REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

NOUVELLE OFFRE DE FORMATION
MASTER ACADEMIQUE HARMONISE

Etablissement	Faculté	Département

Domaine : Mathématiques & Informatique.

Filière : Informatique.

Spécialité : Génie Logiciel.

Le 02/09/2026

رئيس اللجنة البيداغوجية الوطنية
لميدان الرياضات والإعلام الآلي
أ.د بابا حنيني محمد شوقي

Année universitaire : 2026/2027

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

عرض تكويني جديد
ماستر أكاديمي موائم

المؤسسة	الكلية	القسم

الميدان: رياضيات وإعلام آلي.

الشعبة إعلام آلي.

التخصص: هندسة البرمجيات.

Le 02/09/2026

رئيس اللجنة البيداغوجية الوطنية
لميدان الرياضيات والإعلام الآلي
أ.د بابا حنيني محمد شوقي

السنة الجامعية: 2027/2026

SOMMAIRE

I- Fiche d’identité du Master.

Spécialité : Génie Logiciel.

1 - Localisation de la formation.

- Faculté :

- Département :

2- Partenaires de la formation. *

- Autres établissements universitaires :

- Entreprises et autres partenaires socio-économiques :

- Partenaires internationaux :

* Présenter les conventions en annexe de la formation.

3- Contexte et objectifs de la formation.

A– Conditions d'accès.

B - Objectifs de la formation.

C – Profils et compétences métiers visés.

D- Potentialités régionales et nationales d'employabilité.

E – Passerelles vers d'autres spécialités.

F – Indicateurs de suivi de la formation.

G – Capacités d'encadrement.

La capacité d'encadrement est estimée à **……. Étudiants**.

4- Moyens humains disponibles.

A- Enseignants intervenants dans la spécialité.

Nom et prénoms	Diplôme	Grade	Laboratoire	Spécialité	Type d'intervention [1]	Signature

[1] Cours, TD, TP, encadrement de stage, encadrement de mémoire, autres (à préciser).

5- Moyens matériels spécifiques disponibles.

A- Laboratoires pédagogiques et équipements.

Intitulé du laboratoire :

N°	Intitulé de l'équipement	Nombre	Observations

Intitulé du laboratoire :

N°	Intitulé de l'équipement	Nombre	Observations

B- Terrains de stage et formations en entreprise.

Lieu du stage	Nombre d'étudiants	Durée du stage

C- Laboratoire de recherche de soutien à la formation.

Directeur du laboratoire :
No. D'agrément :
Date : **Avis du directeur du laboratoire :**

D- Projets de recherche de soutien au Master.

Intitulé du projet de recherche	Code du projet	Date du début du projet	Date de fin du projet

E- Espaces de travaux personnels et TIC.

II- Fiche d’organisation semestrielle des enseignements.

Spécialité : Génie Logiciel.

1- Semestre 1.

Semestre	Unités d'Enseignement				Matière (s) constituant l'unité							
	Nature	Code	Crédits	Coeff.	Intitulés des matières	VHS	V.H Hebdomadaire				Crédits	Coeff.
						14 Sem	Cours	TD	TP	Autres		
S1	Unité d'Enseignement Fondamentale	UEF11	10	6	Algorithmique Avancée et Complexité	63h00	1h30	1h30	1h30	3h00	5	3
					Bases de Données Avancées	63h00	1h30	1h30	1h30	3h00	5	3
		UEF12	8	6	Ingénierie des Exigences	42h00	1h30	1h30	-	1h30	4	3
					Méthodes de Conception de Logiciels et Design Patterns	63h00	1h30	1h30	1h30	3h00	4	3
	Unité d'Enseignement de Méthodologie	UEM1	9	4	Compilation Avancée	42h00	1h30	-	1h30	1h30	4	2
					Paradigmes de Programmation	42h00	1h30	-	1h30	1h30	5	2
	Unité d'Enseignement Transversale	UET1	2	1	Réseaux Avancés	42h00	1h30	-	1h30	1h30	2	1
	Unité d'Enseignement de Découverte	UED1	1	1	Une Matière au Choix	21h00	1h30	-	-	-	1	1
	Total du semestre 1		**30**	**18**		**378h00**	**12h00**	**6h00**	**9h00**	**15h00**	**30**	**18**

2- Semestre 2.

Semestre	Unités d'Enseignement				Matière (s) constituant l'unité							
	Nature	Code	Crédits	Coeff.	Intitulés des matières	VHS	V.H Hebdomadaire				Crédits	Coeff.
						14 Sem	Cours	TD	TP	Autres		
S2	Unité d'Enseignement Fondamentale	UEF21	10	6	Architectures Logicielles	63h00	1h30	1h30	1h30	3h00	5	3
					Architectures Orientées Services	63h00	1h30	1h30	1h30	3h00	5	3
		UEF22	8	6	Systèmes d'Information Coopératifs	42h00	1h30	1h30	-	1h30	4	3
					Sémantiques Formelles des Langages de Programmation	42h00	1h30	1h30	-	1h30	4	3
	Unité d'Enseignement de Méthodologie	UEM2	9	4	Systèmes Multi Agents	42h00	1h30	-	1h30	1h30	4	2
					Apprentissage Automatique	63h00	1h30	1h30	1h30	3h00	5	2
	Unité d'Enseignement Transversale	UET2	2	1	Lean Startup	42h00	1h30	-	1h30	1h30	2	2
	Unité d'Enseignement de Découverte	UED2	1	1	Une Matière au Choix	21h00	1h30	-	-	-	1	1
	Total du semestre 2		**30**	**18**		**378h00**	**12h00**	**7h30**	**7h30**	**15h00**	**30**	**18**

3- Semestre 3.

