Master Sciences et Technologies

Mention Informatique

1re année

En Alternance

Université de Lille

FACULTÉ DES SCIENCES ET TECHNOLOGIES
Département Informatique

Objectifs

La mention **Informatique** du Master Sciences et Technologies de l'université de Lille a pour objectifs d'apporter aux étudiants des compétences de haut niveau, à la fois sur le plan théorique et pratique, dans le secteur des sciences de l'information et des technologies des communications.
Cette mention **Informatique** est la poursuite d'études naturelle des étudiants ayant obtenu la licence mention informatique de l'université de Lille ou une licence à forte dominante informatique dans un autre établissement (ou tout autre diplôme de même niveau, sous réserve de la validation des études).
Cette mention est un cycle de deux ans organisé autour d'une première année commune se dérivant en cinq parcours.

ACCÈS À LA FORMATION

- Étudiant ayant suivi et validé un parcours de Licence Informatique
- Après un dispositif de VAE ou de formation continue

Pré-requis

Avoir une licence d'informatique et les connaissances suivantes :

- Programmation impérative,
- Programmation objet,
- Base de données,
- Algorithmique,
- Système et réseaux.

Suivi individuel d'alternance

Le livret électronique permettant à chaque étudiant en alternance la saisie des principales données le concernant (entreprise, mission, référent entreprise, etc.). Le suivi est effectué par un tuteur universitaire et un tuteur en entreprise. Ce suivi se concrétise par un minimum de 2 visites en entreprise entre les tuteurs et l'alternant, de rencontres régulières entre le tuteur universitaire et l'alternant et d'une soutenance.

Détail de la formation

Le volume effectif d'enseignement est de 448 heures en présentiel dont 26 heures d'examens.

- Algorithmique et complexité (5 ECTS, 51h) : 12 séances de 1h de Cours, 1h30 de TD, 1h30 de TDM + 3 heures d'examen
- Anglais (4 ECTS, 26h) : 12 séances de 2h de cours + 2 heures d'examen,
- Architecture évoluée des ordinateurs (5 ECTS, 57h) : 12 séances de 1h30 de Cours, 1h30 de TD, 1h30 de TDM + 3 heures d'examen,
- Architecture et conception des systèmes d'exploitation (5 ECTS, 57h) : 12 séances de 1h30 de Cours, 1h30 de TD, 1h30 de TDM + 3 heures d'examens
- Construction d'applications réparties (5 ECTS, 57h) : 12 séances de 1h30 de Cours, 1h30 de TD, 1h30 de TDM + 3 heures d'examen,
- Génie logiciel (GL) (5 ECTS, 57h) : 12 séances de 1h30 de Cours, 1h30 de TD, 1h30 de TDM + 3 heures d'examen,

- 3 options (3 fois 5 ECTS, 3*45h comprenant 13 séances de 2h de CTD ou 1h30 de Cours et 1h30 de TD/TP et 2 à 3 heures d'examens) à choisir parmi (liste non exhaustive):
  Apprentissage par l'exemple -Bases de données avancées - Concepts avancés des langages de programmation - Business Intelligence - Interface homme-machine - Modélisation 3D et synthèse - Principes et algorithmes cryptographiques - Programmation parallèle et distribuée - Reconnaissance de formes -Spécification et vérification du logiciel - Traitement d'images ...

Algorithmiques et complexité (ACT)				**Total heures :**	**51**	**Total ECTS :**	**5**
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	ACT	12	18	18		3	5
Architecture évoluée des ordinateurs (AeO)				**Total heures :**	**57**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	AEO	18	18	18		3	5
Architecture et conception des systèmes d'exploitation (ASE)				**Total heures :**	**57**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	ASE	18	18	18		3	5
Construction d'applications réparties				**Total heures :**	**57**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	CAR	18	18	18		3	5
Génie Logiciel (GL)				**Total heures :**	**57**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	GL	18	18	18		3	5
Suivi Pédagogique d'Alternance (SPA)				**Total heures :**	**8**	**Total ECTS :**	0
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	SPA		8				0

Option 1				**Total heures :**	**45**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	Option 1	18		24		3	5
Option 2				**Total heures :**	**45**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	Option 2	18		24		3	5
Option 3				**Total heures :**	**45**	**Total ECTS :**	5
		C	**TD**	**TDM**	**TPA**	**Exam**	**ECTS**
	Option 3	18		24		3	5
Anglais				**Total heures :**	**26**	**Total ECTS :**	**4**
		C	**TDM**	**TD**	**TPA**	**Exam**	**ECTS**
	Anglais	24				2	4

Descriptif des UEs du 1er semestre

ACT : Algorithmes et ComplexiTé (5 ECTS, 51h)

Volume horaire

Cet enseignement obligatoire a lieu au premier semestre du Master1. Il est organisé sur douze semaines, en une séance hebdomadaire de cours d'une durée d'1h, une séance hebdomadaire de Travaux Dirigés d'une durée de 1h30 et TP d'une durée de 1h30. L'examen final est de 3 heures.

Objectifs

A l'issue de ce module les étudiants doivent être capables d'analyser un problème algorithmique - modélisation, complexité, réduction à un problème connu -, de proposer une solution et de l'analyser

- Connaitre et savoir appliquer a bon escient les paradigmes classiques de conception d'algorithmes (programmation dynamique, diviser et régner, algorithmes gloutons, backtracking, ...).
- Connaitre et comprendre la notion de complexité de problèmes, de classes de complexité - P, NP et ExpTime -, de réduction polynomiale, savoir faire des preuves élémentaires de NP-dureté.
- Connaitre et savoir utiliser quelques méthodes avancées pour résoudre de manière exacte ou approchée des problèmes NP (Branch and Bound, algorithmes probabilistes, heuristiques, méta-heuristiques, …)

Savoir répondre à ces questions est souvent dur. Le cours ne se veut ni un cours "formel" sur la correction, l'analyse d'algorithmes et la complexité de problèmes, ni un cours encyclopédique sur les algorithmes. L'objectif du cours est simplement de vous donner quelques éléments de réponse. Dans la mesure du possible, l'accent sera plutôt mis sur les méthodes, même si le cours sera illustré avec des algorithmes "classiques".

