REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

OFFRE DE FORMATION MASTER

ACADÉMIQUE

Etablissement	Faculté / Institut	Département

Domaine : Mathématiques Informatique

Filière : Informatique

Spécialité : Génie Logiciel

Année universitaire : 2025- 2026

II – Fiche d'organisation semestrielle des enseignements

(Prière de présenter les fiches des 4 semestres)

1- Semestre 1 :

Unité d'Enseignement	VHS	V.H hebdomadaire				Coeff	Crédits	Mode d'évaluation	
	14 sem	C	TD	TP	Travail pers.			Continu	Examen
UE Fondamentales						**12**	**18**		
UEF11	126h	3h	3h	3h	6h				
Algorithmique Avancée & Complexité	63h	1H30	1H30	1H30	3H00	3	5	40%	60%
Bases de Données Avancées	63h	1H30	1H30	1H30	3H00	3	5	40%	60%
UEF12	105h	3h	3h	1h30	4h30				
Ingénierie des Exigences	42h	1H30	1H30	-	1H30	3	4	40%	60%
Méthodes de Conception de logiciels & Design Patterns	63h	1H30	1H30	1H30	3H00	3	4	40%	60%
UE Méthodologique						**4**	**9**		
UEM1	84h	3h	-	3h	3h				
Compilation Avancée	42h	1H30	-	1H30	1H30	2	4	40%	60%
Paradigmes de Programmation	42h	1H30	-	1H30	1H30	2	5	40%	60%
UE Transversale						**1**	**2**		
UET1	42h	1h30	-	1h30	1h30				
Réseaux avancés	42h	1H30	-	1H30	1H30	1	2	40%	60%
UE Découverte						**1**	**1**		
UED1	21h	1h30	-	-	-				
Un module au choix	21h	1H30	-	-	-	1	1		100%
Total Semestre 1	378H	12H	6H	9H00	15h	**18**	**30**		

2- Semestre 2 :

Unité d'Enseignement	VHS	V.H hebdomadaire				Coeff	Crédits	Mode d'évaluation	
	14 sem	C	TD	TP	Travail pers.			Continu	Examen
UE Fondamentales						**12**	**18**		
UEF21	126h	3h	3h	3h	6h				
Architectures Logicielles	63h	1H30	1H30	1H30	3H00	3	5	40%	60%
Architectures Orientées Services	63h	1H30	1H30	1H30	3H00	3	5	40%	60%
UEF22	84h	3h	3h	-	3h				
Systèmes d'Information Coopératifs	42h	1H30	1H30	-	1H30	3	4	40%	60%
Sémantiques Formelles des Langages de programmation	42h	1H30	1H30	-	1H30	3	4	40%	60%
UE Méthodologique						**4**	**9**		
UEM2	105h	3h	1h30	3h	4h30				
Systèmes Multi Agents	42h	1H30	-	1H30	1H30	2	4	40%	60%
Machine Learning	63h	1H30	1H30	1H30	3H00	2	5	40%	60%
UE Transversale						**1**	**2**		
UET2	42h	1h30	-	1h30	1h30				
Gestion de Projets Informatiques	42h	1H30	-	1H30	1H30	1	2	40%	60%
UE Découverte						**1**	**1**		
UED2	21h	1h30	-	-	-				
Un module au choix	21h	1H30	-	-	-	1	1		100%
Total Semestre 2	378H	12H	7H30	7H30	15h	**18**	**30**		

3- Semestre 3 :

Unité d'Enseignement	VHS	V.H hebdomadaire				Coeff	Crédits	Mode d'évaluation	
	14 sem	C	TD	TP	Travail pers.			Continu	Examen
UE Fondamentales						**12**	**18**		
UEF31	105h	3h	1h30	3h	4h30				
Assurance Qualité & Tests des Logiciels	63h	1H30	1H30	1H30	3H00	3	5	40%	60%
DevOps & Maintenance des Logiciels	42h	1H30	-	1H30	1H30	3	4	40%	60%
UEF32	105h	3h	1h30	3h	4h30				
Ingénierie Dirigée par les modèles	42h	1H30	-	1H30	1H30	3	5	40%	60%
Spécification et Vérification Formelle	63h	1H30	1H30	1H30	3h00	3	4	40%	60%
UE Méthodologique						**4**	**9**		
UEM3	126h	4h30	3h	1h30	4h30				
Modélisation et évaluation des performances des systèmes	42h	1H30	1H30	-	1H30	2	4	40%	60%
Big Data Analytics	84h	3H00	1H30	1H30	3H00	2	5	40%	60%
UE Transversale						**1**	**2**		
UET3	21h	1h30	-	-	-				
Méthodologie de recherche	21h	1H30	-	-	-	1	2		100%
UE Découverte						**1**	**1**		
UED3	21h	1h30	-	-	-				
Un module au choix	21h	1H30	-	-	-	1	1		100%
Total Semestre 3	378H	13h30	7h30	6H	13h30	**18**	**30**		

4- Semestre 4 :

Domaine : MI
Filière : Informatique
Spécialité : Génie Logiciel

	VHS	Coeff	Crédits
PFE avec Mémoire	**750h00**	**18**	**30**
Stage dans l'entreprise			
Ateliers			
Travail Personnel			
Autres			
Total Semestre 4	**750h00**	**18**	**30**

5- Récapitulatif global de la formation : (indiquer le VH global séparé en cours, TD, pour les 04 semestres d'enseignement, pour les différents types d'UE)

UE / VH	UEF	UEM	UED	UET	S4	Total
Cours	252	147	63	42		504
TD	210	0	0	0		210
TP	189	21	0	21		231
Mémoire	-	-	-	-	750	750
Stage dans l'entreprise	-	-	-	-		
Ateliers	-	-	-	-		
Travail Personnel	399	168	0	42		609
Autres	-	-	-	-		
Total	1050	336	63	105	750	**2304**
Crédits	54	27	3	6	30	**120**
% en crédits pour chaque UE	45	22,5	2,5	5	25	100,00

Corbeille des modules de découverte au choix (UED)

- Informatique verte
- Gouvernance et transformation digitale
- Philosophie des sciences et de la technologie
- Droit du numérique et protection des données (RGPD)
- Psychologie cognitive
- Technologies émergentes (Blockchain, IOT, …)
- Découverte des spécialités de master (une première partie qui permet de découvrir sommairement les autres spécialités du master et une seconde partie qui permet de découvrir en détails les applications potentielles de la spécialité en cours).

…..

Liste des modules communs entre les spécialités

		STANDARS							SPECIFIQUES
#	MODULES / SPECIALITES	IF	GL	ISI	SD	IA	SEI	RSD	.....
1	Algorithmique Avancée & Complexité	X	X	X	X	X	X	X	X
2	Bases de Données Avancées	X	X	X	X	X	X	X	X
3	Ingénierie des Exigences		X	X					
4	Machine Learning (UEF)				X	X			X
5	Méthodes de Conception de log & Design Patterns	X	X	X					X
6	Analyse de données			X	X	X		X	
7	Paradigmes de Programmation	X	X						X
8	Méthodes d'optimisation					X			X
9	Réseaux avancés	X	X	X	X	X			X
10	DevOps						X		X
11	Deep Learning					X			X
12	Big data Analytics				X				X
13	Vision par ordinateur					X			X
14	Systèmes d'information coopératifs		X	X					
15	Sémantiques Formelle des Langages de Prog	X	X						
16	Systèmes Multi-Agents	X	X						
17	Machine learning (UEM)	X	X	X				X	X
18	Virtualisation et Cloud Computing				X	X			
19	Gestion des projets informatiques	X	X	X	X	X	X	X	X
20	Intelligence artificielle générative					X			X
21	Virtualisation, Cloud Computing et sécurité						X	X	
22	Ingénierie Dirigée par les Modèles	X	X						
23	Spécification et Vérification Formelle	X	X						
24	Reconnaissance des formes					X			X
25	Modélisation et éval des performances des syst.	X	X						
26	Natural Language Processing				X	X			
27	Méthodologies de recherche	X	X	X	X	X	X	X	X

III - Programme détaillé par matière

(1 fiche détaillée par matière)

Programme détaillé des enseignements du semestre 1 (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Algorithmique avancée et complexité **Semestre: 1 Type : UEF1**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP : 1**h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 6

Objectifs de l'enseignement :

- Comprendre les notions de complexité algorithmique (temps, espace)
- Maîtriser les techniques avancées de conception d'algorithmes
- Classer les problèmes selon leur difficulté (P, NP, NP-complet, etc.)
- Appliquer des méthodes exactes et approchées à des problèmes complexes

Connaissances préalables recommandées : Algorithmique, structures de données avancées et programmation.

Contenu de la matière
Cours : 21h

1. Complexité algorithmique
   - Notations asymptotiques (Big O, Ω, Θ)
   - Analyse dans le meilleur des cas, le pire cas, le cas moyen
   - Équations de récurrence
2. Structures de données avancées et complexité
   - Arbres binaires (rappel), Arbres AVL, Arbres Rouge-Noir
   - Tables de hachage
3. Algorithmique textuelle et complexité
   - Recherche de motifs
   - Algorithmes de compression
4. Paradigmes de conception
   - Diviser pour régner
   - Programmation dynamique
   - Algorithmes gloutons
   - Algorithmes probabilistes
5. Classes de problèmes
   - Problèmes de décision et d'optimisation
   - Déterminisme vs non-déterminisme
   - NP-complétude et réduction de problèmes
6. Algorithmes avancés
   - Algorithmes parallèles et distribués
   - Algorithmes en ligne
   - Algorithmes randomisés

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** contrôle continus, travail personnel.

Références bibliographiques

-T. Cormen, C. Leiserson, R. Rivest & C. Stein (2022). Introduction to Algorithms. The MIT Press, 5th Edition.

-P. Brass (2008). Advanced Data Structures. Cambridge University Press, City College of New York.

-C. A. Shaffer (2010). A Practical Introduction to Data Structures and Algorithm Analysis. Department of Computer Science, Virginia Tech.

-Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley.

-Kozen, D. C. (1991). The Design and Analysis of Algorithms. Springer.

-Motwani, R., & Raghavan, P. (1995). Randomized Algorithms. Cambridge University Press.