Semestre	Unités d'Enseignement				Matière (s) constituant l'unité							
	Nature	Code	Crédits	Coeff.	Intitulés des matières	VHS	V.H Hebdomadaire				Crédits	Coeff.
						14 Sem	Cours	TD	TP	Autres		
S3	Unité d'Enseignement Fondamentale	UEF31	9	6	Assurance Qualité & Tests des Logiciels	63h00	1h30	1h30	1h30	3h00	5	3
					DevOps & Maintenance des Logiciels	42h00	1h30	-	1h30	1h30	4	3
		UEF32	9	6	Ingénierie Dirigée par les Modèles	42h00	1h30	-	1h30	1h30	5	3
					Spécification et Vérification Formelle	63h00	1h30	1h30	1h30	3h00	4	3
	Unité d'Enseignement de Méthodologie	UEM3	9	4	Modélisation et Evaluation des Performances des Systèmes	42h00	1h30	1h30	-	1h30	4	2
					Analyse des Données Massives	84h00	3h00	1h30	1h30	3h00	5	2
	Unité d'Enseignement Transversale	UET3	2	1	Méthodologie de Recherche	21h00	1h30	-	-	-	2	1
	Unité d'Enseignement de Découverte	UED3	1	1	Une Matière au Choix	21h00	1h30	-	-	-	1	1
	Total du semestre 3		**30**	**18**		**357h00**	**10h30**	**6h00**	**9h00**	**18h00**	**30**	**18**

4- Semestre 4.

Domaine : Mathématiques et Informatique.
Filière : Informatique.
Spécialité : Génie Logiciel.

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

5- Récapitulatif global de la formation.

UE → / **VH**	**UEF**	**UEM**	**UET**	**UED**	**S4**	**Total**
Cours	252 h	147 h	63 h	63 h		**525 h**
TD	210 h	63 h	-	-		**273 h**
TP	189 h	105 h	42 h	-		**336 h**
Mémoire	-	-	-	-	750 h	**750 h**
Stage dans l'entreprise	-	-	-	-		
Ateliers	-	-	-	-		
Travail Personnel	399 h	168 h	42 h	-		**609 h**
Autres	-	-	-	-		
Total	**1050 h**	**483 h**	**147 h**	**63 h**	**750 h**	**2493 h**
Crédits	**54**	**27**	**6**	**3**	**30**	**120**
% en crédits pour chaque UE	**45 %**	**22.50 %**	**5 %**	**2.50 %**	**25 %**	**100 %**

6- السداسي الأول:

وحدة التعليم	عنوان المواد	الرصيد	المعامل	الحجم الساعي الأسبوعي			الحجم الساعي للسداسي (14 أسبوع)	أخرى	نوع التقييم	
				دروس	أعمال موجهة	أعمال تطبيقية			مراقبة مستمرة	امتحان
وحدة تعليم أساسية الرمز: وت أس 11 الأرصدة:10 المعامل: 6	الخوارزميات المتقدمة والتعقيد	5	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
	قواعد البيانات المتقدمة	5	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
وحد تعليم أساسية الرمز: وت أس 12 الأرصدة: 8 المعامل: 6	هندسة المتطلبات	4	3	1سا30	1سا30	-	42سا00	21سا00	40%	60%
	طرق تصميم البرمجيات ونماذج التصميم	4	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
وحدة تعليم منهجية الرمز: وت م 1 الأرصدة: 9 المعامل: 4	الترجمة المتقدمة	4	2	1سا30	-	1سا30	42سا00	21سا00	40%	60%
	نماذج البرمجة	5	2	1سا30	-	1سا30	42سا00	21سا00	40%	60%
وحدة تعليم أفقية الرمز: وت أف 1 الأرصدة: 2 المعامل: 1	الشبكات المتقدمة	2	1	1سا30	-	1سا30	42سا00	21سا00	40%	60%
وحدة تعليم استكشافية الرمز: وت إس 1 الأرصدة: 1 المعامل: 1	مقياس اختياري	1	1	1سا30	-	-	21سا00	-	-	100%
مجموع السداسي 1		**30**	**18**	**12سا00**	**6سا00**	**9سا00**	**378سا00**	**210سا00**		

7- السداسي الثاني:

وحدة التعليم	عنوان المواد	الرصيد	المعامل	الحجم الساعي الأسبوعي			الحجم الساعي للسداسي (14 أسبوع)	أخرى	نوع التقييم	
				دروس	أعمال موجهة	أعمال تطبيقية			مراقبة مستمرة	امتحان
وحدة تعليم أساسية الرمز: وت أس21 الأرصدة: 10 المعامل: 6	معمارية البرمجيات	5	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
	المعماريات الموجهة بالخدمات	5	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
وحد تعليم أساسية الرمز: وت أس 22 الأرصدة: 8 المعامل: 6	نظم المعلومات التعاونية	4	3	1سا30	1سا30	-	42سا00	21سا00	40%	60%
	الدلالات الشكلية للغات البرمجة	4	3	1سا30	1سا30	-	42سا00	21سا00	40%	60%
وحدة تعليم منهجية الرمز: و ت م 2 الأرصدة: 9 المعامل: 4	النظم متعددة الوكلاء	4	2	1سا30	-	1سا30	42سا00	21سا00	40%	60%
	تعلم الآلة	5	2	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
وحدة تعليم أفقية الرمز: وت أف 2 الأرصدة: 2 المعامل: 1	المؤسسات الناشئة المرنة	2	2	1سا30	-	1سا30	42سا00	21سا00	40%	60%
وحدة تعليم استكشافية الرمز: وت إس 2 الأرصدة: 1 المعامل: 1	مقياس اختياري	1	1	1سا30	-	-	21سا00	-	-	100%
مجموع السداسي 2		**30**	**18**	**12سا00**	**7سا30**	**7سا30**	**378سا00**	**210سا00**		

8- السداسي الثالث:

