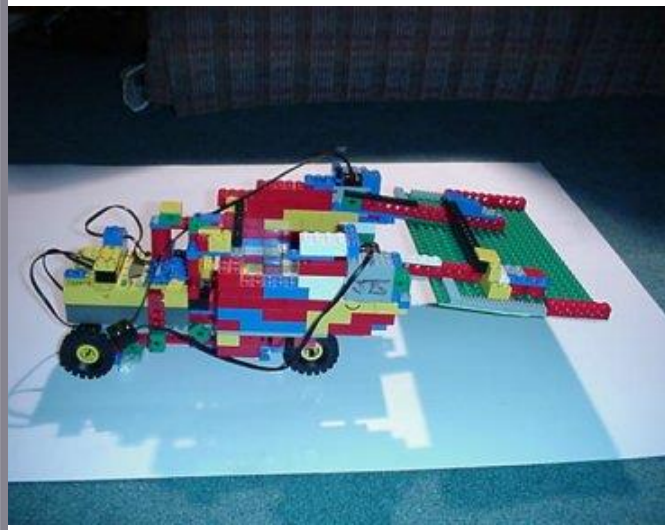
Robo sapiens

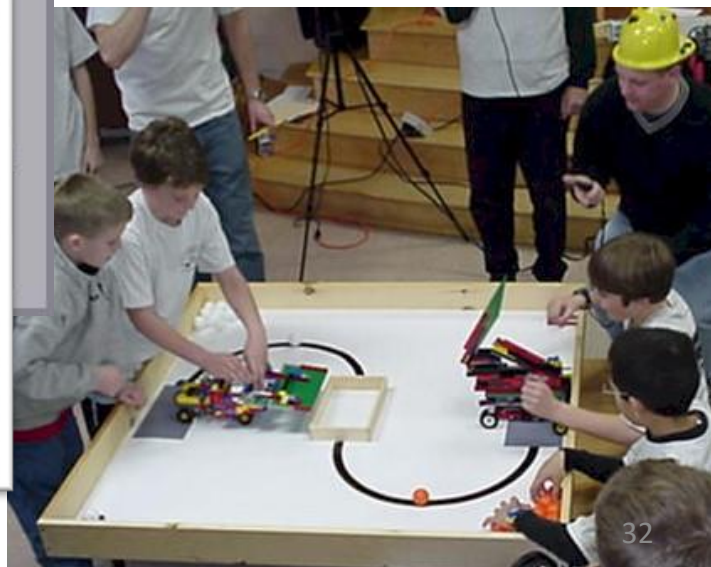
OCTAS




```
; La procédure « Longe_un_mur » fait ceci
; ▪ le robot avance jusqu'à rencontrer un
;   obstacle (interrupteur 1)
; ▪ il se tasse d'une largeur et recommence
;   jusqu'a ce qu'il se retrouve a l'extrémité
;   de l'obsacle

; Ce fichier contient d'autres procédure
; intéressantes (utilisées par Longe_un_mur)
; Avance_Un_Peu, Recule_Un_Peu
; FonceDansObstacle, Avance_D'une_Longueur
; Tasse_D'une_Largeur

[Longe_Un_mur]
FonceDansObstacle
Tant que il actif Fais Tasse_d'une_largeur

[TasseD'uneLargeur]
ReculeUnPeu
Droite90
attends 0,5
AvanceD'uneLongueur
Gauche90
AvanceUnPeu
```