-S. Arora & Barak B. (2006). Computational Complexity: A Modern Approach. Cambridge University Press.

Programme détaillé des enseignements du semestre 1 (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Bases de données avancées **Semestre : 1 Type : UEF1**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP : 1**h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 4

Objectifs de l'enseignement : Cette matière, en l'accent sur les concepts et technologies essentiels pour la conception, la gestion et l'exploitation de systèmes de bases de données modernes, en abordant à la fois les aspects théoriques et pratiques, permet à l'étudiant d'actualiser et d'approfondir ses connaissances des bases de données.

Connaissances préalables recommandées : Concepts de bases sur les bases de données, Langage SQL, Algèbre relationnelle.

Contenu de la matière
Cours : 21h

1-Introduction aux bases de données
- Rappels sur les concepts fondamentaux: modèle relationnel, SGBD, langages de requêtes.
- Types de bases de données: relationnelles, NoSQL, objets, etc.
- Architecture d'un SGBD: client-serveur, architectures distribuées.
- Concepts de transaction, concurrence, et récupération de données.

2-Programmation SQL avancée
- SQL(rappels): jointures, sous-requêtes, fonctions d'agrégation, vues, procédures stockées.
- SQL avancé : Les Triggers, les fonctions, traitement et gestion des erreurs.
- Langages de manipulation de données pour NoSQL: JSONiq, Cypher.

3-Le modèle Objet-Relationnel
- Présentation du modèle Objet
- Présentation du modèle Relationnel-Objet
- Concepts du modèle RO (types complexes, héritage…)
- Interrogation des BDD Relationnelles-Objet (SQL3)

4-Bases de données NoSQL
- Introduction aux bases de données NoSQL: types de bases de données (clés-valeurs, documents, graphes, colonnes).
- Modèles de données NoSQL: avantages et inconvénients.
- Études de cas: MongoDB, Cassandra, Neo4j.

5-Bases de données distribuées
- Architecture des systèmes distribués.
- Techniques de réplication et de partitionnement.
- Consistance des données dans un environnement distribué: ACID vs BASE.

6-Bases de données dans le cloud
- Modèles de service: IaaS, PaaS, SaaS.
- Cloud computing et bases de données.
- Considérations de sécurité dans le cloud.

7-Performance et sécurité des bases de données
- Optimisation des requêtes SQL.
- Indexation et stratégies d'accès.
- Sécurité: contrôle d'accès, chiffrement des données, audit.
- Gestion de la sécurité dans les bases de données distribuées.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** contrôles continus, travail personnel.

Références bibliographiques

-R. Elmasri & S. Navathe (2016). Fundamentals of Database Systems, 7th Edition, Pearson.

- R. Elmesri, B. Navate (2016). Fundamentals of Database Systems. 7th edition, Pearson Editions

-T. Connolly & C.Begg (2014). Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson.

-Christian Soutou (2013). SQL pour Oracle. Editions Eyrolles.

-Silberschatz, H. Korth & S. Sudarshan (2019). Database System Concepts, 7th Edition, McGraw-Hill.

-Mohamed Fadhel SAAD (2016). PL/SQL sous Oracle 12c. Guide du développeur.

-R.G.G. Cattell (1994). Object Data Management. Addison-Wesley.

-P. Selmer (2012). NOSQL stores and Data analytics tools. Advances in Data Management.

Programme détaillé des enseignements du semestre 1 (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Ingénierie des exigences **Semestre : 1 Type : UEF2**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** 1h30 **TP :**
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Le programme de cette matière permet à l'étudiant :

- De comprendre les concepts fondamentaux de l'ingénierie des exigences.
- De découvrir les différentes étapes du processus d'ingénierie des exigences : identification, analyse, spécification, validation et gestion.
- D'apprendre à utiliser les techniques et outils appropriés pour chaque étape.
- De développer des compétences en communication et en collaboration pour travailler efficacement avec les parties prenantes.
- D'appliquer les principes de l'IE dans des projets de développement logiciel.

Connaissances préalables recommandées : Génie logiciel, UML.

Contenu de la matière
Cours : 21h

1. Introduction à l'ingénierie des exigences (IE)
   - Définition de l'IE.
   - Rôle de l'IE dans le cycle de vie du logiciel.
   - Concepts clés : besoins, exigences fonctionnelles et non fonctionnelles, contraintes, etc.
2. Processus d'ingénierie des exigences
   - Identification des exigences : techniques de collecte des besoins
   - Analyse des exigences : modélisation des besoins, gestion des conflits, priorisation.
   - Spécification des exigences : rédaction de spécifications claires et non ambigües (UML, use cases, etc.).
   - Validation des exigences : revue, prototypage, tests.
   - Gestion des exigences : contrôle de version, traçabilité, analyse d'impact.
3- Techniques et outils
   - Techniques de collecte des besoins : entretiens, ateliers, observations, analyse documentaire.
   - Modélisation des besoins : diagrammes de cas d'utilisation (UML), diagrammes de classes (UML), modèles de données.
   - Spécification formelle : langages de spécification (Alloy, Z).
   - Outils de gestion des exigences : DOORS, RequisitePro, etc.
4. Qualité des exigences
   - Critères de qualité : complétude, cohérence, faisabilité, etc.
   - Techniques d'évaluation de la qualité.
   - Gestion de la qualité des exigences.
5- Ingénierie des exigences dans des contextes spécifiques
   - Agile development.
   - Systèmes embarqués.
   - Applications web.
   - Ingénierie des systèmes complexes.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôle continus, travail personnel.

Références bibliographiques

-Karl Wiegers, Joy Beatty (2013). Software Requirements. Microsoft Press, 3e éd.

- Karl Wiegers and Joy Beatty (2013). Software Requirements. Microsoft, Karl Wiegers and Seilevel.

-Elizabeth Hull, Ken Jackson, Jeremy Dick (2017). Requirements Engineering. Springer, 4e éd.

- Klaus Pohl and P.A. Börstler (2010). Requirements Engineering: Processes and Techniques.

- Stéphane Badreau, Jean-Louis Boulanger (2014). Ingénierie des exigences : Méthodes et bonnes pratiques pour construire et maintenir un référentiel. Dunod.

-Martin Fowler (2000). UML Distilled.

-Sommerville I (2013). Software Engineering. Pearson Education.

-Roel Wieringa Anne Persson (2010). Requirements Engineering: Foundation for Software Quality. Spinger.

Programme détaillé des enseignements du semestre (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière: Méthodes de conception de logiciels & design patterns
Semestre: 1 **Type :** UEF2
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 3 **Crédit :** 3

Objectifs de l'enseignement : Ce cours vise à approfondir les connaissances et compétences des étudiants dans les principes, méthodes et outils de conception logicielle de haut niveau et pour des environnements divers. A l'issue de ce module, l'étudiant doit :

- Maîtriser les concepts fondamentaux de la conception logicielle.
- Comprendre et appliquer les différentes méthodes de conception avancées.
- Savoir choisir et utiliser les outils de modélisation et de conception appropriés au type d'application à développer.

- Comprendre les enjeux de qualité logicielle et des aspects liés à la sécurité et à la maintenabilité.

Connaissances préalables recommandées : GL et Programmation orientée objet.

Contenu de la matière
Cours : 21h

1. Rappels sur les méthodes de conception
   - Modèles de cycle de vie du logiciel (cycle en V, agile, transformations formelles, etc.).
   - Notions de qualité logicielle (fiabilité, maintenabilité, performance).
2. Conception Orientée Objet Avancée
   - Rappels des concepts de l'OO (héritage, polymorphisme, encapsulation, etc.).
   - Patrons de conception (design patterns) pour la réutilisation et la modularité.
   - Outils de modélisation UML (Unified Modeling Language), UP (Processus unifié) RUP, 2TP.
   - Outils CASE (AGL) dans le développement (exemples)
3. Modélisation et Conception Basée sur des Modèles (MDA)
   - Model-Driven Architecture (MDA).
   - Modèles de plateformes (PIM, PSM).
   - Transformation de modèles.
   - Outils de génération de code à partir de modèles.
4. Qualité et Sécurité
   - Analyse statique et dynamique du code.
   - Tests unitaires et d'intégration.
   - Techniques de sécurisation des logiciels (prévention des failles, chiffrement, etc.).
   - Maintenabilité (modularité, couplage, cohésion).
5. Méthodes Agiles Avancées
   - Scrum avancé (sprints, rôles, planification, etc.).
   - Kanban (visualisation, flux, etc.).
   - Adaptation des méthodes agiles à différents contextes.
   - Intégration de l'agilité dans les cycles de développement classiques.
6. Conception de logiciels spécifiques : généralités
   - Les systèmes embarqués.
   - Le cloud.

- Les systèmes distribués.
- Les bases de données.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôle continus, travail personnel.

Références bibliographiques

-Booch, G., Rumbaugh, J., Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison-Wesley.

- Rolland, Collette (1996). Conception des BBD : méthodes orientées objet et évènements. Techniques de l'ingénieur, réf : H3248 v1

-Fowler, M. (2003). Patterns of Enterprise Application Architecture. Addison-Wesley.

-Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

-OMG (2017). OMG Unified Modeling Language (OMG UML), Version 2.5.1. Object Management Group. Object Management Group.

-Kruchten P. (2001). Introduction au Rational Unified Process (RUP), Editions Eyrolles, Paris.

-L Thiry, B Thirion, M Hassenforder (2008). Dsls pour le développement agile de transformations. IDM'08.

-A Rasse, JL Boulanger, G Mariano, L Thiry, JM Perronne (2008). Approche orientée modèles pour une conception UML certifiée des systèmes logiciels critiques. Actes de la Conférence Internationale Francophone d'Automatique (CIFA 08).

Programme détaillé des enseignements du semestre 1 (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Compilation avancée **Semestre : 1 Type : UEM1**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Cette matière permet à l'étudiant :

- De comprendre les principes fondamentaux de la conception et de l'implémentation des compilateurs.
- De connaitre les techniques d'analyse lexicale, syntaxique et sémantique.
- De pouvoir concevoir et mettre en œuvre des étapes de génération de code intermédiaire et final.
- D'acquérir des compétences dans l'optimisation du code et la gestion des erreurs.
- De découvrir les aspects avancés de la compilation, tels que la compilation incrémentale, la compilation parallèle, et les langages spécifiques à la compilation.