وحدة التعليم	عنوان المواد	الرصيد	المعامل	الحجم الساعي الأسبوعي			الحجم الساعي للسداسي (14 أسبوع)	أخرى	نوع التقييم	
				دروس	أعمال موجهة	أعمال تطبيقية			مراقبة مستمرة	امتحان
وحدة تعليم أساسية الرمز: وت أس 31 الأرصدة: 9 المعامل: 6	ضمان الجودة واختبار البرمجيات	5	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
	DevOps وصيانة البرمجيات	4	3	1سا30	-	1سا30	42سا00	21سا00	40%	60%
وحد تعليم أساسية الرمز: وت أس 32 الأرصدة: 9 المعامل: 6	الهندسة الموجهة بالنماذج	5	3	1سا30	-	1سا30	42سا00	21سا00	40%	60%
	المواصفات والتحقق الشكلي	4	3	1سا30	1سا30	1سا30	63سا00	42سا00	40%	60%
وحدة تعليم منهجية الرمز: وت م 3 الأرصدة: 9 المعامل: 4	نمذجة وتقييم أداء الأنظمة	4	2	1سا30	1سا30	-	42سا00	21سا00	40%	60%
	تحليل البيانات الضخمة	5	2	**3سا00**	1سا30	1سا30	63سا00	42سا00	40%	60%
وحدة تعليم أفقية الرمز: وت أف 3 الأرصدة: 2 المعامل: 1	منهجية البحث العلمي	2	1	1سا30	-	-	21سا00	21-سا00	-	100%
وحدة تعليم استكشافية الرمز: وت إس 3 الأرصدة: 1 المعامل: 1	مقياس اختياري	1	1	1سا30	-	-	21سا00	-	-	100%
مجموع السداسي 3		**30**	**18**	**13سا30**	**6سا00**	**7سا30**	**378سا00**	**189سا00**		

III- Programme détaillé par matière.

Spécialité : Génie Logiciel.

Intitulé de la matière : Algorithmique Avancée et Complexité. **Semestre :** 1. **Type :** UEF11.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **Coefficient :** 3. **Crédit :** 5.

Objectifs de la matière : Cette matière couvre des concepts fondamentaux et des techniques avancées en conception et analyse d'algorithmes en approfondissant les structures de données, les algorithmes de recherche et de tri, ainsi que des approches comme la programmation dynamique et les algorithmes gloutons. Elle fournit aussi une compréhension approfondie de la complexité algorithmique, en mettant l'accent sur les techniques d'analyse, l'importance de la complexité pour l'optimisation et les méthodes pour prouver la correction des algorithmes.

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

Contenu de la matière :

Cours : 21h.

1- Introduction à la complexité algorithmique et notions préliminaires.
- Définition d'algorithme et d'algorithmique.
- Notion de problème algorithmique et sa résolution.
- Pourquoi analyser la complexité ? Comparaison d'algorithmes pour un même problème.
- Notions de base et types de complexité : temporelle et spatiale.

2- Analyse de la complexité temporelle.
- Notation grand O (O) : Définition, interprétation et utilité pour exprimer la complexité asymptotique.
- Calcul de la complexité.
  - Identification des opérations élémentaires (comparaisons, affectations, etc.).
  - Analyse des boucles et des fonctions récursives.
  - Cas le pire, le meilleur, et moyen.
  - Exemples de complexités courantes : O(1), O(log n), O(n), O(n log n), O(n^2), etc.

3- Structures de données et complexité.
- Tableaux, listes chaînées, arbres, graphes, Tas : opérations courantes et complexité associée.
- Impact de la structure de données sur la complexité d'un algorithme.

4- Techniques d'analyse de complexité.
- Diviser pour régner : Stratégie de base, analyse de la complexité (exemple, tri par fusion).
- Programmation dynamique : Principe et optimisation (exemples : problème du sac à dos, plus court chemin).
- Algorithmes gloutons : Principes et analyse de la complexité. (Exemples voyageur de commerce, algorithme de Dijkstra).

5- Complexité spatiale.
- Mesure de l'utilisation de la mémoire par un algorithme.
- Lien avec la complexité temporelle et compromis possibles.

6- Complexité et Preuve de Correction.
  - Notions de P, NP, NP-complet : Définitions et exemples de problèmes NP-complets.
  - Techniques de Preuve de Correction : Preuve par induction, preuves par contradiction.

7- Optimisation d'algorithmes.
- Utilisation des concepts de complexité pour identifier les points faibles d'un algorithme.

Techniques d'optimisation (par exemple, simplification de boucles, réduction de la récursivité).

Mode d'évaluation : Contrôle continu et examen.

Références bibliographiques :

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein (2009). Introduction to Algorithms. MIT Press, 3rd Edition.
2. Ivan Lavallée (2008). Complexité et algorithmique avancée : une introduction. Editions Hermann.
3. Jon Kleinberg, Éva Tardos (2005). Algorithm Design. Pearson Publisher, 1st edition.
4. Nicolas Hermann et Pierre Lescanne (2005). Est-ce que P = NP ? Les Dossiers de La Recherche, 20 : 64–68, août-octobre.
5. Robert Sedgewick, Kevin Wayne (2011). Algorithms, Addison-Wesley, 4th edition.
6. Förg Flum, Martin Grohe. Parameterized Complexity Theory. Springer Verlag, 2006.
7. Udi Manber 1989). Introduction to Algorithms: A Creative Approach. Addison-Wesley.
8. J. J. McConnell (2001). Analysis of algorithms: an active learning approach. Jones and Barlett Publishers.
9. Cormen, Leiserson, Rivest Stein (2010). Algorithmique, cours exercices et problèmes, 3ème édition, Dunod.
10. Sylvain Perifel (2014). Complexité algorithmique. Ellipses.
11. Sanjeen Arora and Boaz Barak (2006). Computational Complexity: A Modern Approach.

Intitulé de la matière : Bases de Données Avancées. **Semestre :** 1. **Type :** UEF11.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **Coefficient :** 3. **Crédit :** 5.

Objectifs de la matière : 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 avancé : Jointures, sous-requêtes, fonctions d'agrégation, vues, procédures stockées.
- Les Triggers.
- Les fonctions et procédures stockées.
- 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- 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 : Contrôle continu et examen.