Contenu

- Présentation. Exemples. Rappel sur la complexité des algorithmes.
- Quelques schémas "classiques" d'algorithmes (2-3 cours)} Nous étudierons deux ou trois paradigmes : la programmation dynamique, les algorithmes gloutons (et éventuellement "Diviser pour Régner").
- Complexité de problèmes : Qu'est-ce que la complexité intrinsèque d'un problème ? Qu'est-ce qu'un problème "dur" ? Nous aborderons : Les classes P et NP, la notion de réductions polynômiales, les propriétés NP-dures, le théorème de Cook, la problématique P=NP...
- Un peu d'algorithmique "avancée" : Nous (re)verrons d'abord quelques méthodes classiques de recherche de solution, certaines permettant d'appréhender des problèmes "durs" :
- Méthodes classiques de recherche : backtracking, minmax, séparation-évaluation...
- Heuristiques et leur garantie
- Ensuite, selon le temps disponible on évoquera les métaheuristiques, les algorithmes probabilistes,
- La calculabilité : Qu'est-ce qu'un algorithme ? Quels problèmes peut-on "résoudre par ordinateur" ? Nous aborderons la notion de modèle de calcul, et montrerons les limites de la calculabilité.

Remarque : En cours comme en TD, les algorithmes seront écrits en pseudo-language. Le langage utilisé en TP sera JAVA.

AEO : Architecture évoluée des ordinateurs (5 ECTS, 57h)

Volume horaire

Cet enseignement obligatoire a lieu au premier semestre du Master1. Il est organisé sur douze semaines, en une séance hebdomadaire de cours d'une durée d'1h30, une séance hebdomadaire de Travaux Dirigés d'une durée de 1h30 et 12 séances de TP d'une durée de 1h30. L'examen final est de 3 heures.

Objectifs

Ce cours vise à comprendre le fonctionnement des architectures, évoluées, des ordinateurs (architectures parallèles). Il permet également de se familiariser avec les méthodes et outils modernes de conception de systèmes embarqués sur FPGAs.

Contenu

A l'issue de ce module les étudiants doivent :

- Lire une spéciation d'un ordinateur et en comprendre les caractéristiques matérielles voire de comparer les solutions proposées par les constructeurs. Cela recouvre le fonctionnement du cœur du processeur : modèle de fonctionnement RISC/CISC, pipeline, multi cœur, partage mémoire, bus internes ; la mémoire hiérarchique : cache, entrelacement des bancs, protocoles de cohérences ; et communications : réseaux dynamiques et statiques, routage.
- Tirer partie au mieux des solutions matérielles lors de la mise en oeuvre d'applications. Des notions de placement de données et de répartition du calcul sont abordées en termes d'exploitation des performances matérielles de la machine. ´

— Intégrer les nouvelles architectures multi cœur et cluster dans l'entreprise, de maîtriser la complexité des machines parallèles. Les fonctionnements multi cœurs, multi-thread et multi-pipeline sont introduits en s'appuyant a des solutions commerciales parallèles et massivement parallèles. Sur ce dernier point, le TOP 500 sert de référence chaque année.
— Pour des processeurs embarqués, assembler des composants élémentaires matériels comme accélérateur de performances. Dans le cadre de l'opération une carte/un étudiant développée par le GDR SoC/SiP, chaque binôme d'étudiant reçoit pour la durée du semestre une carte FPGA Nexys3 (Xilinx Spartan 6) sur laquelle ils réalisent leurs TPs en VHDL. Lors de ces séances ils manipulent un processeur softcore (processeur HoMade développé par l'enseignant) avec lequel il est possible d'intégrer des unités calculatoires comme accélérateurs matériels.

ANG : Anglais (4 ECTS, 26h)

Volume horaire
Cette unité se déroule MS1 du master science - mention informatique. Il s'agit d'une UE obligatoire de cette mention. L'enseignement est organisé sous la forme d'une séance hebdomadaire de 2 heures durant 12 semaines. L'examen final est de 2 heures.

Objectifs
Développer des compétences indispensables dans la vie professionnelle, en particulier l'expression écrite et orale. La compréhension de l'écrit et de l'oral sera aussi pratiquée.

Contenu
Le travail écrit se fera à partir de textes étudiés en commun, avec exercices de compréhension et un travail de réflexion sur la langue (vocabulaire et grammaire). Les étudiants s'entraîneront à réécrire chaque texte de manière synthétique, fidèle et structurée, c'est-à-dire à faire en faire un résumé.

Le travail d'expression orale visera à développer la capacité à prendre la parole en public de manière prolongée, à travers des exposés. Dans un premier temps, des techniques de présentation orale seront étudiées. Ensuite, les étudiants prépareront des exposés courts sur des sujets techniques vulgarisés, ou touchant à un centre d'intérêt personnel. Les exposés devront être accompagnés d'un diaporama.

ASE : Architecture des systèmes d'exploitation (5 ECTS, 57h)

Volume horaire
Cette unité se déroule au premier semestre du master info. Il s'agit d'une UE obligatoire.
L'enseignement est organisé sous la forme d'une séance hebdomadaire d'1h30 de cours, d'une séance hebdomadaire d'1h30 de TD et d'une séance hebdomadaire de 1h30 de travaux pratiques. L'examen finale est de 3 heures.