Connaissances préalables recommandées : Théorie des langages, Architecture des ordinateurs, Algorithmique, Système d'exploitation, Logique Mathématique.

Contenu de la matière
Cours : 21h

1- Analyses lexicales et syntaxiques : rappels

- Analyseur lexical et les expressions régulières (Lex)
- Analyseur syntaxique et les grammaires à contexte libre (type 2)
- Notions de langages formels et d'automates.
- Introduction aux outils de construction de compilateurs (lex, yacc, etc.).

2. Analyse lexicale

- Expressions régulières et automates finis déterministes (AFD).
- Construction d'analyseurs lexicaux avec des outils tels que `lex`.
- Reconnaissance des jetons et génération de leur représentation.
- Algorithme d'analyse
- Création et gestion de la table des symboles
- Gestion des erreurs lexicales

3. Analyse syntaxique

- Les concepts de base: formes de grammaires, factorisation.
- Grammaires hors-contexte (GHC) et leur utilisation pour la spécification du langage.
- Méthodes d'analyse descendantes: non déterministes, déterministes (LL, récursive)
- Méthodes d'analyse ascendantes déterministes: analyse LR par les contextes et les items, analyse simple LR par les contextes et les items, analyse LALR par les contextes et les items
- Mise à jour de la table des symboles.
- Gestion des conflits dans le cas des tables d'analyse multi-définies
- Relations entre les méthodes de l'analyse syntaxique.
- Gestion des ambiguïtés et des erreurs syntaxiques.
- Mode panique et récupération.
- Construction d'analyseurs syntaxiques avec des outils tels que yacc.

4. Analyse sémantique et traduction dirigée par la syntaxe

  - Arbres syntaxiques abstraits (AST) et leur utilisation pour la représentation du code source.

- Langages intermédiaires
- Table des symboles et gestion des contextes d'exécution.
- Schémas de traduction: analyse ascendante, analyse descendante.
- Contrôle de type et détection des erreurs sémantiques.
- Génération d'instructions intermédiaires (code 3 adresses, bytecode, etc.).
- Analyse sémantique.

5. Génération de code
   - Stratégies de génération de code (direct, indirect).
   - Optimisation du code au niveau intermédiaire.
   - Gestion des registres et allocation mémoire.
   - Génération de code machine pour différentes architectures.
6. Optimisation du code
   - Optimisations locales et globales.
   - Analyse de flot de données.
   - Transformations de code (élimination des code morts, réduction de la complexité).
   - Analyse et transformation des boucles.
7. Compilation avancée
   - Compilation incrémentale.
   - Compilation parallèle et distribuée.
   - Langages spécifiques à la compilation (DSL).
   - Techniques de débuggage des compilateurs.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel..

Références bibliographiques

-Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman (2007). Compilers: Principles, Techniques and Tools. Pearson Addison Wesley.

-Anthony J. Dos Reis (2011). Compiler Construction Using Java, JavaCC, and Yacc. Wiley-IEEE Computer Society Press.

-Andrew W Appel (1998). Modern Compiler Implementation in ML. Published by Cambridge university Press.

-Ronald Mak (2009). Writing Compilers and Interpreters: A Software Engineering Approach. John Wiley and Sons.

-Sebastian Hack, Reinhard Wilhelm and Helmut Seidl (2013). Compiler Design: Code Generation and Machine-Level Optimization. Springer.

-Keith Cooper & Linda Torczon (2011). Engineering a Compiler. 2nd edition, Elsevier Morgan-Kauffman.

-Dick Grune, Kees van Reeuwijk, Henri E. Bal and Ceriel J.H. Jacobs (2012). Modern Compiler Design. Wiley. .

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Paradigmes de programmation **Semestre: 3 Type : UEM1**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 5

Objectifs de l'enseignement : cette matière explore les différents styles de programmation et leurs applications afin de permettre à l'étudiant de:

- Comprendre les fondements des principaux paradigmes de programmation.
- Développer une capacité à choisir le paradigme approprié pour résoudre un problème donné.
- Découvrir les concepts clés de chaque paradigme.
- Développer des compétences en programmation dans différents langages et styles.

Connaissances préalables recommandées : Algorithmique, Compilation, Théorie des langages, logique.

Contenu de la matière
Cours : 21h

Chapitre 1 : Introduction aux Paradigmes des Langages de Programmation

- Définitions et concepts fondamentaux.
- Classification des langages de programmation selon les paradigmes : impératifs, déclaratif, etc.
- Importance des paradigmes dans le développement logiciel.

Chapitre 2 : Paradigme impératif

- Principe du paradigme impératif
- Concepts clés de la programmation impérative
- Programmation procédurale:
  - Concepts clés : procédures, fonctions, variables, contrôle de flux.
  - Langages : C, Pascal, BASIC.
  - Exemple de mise en œuvre de la programmation procédurale.
  - Avantages et inconvénients.
- Programmation orientée objet (POO):
  - Concepts clés : objets, classes, encapsulation, héritage, polymorphisme, abstraction.
  - Langages : Java, C++, Python.
  - Exemple de mise en œuvre de la programmation objet
  - Avantages et inconvénients, applications.
- Programmation structurée

Chapitre 3 : Paradigme déclaratif

- Programmation fonctionnelle
  - Concepts clés : fonctions comme citoyens de première classe, fonctions pures, récursivité, évaluation paresseuse, types polymorphes.
  - Principe du paradigme fonctionnel
  - Filtrage par motif.
  - Listes et récursivité sur les listes : algorithmes sur les listes.
  - Concepts avancés de la programmation fonctionnelle : types unions, fonctions sur les types unions, structures de données, structures immuables et persistantes, évaluation.
  - Etude de cas avec un langage déclaratif (Lisp, Haskell, Scala): développement d'un interpréteur ou d'une machine à pile.
  - Avantages et inconvénients, applications.

- Programmation logique
    - Concepts clés : faits, règles, clauses, inférence.
    - Langages : Prolog.
    - Arithmétique de base, prédicats intégrés.
    - Représentation des structures : listes, relations.
    - Unification, Selected Lineary Defined Resolution(SLD), contrôle d'exécution.
    - Etude de cas avec le langage Prolog: génération de planning ou un noyau d'un système expert.
    - Avantages et inconvénients, applications.
- Programmation descriptive (HTML, LaTeX).

Chapitre 4 : Paradigmes de programmation avancés

- Programmation parallèle et concurrente.
    - Concepts clé de la programmation concurrente.
    - Exemple de mise en œuvre de la programmation concurrente.
- Programmation orientée prototype.
- Programmation réactive.
- Calcul quantique.

Chapitre 5 : Comparaison et choix des paradigmes

- Critères de sélection d'un paradigme.
- Avantages, inconvénients, contextes d'usage.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Bansal A. K. (2014). Introduction to programming languages. CRC Press.

-Gabbrielli M., Martini S. (2010). Programming languages: Principles and paradigms. Springer.

-Van Roy, P., & Haridi, S. (2004). Concepts, techniques, and models of computer programming. MIT press.

-David A Watt (2004). Programming language design concepts. John Wiley & sons.

-Daniel P. Friedman, Mitchell Wand, Christopher T. Haynes (2001). Essentials of Programming Languages. MIT Press.

-Clocksin, W. F., & Mellish, C. S. (2003). Programming in PROLOG. Springer Science & Business Media.

-Louden K. C., Lambert K. A. (2012). Programming languages: Principles and practice. Course Technology, Cengage Learning.

-Scott M. L. (2009). Programming Language Pragmatics. Elsevier.

-Sebesta R. W. (2006). Concepts of Programming Languages. Addison Wesley.

-Minsky, Y., Madhavapeddy, A., & Hickey, J. (2013). Real World OCaml: Functional programming for the masses. " O'Reilly Media, Inc.".

Programme détaillé des enseignements du semestre (S1)
Master académique
Spécialité : Génie logiciel **(Filière : Informatique)**

Intitulé de la matière: Réseaux avancés **Semestre :** 1 **Type :** UET1
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP:** 1h30
VHS travail personnel : 21h **Coefficient :** 1 **Crédit :** 2

Objectifs de l'enseignement : Cette matière vise à consolider et approfondir les connaissances des étudiants sur le fonctionnement des réseaux informatiques, en abordant des notions avancées telles que le routage dynamique, les protocoles de transmission, les protocoles applicatifs ainsi que la configuration IPv6. L'objectif est de préparer les étudiants à analyser, concevoir et gérer des infrastructures réseau plus complexes dans des contextes professionnels ou de recherche.

Connaissances préalables recommandées : Réseaux informatiques (licence)

Contenu de la matière
Cours : 21h

1- Rappels sur les réseaux
- Modèles OSI et TCP/IP
- Adressage MAC/IP (Protocole ARP)
- Adressage dans les sous-réseaux (FLSM, VLSM, CIDR ..)

2- Protocoles de routage
- Routage statique
- Routage dynamique (RIP, OSPF, BGP)
  - Vecteurs de distance
  - Etat des liens

3- Réseaux locaux et réseaux étendues
- Réseaux locaux Virtuels (Vlan), Protocoles VTP, DTP
- Routage inter-Vlan
- Redondances dans les LANs (STP, EtherChannel, Protocole HSRP)
- Protocole PPP
- VPN (Protocole GRE)

4- Protocoles de transmission
- Les protocoles TCP/UDP
- Gestion des états de la connexion TCP
- Contrôle de flux
- Contrôle de congestion
- Contrôle d'erreur

5- Protocoles applicatifs
- Protocoles web (HTTP et HTTPS)
- Protocoles de messagerie électronique (SMTP, POP et IMAP)
- Services de partage de fichiers (FTP et SMB)
- DHCP
- DNS
- Telnet et SSH

6- Protocole IPv6
- Adressage
- Transition IPv4/IPv6
- Services IPv6…

7- Gestion des réseaux

- LLDP, CDP, NTP
- SNMP
- Qualité de Service (QoS)
- Automatisation de gestion de la configuration (Ansible…)

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôle continu, travail personnel.

Références bibliographiques

-. Tanenbaum, A., Feamster, N., & Witherall, D. (2021). Computer networks (6th ed.). Pearson Publisher.