Références bibliographiques :

1. R. Elmasri & S. Navathe (2016). Fundamentals of Database Systems, 7th Edition, Pearson.
2. R. Elmesri, B. Navate (2016). Fundamentals of Database Systems. 7th edition, Pearson Editions.
3. T. Connolly & C. Begg (2014). Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson.
4. Christian Soutou (2013). SQL pour Oracle. Editions Eyrolles.
5. Silberschatz, H. Korth & S. Sudarshan (2019). Database System Concepts, 7th Edition, McGraw-Hill.
6. Mohamed Fadhel SAAD (2016). PL/SQL sous Oracle 12c. Guide du développeur.
7. R.G.G. Cattell (1994). Object Data Management. Addison-Wesley.
8. P. Selmer (2012). NOSQL stores and Data analytics tools. Advances in Data Management.

Intitulé de la matière : Ingénierie des Exigences. **Semestre :** 1. **Type :** UEF12.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 1h30. **TP :** 0h00.
VHS travail personnel : 21h00. **Coefficient :** 3. **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 et 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, Requisite Pro, 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 :

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

Références bibliographiques :

1. Karl Wiegers, Joy Beatty (2013). Software Requirements. Microsoft Press, 3e éd.
2. Elizabeth Hull, Ken Jackson, Jeremy Dick (2017). Requirements Engineering. Springer, 4e éd.
3. Klaus Pohl and P.A. Börstler (2010). Requirements Engineering: Processes and Techniques.
4. Stéphane Badreau, Jean-Louis Boulanger (2014). Ingénierie des exigences : Méthodes et bonnes pratiques pour construire et maintenir un référentiel. Dunod.
5. Martin Fowler (2000). UML Distilled.
6. Sommerville I (2013). Software Engineering. Pearson Education.
7. Roel Wieringa Anne Persson (2010). Requirements Engineering: Foundation for Software Quality. Spinger.

Intitulé de la matière : Méthodes de Conception de logiciels et Design Patterns. **Semestre :** 1. **Type :** UEF12.

VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.

VHS travail personnel : 42h00. **Coefficient :** 3. **Crédit :** 4.

Objectifs de l'enseignement : Cette matière 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 : Génie Logiciel 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 :

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

Références bibliographiques :

1. Booch, G., Rumbaugh, J., Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison-Wesley.
2. Rolland, Collette (1996). Conception des BBD : méthodes orientées objet et évènements. Techniques de l'ingénieur, réf : H3248 v1.
3. Fowler, M. (2003). Patterns of Enterprise Application Architecture. Addison-Wesley.
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
5. OMG (2017). OMG Unified Modeling Language (OMG UML), Version 2.5.1. Object Management Group. Object Management Group.
6. Kruchten P. (2001). Introduction au Rational Unified Process (RUP), Editions Eyrolles, Paris.
7. L Thiry, B Thirion, M Hassen forder (2008). Dsls pour le développement agile de transformations. IDM'08.
8. 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).

Intitulé de la matière : Compilation Avancée. **Semestre :** 1. **Type :** UEM1.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **Coefficient :** 2. **Crédit :** 4.

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

- De comprendre les principes de la conception et de l'implémentation des compilateurs.
- De connaitre les techniques d'analyse lexicale, syntaxique et sémantique.
- De pouvoir 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 et 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 débogage des compilateurs.

Mode d'évaluation :

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

Références bibliographiques :

1. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman (2007). Compilers: Principles, Techniques and Tools. Pearson Addison Wesley.
2. Anthony J. Dos Reis (2011). Compiler Construction Using Java, Java CC, and Yacc. Wiley-IEEE Computer Society Press.
3. Andrew W Appel (1998). Modern Compiler Implementation in ML. Published by Cambridge university Press.
4. Ronald Mak (2009). Writing Compilers and Interpreters: A Software Engineering Approach. John Wiley and Sons.
5. Sebastian Hack, Reinhard Wilhelm and Helmut Seidl (2013). Compiler Design: Code Generation and Machine-Level Optimization. Springer.
6. Keith Cooper & Linda Torczon (2011). Engineering a Compiler. 2nd edition, Elsevier Morgan-Kauffman.
7. Dick Grune, Kees van Reeuwijk, Henri E. Bal and Ceriel J.H. Jacobs (2012). Modern Compiler Design. Wiley.

Intitulé de la matière : Paradigmes de Programmation. **Semestre :** 1. **Type :** UEM1.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **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 et logique.

Contenu de la matière :

Cours : 21h.

1. **Introduction aux Paradigmes des Langages de Programmation**
   - Définitions et concepts fondamentaux.
   - Classification des langages de programmation selon les paradigmes : impératif, déclaratif, etc.
   - Importance des paradigmes dans le développement logiciel.
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
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).

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.

5. Comparaison et choix des paradigmes

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

Mode d'évaluation :

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

Références bibliographiques :

1. Bansal A. K. (2014). Introduction to programming languages. CRC Press.
2. Gabbrielli M., Martini S. (2010). Programming languages : Principles and paradigms. Springer.
3. Van Roy, P., & Haridi, S. (2004). Concepts, techniques, and models of computer programming. MIT press.
4. David A Watt (2004). Programming language design concepts. John Wiley & sons.
5. Daniel P. Friedman, Mitchell Wand, Christopher T. Haynes (2001). Essentials of Programming Languages. MIT Press.
6. Clocksin, W. F., & Mellish, C. S. (2003). Programming in PROLOG. Springer Science & Business Media.
7. Louden K. C., Lambert K. A. (2012). Programming languages: Principles and practice. Course Technology, Cengage Learning.
8. Scott M. L. (2009). Programming Language Pragmatics. Elsevier.
9. Sebesta R. W. (2006). Concepts of Programming Languages. Addison Wesley.
10. Minsky, Y., Madhavapeddy, A., & Hickey, J. (2013). Real World OCaml: Functional programming for the masses. " O'Reilly Media, Inc.".