Port parallèle:
https://en.wikipedia.org/wiki/Parallel_port



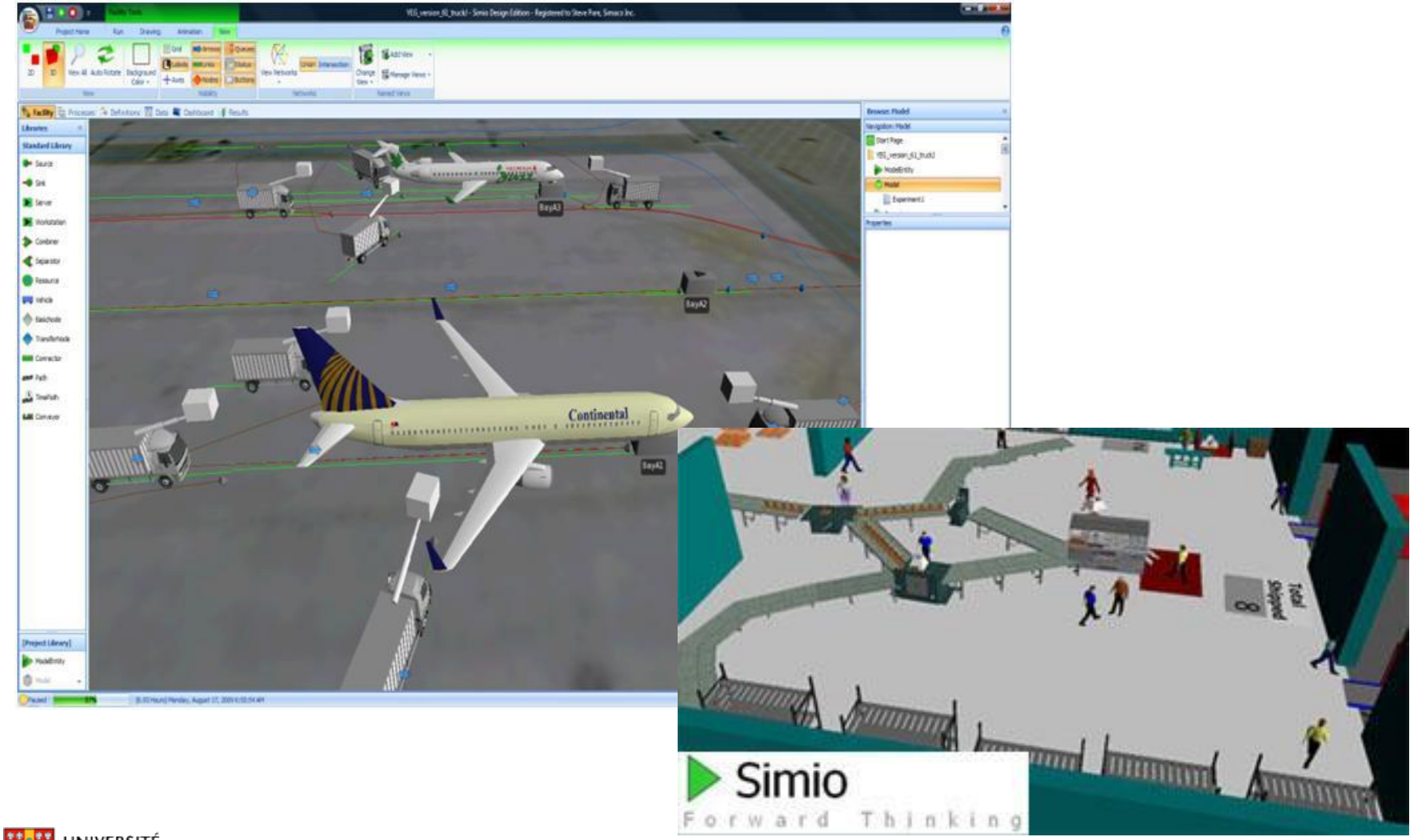


UNIVERSITÉ LAVAL

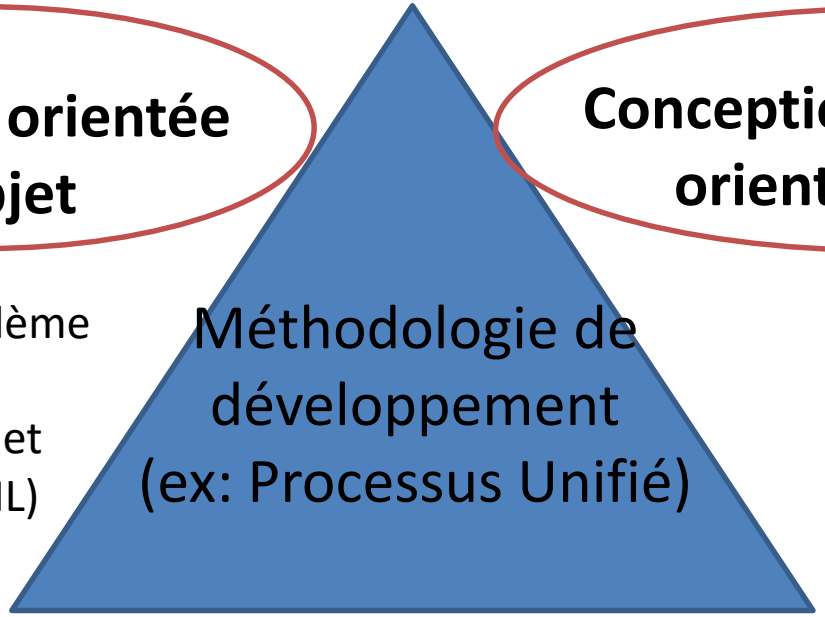
Langage Simula-67 (1967)

- Naissance de l'orienté-objet
  *"The real extension of Simula was the concept that procedures were temporarily stoppable and that the stopped procedures could be referenced. These stoppable procedures were called **classes**. A realization of a class was an **object**. A class could have a number of **attributes** and a number of internal procedures: the **methods** in OO terminology. Classes could be **subclassed**."*
  - http://progopedia.com/language/simula-67/

Logiciel de simulation Simio (2014)



Génie logiciel orienté objet



Analyse orientée objet

- Comprendre le problème
- Décrire la situation à l'aide de documents et diagrammes (ex: UML)

Conception (design) orienté objet

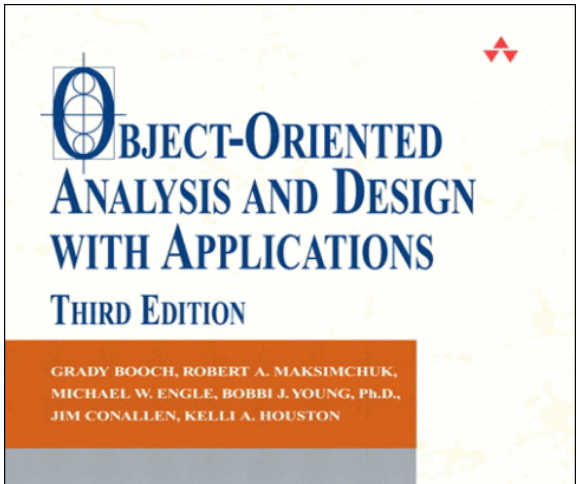
- Concevoir une solution informatique
- Tracer des plans (plus ou moins détaillés) sous la forme de documents et diagrammes (ex: UML)

Programmation orientée objet

Mettre en œuvre la solution à l'aide d'un langage (ex: Java)

Analyse (analysis) vs Conception (design)

- The boundaries between analysis and design are fuzzy.
- In analysis, the focus is to (1) fully analyze the problem at hand, and (2) to model the world by discovering the class/objects that forms the vocabulary of the problem domain.
- In design, (3) we invent the abstractions and mechanisms that provide the design of the solution to be built.



OOA-OOD-OOP

- L'***analyse orientée objet*** (OOA) est une méthode qui examine les exigences d'un projet dans une perspective de ***classes*** et ***d'objets*** tirés du domaine de l'application.
- La ***conception orientée objet*** (OOD) est une méthode de ***design*** comprenant un processus ce ***décomposition*** par ***objets*** et une ***notation*** permettant de dépeindre les ***modèles logiques/physiques*** et ***statiques/dynamiques*** du système en cours de développement.
- La ***programmation par objets*** (OOP) est une méthode ***d'implantation*** par laquelle les programmes sont organisés en ***un ensemble d'objets coopératifs***, chaque objet représentant une instance d'une classe, chaque classe faisant partie d'une ***hiérarchie*** de classes unies par des ***relations d'héritage***.