-. White, R., & Banks, E. (2017). Computer networking problems and solutions: An innovative approach to building resilient, modern networks. Addison-Wesley Professional.

-. Englander, I., & Wong, W. (2021). The architecture of computer hardware, systems software, and networking: An information technology approach. John Wiley & Sons.

-. Bonaventure, O. (2021). Computer networking: Principles, protocols and practice.

-. Dordal, P. L. (2022). An introduction to computer networks. Department of Computer Science, Loyola University Chicago.

Programme détaillé des enseignements du semestre 1 (S1)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Module au choix **Semestre : 1** **Type : UED1**
VHS : 21h **VHH :** 1h30 **Cours :** 1h30 **TD :** **TP :**
VHS travail personnel : **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Cette matière permet aux étudiants d'acquérir une vision plus large de leur domaine d'études et favorise une meilleure compréhension des enjeux sociaux, économiques et éthiques liés à l'informatique, ainsi qu'une capacité à communiquer efficacement et à s'adapter à des situations diverses qui peuvent se présenter. En d'autres termes, cette matière permet :

- Une ouverture à d'autres disciplines en relation avec le quotidien, afin de comprendre l'importance de la collaboration dans un environnement pluridisciplinaire.
- L'élargissement des horizons, ce qui favorise la compréhension de la pensée humaine dans le temps et l'espace.
- Développement de compétences transversales : communication, vision holistique, analyse et synthèse, capacité d'adaptation, etc.

Connaissances préalables recommandées

Contenu de la matière
Cours : 21h

1- Introduction à la discipline concernée (matière choisie).
2- Présentation des concepts fondamentaux.
3- Applications croisées avec l'informatique
4- Études de cas.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (100%)**.

Références bibliographiques : L'enseignant propose des références selon la matière choisie.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière: Architectures logicielles **Semestre: 2 Type : UEF21**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 5

Objectifs de l'enseignement : cette matière permet à l'étudiant de :

- Comprendre les principes fondamentaux de l'architecture logicielle.
- Découvrir les différents styles d'architectures logicielles.
- Concevoir et évaluer des architectures logicielles robustes et évolutives.
- Connaitre les bonnes pratiques et les patterns d'architecture.
- Se familiariser avec les outils et technologies utilisés dans l'architecture logicielle.
- Développer une capacité d'analyse critique des architectures existantes et proposer des améliorations.

Connaissances préalables recommandées: Systèmes d'information, génie logiciel.

Contenu de la matière
Cours : 21h

1. Introduction à l'architecture logicielle

- Définition et concepts de base des composants logiciels.
- Programmation modulaire Vs. Programmation par composants.
- Définition et importance de l'architecture logicielle.
- Concepts clés : abstractions, modules, interfaces.
- Exigences fonctionnelles et non fonctionnelles (performance, sécurité, évolutivité).
- Processus de développement centré architecture.
- Relations entre architecture logicielle et architecture matérielle.

2. Styles d'architecture logicielle

- Architecture en couches.
- Architecture orientée services (SOA).
- Architecture micro-services.
- Architecture orientée événements.
- Architecture pilotée par modèle.
- Architecture distribuée.

3. Principes de conception d'architectures logicielles

- Séparation des préoccupations.
- Qualités d'une bonne architecture: Couplage faible, Cohésion forte, Abstraction, et encapsulation.
- Réutilisation et modularité.
- Processus de développement architectural orienté style.

4. Patterns d'architecture

- Patterns de conception (design patterns).
- Patterns d'architecture (architecture patterns).
- Utilisation de patterns pour résoudre des problèmes courants.

5. Langages de description et outils de modélisation d'architectures logicielles

- Frameworks et librairies d'architecture (JEE, Spring, etc.).
- Outils d'analyse et de visualisation d'architectures: Visual Studio, Visual Paradigm, Enterprise Architect, AADL(Architecture Analysis and Design Language).
- Languages de description: Acme, Wright

- Outils de modélisation et de documentation d'architectures: DocGen BOUML.
- Intégration continue et livraison continue (CI/CD).

6. Analyse et évaluation d'architectures

- Méthodologies d'évaluation d'architectures.
- Analyse des forces et faiblesses d'une architecture.
- Identification des risques et des points de vulnérabilité.

7. Cas d'études et projets

- Analyse d'architectures existantes (applications web, systèmes embarqués, etc.).
- Conception d'architectures pour des projets spécifiques.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Richards, M. (2020). Software Architecture Patterns. O'Reilly Media.

-Len Bass, Paul Clements, Rick Kazman (2003). Software Architecture in Practice. Addison-Wesley.

-P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford (2010). Documenting Software Architectures - Views and Beyond », 2nd edition, Addison Wesley.

-Jacques Printz (2009). Architecture logicielle: Concevoir des applications simples, sûres et adaptables. Editions Dunod.

-Chris Richardson (2018). Microservices Patterns: With examples in Java. Manning Publisher.

-M. Lankhorst (2009). Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer.

-Robert C. Martin (1988). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.

-Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley.

-Paul Clements, Rick Kazman, John Klein (2003). Documenting Software Architectures: Views and Beyond. Addison-Wesley.

-B. Bruller (2003). Architectures de Systèmes d'Information Modèles, services et protocoles. Editions Vuibert.

-Martin Fowler (2003). Patterns of Enterprise Application Architecture. Addison-Wesley.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Architectures orientées services **Semestre: 2 Type : UEF21**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 5

Objectifs de l'enseignement : cette matière permet à l'étudiant:

- De comprendre les concepts fondamentaux des architectures orientées services (SOA) et des microservices.
- D'analyser les avantages et les inconvénients de chaque approche.
- De se familiariser avec les technologies et les outils liés à la mise en œuvre de SOA et de microservices.
- De développer des compétences en conception, en implémentation et en déploiement d'applications basées sur ces architectures.
- D'évaluer les défis et les bonnes pratiques pour la migration vers une architecture de microservices.

Connaissances préalables recommandées : développement d'applications web, bases de données,

Contenu de la matière
Cours : 21h

1. Introduction aux architectures orientées services (SOA)
   - Services web : rappels (concepts et technologies impliquées)
   - Définition et principes fondamentaux de SOA.
   - Avantages et inconvénients de SOA par rapport aux architectures monolithiques.
   - Éléments constitutifs d'une SOA : services, bus de services, registre de services, etc.
   - Modèles de communication : API, messaging, etc.
   - Orchestration et chorégraphie de services.
   - Cas d'utilisation de SOA dans différents contextes.
2. Introduction aux microservices
   - Définition et principes fondamentaux des microservices.
   - SOA Vs. microservices.
   - Avantages et inconvénients des microservices.
   - Principes de conception des microservices : autonomie, découplage, scalabilité, résilience, etc.
   - Composants essentiels d'une architecture de microservices : discovery, load balancing, API Gateway, etc.
3. Technologies et outils pour les microservices
   - Frameworks et langages pour le développement de microservices : Java, Python, Go, Node.js, etc.
   - Conteneurisation avec Docker et Kubernetes.
   - Orchestration et gestion de conteneurs.
   - Outils de surveillance et de journalisation.
   - Gestion de configuration et de secrets.
   - Communication inter-services : REST, messaging, gRPC, etc.
   - Bases de données pour microservices : bases de données relationnelles, NoSQL, etc.
4. Conception d'applications basées sur les microservices
   - Identification des domaines de services.
   - Découpage de l'application en microservices.

- Conception d'API robustes et performantes.
- Stratégies de gestion des données : données partagées, cohérence.
- Développement d'une architecture sécurisée.

5. Déploiement et gestion des microservices
   - Déploiement continu et livraison continue (CI/CD).
   - Infrastructure as Code (IaC).
   - Surveillance et gestion des performances.
   - Gestion des erreurs et de la résilience.
   - Tests unitaires, tests d'intégration et tests de bout en bout.
6. Migration vers une architecture de microservices
   - Stratégies de migration : big bang, strangler fig, etc.
   - Identification des services à migrer.
   - Gestion des dépendances et des interactions entre les anciens et les nouveaux services.
   - Impact sur l'organisation et les processus de développement.
   - Etude de cas pratiques.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Papazoglou, M. P., & Georgiou, P. (2007). Service-Oriented Computing: Concepts, Technology, and Middleware. MIT Press.

-Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

-Fowler, M. (2014). Microservices. https://martinfowler.com/articles/microservices.html

-Lewis, J., & Fowler, M. (2020). Microservices: From Design to Deployment. https://martinfowler.com/articles/microservices.html

-Dragoni, N., Ghezzi, C., & Marinoni, M. (2017). Microservices: A survey of approaches, techniques, and tools. Journal of Systems and Software, 130, 209-226.

-Leonard Richardson (2007). RESTful Web Services. O'Reilly Media.

-Thomas Erl (2007). SOA: Principles of Service Design. Prentice Hall.

-JJ Geewax (2021). API Design Patterns. Manning.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Systèmes d'information coopératifs **Semestre : 2 Type : UEF1**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** 1h30 **TP :**
VHS travail personnel : 21h **Coefficient :** 3 **Crédit :** 5

Objectifs de l'enseignement : Cette matière permet aux étudiants:

- De comprendre les concepts fondamentaux des systèmes d'information coopératifs (SIC).
- De maîtriser les architectures et les technologies clés des SIC.
- De développer des compétences en conception, développement et gestion de SIC.
- D'analyser les enjeux de collaboration et de communication dans les SIC.
- D'évaluer l'impact des SIC sur les organisations et les individus.

Connaissances préalables recommandées : Gestion de projet, base de données, systèmes d'information.