Intitulé de la matière : Réseaux Avancés. **Semestre :** 1. **Type :** UET1.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **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.

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 :

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

Références bibliographiques :

1. Tanenbaum, A., Feamster, N., & Witherall, D. (2021). Computer networks (6th ed.). Pearson Publisher.
2. White, R., & Banks, E. (2017). Computer networking problems and solutions: An innovative approach to building resilient, modern networks. Addison-Wesley Professional.
3. Englander, I., & Wong, W. (2021). The architecture of computer hardware, systems software, and networking: An information technology approach. John Wiley & Sons.
4. Bonaventure, O. (2021). Computer networking: Principles, protocols and practice.
5. Dordal, P. L. (2022). An introduction to computer networks. Department of Computer Science, Loyola University Chicago.

Intitulé de la matière : Une Matière au Choix. **Semestre :** 1. **Type :** UED1.
VHS : 21h00. **VHH :** 1h30. **Cours :** 1h30. **TD :** 0h00. **TP :** 0h00.
VHS travail personnel : 0h00. **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 liées 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.

Liste des matières proposées pour le choix (Une matière par semestre).	
Catégorie : ''Sciences et culture''.	**Catégorie : ''Technique''.**
- Informatique verte (Green IT). - Gouvernance et transformation digitale. - L'impact des technologies sur le travail et les relations sociales. - Les technologies émergentes dans les services publics. - Biologie des systèmes et modélisation informatique. - Philosophie des sciences et de la technique. - Systèmes d'information environnementaux. - Droit du Numérique et Protection des Données (RGPD). - Psychologie cognitive.	- Informatique fondamentale. - Science des Données. - Ingénierie des systèmes d'information. - Intelligence Artificielle. - Sécurité Informatique (Cybersécurité). - Réseaux et systèmes distribués. - Systèmes Cyber-Physiques. - Informatique visuelle. - Bio-informatique. - Calcul haute performance. - Informatique quantique.

Mode d'évaluation :

- Examen semestriel en présentiel (100%).

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

Intitulé de la matière : Architectures Logicielles. **Semestre : 2**. **Type :** UEF21.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **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**
   - Framework 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 :

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

Références bibliographiques :

1. Richards, M. (2020). Software Architecture Patterns. O'Reilly Media.
2. Len Bass, Paul Clements, Rick Kazman (2003). Software Architecture in Practice. Addison-Wesley.
3. 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.
4. Jacques Printz (2009). Architecture logicielle : Concevoir des applications simples, sûres et adaptables. Editions Dunod.
5. Chris Richardson (2018). Microservices Patterns: With examples in Java. Manning Publisher.
6. M. Lankhorst (2009). Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer.
7. Robert C. Martin (1988). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.
8. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley.
9. Paul Clements, Rick Kazman, John Klein (2003). Documenting Software Architectures: Views and beyond. Addison-Wesley.
10. B. Bruller (2003). Architectures de Systèmes d'Information Modèles, services et protocoles. Editions Vuibert.
11. Martin Fowler (2003). Patterns of Enterprise Application Architecture. Addison-Wesley.

Intitulé de la matière : Architectures Orientées Services. **Semestre :** 2. **Type :** UEF21.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **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 micro services.
- 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 micro services.
- 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 micro services.

Connaissances préalables recommandées : Développement d'applications web et 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 micro services**
   - Définition et principes fondamentaux des micro services.
   - SOA Vs micro services.
   - Avantages et inconvénients des micro services.
   - Principes de conception des micro services : Autonomie, découplage, scalabilité, résilience, etc.
   - Composants essentiels d'une architecture de micro services : Discovery, Load balancing, API Gateway, etc.
3. **Technologies et outils pour les micro services**
   - Framework et langages pour le développement de micro services : 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 interservices : REST, messaging, gRPC, etc.
   - Bases de données pour micro services : Bases de données relationnelles, NoSQL, etc.

4. **Conception d'applications basées sur les micro services**
   - Identification des domaines de services.
   - Découpage de l'application en micro services.
   - Conception d'API robustes et performantes.
   - Stratégies de gestion des données : données partagées, cohérence.
   - Développement d'une architecture sécurisée.
5. **Déploiement et gestion des micro services**
   - 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 micro services**
   - 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 :

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

Références bibliographiques :

1. Papazoglou, M. P., & Georgiou, P. (2007). Service-Oriented Computing: Concepts, Technology, and Middleware. MIT Press.
2. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
3. Fowler, M. (2014). Micro services. https://martinfowler.com/articles/microservices.html.
4. Lewis, J., & Fowler, M. (2020). Microservices: From Design to Deployment. https://martinfowler.com/articles/microservices.html.
5. Dragoni, N., Ghezzi, C., & Marinoni, M. (2017). Microservices: A survey of approaches, techniques, and tools. Journal of Systems and Software, 130, 209-226.
6. Leonard Richardson (2007). RESTful Web Services. O'Reilly Media.
7. Thomas Erl (2007). SOA: Principles of Service Design. Prentice Hall.
8. JJ Geewax (2021). API Design Patterns. Manning.

Intitulé de la matière : Systèmes d'Information Coopératifs. **Semestre : 2**. **Type :** UEF22.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 1h30. **TP :** 0h00.
VHS travail personnel : 21h00. **Coefficient :** 3. **Crédit :** 4.

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 et 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 :

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

Références bibliographiques :

1. Alquier & al, (2000). Les systèmes d'information coopératifs : Une approche par les collecticiels.
2. Baghdadi Y. (1997). Contribution méthodologique à la conception des systèmes d'information coopératifs. Thèse de doctorat. Toulouse 1.
3. Nurcan, Selmin (1996). Analyse et conception de systèmes d'information coopératifs. TSI'96.
4. Vincent Couturier (2004). L'ingénierie des systèmes d'information coopératifs par réutilisation : une approche à base de patterns. Thèse de doctorat, Lyon 3.
5. Schmidt, K. (2003). Towards a framework for collaborative systems design. HCI International.
6. Archimag (2023). Gestion Électronique des Documents (GED) et Collaboration : il est temps de passer de la gestion à la collaboration autour des documents.
7. Van Der Aalst, Kees Van Hee (2002). Workflow Management: Models, Methods and Systems. MIT Press.