Objectifs
A l'issue de l'enseignement, l'étudiant est capable d'écrire les éléments clefs d'un système d'exploitation, d'en expliquer le comportement pour en tirer le meilleur parti et d'en concevoir ou d'en faire évoluer des parties (pilote de périphérique, ordonnancement spécifique, ...). A travers le cours, l'étudiant découvre le lien entre le matériel et le logiciel et apprend à programmer des matériels à nu.
Plus précisément, l'objectif est de savoir :
— Expliquer le fonctionnement d'un système d'exploitation et d'en prédire le comportement ;
— Tirer le meilleur parti d'un système d'exploitation et d'éviter les maladresses d'usage lors de la conception d'applications ;
— Concevoir des éléments d'un système d'exploitation, tel que les pilotes d'un périphérique ou une politique d'ordonnancement particulière ;
— Avoir programmé un matériel nu, et savoir bâtir des abstractions performantes sur la base d'un matériel conventionnel.

Contenu
Le cours dresse un état de l'art des différentes techniques et algorithmes dédiés à gestion et à l'abstraction des ressources matérielles élémentaires, à savoir, la mémoire de travail la mémoire

persistante, et le microprocesseur. La gestion des entrée/sorties est détaillée dans le cours de réseaux (UE RSX de la licence mention informatique). Un projet de TP vise à reconstituer (par la pratique) les couches composant un système de fichiers. Un autre projet à mettre en oeuvre un ordonnanceur de tâches.
La réalisation de composants système est une activité qui implique de nombreuses heures de manipulation. Ce module comporte la réalisation d'un prototype qui implique un travail personnel substantiel en plus des heures de TD et de TP encadrées.

Génie logiciel (GL) (5 ECTS, 57h)

Volume horaire
12 * Cours (1h30)=18h et 12 * TD (1h30) = 18h+ 12*TP (1h30) = 18h+ travail personnel. L'examen final est de 3 heures.

Objectifs
Ce cours a pour objectif de concevoir des applications orientées objets de façon systématique et reproductible. Dans ce cours, l'étudiant(e) apprendra à rechercher et établir les fonctionnalités d'une application, et à les modéliser sous forme de cas d'utilisation (C.U.) et de scénarios. IL ou elle apprendra à rechercher les classes et les acteurs nécessaires à la conception de l'application. L'étudiant(e) apprendra aussi des bonnes pratiques de conception, comme l'utilisation de patron de conception (design pattern), le découpage en couches de l'architecture, la structuration en paquetages et le maquettage.

Contenu du cours
L'objectif de cette UE est d'apprendre à concevoir des systèmes informatiques (ou applications) complexes en utilisant une approche orientée objets. A l'issue de ce module l'étudiant :

- Sera capable de concevoir une application qui fonctionne de façon systématique et reproductible ;
- Sera capable de spécifier les besoins du système (i.e. : l'application) à partir de documents et d'interviews ; Il exprimera les besoins a l'aide d'acteurs, de scenario et de cas d'utilisation ;
- Sera capable d'analyser les besoins afin de proposer une solution pour le système, sous forme de « modèle de conception ». Il sera capable de proposer une première solution, puis de raffiner les objectifs de conception et de décomposer le système en sous-systèmes plus facilement appréhendable ;
- Maîtrisera la réutilisation de solutions existantes proposées sous forme de patterns (patrons) ;
- Sera capable de spécifier les interfaces des objets du système, d'identifier les attributs et maîtrisera les diagrammes UML permettant la modélisation des différentes phases de la conception du logiciel ;
- Pourra générer le squelette des classes du système à partir des modèles de conceptions, opérations, de spécifier les invariants et les pre et post conditions.
-

Descriptif des UEs du 2nd semestre

Le second semestre est commun à l'ensemble des spécialités du master Informatique et comprend les modules :

Suivi Pédagogique d'Alternance (SPA) (0 ECTS, 8h)

Description
Le suivi est effectué par un tuteur universitaire et un tuteur en entreprise. Ce suivi se concrétise par un minimum de 2 visites en entreprise entre les tuteurs et l'alternant, de rencontres régulières entre le tuteur universitaire et l'alternant et d'une soutenance.

Construction d'applications réparties (CAR) (5 ECTS, 57h)

Volume horaire
12 semaines de cours organisées en 1 cours d'1h30 (=18h), 1 TD d'1h30 (=18h) et un TP de 1h30 (=18h). L'examen final est de 3 heures.

Objectifs
L'objectif de cette UE est d'appréhender et de mettre en pratique les concepts des applications réparties. De part la multiplication des matériels informatiques et le développement des réseaux, il est de plus en plus courant qu'une application mette en oeuvre des interactions sur plusieurs (voire de nombreux) sites afin de rendre son service. Le domaine de l'informatique répartie étant très vaste, ce cours met principalement l'accent sur les styles d'architecture que l'on rencontre : les applications client/serveur sur Internet, les objets distribués avec Java RMI, le standard CORBA de l'OMG, les Web Services du W3C, les composants EJB et Java EE.

Contenu
- Introduction et concepts des applications répartis et client/serveur
  - notions de *middleware*, mode requête/réponse, MOM, RPC, bus,
  - client/serveur 2, 3, n tiers,
  - *proxy*, représentation de données, passage de paramètres en réparti.
- Applications réparties en mode message
  - protocoles client/serveur, modes bloquant/non bloquant, adressage,
  - modes point à point, multiploints,
  - gestion de la concurrence et de la synchronisation.
- Applications Web et servlet
  - traitement côtés client et serveur, aspect du Web,
  - notion de session, gestion de la concurrence.
- Notions objets répartis
  - client/serveur et programmation objet, notion d'interface, de service,
  - nommage, contrôle d'accès, durée de vie des objets en environnement réparti,
  - migration, réplication, ramasse-miettes.
- Java RMI
  - mode de programmation, générateur de souches, souches dynamiques, passage de paramètres,
  - services de nommage, d'activation, ramasse-miettes réparti,
  - RMI-IIOP, JRMP, chargement dynamique de classes, RMI et pare-feux.
- OMG CORBA
  - IDL, modèle de programmation, services, fonctionnalités avancées.
- Web Services
  - principes, XML-RPC, SOAP, WSDL.
- Notions composants répartis
- Java Enterprise Edition (Java EE, J2EE)

- principes, servlet, JSP, JDBC, EJB 3 session, entité,
- services techniques (persistance, nommage, transaction, sécurité),
- patrons de conception Java EE.