Contenu de la matière
Cours : 21h

1. Systèmes d'information coopératifs
   - Fondement des systèmes d'information répartis : aspects, caractéristiques,
   - Définition
   - Principes de la collaboration et de la coordination.
   - Enjeux et défis des SIC dans divers contextes.
2. Architecture des systèmes d'information coopératifs
   - Le travail coopératif.
   - Les formes et les modalités de coopération.
   - Modèles d'architecture pour les SIC (orientés groupe, architectures de référence, basés sur les standards).
   - Les outils du travail coopératif (groupware synchrones, groupware asynchrones, GED, workflow).
3. Ingénierie des systèmes d'information coopératifs
   - Méthodologies de développement de SIC (agile, OSSAD, etc.).
   - Analyse des besoins et modélisation des processus métier.
   - Conception d'interfaces utilisateur collaboratives.
   - Gestion de projet et gestion du changement dans les SIC.
   - Sécurité des systèmes d'information coopératifs.
4. Collaboration et communication dans les SIC
   - Analyse sociologique du travail coopératif/collaboratif
   - Outils de communication synchrone et asynchrone (chat, visioconférence, etc.).
   - Gestion des connaissances et des informations partagées.
   - Outils d'aide à la prise de décision (GDSS, …).
   - Aspects sociaux des SIC.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** contrôles continus, travail personnel.

Références bibliographiques

-Alquier & al, (2000). Les systèmes d'information coopératifs : une approche par les collecticiels.

-Baghdadi Y. (1997). Contribution méthodologique à la conception des systèmes d'information coopératifs. Thèse de doctorat. Toulouse 1.

- Nurcan, Selmin (1996). Analyse et conception de systèmes d'information coopératifs. TSI'96.

- Vincent Couturier (2004). L'ingénierie des systèmes d'information coopératifs par réutilisation : une approche à base de patterns. Thèse de doctorat, Lyon 3.

-Schmidt, K. (2003). Towards a framework for collaborative systems design. HCI International.

- Archimag (2023). Gestion Électronique des Documents (GED) et Collaboration : il est temps de passer de la gestion à la collaboration autour des documents.

- Van Der Aalst, Kees Van Hee (2002). Workflow Management : Models, Methods and Systems. MIT Press.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Systèmes multi agents **Semestre : 2 Type : UEF2**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : La matière "Systèmes Multi-Agents" (SMA) vise à fournir une compréhension approfondie des concepts fondamentaux, des outils et des méthodologies nécessaires à la conception, au développement et à l'évaluation de systèmes complexes basés sur l'interaction d'agents autonomes.

Connaissances préalables recommandées : POO, Bases de l'IA, Algorithmique et structures de données, Notions de systèmes distribués et réseaux.

Contenu de la matière
Cours : 21h

1. Introduction aux Systèmes Multi-Agents
- Définition d'un agent et d'un système multi-agents.
- Différences entre agents, objets et processus.
- Structure des agents : rationalité, structure conceptuelle.
- Architectures d'agents : BDI, réactifs, délibératifs, hybrides.

2. Modèles de Communication et d'Interaction
- Communication inter-agents basée sur les langages **KQML** et **FIPA-ACL**.
- Protocoles d'interaction (contract net, enchères).
- Ontologies et langages pour agents.
- Programmation orientée agents.

3. Coordination, Coopération et Négociation
- Mécanismes de coordination distribuée.
- Coopération entre agents pour atteindre des objectifs globaux.
- Stratégies de négociation et résolution de conflits.

4. Organisation et Structuration
- Organisations des SMA : hiérarchiques, holoniques.
- Modèles organisationnels : MOISE, AGR.

5. Planification et Raisonnement Multi-Agents
- Planification distribuée.
- Raisonnement basé sur croyances, désirs et intentions (**BDI**).

6. SMA et Simulation
- Modélisation et simulation d'environnements complexes.
- Méthodologies de développement : AAII, GAIA, Agent UML, Prometheus, MadKit.
- Plateformes et simulation: JAD, Netlogo, Repast, GAMA
- Exemples : systèmes de transport, réseaux sociaux, smart grids.

7. Applications des SMA
- Génie logiciel.
- Robotique coopérative.
- Commerce électronique et systèmes distribués.
- Simulation sociale, IoT.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.

- **Évaluation continue (CC) (40%) :** contrôles continus, miniprojet.

Références bibliographiques

-Wooldridge, M. J. (2009). An introduction to multi agent systems (2nd ed.). John Wiley& Sons.

-Ferber, J. (1999). Multi-agent systems: An introduction to distributed artificial intelligence. Harlow, UK: Addison-Wesley.

-Weiss, G. (Ed.). (2013). Multiagentsystems: A modern approach to distributed artificial intelligence (2nd ed.). The MIT Press.

-Shoham, Y., &Leyton-Brown, K. (2009). Multiagents ystems:Algorithmic, game-theoretic, and logical foundations. Cambridge, UK: Cambridge UniversityPress.

-Ferber, J. et Gutknecht, O. (2000). MadKit: a generic multi-agent platform, article présenté à Proceedings of the 4th International Conference on Autonomous Agents, Barcelone, Espagne.

-Nicholas R. Jennings, Michael J. Wooldridge (1998). Agent technology. Springer.

-Eric Bonabeau et Guy Theraulaz (1994). Intelligence Collective. Hermès.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Ingénierie des systèmes d'information (Filière : Informatique)
Intitulé de la matière : Sémantique Formelle des langages de programmation
Semestre : 2 Type : UEM2
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** 1h30 **TP :**
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Cette matière explore la signification et le comportement des programmes informatiques. Elle couvre les différentes approches pour décrire la sémantique des langages, telles que la sémantique dénotationnelle, opérationnelle et axiomatique, ainsi que des sujets comme la vérification de programmes et la modélisation de langages.

Connaissances préalables recommandées : Logique

Contenu de la matière
Cours : 21h

1. Introduction à la sémantique des langages de programmation
   - Définition et importance de la sémantique.
   - Syntaxe vs. sémantique.
   - Types de sémantiques : dénotationnelle, opérationnelle, axiomatique.
2. Sémantique dénotationnelle
   - Introduction aux domaines et aux fonctions continues.
   - Modélisation des programmes par des fonctions mathématiques.
   - Applications : dénotation des valeurs, dénotation des constructions de programmes, théorie du point fixe, preuves de correction.
3. Sémantique opérationnelle
   - Modèles abstraits de machines et de calcul.
   - Différentes approches : machines abstraites, sémantique par réduction de termes, sémantique naturelle.
   - Analyse du comportement d'exécution des programmes.
   - Applications : compilation, interprétation, simulation.
4. Sémantique axiomatique:
   - Logique de Hoare et spécification de programmes.
   - Règles d'inférence pour les structures de contrôle.
   - Preuves de correction de programmes.
   - Notion de ''weakest precondition'' de Dijikstra, propriétés des programmes.
   - Applications : vérification formelle, déduction de propriétés.
5. Vérification de programmes: Introduction à la vérification formelle, Outils et techniques pour la vérification, Exemples de vérification de programmes spécifiques.
6. Modélisation de langages
   - Élaboration de modèles formels pour les langages.
   - Étude de la sémantique de langages spécifiques (ex : langage fonctionnel, impératif).
   - Conception de nouveaux langages de programmation.
7. Introduction à la compilation
   - Processus de compilation.
   - Analyse lexicale, syntaxique et sémantique.
   - Génération de code.
   - Optimisation.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Pettorossi, A. (2010). Semantics of programming languages. Aracne Editrice.

-Winskel, G. (1993). The formal semantics of programming languages: an introduction. MIT press.

-Winskel, G. (2005). Denotational Semantics. CS Lecture Notes, Cambridge University.

-Schmidt, D. A. (1994). The structure of typed programming languages. MIT press.

- Lalement, R. (1990). Logique, réduction, résolution. Editions Masson.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Machine learrning **Semestre : 2 Type : UEM2**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 42h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Cette matière comprend une introduction au domaine, suivie d'un aperçu des techniques d'apprentissage supervisé et non supervisé, puis d'une exploration de sujets plus avancés tels que l'apprentissage profond et le traitement automatique du langage naturel. La matière aborde aussi les fondements mathématiques essentiels (algèbre linéaire, statistiques), la programmation Python et des aspects pratiques tels que le prétraitement des données, l'ingénierie des caractéristiques, la sélection de modèles, l'entraînement, l'évaluation et le déploiement.

Connaissances préalables recommandées : Mathématiques : Algèbre linéaire, probabilités, statistiques, etc.

Contenu de la matière
Cours : 21h

1. Introduction et fondamentaux de l'apprentissage automatique
   -Apprentissage automatique ; définition et domaines d'application
   -Types d'apprentissage (supervisé, non supervisé, semi supervisé, par renforcement, autonome)
   -Fondements mathématiques : algèbre linéaire, statistiques et probabilités, variables aléatoires et distributions, pensée Baysienne.
   -Flux de travail d'apprentissage automatique: les étapes d'un projet d'apprentissage automatique (de la collecte des données au déploiement).
2. Apprentissage supervisé
   -Régression : linéaire, polynomiale, etc.
   -Classification : Régression logistique, machines à vecteurs de support (SVM), arbres de décision et méthodes d'ensemble (forêts aléatoires, boosting de gradient).
   -Évaluation de modèles : Mesures permettant d'évaluer les performances des modèles de régression et de classification (RMSE, exactitude, précision, rappel, score F1).
3. Apprentissage non supervisé
   -Clustering : Clustering K-means, clustering hiérarchique (agglomératif, divisif), basé sur la densité (DBSCAN, OPTICS).
   -Métriques d'évaluation : score de silhouette, indice de Davies-Bouldin, indice de Rand ajusté, information mutuelle
   -Réduction de dimensionnalité : Analyse en composantes principales (ACP), t-SNE, Analyse Discriminante Linéaire (LDA), Analyse en Composantes Indépendantes (ICA), UMAP, Analyse Factorielle.
   -Métriques d'évaluation : support, confiance, levier, conviction.
   -Détection d'anomalies : Identification des valeurs aberrantes
   - Apprentissage des Règles d'Association: Analyse du panier d'achat, Algorithme Apriori, Algorithme Eclat, Croissance de motif fréquent (FP-Growth), Métriques d'évaluation (support, confiance, levier, conviction)
4. Apprentissage profond
   -Introduction aux réseaux de neurones : architecture de base, principes de fonctionnement.
   -Architectures d'apprentissage profond : différents types de réseaux de neurones (réseaux de neurones convolutifs (CNN), réseaux de neurones récurrents (RNN).
   -Entraînement de modèles d'apprentissage profond : TensorFlow, PyTorch.

5. Sujets avancés

-Traitement automatique du langage naturel (TALN)
-Analyse des séries temporelles
-Systèmes de recommandation
-Apprentissage par renforcement

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** contrôles continus, travail personnel.