Intitulé de la matière : Sémantiques Formelles des Langages de Programmation. **Semestre : 2**. **Type :** UEF22.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 1h30. **TP :** 0h00.
VHS travail personnel : 21h00. **Coefficient :** 3. **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.
   - 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.
3. **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.
4. **Vérification de programmes : Introduction à la vérification formelle, Outils et techniques pour la vérification, Exemples de vérification de programmes spécifiques.**
5. **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.
6. **Introduction à la compilation**
   - Processus de compilation.
   - Analyse lexicale, syntaxique et sémantique.
   - Génération de code.
   - Optimisation.

Mode d'évaluation :

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

Références bibliographiques :

1. Pettorossi, A. (2010). Semantics of programming languages. Aracne Editrice.
2. Winskel, G. (1993). The formal semantics of programming languages: an introduction. MIT Press.
3. Winskel, G. (2005). Denotational Semantics. CS Lecture Notes, Cambridge University.
4. Schmidt, D. A. (1994). The structure of typed programming languages. MIT press.
5. Lalement, R. (1990). Logique, réduction, résolution. Editions Masson.

Intitulé de la matière : Systèmes Multi Agents. **Semestre : 2**. **Type :** UEM2.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **Coefficient :** 2. **Crédit :** 4.

Objectifs de l'enseignement : Cette matière 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 :

- Examen semestriel en présentiel (60%).
- Évaluation continue (CC) (40%) : contrôles continus, mini projet.

Références bibliographiques :

1. Wooldridge, M. J. (2009). An introduction to multi agent systems (2nd ed.). John Wiley& Sons.
2. Ferber, J. (1999). Multi-agent systems: An introduction to distributed artificial intelligence. Harlow, UK : Addison-Wesley.
3. Weiss, G. (Ed.). (2013). Multiagent systems: A modern approach to distributed artificial intelligence (2nd ed.). The MIT Press.
4. Shoham, Y., &Leyton-Brown, K. (2009). Multiagents systems: Algorithmic, game-theoretic, and logical foundations. Cambridge, UK: Cambridge University Press.
5. 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.
6. Nicholas R. Jennings, Michael J. Wooldridge (1998). Agent technology. Springer.
7. Eric Bonabeau et Guy Theraulaz (1994). Intelligence Collective. Hermès.

Intitulé de la matière : Apprentissage Automatique. **Semestre : 2**. **Type :** UEM2.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **Coefficient :** 2. **Crédit :** 5.

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 Bayésienne.
   - 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 :

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

Références bibliographiques :

1. Andreas Müller and Sarah Guido (2016). Introduction to Machine Learning with Python.
2. 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.
3. Gullitti, T. et LLC, R.B. (2017). Application Of Machine Learning Algorithms to On-Board Diagnostics (Obd Ii) Threshold Determination.
4. 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.
5. Batta, M. (2018). Machine Learning Algorithms: A Review. https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review.

Intitulé de la matière : **Lean Startup** Semestre : 2 Type : UET21

VHS : 42h **VHH :** 3h00 **Cours :** 1h30 **TD :** **TP :** 1h30

VHS travail personnel : 21h **Coefficient :** 2 **Crédit :** 2

Objectifs de l'enseignement

Ce module initie les étudiants à la méthodologie Lean Startup. Axée sur l'expérimentation rapide, l'apprentissage continu et la prise de décision basée sur les données, cette approche vise à réduire le gaspillage de temps et de ressources. Le cours offre des outils pratiques, particulièrement adaptés aux profils scientifiques et technologiques, pour transformer des idées innovantes ou des résultats de recherche en projets économiques viables. À son issue, les étudiants sauront formuler des hypothèses, concevoir un Produit Minimum Viable (MVP) et aligner l'innovation avec les besoins réels du marché.

Connaissances préalables recommandées

Principes de l'entrepreneuriat ; Fondamentaux de la gestion de projet ;

Contenu de la matière

Cours : 21h

Chapitre 1 : Introduction aux startups

- Définition et caractéristiques d'une startup ;
- Différences entre une entreprise traditionnelle et une startup ;
- Innovation et création de valeur ;
- Particularités des projets technologiques dans un contexte d'incertitude.

Chapitre 2 : La méthodologie Lean Startup

- Origine et évolution du Lean Startup ;
- Principes fondamentaux de la méthode ;
- Cycle d'apprentissage continu : Build – Measure – Learn ;
- Concept de « Validated Learning » ;
- Optimisation des ressources et réduction du gaspillage.

Chapitre 3 : Développement client (Customer Development)

- Méthodologie de Steve Blank ;
- Identification des problèmes réels des clients ;
- Conduite d'entretiens terrain et analyse des résultats ;
- Construction et validation des hypothèses de valeur.

Chapitre 4 : Modèle économique (Business Model)

- Définition et importance du modèle économique ;
- Business Model Canvas ;
- Segmentation des clients ;
- Proposition de valeur ;
- Sources de revenus et structure des coûts.

Chapitre 5 : Développement du MVP

- Concept du Minimum Viable Product (MVP) ;
- Importance du MVP dans la réduction des risques ;
- Différents types de MVP ;
- Applications du MVP dans les projets numériques, scientifiques et technologiques.

Chapitre 6 : Mesure et analyse

- Indicateurs clés de performance (KPIs) ;
- Analyse de la croissance et du comportement des utilisateurs ;
- Modèle AARRR (Acquisition, Activation, Rétention, Recommandation, Revenu) ;
- Prise de décision fondée sur les données.

Chapitre 7 : Adaptation stratégique et prise de décision

- Concepts de Pivot et Persevere ;
- Quand changer de direction stratégique ?
- Différents types de pivots ;
- Analyse d'études de cas internationales.