AeA : Applications et algorithmes (5ECTS, 45h)

Volume horaire

- 12 Cours C-TD de 1h30 (=18h)
- 12 TP de 2h (=24h)
- L'examen final est de 3 heures.

Objectifs

A l'issue de ce module les étudiants doivent pouvoir identifier les défis algorithmiques qui se cachent derrière des problèmes issus de la vie réelle et de les résoudre en adaptant les outils informatiques qu'ils auront appris tout au long du module. En particulier, ils seront capables d'appliquer des outils fondamentaux issus de l'algorithmique des mots, des arbres et des graphes. Ils auront des compétences pour comprendre et proposer des modèles informatiques pertinents qui permettent d'appréhender des problématiques récurrentes dans plusieurs domaines d'application.

Contenu du cours

Ce module vise l'apprentissage d'outils algorithmiques transversaux en informatique et la découverte du rôle crucial qu'ils jouent dans le contexte d'applications concrètes. Plusieurs domaines (bio-informatique, réseaux de communications, réseaux du web, ou encore réseaux sociaux et collaboratifs) ont en effet en commun de faire appel a des modèles et des techniques algorithmiques avancées, qui s'articulent autour des mots, des arbres, et des graphes. Il s'agit de présenter ces différents outils et de montrer de façon pratique leurs intérêts et leur interaction forte avec des applications réelles.
Tout au long des cours, les étudiants sont stimulés et sensibilises à des contextes applicatifs particuliers en montrant comment ils peuvent être modélisés de façon informatique. En parallèle, des techniques et des algorithmes avancés qui sont souvent utilisés pour résoudre ces problèmes sont présentés et analysés :

- Des algorithmes sur les arbres et sur les mots (applications en Bio-informatique et en informatique musicale) : alignement de séquences, arbres de phylogénie, recherche de motifs.
- Des algorithmes sur les graphes (applications aux réseaux de communication) : algorithmes de calcul sous-structures, colorations, stable, et couplages.
- Des algorithmes sur les grands graphes (application aux réseaux du web et aux réseaux sociaux) : modèles pour l'internet, algorithme PageRank, algorithmes pour le codage, le partage, la recherche de données à grande échelle.

ASE++/ASEa : Architecture et conception avancées des systèmes d'exploitation (5ECTS, 45h)

Volume horaire

- 12 Cours TD de 1h30 (=18h)
- 12 TP de 2h (=24h)
- L'examen final est de 3 heures.

Objectifs

Ce module est la continuité de l'**unité ASE** du premier semestre. Nous nous efforcerons ici à

1-- intégrer les différents aspects abordés lors des séances de TD/TP d'ASE,

2-- à prendre en compte les architectures matérielles actuelles, en particulier multicoeur, et
3-- à développer quelques composants dans un véritable noyau de système d'exploitation.

Les enseignements se dérouleront en "mode projet" et solliciteront particulièrement l'implication des étudianst dans l'avancement.

Contenu du cours

- dans un premier temps, l'**intégration des différents "modules"** développés en ASE permetra de
    - construire un réel coeur de système d'exploitation
    - associant la gestion de contextes réalisant des entrées/sorties et s'exécutant dans des environnements mémoires isolés.
- nous nous attacherons ensuite à **exploiter les architectures multi-coeurs** en adaptant les services du système et en particulier l'ordonnancement et le changement de contextes. Il s'agira principalement de
    - gérer le redirection des interruptions matérielles entre les différents coeurs,
    - de partager au sein du noyau des structures de données gérant l'activité de chacun des coeurs,
    - de mettre en oeuvre des algorithmes d'oronnacement adaptés aux multi-coeurs.
- enfin, nous apprendrons à **travailler sur le noyau Linux**. Il s'agira
    - de comprendre à haut niveau la structure du noyau Linux,
    - de savoir comment compiler, installer et exécuter Linux sur une machine virtuelle QEMU pour développer et déboguer,
    - de produire et interpréter la trace des événements du système,
    - pour enfin modifier un module Linux existant tel l'ordonnanceur, ou un mécanisme de synchronisation comme celui des spin-locks.

Business intelligence (BI) (5ECTS, 45h)

Volume horaire

- 12 Cours de 1h30 (=18h)
- 12 TP de 2h (=24h)
- L'examen final est de 3 heures.

Objectifs

Ce cours a pour objectif de présenter des outils et des techniques liés à l'informatique décisionnelle. De nombreux moyens informatiques sont aujourd'hui mis en œuvre pour aider les organes de décision des entreprises ; de l'entrepôt de données qui définit un support au système d'information décisionnel, aux outils de fouille de données permettant d'en extraire de nouvelles connaissances. Les entrepôts de données (datawarehouse) et la fouille de données (datamining) sont les éléments d'un domaine de recherche et de développement très actif actuellement.
Le but du cours est de familiariser l'étudiant avec les concepts d'entrepôt de données et d'extraction de connaissances à partir de gros volumes de données. L'étudiant apprendra à utiliser les méthodes d'organisation de grands ensembles de données ainsi que les outils et techniques permettant de les explorer et d'en extraire des connaissances.