Références bibliographiques

- Andreas Müller and Sarah Guido (2016). Introduction to Machine Learning with Python. O'Reilly.

-Bastien, L. (2024). Machine Learning et Big Data : définition et explications de la combinaison. https://www.lebigdata.fr/machine-learning-et-big-data.

-Gullitti, T. et LLC, R.B. (2017). Application Of Machine Learning Algorithms To On-Board Diagnostics (Obd Ii) Threshold Determination.

-Chapelle, O., Scholkopf, B. et Zien, Eds., A. (2009). Semi-Supervised Learning. IEEE Transactions on Neural Networks, 20(3), p: 542–542. https://doi.org/10.1109/TNN.2009.2015974

-Batta, M. (2018). Machine Learning Algorithms : A Review. https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Gestion de projets informatiques **Semestre : 2 Type : UET2**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD : -** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 1 **Crédit :** 2

Objectifs de l'enseignement: cette matière permet à l'étudiant :

- De comprendre les principes et les méthodes de gestion de projets informatiques.
- D'exécuter les différentes phases d'un projet informatique, de la conception à la clôture.
- D'acquérir les compétences en planification, organisation, suivi et contrôle de projets.
- D'identifier et gérer les risques liés aux projets informatiques.
- D'utiliser des outils et des techniques de gestion de projets informatiques.
- De travailler en équipe dans un contexte de gestion de projet.

Connaissances préalables recommandées : génie logiciel, systèmes d'information.

Contenu de la matière
Cours : 21h

1. Gestion de projets informatiques : rappels
   - Définition et enjeux de la gestion de projets informatiques.
   - Processus de production d'un projet informatique : processus de réalisation, processus de gestion, processus qualité.
   - Les différents types de projets informatiques (développement logiciel, infrastructure, etc.).
   - Les acteurs impliqués dans un projet informatique.
   - Les relations entre gestion de projet et systèmes d'information.
   - Méthodologies traditionnelles (cycle en V).
   - Méthodologies agiles (Scrum, Kanban).
2. Planification et estimation
   - Définition du périmètre et des objectifs du projet.
   - Découpage du projet en tâches (WBS).
   - Établissement du planning (diagramme de Gantt, PERT).
   - Estimation des ressources (temps, budget, personnel, coûts, COCOMO).
   - Utilisation d'outils de planification (Microsoft Project, Asana, etc.).
3. Gestion des risques et suivi
   - Identification et analyse des risques (matrice SWOT, diagramme d'Ishikawa)
   - Évaluation de la probabilité et de l'impact des risques
   - Suivi de l'avancement du projet
   - Gestion des changements
   - Communication et reporting
   - Tableaux de bord et de métriques clés de performance (KPI)
4. Gestion des ressources
   - Identification des ressources nécessaires
   - Allocation des ressources aux différentes tâches
   - Gestion des conflits et des priorités
5. Gestion de la qualité
   - Définition des critères de qualité.
   - Mise en place de procédures de contrôle qualité
   - Réalisation de tests de vérification et de validation.
6. Clôture du projet

- Documentation de la clôture du projet.
- Transfert des connaissances et des compétences.
- Évaluation de la satisfaction des parties prenantes.
- Bilan du projet et identification des leçons apprises.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** contrôles continus, travail personnel..

Références bibliographiques

-Project Management Institute (PMI) 52021). A Guide to the Project Management Body of Knowledge (PMBOK Guide). https://www.pmi.org/

-Andrew Stellman, Jennifer Greene (2005). Applied Software Project Management . Series: Theory In Practice.

-Kerzner, Harold 2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley. https://www.wiley.com/

-S. Berkun (2008). Making Things Happen: Mastering Project Management (Theory I Practice). O'Reilly.

- R. S. Pressman, B. R. Maxim (2014). Software Engineering: a Practionner's Approach. McGraww Hill.

- McConnell, Steve (2006). Software Estimation: Demystifying the Black Art. Microsoft Press. https://www.microsoftpressstore.com/

-C. Aubry (2022). Scrum un outil convivial pour une agilité radicale. Editions Dunod.

-Stephen H.Kan (2010). Metrics and Models in Software Quality Engineering (2nd Edition), Addison-Wesley Professional.

-Linda Westfall (2009). The Certified Software Quality Engineer Handbook. Quality Press.

-Murali Chemuturi (2010). Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers. J. Ross Publishing.

Programme détaillé des enseignements du semestre 2 (S2)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Au choix **Semestre : 2 Type : UED2**
VHS : 21h **VHH :** 1h30 **Cours :** 1h30 **TD : TP :**
VHS travail personnel : **Coefficient :** 1 **Crédit :** 1

Objectifs de l'enseignement
Cette matière permet aux étudiants d'acquérir une vision plus large de leur domaine d'études et favorise une meilleure compréhension des enjeux sociaux, économiques et éthiques liés à l'informatique, ainsi qu'une capacité à communiquer efficacement et à s'adapter à des situations diverses qui peuvent se présenter. En d'autres termes, cette matière permet :

- Une ouverture à d'autres disciplines en relation avec le quotidien, afin de comprendre l'importance de la collaboration dans un environnement pluridisciplinaire.
- L'élargissement des horizons, ce qui favorise la compréhension de la pensée humaine dans le temps et l'espace.
- Développement de compétences transversales : communication, vision holistique, analyse et synthèse, capacité d'adaptation, etc.

Connaissances préalables recommandées

Contenu de la matière
Cours : 21h

5- Introduction à la discipline concernée (matière choisie).
6- Présentation des concepts fondamentaux.
7- Applications croisées avec l'informatique
8- Études de cas.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (100%)**.

Références bibliographiques : L'enseignant propose des références selon la matière choisie.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Assurance qualité et tests de logiciels **Semestre: 3 Type : UEF31**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 5

Objectifs de l'enseignement : Cette matière vise à fournir aux étudiants les connaissances et compétences nécessaires pour garantir la qualité des logiciels tout au long de leur cycle de vie, afin de répondre aux besoins des clients.

Connaissances préalables recommandées : algorithmique, génie logiciel, systèmes d'information.

Contenu de la matière
Cours : 21h

1. Introduction à l'assurance qualité logicielle
   - Définition et objectifs de l'assurance qualité logicielle (AQL)
   - Rôle de l'AQL dans le cycle de vie du logiciel
   - Qualité logicielle : mesures et indicateurs (Correctitude, fiabilité, efficacité, etc.)
   - Normes et standards en AQL: ISO/IEC 25000, IEEE 829, etc. et leurs application.
2. Gestion de la qualité logicielle
   - Planification de l'assurance qualité: Plan d'AQL.
   - Suivi et contrôle de la qualité
   - Gestion des risques qualité: identification et mise en place des mesures préventives et/ou correctives.
   - Gestion de la configuration logicielle: suivi des versions, traçabilité et cohérence.
   - Gestion des défauts
   - Qualité du logiciel et développement agile.
3. Tests de logiciels
   - Introduction aux tests : Définition et rôle dans l'AQL.
   - Types de tests
     - Tests unitaires.
     - Tests d'intégration:.
     - Tests système.
     - Tests d'acceptation.
     - Tests de performance.
   - Stratégies de test: tests boîte noire, tests boîte blanche, etc..
   - Outils de test: Selenium, JUnit, etc. et de leur utilisation.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques
- Pressman, Roger S., & Maxim, Bruce R. (2025). Software engineering: a practitioner's approach. 7th edition, McGraw-Hill.

- Ian Sommerville (2016). Software engineering. Pearson Education Limited.

- Stephen H.Kan (2010). Metrics and Models in Software Quality Engineering. 2nd Edition. Addison-Wesley Professional, ISBN-10: 0201729156.

- Linda Westfall (2009). The Certified Software Quality Engineer Handbook. Quality Press, ISBN-10: 0873897307.

- Murali Chemuturi (2010). Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers. J. Ross Publishing. ISBN-10: 1604270322.

- Myers, Glenford J., Sandler, Corey, Badgett, Tom, & Thomas, Todd M. (2004). The art of software testing. 2nd edition, John Wiley & Sons.

- Kaner, Cem, Bach, James, & Pettichord, Bret (2002). Lessons learned in software testing, a context-driven approach. John Wiley & Sons.

- Beizer, Boris (1990). Software testing techniques. Van Nostrand Reinhold.

- Pezzè, Mauro, & Young, Michal (2008). Software testing and analysis: process, principles, and techniques. John Wiley & Sons.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Ingénierie dirigée par les modèles **Semestre : 3 Type : UEF31**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 3 **Crédit :** 5

Objectifs de l'enseignement : Suite à l'initiative MDA du consortium OMG, l'ingénierie dirigée par les modèles (MDE) a vu le jour. Cette approcheest vue comme une démarche prometteuse à la fois dans la garantie des qualités de service du produit logiciel quedans la maîtrise du processus de développement et de maintenance. Par ailleurs, l'ingénierie dirigée par les modèlesvise la capitalisation du savoir-faire par son explicitation au niveau des méta-modèles et des transformations. Ce cours a pour but de permettre aux étudiants de maîtriser les concepts clés de la MDE et sa démarche.

Connaissances préalables recommandées : Langage orienté objets, génie logiciel, systèmes d'exploitation, compilation.

Contenu de la matière
Cours : 21h

Chapitre 1 : Introduction à l'ingénierie des modèles
- Historique et contexte
- Principes fondamentaux de l'IDM
- Objectifs et avantages de l'IDM
- Concepts clés (modèle, méta-modèle, transformation)

Chapitre 2 : L'initiative MDA du consortium OMG
- Principes de l'architecture MDA
- Standards de l'OMG (UML, OCL, MOF, Corba, XML-XMI, QVT, CWM)

Chapitre 3 : Modèles pour la définition des applications logicielles
- Types de modèles logiciels
- Modèles métiers(CIM)
- Modèles Indépendants des Plateformes (PIM)
- Modèles Spécifiques aux Plateformes (PSM)

Chapitre 4 : Transformation des modèles et génération du code
- Principe du mapping entre modèles
- Caractéristiques des transformations
- Modélisation des transformations
- Traçabilité des transformations

Chapitre 5 : Outils
- Catégories d'outils IDM et caractéristiques communes
- Présentation d'EMF (Eclipse Modeling Framework)
- Présentation du langage ATL (Atlas Transformation Language)
- La plateforme Eclipse pour l'IDM
- Aperçu d'autres outils IDM (cette partie fera l'objet de petits travaux avec les étudiants)

Chapitre 6 : Développement Formel par Raffinement
- La méthode B
- Raffinement d'une spécification abstraite B à une implémentation B :
- Méthodologie de développement d'applications sûres

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Kleppe, A., Warmer, J., &Basten, W. (2003). MDA explained: The model driven architecture: Practice and promise. Addison-Wesley Professional.

-Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S., &Jungmayr, S. J. (2006). Model-driven software development: Technology, engineering, management. John Wiley & Sons.

-Booch, G., Rumbaugh, J., & Jacobson, I. (1999). The unified modeling language user guide. Addison-Wesley Professional.

-Fowler, M. (2003). UML distilled: A brief guide to the standard object modeling language. Addison-Wesley Professional.

-Object Management Group. (n.d.). OMG SysML specification (Version 2.0 Beta 1). Retrieved from https://www.omg.org/spec/SysML/2.0/Beta1/About-SysML

-Holt, J., & Perry, S. (2019). SysML for systems engineering: A model-based approach. Institution of Engineering and Technology.

-Abrial, J.-R. (1996). The B-book: Assigning programs to meanings. Cambridge University Press.

Ressources logicielles:

- Eclipse Modeling Framework (**EMF**): Plateforme de référence pour la modélisation et la génération de code, https://www.eclipse.org/modeling/emf/
- **Papyrus**: Outil open source pour UML et SysMLbasé sur Eclipse, https://www.eclipse.org/papyrus/
- **ATL** (Atlas Transformation Language): Langage pour la transformation de modèles (M2M), https://www.eclipse.org/atl/

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Ingénierie des systèmes d'information (Filière : Informatique)

Intitulé de la matière : DevOpps et maintenance logicielle **Semestre: 3 Type : UEF31**
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** **TP :** 1h30
VHS travail personnel : 21h **Coefficient :** 3 **Crédit :** 4

Objectifs de l'enseignement: Ce programme vise à fournir aux étudiants les compétences et les connaissances nécessaires pour aborder les défis du développement logiciel, en mettant l'accent sur les aspects DevOps et la maintenance logicielle. Il permet aux étudiants :

- De comprendre les principes fondamentaux de DevOps et son impact sur le cycle de vie du développement logiciel.
- D'utiliser les outils et les pratiques DevOps pour l'automatisation, l'intégration continue, le déploiement continu et la surveillance des applications.
- D'acquérir des compétences en maintenance logicielle, notamment la gestion des bogues, la refactorisation et l'évolution des logiciels.
- De s'initier à l'approche collaborative et agile pour la gestion des projets logiciels.

Connaissances préalables recommandées : génie logiciel, gestion de projets.

Contenu de la matière
Cours : 21h

1- Introduction à DevOps et à la Maintenance Logicielle
- Définition et principes de DevOps.
- L'évolution du développement logiciel: de la cascade à l'agilité.
- Le cycle de vie DevOps : planification, développement, intégration, livraison, exploitation et surveillance.
- La maintenance logicielle: définition, enjeux, objectifs, types, cycle de vie et maintenance, qualité logicielle et maintenance, cadre réglementaire et normes (ISO/CEI 14764, etc.).
- La relation entre DevOps et la maintenance logicielle.

2- Automatisation et Intégration Continue
- Introduction à l'automatisation dans le développement logiciel.
- Outils d'automatisation : systèmes de gestion de versions (Git), outils d'intégration continue (Jenkins, GitLab CI, CircleCI).
- Conception et mise en œuvre de pipelines d'intégration continue.
- Gestion de la configuration logicielle.

3- Déploiement Continu et Infrastructure as Code
- Introduction au déploiement continu.
- Outils de déploiement continu : Docker, Kubernetes, Ansible, Terraform.
- Infrastructure as Code (IaC): avantages et mise en œuvre.
- Gestion des environnements de déploiement.

4- DevOps et Collaboration
- Rôle de la communication et de la collaboration dans DevOps.
- Méthodologies agiles et DevOps.
- Gestion des connaissances et documentation.

5- Outils de surveillance et techniques de maintenance
- Introduction à la surveillance des applications.
- Outils de surveillance : ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, Grafana.
- Gestion des incidents et des problèmes.
- Analyse de la cause racine des problèmes.

- Analyse d'impact et de faisabilité.
- Analyse et compréhension du code existant (reverse engineering).
- Techniques de débogage et de résolution de problèmes.
- Tests et validation de logiciels maintenus (tests unitaires, d'intégration, système, etc.).
- Outils d'aide à la maintenance (outils d'analyse statique, débogueurs, etc.).

6- Gestion de la configuration logicielle

- Principes et objectifs de la gestion de configuration.
- Identification et suivi des versions logicielles.
- Contrôle des changements et gestion des versions.
- Outils de gestion de configuration (Git, SVN, etc.).

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

- Pressman, Roger S., & Maxim, Bruce R. (2025). Software engineering: a practitioner's approach. 7th edition, McGraw-Hill.

- Gene Kim, Kevin Behr, George Spafford (2018). The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win. IT Revolution Press

- Jez Humble, David Farley (2010). Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison Wesley.

-Scott Chacon and Ben Straub (2014). Git Pro. 2nd edition. Apress publisher.

- John Ferguson Smart (year). Jenkins: The Definitive Guide.

- Diomidis Spinellis (2016). Effective Debugging: 66 Specific Ways to Debug Software and Hardware. Addison-Wesley.

- Martin Fowler (2018). Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional.

- Gene Kim, Jez Humble, Patrick Debois, John Willis (2016). The DevOps Handbook: How to Simplify Engineering and Operations to Maximize Business Value. IT Revolution

- Ken Schwaber (2004). Agile Project Management with Scrum. Microsoft Press.

- Pressman, Roger S., & Maxim, Bruce R. (2025). Software engineering: a practitioner's approach. 7th edition, McGraw-Hill.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Spécification et vérification formelle **Semestre : 3 Type : UEF32**
VHS : 63h **VHH :** 4h30 **Cours :** 1h30 **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 42h **Coefficient :** 3 **Crédit :** 4

Objectifs de l'enseignement : Les compétences développées dans le cadre de ce cours sont :

- Connaître les langages de description formelle des systèmes logiciels,
- Appliquer les techniques de modélisation aux systèmes critiques,
- Connaître les langages de spécification formelle des propriétés et des exigences,
- Exprimer les propriétés à satisfaire par les systèmes logiciels,
- Apprendre à appliquer les techniques de vérification formelle.

Connaissances préalables recommandées : Génie Logiciel, Logique Mathématique, Probabilités & Statistiques.

Contenu de la matière
Cours : 21h

1. Introduction aux méthodes formelles
    - Processus de modélisation et de vérification formelle
    - Langages formels de description des systèmes
    - Langages formels de spécification des systèmes
    - Techniques formelles de vérification
2. Modèles formels à base d'automates
    - Systèmes de transitions
    - Automates communicants et leurs types
    - Sémantiques
3. Algèbres de processus
    - Syntaxe
    - Sémantique Opérationnelle structurée
4. Réseaux de Petri
    - Syntaxe
    - Propriétés des graphes de marquages
    - Variantes et Extensions
5. Logiques temporelles & Technique de Model-Checking
    - Logique Temporelle à temps linéaire
    - Logique Temporelle à temps arborescent
    - Algorithmes de model-checking
6. Spécification Comportementale
    - Relations d'équivalence comportementale
    - Relations basées sur les modèles des traces, failures et raidies
    - Relations de Raffinement
7. Descriptions Formelles des Données
    - Types Abstrait de Données : Syntaxe & Sémantique axiomatique
    - Langage Z.
8. Modèles probabilistes
    - Chaînes de Markov

- Evaluation des performances.
- Model-Checking Probabiliste.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôle continu, mini projet.

Références bibliographiques

-Roggenbach, M., Cerone, A., Schlingloff, B.-H., Schneider, G., & Shaikh, S. A. (2021). Formal methods for software engineering: Languages, methods, application domains. Springer.
-Hoare, C. A. R. (1985). Communicating sequential processes. Prentice-Hall.
-Diaz, M. (Ed.). (2003). Vérification et mise enœuvre des réseaux de Petri. Hermes-Science.
-Reisig, W. (2013). Understanding Petri nets: Modeling techniques, analysis methods, case studies. Springer.
-Milner, R. (1989). Communication and concurrency. Prentice-Hall.
-Hennessy, M. (1988). Algebraic theory of processes. MIT Press.
-Baeten, J. C. M., &Weijland, W. P. (1990). Process algebra. Cambridge University Press.
-Roscoe, B. (2005). The theory and practice of concurrency. Prentice-Hall. (Original work published 1998)
-Fokkink, W. J. (2000). Introduction to process algebra (Texts in Theoretical Computer Science: An EATCS Series). Springer.
-Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., & Roscoe, B. (2000). Modelling and analysis of security protocols. Addison-Wesley.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Modélisation et évaluation des performances des systèmes
Semestre : 3 Type : UEM3
VHS : 42h **VHH :** 3h **Cours :** 1h30 **TD :** 1h30 **TP :**
VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 4

Objectifs de l'enseignement : Les objectifs de cette matière sont multiples. Le premier objectif serait de familiariser les étudiants aux principes de la modélisation et de l'évaluation des performances des systèmes de manière générale et plus particulièrement des systèmes informatiques logiciels et matériels. Un autre objectif consiste à sensibiliser l'étudiant au fait que le développement de tout système informatique ne doit aboutir à un système sous-dimensionné tout en évitant le surdimensionnement. Pour cela, développer un système adapté, en respectant le plus possible les objectifs du cahier des charges, est une démarche qui passera obligatoirement, au cours de la phase de conception, par une étape de modélisation en choisissant le formalisme le plus adéquat, suivie d'une étape d'analyse et d'évaluation des performances.