Unified Modeling Language (UML)

- Langage / formalisme permettant de décrire des systèmes à l'aide des concepts orienté objet de manière visuelle
- Issu de la fusion de différents formalisme compétiteurs (Booch, Rumbaugh, Jacobson) qui ont décidé de s'associer
- Maintenant géré par une organisation internationale (Object Management Group, OMG)

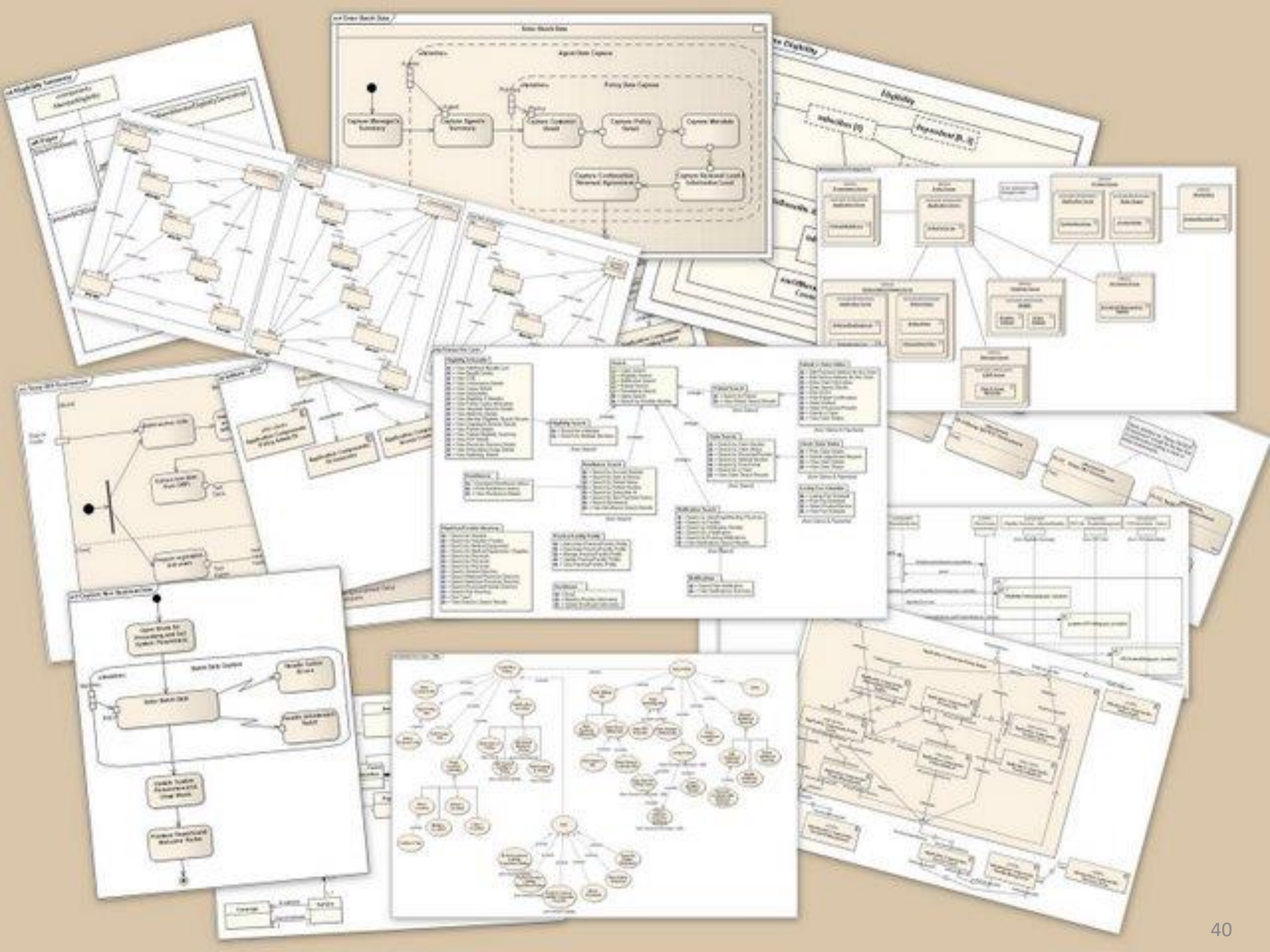
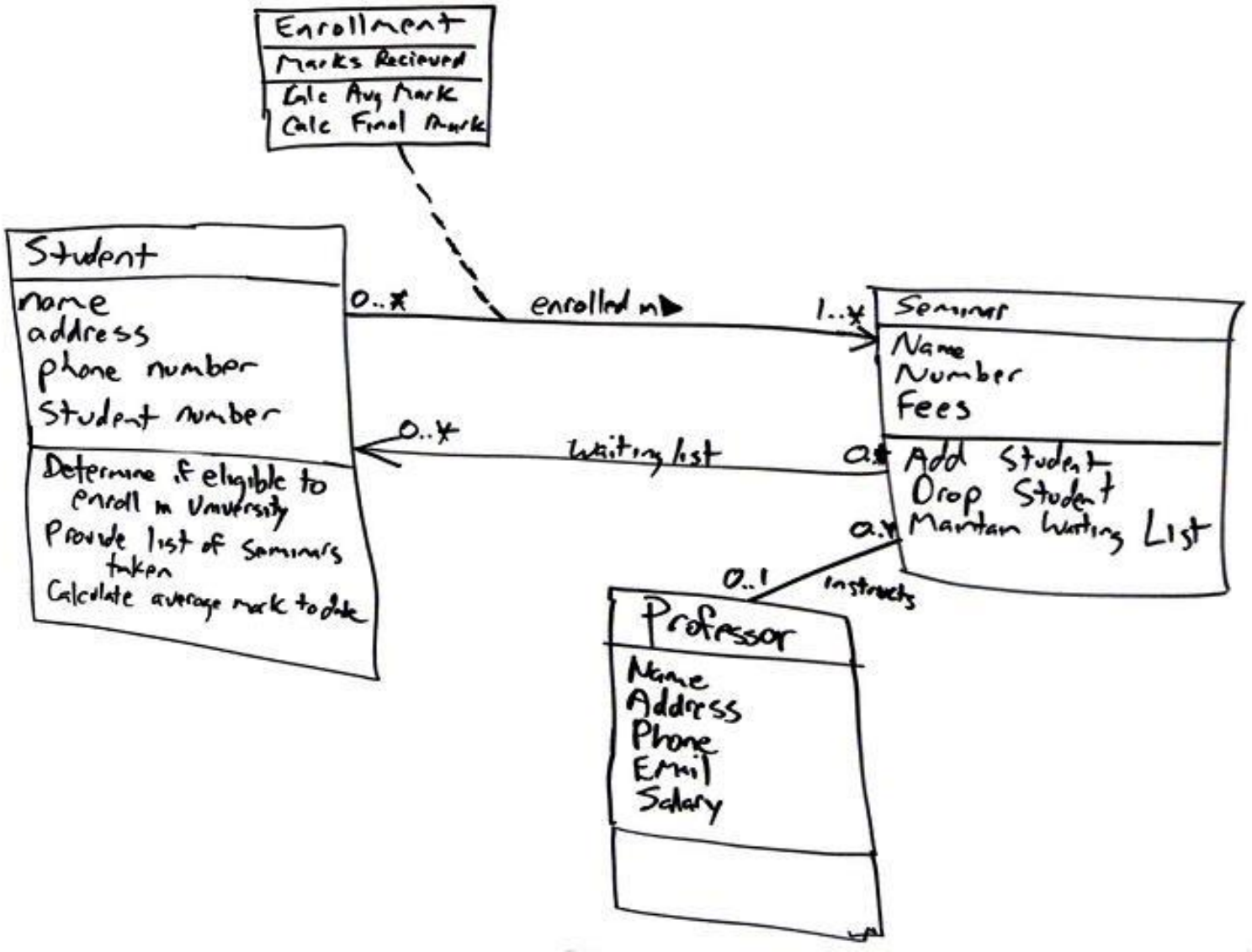