Chapitre 8 : Financement et stratégies de croissance

- Sources de financement en phase de démarrage ;
- Business Angels et Capital-risque ;
- Stratégies de croissance rapide ;
- Fondamentaux du marketing et de la croissance des startups.

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

- **Examen semestriel en présentiel (20%)**.
- **Évaluation continue (CC) (80%) :** contrôles continus, travail personnel.

Références bibliographiques

- **Ries, E. (2011).** *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.
- **Blank, S., & Dorf, B. (2012).** *The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company*. K&S Ranch Publishing.
- **Osterwalder, A., Pigneur, Y., Bernarda, G., & Smith, A. (2014).** *Value Proposition Design: How to Create Products and Services Customers Want*. John Wiley & Sons.
- **Osterwalder, A., & Pigneur, Y. (2010).** *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons.

Intitulé de la matière : Une Matière au Choix. **Semestre :** 2. **Type :** UED2.
VHS : 21h00. **VHH :** 1h30. **Cours :** 1h30. **TD :** 0h00. **TP :** 0h00.
VHS travail personnel : 00h00. **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 liées 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.

Liste des matières proposées pour le choix (Une matière par semestre).	
Catégorie : ''Sciences et culture''.	**Catégorie : ''Technique''.**
- Informatique verte (Green IT). - Gouvernance et transformation digitale. - L'impact des technologies sur le travail et les relations sociales. - Les technologies émergentes dans les services publics. - Biologie des systèmes et modélisation informatique. - Philosophie des sciences et de la technique. - Systèmes d'information environnementaux. - Droit du Numérique et Protection des Données (RGPD). - Psychologie cognitive.	- Informatique fondamentale. - Ingénierie des Systèmes d'Informations. - Intelligence Artificielle. - Science des données. - Sécurité Informatique (Cybersécurité). - Réseaux et systèmes distribués. - Systèmes Cyber-Physiques. - Informatique visuelle. - Bio-informatique. - Calcul haute performance. - Informatique quantique.

Mode d'évaluation :

- Examen semestriel en présentiel (100%).

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

Intitulé de la matière : Assurance Qualité & Tests des Logiciels. **Semestre :** 3. **Type :** UEF31.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **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 et 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 applications.
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 : Sélénium, JUnit, etc. et de leur utilisation.

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

Références bibliographiques :

1. Pressman, Roger S., & Maxim, Bruce R. (2025). Software engineering: a practitioner's approach. 7th edition, McGraw-Hill.
2. Ian Sommerville (2016). Software engineering. Pearson Education Limited.
3. Stephen H.Kan (2010). Metrics and Models in Software Quality Engineering. 2nd Edition. Addison-Wesley Professional, ISBN-10 : 0201729156.
4. Linda Westfall (2009). The Certified Software Quality Engineer Handbook. Quality Press, ISBN-10 : 0873897307.
5. Murali Chemuturi (2010). Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers. J. Ross Publishing. ISBN-10 : 1604270322.
6. Myers, Glenford J., Sandler, Corey, Badgett, Tom, & Thomas, Todd M. (2004). The art of software testing. 2nd edition, John Wiley & Sons.
7. Kaner, Cem, Bach, James, & Pettichord, Bret (2002). Lessons learned in software testing, a context-driven approach. John Wiley & Sons.
8. Beizer, Boris (1990). Software testing techniques. Van Nostrand Reinhold.
9. Pezzè, Mauro, & Young, Michal (2008). Software testing and analysis: process, principles, and techniques. John Wiley & Sons.

Intitulé de la matière : DevOps & Maintenance des Logiciels. **Semestre :** 3. **Type :** UEF31.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **Coefficient :** 3. **Crédit :** 4.

Objectifs de l'enseignement : Cette matière 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 :

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

Références bibliographiques :

1. Pressman, Roger S., & Maxim, Bruce R. (2025). Software engineering: A practitioner's approach. 7th edition, McGraw-Hill.
2. Gene Kim, Kevin Behr, George Spafford (2018). The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win. IT Revolution Press.
3. Jez Humble, David Farley (2010). Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison Wesley.
4. Scott Chacon and Ben Straub (2014). Git Pro. 2nd edition. A press publisher.
5. John Ferguson Smart (year). Jenkins: The Definitive Guide.
6. Diomidis Spinellis (2016). Effective Debugging: 66 Specific Ways to Debug Software and Hardware. Addison-Wesley.
7. Martin Fowler (2018). Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional.
8. Gene Kim, Jez Humble, Patrick Debois, John Willis (2016). The DevOps Handbook: How to Simplify Engineering and Operations to Maximize Business Value. IT Revolution.
9. Ken Schwaber (2004). Agile Project Management with Scrum. Microsoft Press.

Intitulé de la matière : Ingénierie Dirigée par les Modèles. **Semestre :** 3. **Type :** UEF32.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 0h00. **TP :** 1h30.
VHS travail personnel : 21h00. **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 approche est vue comme une démarche prometteuse à la fois dans la garantie des qualités de service du produit logiciel que dans la maîtrise du processus de développement et de maintenance. Par ailleurs, l'ingénierie dirigée par les modèles vise 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 et compilation.

Contenu de la matière :
Cours : 21h.

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).
2. **L'initiative MDA du consortium OMG**
   - Principes de l'architecture MDA.
   - Standards de l'OMG (UML, OCL, MOF, Corba, XML-XMI, QVT, CWM).
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).
4. **Transformation des modèles et génération du code**
   - Principe du mapping entre modèles.
   - Caractéristiques des transformations.
   - Modélisation des transformations.
   - Traçabilité des transformations.
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).
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 :

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

Références bibliographiques

1. Kleppe, A., Warmer, J., &Basten, W. (2003). MDA explained: The model driven architecture: Practice and promise. Addison-Wesley Professional.
2. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S., & Jungmayr, S. J. (2006). Model-driven software development: Technology, engineering, management. John Wiley & Sons.
3. Booch, G., Rumbaugh, J., & Jacobson, I. (1999). The unified modeling language user guide. Addison-Wesley Professional.
4. Fowler, M. (2003). UML distilled: A brief guide to the standard object modeling language. Addison-Wesley Professional.
5. 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.
6. Holt, J., & Perry, S. (2019). SysML for systems engineering: A model-based approach. Institution of Engineering and Technology.
7. Abrial, J.-R. (1996). The B-book: Assigning programs to meanings. Cambridge University Press.