Connaissances préalables recommandées : Probabilités et Statistiques

Contenu de la matière
Cours : 21h

1. Introduction à la modélisation des systèmes

2. Principes de la modélisation et de l'évaluation des performances

3. Processus stochastiques

4. Chaînes de Markov
- Chaînes de Markov à temps discret
- Chaînes de Markov à temps continu

5. Files d'attente
- Files d'attente mono-serveur
- Files d'attente multi-serveurs
- Files d'attente à capacité limitée

6. Les réseaux de Petri

8. Les réseaux de Petri stochastiques

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Références bibliographiques

-Douc, R., Moulines, E., Priouret, P., & Soulier, P. (2018). Markov chains. Springer.

-Choquet-Geniet, A. (2006). Les réseaux de Petri : Un outil de modélisation : Cours et exercices corrigés - Licence, Master, écoles d'ingénieur. Dunod.

-Baynat, B. (2000). Théorie des files d'attente. Hermes.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Ingénierie des systèmes d'information (Filière : Informatique)

Intitulé de la matière : Big data analytics **Semestre: 3 Type : UEM3**
VHS : 84h **VHH :** 6h **Cours :** 3h **TD :** 1h30 **TP :** 1h30
VHS travail personnel : 63h **Coefficient :** 2 **Crédit :** 5

Objectifs de l'enseignement : ce programme vise à donner aux étudiants les compétences nécessaires pour traiter, analyser et interpréter de grands ensembles de données. Il permet aux étudiants :

- De comprendre les concepts fondamentaux du Big Data et de l'analytique.
- De découvrir les techniques de stockage, de traitement et d'analyse des données volumineuses.
- De développer des compétences en visualisation de données pour la prise de décision.
- De se familiariser avec les outils et technologies clés du Big Data.

Connaissances préalables recommandées : Bases de données, machine learning, statistiques, systèmes d'aide à la décision.

Contenu de la matière
Cours : 42h

1. Introduction au Big Data et à l'analytique
   - Définition et enjeux du Big Data: Volume, Vélocité, Variété, Véracité (les 4V), etc.
   - Architecture du Big Data: Systèmes distribués, Cloud computing, architectures Hadoop et Spark.
   - Types d'analytique: Descriptive, diagnostique, prédictive et prescriptive.
   - Applications du Big Data: Secteurs variés (finance, santé, marketing, etc.).
   - Outils et technologies: Hadoop, Spark, NoSQL, etc.
2. Stockage et traitement des données
   - Bases de données NoSQL: Types (clé-valeur, document, graphe, etc.), avantages et inconvénients.
   - Systèmes de fichiers distribués: HDFS (Hadoop Distributed File System).
   - Traitement batch et streaming: Apache Spark, Apache Kafka.
   - Langages de programmation: Python, Scala.
3. Analyse exploratoire des données et apprentissage automatique
   - Statistiques descriptives et inférentielles.
   - Nettoyage et prétraitement des données.
   - Techniques d'apprentissage automatique: Régression, classification, clustering, réduction de dimension.
   - Outils: Scikit-learn, TensorFlow, PyTorch.
4. Visualisation de données et prise de décision
   - Principes de la visualisation de données.
   - Outils de visualisation: Tableau, Power BI, matplotlib, seaborn.
   - Storytelling avec les données.
   - Intégration des résultats de l'analyse dans le processus décisionnel.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)

- **Examen semestriel en présentiel (60%)**.
- **Évaluation continue (CC) (40%) :** Contrôles continus, travail personnel.

Les travaux pratiques peuvent porter sur :

- L'analyse de données issues de différents domaines (finance, santé, marketing...).
- Conception et mise en œuvre d'une solution d'analyse de Big Data, sous forme de mini-projet en groupe d'étudiants.

Références bibliographiques

-Venkat Ankam (2016). Big data analytics. Packt publishing.

- Viktor Mayer-Schönberger et Kenneth Cukier (2013). Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt publisher.

-Kaisler S, Armour F, Espinosa J (2013) Big data: issues and challenges moving forward, Wailea, Maui, HI, s.n, pp 995–1004

- Nathan Marz and James Warren (2015). Big Data: Principles and best practices of scalable realtime data systems. Manning publisher.

-Bai, X., Duan, J., Li, B., Fu, S., Yin, W., Yang, Z., & Qu, Z. (2024). Global quantitative analysis and visualization of big data and medical devices based on bibliometrics. Expert Systems with Applications, 254, 124398.

- Tom White (2012). Hadoop: The Definitive Guide. Yahoo Press.

-Yadav S, Sohal A (2017) Review paper on big data analytics in Cloud computing. Int J Comp Trends Technol (IJCTT) IX. 49(3);156-160

- Matei Zaharia, Bill Chambers(2018). Spark: The Definitive Guide. O'Reilly Media publisher.

- Cole Nussbaumer Knaflic (2015). Storytelling with Data: A Data Visualization Guide for Business Professionals. Wiley publisher.

- Trevor Hastie, Robert Tibshirani et Jerome Friedman (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd edition, Springer.

- Edward R. Tufte (1997). The Visual Display of Quantitative Information. Graphics Pr publisher.

- Aurélien Géron (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. 2nd edition. O'Reilly Media publisher.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)
Intitulé de la matière : Méthodologie de recherche **Semestre : 3 Type : UET3**
VHS : 21h **VHH :** 1h30 **Cours :** 1h30 **TD : -** **TP : -**
VHS travail personnel : **Coefficient :** 1 **Crédit :** 2

Objectifs de l'enseignement : Cette matière vise à enseigner aux étudiants les compétences et les connaissances nécessaires pour mener des recherches scientifiques efficaces. Il couvre les étapes clés de la recherche, de la formulation du sujet à la rédaction du rapport final, en passant par la revue de littérature, la conception de la recherche, la collecte et l'analyse des données, ainsi que la présentation orale des résultats.

Connaissances préalables recommandées : rédaction scientifique, langue utilisée.

Contenu de la matière
Cours : 21h

1. Introduction à la recherche scientifique
   - Définition et objectifs de la recherche scientifique.
   - Types de recherche (fondamentale, appliquée, etc.).
   - Rôle de la méthodologie dans la recherche.
   - L'importance de la maitrise de la langue utilisée.
   - Éthique de la recherche en informatique : plagiat, auto plagiat, IA générative (ChatGpt, Deepseek, etc.)
2. Formulation du sujet de recherche
   - Identification et sélection d'un sujet de recherche pertinent.
   - Revue de littérature :
     - Recherche de sources d'information pertinentes (articles scientifiques, livres, etc.).
     - Analyse critique de la littérature existante (état de l'art).
     - Identification des lacunes dans l'état de l'art.
   - Définition de la problématique
     - Formulation d'une question de recherche claire et précise.
     - Définition d'hypothèses de recherche.
   - Objectifs de la recherche
     - Définition des objectifs spécifiques de la recherche cible.
     - Identification des résultats attendus.
3. Conception de la recherche
   - Choix des méthodes de recherche appropriées (expérimentale, qualitative, quantitative, etc.).
   - Planification de la collecte des données (échantillonnage, outils de collecte, etc.).
   - Élaboration d'un plan de recherche.
4. Collecte des données
   - Mise en œuvre du plan de recherche.
   - Utilisation des outils et techniques de collecte de données spécifiques à l'informatique.
   - Gestion et organisation des données collectées.
5. Analyse des données
   - Utilisation de logiciels d'analyse de données.
   - Traitement et analyse des données collectées.
   - Utilisation de méthodes statistiques ou d'autres techniques d'analyse appropriées.
   - Interprétation des résultats et discussion des implications.
6. Rédaction du rapport de recherche
   - Modes et structures de publication : article, brevet, thèse, livre, poster, communication orale…
   - Utilisation de logiciels de rédaction : LaTex, overleaf, etc.
   - Structure et organisation d'un rapport de recherche scientifique.

- Rédaction claire et concise des différentes parties du rapport.
- Présentation des résultats et discussion des conclusions.
- Rédaction et outils de gestion des références bibliographiques (APA, IEEE, Zotero, EndNote, etc.) et des annexes.

7. Présentation orale des résultats
   - Préparation et réalisation d'une présentation orale des résultats de recherche.
   - Maîtrise de la communication scientifique.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (100%)**.

Références bibliographiques

-Beaud, Michel (2020). L'art de la thèse. La Découverte.

-Stefan Kottwitz (2021). LaTeX Beginner's Guide: Create visually appealing texts, articles, and books for business and science using LaTeX. 2nd Edition, Packt Publishing.

-Jean-Marie Dubois (2005). La rédaction scientifique : mémoires et thèses : formes régulières et par articles. Estem.

-Michèle Lenoble-Pinson (1996). La rédaction scientifique : conception, rédaction, présentation. Signalétique, De Boeck Université.

-Christine Gérard, Jean Germain (1985). Recherche bibliographique et documentaire : généralités. Faculté de philosophie et Lettres.

Programme détaillé des enseignements du semestre 3 (S3)
Master académique
Spécialité : Génie logiciel (Filière : Informatique)

Intitulé de la matière : Module au choix **Semestre : 3 Type : UED3**
VHS : 21h **VHH :** 1h30 **Cours :** 1h30 **TD : TP :**
VHS travail personnel : **Coefficient :** 1 **Crédit :** 1

Objectifs de l'enseignement
Cette matière permet aux étudiants d'acquérir une vision plus large de leur domaine d'études et favorise une meilleure compréhension des enjeux sociaux, économiques et éthiques liés à l'informatique, ainsi qu'une capacité à communiquer efficacement et à s'adapter à des situations diverses qui peuvent se présenter. En d'autres termes, cette matière permet :

- Une ouverture à d'autres disciplines en relation avec le quotidien, afin de comprendre l'importance de la collaboration dans un environnement pluridisciplinaire.
- L'élargissement des horizons, ce qui favorise la compréhension de la pensée humaine dans le temps et l'espace.
- Développement de compétences transversales : communication, vision holistique, analyse et synthèse, capacité d'adaptation, etc.

Connaissances préalables recommandées

Contenu de la matière
Cours : 21h

9- Introduction à la discipline concernée (matière choisie).
10- Présentation des concepts fondamentaux.
11- Applications croisées avec l'informatique
12- Études de cas.

Mode d'évaluation (doit être porté à la connaissance des étudiants en début de chaque semestre)
- **Examen semestriel en présentiel (100%)**.

Références bibliographiques : L'enseignant propose des références selon la matière choisie.