Contenu

Au niveau du contenu :
1. Introduction a la Business Intelligence
2. Le datawarehouse/Entrepôt de données

- Définition de l'entrepôt de données
- Alimentation, ETL, Metadata
- Conception de l'entrepôt de données
  - Modélisation : schéma en étoile, en flocon, en constellation
  - Dimensionnement
- Exemples issus du monde industriel
- L'analyse multidimensionnelle et le OLAP
  - Définition
  - OLAP, ROLAP, MOLAP
  - SQL-OLAP + exemple en Oracle
- Outils du marché décisionnel

3. Datamining/Fouille de données
   - Les données, leur représentation
   - Définition et introduction au datamining
   - Le Clustering
     - Clustering par voisinage
     - Clustering hiérarchique
   - Règles d'association `
   - Classification
     - Arbre de décision
     - Gain, Indice de Gini etc.
     - Classification baysienne

En travaux pratiques, nous verrons comment concevoir un entrepôt de données et l'utiliser pour faire de la fouille de données. Nous utiliserons également les méthodes classiques d'exploration de données. L'environnement de TP est sous Oracle pour la partie entrepôt de données et Weka, Orange et RapidMiner pour la partie fouille de données.

COMPIL (5ECTS, 45h)

Volume horaire
1h30 de Cours et 2h de TD/TP par semaine, sur 12 semaines. L'examen final est de 3 heures.

Objectifs
Cette UE vise à donner une culture de la compilation aux étudiants et de leur donner les outils conceptuels leur permettant d'écrire un compilateur (au sens de traducteur d'un langage dans un autre).

Contenu
A l'issue de ce module les étudiants doivent :

- Connaître l'organisation des différentes phases de la compilation : analyse lexicale, analyse syntaxique, analyses statiques, optimisations, génération de code ;
- Savoir reconnaître et comprendre l'intérêt des différentes représentations intermédiaires d'un compilateur (grammaire, arbre abstrait, forme à assignation unique statique, code à 3 adresses, etc.) ;
- Savoir contrôler un compilateur de référence tel GCC ou LLVM par l'utilisation de ses options ;
- Connaître les différents types de compilateur (vers du code natif, vers une machine virtuelle, à la volée, source à source) et leurs particularités ;
- Connaître et savoir implémenter les principales optimisations classiques ;
- Etre capable d'écrire un compilateur pour un langage simple.

Fondement de l'apprentissage automatique (FAA) (5ECTS, 45h)

Volume horaire 1h30 de Cours et 2h de TD/TP par semaine, sur 12 semaines. L'examen final est de 3 heures.

Objectifs

L'objectif principal de cette UE est de former les étudiants aux méthodes modernes d'apprentissage qui sont aujourd'hui au cœur d'un grand nombre de systèmes utilisés quotidiennement (les moteurs de recherche, les services de recommandation, les appareils photo détectant les visages, les téléphones portables apprenant des profils, les services vocaux reconnaissant la parole, *etc.*). Ainsi, les étudiants auront les compétences pour comprendre le monde numérique qui les entoure et l'utilisation qui peut être faite de données (données laissées sur le web, collectées par les opérateurs réseau ou de téléphonie *etc.*). Ils auront aussi les bases pour concevoir eux-mêmes des systèmes traitant automatiquement des données, pour constituer des bases de données pouvant être utilisées par un système d'apprentissage (comprenant l'importance d'une collecte et d'une annotation de qualité et la valeur de ces données). On s'attachera aussi à comprendre les qualités et les défauts de différents algorithmes afin d'associer le bon algorithme (et ses paramètres) à un problème donné. Un autre objectif sera de permettre aux étudiants de se forger un esprit critique quant aux résultats produits par les systèmes d'apprentissage (notion de risque, de validation croisée, de sur-apprentissage, de biais etc.). Toutes ces notions seront assimilées très pratiquement sur des cas concrets.

Contenu

L'apprentissage automatique (*Machine Learning* en anglais) décrit un champs de l'intelligence artificielle dans lequel un agent (un ordinateur, un robot, un avatar virtuel, un jeu vidéo *etc.*) effectue un traitement d'information en s'inspirant d'exemples (de données) et non pas à partir d'une description explicite de ce traitement. Il s'agit donc de permettre à la machine de généraliser un traitement à des données qu'elle n'a jamais vues. Il existe plusieurs types d'apprentissage dont 3 seront particulièrement développés dans le cours. On s'intéressera tout d'abord à l'apprentissage supervisé qui comprend par exemple la classification et la régression de fonctions (par réseaux de neurones, Machines à Vecteurs Supports (SVM), arbres de décision). On étudiera ensuite l'apprentissage non-supervisé dont l'objectif est de partitionner un jeu de données en groupes de données similaires (K-means, Neural Gas). On terminera par l'apprentissage actif qui permet à la machine de sélectionner les exemples dont elle a besoin pour apprendre et en réduire le nombre.

HECI : Histoire et épistémologie du calcul et de l'informatique (5ECTS, 45h)

Volume horaire
L'enseignement est organisé sous la forme de deux séances hebdomadaires d'1 heure 30 de cours (18h) et de 2 heures de TD (24h) durant 12 semaines. L'examen final est de 3 heures.