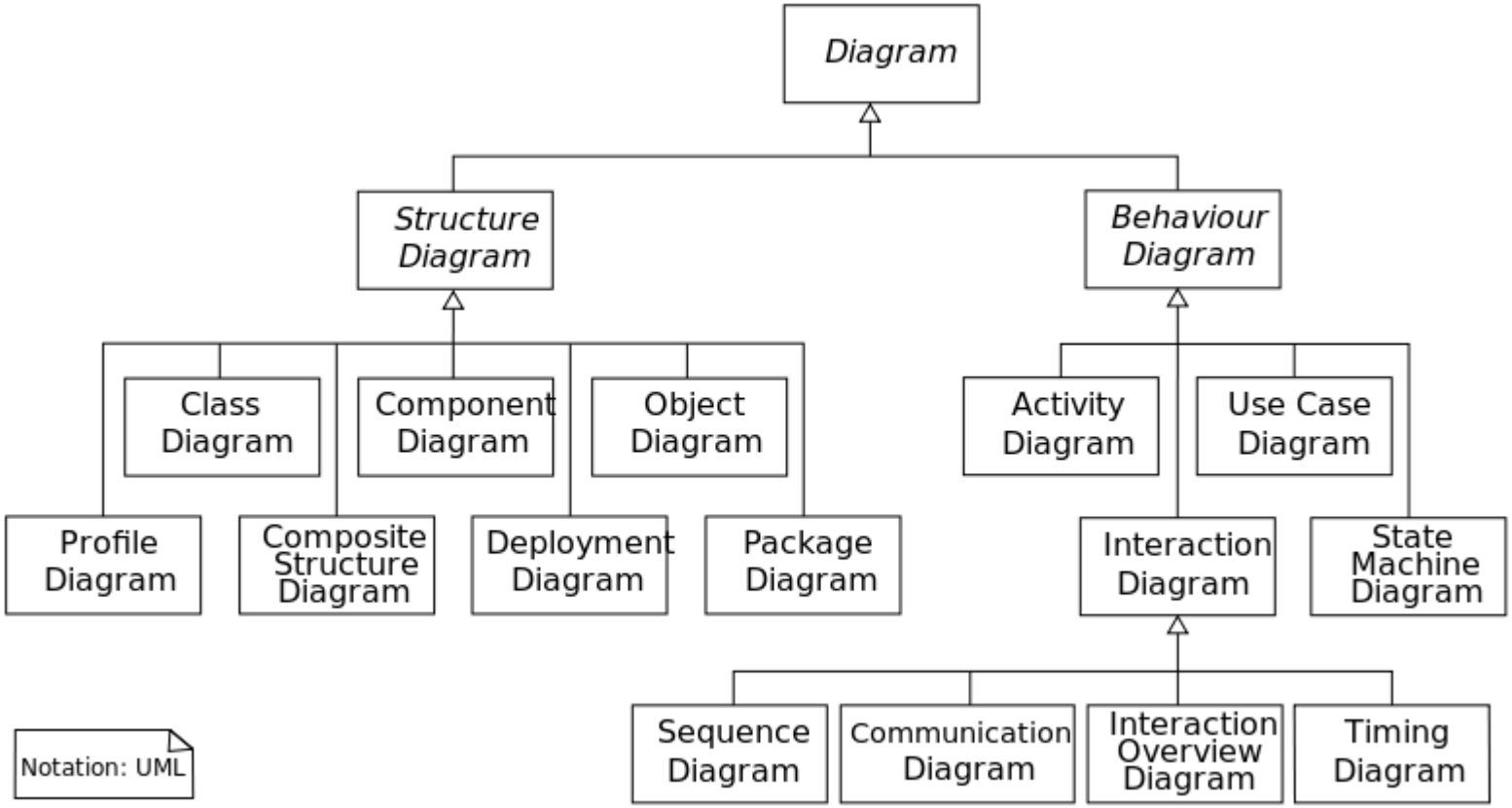
Diagramme de classes



UML: les mêmes diagrammes utilisés à différentes sauces

	ESQUISSE À LA MAIN POUR EXPLICITER LES PARTIES DÉLICATES	Diagrammes plus détaillés créés avec un outil
Modélisation conceptuelle du domaine d'affaires	*Pour l'analyse, on fait UN des deux*	
Modélisation solution (Design) – indépendant du langage de programmation qui sera utilisé	*Pour le design, on choisi généralement UNE*	
Modélisation de la solution tenant compte du langage de programmation qui sera utilisé	On ne fait jamais ça	*de ces trois avenues*

UML : 14 types de diagrammes (ouf!)



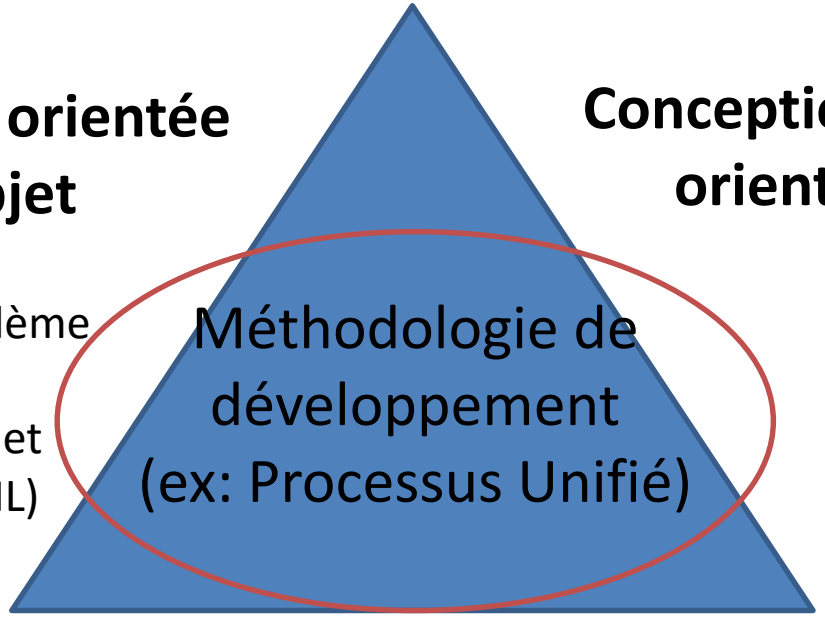
Génie logiciel orienté objet

Analyse orientée objet

- Comprendre le problème
- Décrire la situation à l'aide de documents et diagrammes (ex: UML)

Conception (design) orienté objet

- Concevoir une solution informatique
- Tracer des plans (plus ou moins détaillés) sous la forme de documents et diagrammes (ex: UML)



Programmation orientée objet

Mettre en œuvre la solution à l'aide d'un langage (ex: Java)

Méthodologie / processus de développement logiciel

- Précise une méthodologie à suivre pour développer un logiciel
- Spécifie les activités à réaliser et les livrables (« artefacts ») à produire
- Décrit notamment quand et comment procéder à l'analyse, à la conception, à la programmation, etc..

Processus Unifié (PU)

- Processus de développement itératif reposant sur UML et proposé à l'origine par Booch, Rumbaugh et Jacobson (livre publié en 1999)
- De nouvelles versions (Rational Unified Process) ont été publiées depuis (propriété de IBM)
- Le processus original (UP) peut être vu comme un tronc commun à plusieurs méthodes itératives

Changement cette session

- L’utilisation d’un dépôt git de l’université est obligatoire
- Contribution des coéquipiers et projet pilote
  - Anecdote
  - Pour le bien de tous!
- Offre un nouveau laboratoire “ obligatoire ”

Présentation du projet

VisuaLigue

!!! Important !!!

- Création des équipes sur MonPortail, avec un nom d'équipe
- Demande de création de dépôt git sur pixel
- Laboratoire de la semaine prochaine

- Bonne session à tous!