Objectifs
Cette UE a essentiellement deux objectifs :
Sensibiliser les étudiants aux principales étapes de l'informatique, à la fois du coté matériel (machines à calculer, premiers ordinateurs, lampes, transistors, cartes perforées, etc.) que du point de vue de l'évolution des idées (programme de Hilbert, mécanisation du raisonnement, résultat d'impossibilité de Gödel, thèse de Church sur l'équivalence des modèles de calcul, ...)
Apporter des éléments de réponse à certaines questions philosophiques ou sociétales posées par le développement de l'informatique : est-il, par exemple, possible d'empêcher la duplication incontrôlée de données numériques (le "piratage") ? Les ordinateurs pourront-ils à terme supplanter les êtres humains dans tous les domaines (échecs, jeopardy,...) ? Est-ce qu'une machine peut être « intelligente » ? Est-ce qu'une preuve mathématique, qu'on ne peut pas vérifier à la main, est "vraie" ? Est-ce que les ordinateurs peuvent produire du hasard ? Etc.
A l'issue de l'enseignement, l'étudiant est capable de :

- Naviguer dans les grandes étapes de l'histoire du développement de l'informatique,

— Raisonner en scientifique sur les problèmes sociaux ou philosophiques posés par le développement de l'informatique,
— Lire, comprendre et exploiter des articles scientifiques,
— Présenter à l'oral son travail devant un groupe,
— Etre en capacité de réinvestir les connaissances acquises dans un contexte professionnel pour être un leader dans l'innovation ("hi-tech innovation leader").

Contenu

- Histoire du calcul et de l'informatique
  - Instruments de calcul anciens.
  - De Pascal à Babbage.
  - La mécanographie.
  - La naissance du calcul électronique, l'Eniac, Turing et Enigma.
- Problèmes philosophiques liés à l'IA, aux théories du calcul, de l'information et de la complexité
  Thèmes possibles parmi d'autres.
  Discussions autour de l'intelligence artificielle, le test de Turing, les sciences cognitives et le computationnalisme. L'objection de Lucas. Penrose.
  Effectivité et praticabilité. Thèse de Church (diverses versions). Classes de complexité, notion de faisabilité. Modèles de calcul classique et quantique. Franchissement de la barrière de Turing. Rapports avec la physique.
  La théorie du calcul et la logique (le concept de système formel, l'opposition vrai/démontrable, l'indécidabilité et son interprétation, la notion de modèle, etc.)
  Réflexion sur la notion de démonstration automatique : une démonstration faite par ordinateur et trop complexe pour être vérifiée par l'homme est-elle valide ?
  La théorie algorithmique de l'information et son utilisation en épistémologie (le principe du rasoir d'Occam), en physique (entropie, calculs réversibles, etc.), en philosophie des mathématiques (les nombres oméga de Chaitin).
  La notion de hasard en informatique (les générateurs pseudo-aléatoires, les générateurs pour la cryptographie, les suites aléatoires au sens de Martin-Löf).

IHM : Interaction homme-machine (5ECTS, 45h)

Volume horaire
Cet enseignement est une option qui a lieu au second semestre du Master 1. Il est organisé sur 12 semaines, en une séance de Cours TD hebdomadaire d'une durée de 1H30 et une séance de TP hebdomadaire d'une durée de 2H. L'examen final est de 3 heures.

Objectifs
L'objectif de cette option est d'étudier la conception et le développement d'interfaces graphiques, de la traduction des besoins utilisateurs en cahier des charges, en suivant la démarche de conception centrée utilisateur, puis du cahier des charges à l'application, par l'utilisation de composants, la gestion d'événements et l'utilisation de techniques d'interaction avancées.

Contenu
Développement d'interfaces

- Langage support : Java
- Bibliothèque support : Swing (et AWT)
- Modèles d'architectures, architecture MVC
- Patrons de conception Observateur, Composite, Stratégie, Commande et Memento
- Composants, fenêtres, conteneurs, gestionnaires de placement
- Gestion des événements
- Transfert de données : Drap and Drop et (couper, copier) - coller
- Actions, raccourcis clavier
- Annulation / Restauration (Undo/Redo)
- Création de composants

- Interfaces gestuelles
- Conception Centrée Utilisateurs
- Démarche de conception centrée utilisateurs
- Prototypage
- Evaluation

Langages avancés pour les bases de données (LABD) (5ECTS, 45h)

Volume horaire
L'enseignement est organisé sous la forme de deux séances hebdomadaires d'1 heure 30 de cours (18h) et de 2 heures de TD (24h) durant 12 semaines. L'examen final est de 3 heures.

Objectifs
A l'issue de ce module les étudiants doivent avoir acquis les compétences suivantes dans la cadre des données du Web ´ et du Web des données : ils doivent savoir
1. Données du Web : Interroger des données XML existantes – Typer des données et définir un schéma de données XML – Transformer des données XML sous l'angle transformation de document et sous l'angle transformation de données – Gérer une base de données XML.
2. Web des données : Modéliser des relations et des règles sémantiques – Représenter des connaissances – Annoter des données sémantiquement – Faire des requêtes sémantiques ou complexes sur les données du web – Organiser des représentations des connaissances évoluées en construisant ses propres ontologies.

Contenu
Les standards et langages du W3C pour les données du Web :
1. XML : DTD – XPath – XML-Schema – XSLT –Xquery – XQuery Update Facility.
2. Web sémantique ´ : RDF – RDFS – RDFa – SPARQL – OWL.

Modélisation 3D et synthèse d'images (M3DS) (5ECTS, 45h)

Volume horaire
- 1H30 de cours par semaine (=18h)
- 2H de TP par semaine (=24)
- L'examen final est de 3 heures.

Objectifs
L'objectif de l'unité M3DS est d'apporter les fondements de la représentation numérique des mondes virtuels 3D et de leur visualisation. L'unité présente les structures de données et les algorithmes fondamentaux nécessaires au développement des applications 3D.
A l'issue de ce module les étudiants doivent :
- Maîtriser les fondations d'une application 3D : la représentation des objets 3D, leur positionnement dans une scène, et leur visualisation avec une librairie de programmation 3D basée sur les shaders.
- Acquérir les fondements de la visualisation 3D : l'élimination des parties cachées, l'éclairage et les textures.
- Connaitre différentes méthodes pour représenter les objets 3D : les modèles polygonaux, les surfaces polynomiales et implicites, les volumes.
- Savoir mettre en œuvre des outils pour l'interaction 3D : la sélection et la manipulation.
- Acquérir les notions de l'animation basée sur la physique : le mouvement d'objets rigides et la collision.

Contenu
Dans un premier temps, les notions, telles que maillage, positionnement 3D, éclairement, texture et ombres, sont abordées de manière pratique en s'appuyant sur la librairie OpenGL (www.opengl.org),

librairie professionnelle, alternative à Direct3D, et qui permet de programmer des visualisations 3D en temps-réel. Ensuite, les réponses aux questions suivantes sont développées :

- Comment positionner des objets 3D et naviguer dans les mondes 3D ? (changements de repères, graphes de scène)
- Comment représenter un objet 3D ? (structure de données des maillages, courbes, surfaces et volumes)
- Comment visualiser une scène 3D ? (algorithmes de rendu projectif, lancer de rayons, introduction à la radiosité, texturage, éclairement)
- Comment animer des objets 3D et interagir avec eux ? (aperçu des algorithmes d'animation, détection des collisions)

Les aspects théoriques, illustrés en cours, seront mis en pratique lors des séances de TP.

Principes et algorithmes cryptographiques (PAC) (5ECTS, 45h)

Volume horaire
Il est organisé en une séance de 1h30 cours (=18) et une séance de 1h30 TD hebdomadaire (=18h). Quatre séances de 1h30 TP permettent la mise en pratique des notions abordées. L'examen final est de 3 heures.

Objectifs
A l'issue de ce module les étudiants doivent :

- Connaître les différentes missions de la cryptographie contemporaine : confidentialité, authentification, intégrité,
- Connaître la différence entre systèmes de chiffrements symétrique et asymétrique,
- Connaître les problèmes mathématiques sur lesquels repose la sécurité des systèmes cryptographiques,
- Mettre en oeuvre des notions d'arithmétique, d'alea et de complexité,
- Implémenter des techniques indispensables à la sécurité informatique,
- Etre en capacité d'assurer une veille dans certains domaines technologiques/scientifiques,
- Utiliser une API de primitives cryptographiques (OpenSSL),
- Utiliser un logiciel de calcul formel.

Contenu
Systèmes cryptographiques classiques : méthodes de substitution et de transposition. Cryptanalyse de ces systèmes.
Éléments de théorie de l'information, notion d'entropie. Le masque jetable comme seul système de chiffrement inconditionnellement sûr.

- Génération d'alea.
- Systèmes de chiffrement contemporains. Systèmes par blocs et modes opératoires. Systèmes par flot.
- Cryptographie à clef publique.
- Fonctions de hachage. Signature numérique.
- Problèmes d'identification et d'authentification.
- Certification des clés
- Partage de secret.
- Preuve sans transfert de connaissance.

PPD : Programmation parallèle et distribuée (5ECTS, 45h)

Volume horaire
Programmation parallèle et distribuée est une option proposée au second semestre de la première année du master informatique de Lille.
PPD est organisée sous la forme d'une séance hebdomadaire de cours-TD et d'une séance hebdomadaire de TP sur 12 semaines. L'examen final est de 3 heures.

Objectifs
Le but du cours est d'initier l'étudiant à la programmation parallèle distribuée. Il s'agit d'apprendre à implémenter des applications parallèles et/ou distribuées et de découvrir les outils nécessaires pour leur déploiement, exécution et évaluation de leurs performances sur des machines parallèles/distribuées à petite et à grande échelle.
Sont traites en cours-TD :

- Revue des machines parallèles/distribuées (réseaux de stations, grappes de processeurs, processeurs multi-cœurs/GPU, environnements hiérarchiques).
- Etude des paradigmes de programmation parallèle et distribuée (parallélisme de taches, parallélisme de données, mémoire partagée, communication par messages).
- Problèmes fondamentaux de la programmation parallèle distribuée (partitionnement de taches/données, régulation de charge, ordonnancement, tolérance aux pannes, mesure de performance).
- Environnements et outils de la programmation parallèle distribuée : MPI et OpenMP.
- Programmation sur grilles de calcul (concept de grille, la plate-forme Grid5000, modèles de programmation, intergiciels et outils pour les grilles).

En Travaux Pratiques, il s'agit d'apprendre à programmer avec MPI et OpenMP en utilisant le réseau de stations de travail (processeurs multi-coeurs) de la salle de Travaux Pratiques. Les programmes développés sont passés ensuite sur la grille afin d'évaluer leurs performances sur un cluster puis sur une grille de clusters. La grille Grid5000 d'échelle nationale est utilisée comme plate-forme d'expérimentation à grande échelle. L'étudiant apprend à se connecter à une grille/cluster, réserver les ressources (avec OAR), déployer des programmes (avec Kadeploy) et collecter des statistiques sur l'utilisation des ressources avec les outils de Grid5000 (Kaspied, Monica, etc.).

Contenu
A l'issue de ce module les étudiants doivent avoir appris à :

- Implémenter avec MPI et évaluer les performances des algorithmes parallèles et/ou distribués pour des environnements à mémoire distribuée de type réseaux de stations de travail (NOWs), clusters (COWs) et grilles de calcul
- Implémenter avec OpenMP et évaluer les performances des algorithmes parallèles pour des environnements à mémoire partagée de type processeurs multi-coeurs.
- Se servir des outils de réservation de ressources, de déploiement et de supervision d'applications parallèles et/ou distribuées sur clusters de processeurs ainsi que sur environnements à grande échelle de type grille de calcul.

RdF (5ECTS, 45h)

Volume horaire

- 18h de cours-TD (1h30/semaine)
- 24h de travaux pratiques (2h/semaine)

- L'examen final est de 3 heures.

Objectifs
Comprendre les notions et techniques de base de l'acquisition, de la représentation et du traitement des images numériques.

Contenu
La finalité de la Reconnaissance de Formes (traduction de l'anglais "pattern recognition") est d'identifier des motifs à partir de données brutes, afin de prendre une décision dépendant de la catégorie attribuée à ce motif. Cette discipline, qui a des domaines d'application divers comme l'annotation d'images ou la reconnaissance de l'écriture manuscrite, regroupe l'ensemble des techniques et méthodes visant à automatiser ce processus d'identification. L'objectif du module est de maîtriser les notions et techniques de base d'analyse de données multidimensionnelles, de classification et de décision. Nous aborderons notamment les aspects suivants :

— Extraction d'attributs géométriques, topologiques, d'apparence - Représentation et codage des attributs.
— Méthodes statistiques en reconnaissance des formes - Théorie de la décision, classement, classification et discrimination de données multidimensionnelles. ´
— Méthodes syntaxiques en reconnaissance des formes - Structuration en chaines et arbres, langages formels, comparaison d'arbres.

SVL : Spécification et Validation du Logiciel (5ECTS, 45h)

Volume horaire
L'unité comporte une séance de cours-TD hebdomadaire de 1h30 et une séance de TD-machine hebdomadaire de 2h. L'examen final est de 3 heures.

Objectifs
A l'issue de ce module, pour la partie test, les étudiants doivent :

- Etre conscients de la nécessité de tester le code qu'ils écrivent, quel que soit son contexte ;
- Pouvoir discuter des intérêts de l'approche dirigée par les tests (TDD), savoir spécifier un programme par ses tests ;
- Connaître les concepts du test unitaire/d'interaction, être capables d'utiliser une bibliothèque dédiée pour un langage objet ;
- Connaître quelques critères de couverture de code, être conscients de leurs limites, être capables d'utiliser une bibliothèque de calcul de couverture ;
- Savoir appliquer la notion de test aux limites ;
- Savoir adapter une bibliothèque à leurs besoins en utilisant des matchers ;
- Connaître les principes du test d'acceptation, savoir utiliser un outil de test d'acceptation pour application web ;
- Connaître les principes de l'intégration continue ;
- Etre conscients des limites intrinsèques du test.

A l'issue de ce module, pour la partie approches formelles, les étudiants doivent :

- Savoir énumérer les grands types de méthodes formelles existants, être conscients de leurs avantages et inconvénients (notamment par rapport au test) ;
- Avoir intégré la nécessité d'une abstraction des programmes, la plupart des méthodes formelles s'appliquant a un modèle du programme et non à son implémentation, trop bas niveau et détaillée ;
- Connaître les concepts d'un formalisme de spécification formelle étudié dans l'UE (par exemple Promela ou Alloy), avoir compris le lien avec les cas d'étude et langages de programmation qu'ils utilisent au quotidien ;
- Savoir utiliser un outil d'analyse formelle lié à ce formalisme ;
- Avoir compris la logique du premier ordre, la notion de logique décidable/indécidable, de complétude et de sureté d'une analyse.

Contenu
Test du logiciel
Un des buts de l'UE est de familiariser les étudiants avec la nécessité de tester le code qu'ils développent. Ils apprendront à maîtriser le vocabulaire, les concepts et les techniques de base pour le test des programmes objets (test unitaire, d'interaction, couverture de code) ainsi que des aspects plus avancés (contenu à adapter en fonction des acquis présentés par les étudiants en début d'UE). Le travail personnel portera principalement sur la pratique : en plus des TPs d'apprentissage, les étudiants seront fortement incités à réutiliser leurs compétences dans les autres UEs et projets du semestre.
Approches formelles
Le but de cette partie plus théorique est, après avoir amené les étudiants à réaliser les limites du test, de leur faire découvrir un moyen de spécification formel et les capacités d'analyses associées. L'idée est d'illustrer l'application des méthodes formelles à des exemples de développement logiciel ou système, et étant données les 6 semaines imparties, de se restreindre à des analyses automatiques. On choisira par exemple le model-checking avec Spin (lien avec les machines à état d'UML, avec la

programmation système multi-processus) qui a déjà été expérimenté dans la précédente maquette, ou la description de modèle et la preuve de propriétés avec Alloy (lien avec la logique OCL, proximité avec le formalisme objet). Le travail personnel portera principalement sur la compréhension de la théorie qui sous-tend le formalisme choisi (CTD), les TPS venant plus à titre illustratif.

Traitement d'Images - TI (5ECTS, 45h)

Volume horaire
L'unité comporte une séance de cours-TD hebdomadaire de 1h30 et une séance de TD-machine hebdomadaire de 2h. L'examen final est de 3 heures.

Objectifs
A l'issue de ce module, les étudiants doivent savoir :

- Paramétrer le couple caméra- éclairage pour l'acquisition d'images d'une scène,
- Concevoir et mettre en œuvre une chaîne de traitements bas niveau pour l'amélioration des images acquises,
- Interpréter et exploiter le contenu fréquentiel d'une image,
- Choisir et implémenter une méthode de détection des contours des objets présents dans l'image.

Contenu
Cette unité aborde les différents éléments composant une chaine d'acquisition et de traitement bas niveau des images :

- Acquisition d'images numériques : éclairage, caméras, géométrie d'acquisition, échantillonnage, dématricage couleur,
- Représentations spatiale et fréquentielle des images numériques, application à la compression JPEG,
- Opérations de base en traitement des images : transformations ponctuelles, filtrage, détection de contours.

Lors des TPs, les étudiants utilisent le logiciel libre ImageJ et développent des scripts et des modules additionnels en java.

Contacts

Secrétariat pédagogique
m1m2-info@univ-lille1.fr

Directeur des études
Romain Rouvoy romain.rouvoy@univ-lille1.fr

Responsable alternance
Yves Roos yves.roos@univ-lille1.fr