Ressources logicielles :

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

Intitulé de la matière : Spécification et Vérification Formelle. **Semestre :** 3. **Type :** UEF32.
VHS : 63h00. **VHH :** 4h30. **Cours :** 1h30. **TD :** 1h30. **TP :** 1h30.
VHS travail personnel : 42h00. **Coefficient :** 3. **Crédit :** 4.

Objectifs de l'enseignement : Les compétences développées dans le cadre de cette matière 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 et 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 :

- Examen semestriel en présentiel (60%).
- Évaluation continue (CC) (40%) : Contrôle continu, mini projet.

Références bibliographiques :

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

Intitulé de la matière : Modélisation et Evaluation des Performances des Systèmes. **Semestre :** 3. **Type :** UEM3.
VHS : 42h00. **VHH :** 3h00. **Cours :** 1h30. **TD :** 1h30. **TP :** 0h00.
VHS travail personnel : 21h00. **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 multiserveurs.
   - Files d'attente à capacité limitée.
6. **Les réseaux de Petri**
7. **Les réseaux de Petri stochastiques**

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

Références bibliographiques :

1. Douc, R., Moulines, E., Priouret, P., & Soulier, P. (2018). Markov Chains. Springer.
2. Choquet-Geniet, A. (2006). Les réseaux de Petri : Un outil de modélisation : Cours et exercices corrigés - Licence, Master, écoles d'ingénieur. Dunod.
3. Baynat, B. (2000). Théorie des files d'attente. Hermes.

Intitulé de la matière : Analyse des Données Massives. **Semestre :** 3. **Type :** UEM3.
VHS : 84h00. **VHH :** 6h00. **Cours :** 3h00. **TD :** 1h30. **TP : 1**h30.
VHS travail personnel : 42h00. **Coefficient :** 2. **Crédit :** 5.

Objectifs de l'enseignement : Cette matière 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, Apprentissage Automatique, statistiques et 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 :

- 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 :

1. Venkat Ankam (2016). Big data analytics. Packt Publishing.
2. Viktor Mayer-Schönberger et Kenneth Cukier (2013). Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt publisher.
3. Kaisler S, Armour F, Espinosa J (2013) Big data: issues and challenges moving forward, Wailea, Maui, HI, s.n, pp 995–1004.
4. Nathan Marz and James Warren (2015). Big Data: Principles and best practices of scalable realtime data systems. Manning Publisher.
5. 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.
6. Tom White (2012). Hadoop: The Definitive Guide. Yahoo Press.
7. 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.
8. Matei Zaharia, Bill Chambers (2018). Spark: The Definitive Guide. O'Reilly Media publisher.
9. Cole Nussbaumer Knaflic (2015). Storytelling with Data: A Data Visualization Guide for Business Professionals. Wiley publisher.
10. Trevor Hastie, Robert Tibshirani et Jerome Friedman (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd edition, Springer.
11. Edward R. Tufte (1997). The Visual Display of Quantitative Information. Graphics Pr publisher.
12. Aurélien Géron (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. 2nd edition. O'Reilly Media publisher.

Intitulé de la matière : Méthodologie de Recherche. **Semestre :** 3. **Type :** UET3.
VHS : 21h00. **VHH :** 1h30. **Cours :** 1h30. **TD :** 0h00. **TP :** 0h00.
VHS travail personnel : 00h00. **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 et 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 styles des références bibliographiques (APA, IEEE, 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 :

- Examen semestriel en présentiel (100%).

Références bibliographiques :

1. Beaud, Michel (2020). L'art de la thèse. La Découverte.
2. Stefan Kottwitz (2021). LaTeX Beginner's Guide: Create visually appealing texts, articles, and books for business and science using LaTeX. 2nd Edition, Packt Publishing.
3. Jean-Marie Dubois (2005). La rédaction scientifique : mémoires et thèses : formes régulières et par articles. Estem.
4. Michèle Lenoble-Pinson (1996). La rédaction scientifique : conception, rédaction, présentation. Signalétique, De Boeck Université.
5. Christine Gérard, Jean Germain (1985). Recherche bibliographique et documentaire : généralités. Faculté de philosophie et Lettres.

Intitulé de la matière : Une Matière au Choix. **Semestre :** 3. **Type :** UED3.
VHS : 21h00. **VHH :** 1h30. **Cours :** 1h30. **TD :** 0h00. **TP :** 0h00.
VHS travail personnel : 0h00. **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 liées 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.

Liste des matières proposées pour le choix (Une matière par semestre).	
Catégorie : ''Sciences et culture''.	**Catégorie : ''Technique''.**
- Informatique verte (Green IT). - Gouvernance et transformation digitale. - L'impact des technologies sur le travail et les relations sociales. - Les technologies émergentes dans les services publics. - Biologie des systèmes et modélisation informatique. - Philosophie des sciences et de la technique. - Systèmes d'information environnementaux. - Droit du Numérique et Protection des Données (RGPD). - Psychologie cognitive.	- Informatique fondamentale. - Science des Données. - Ingénierie des systèmes d'information. - Intelligence Artificielle. - Sécurité Informatique (Cybersécurité). - Réseaux et systèmes distribués. - Systèmes Cyber-Physiques. - Informatique visuelle. - Bio-informatique. - Calcul haute performance. - Informatique quantique.

Mode d'évaluation :

- Examen semestriel en présentiel (100%).

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

IV- Accords ou conventions.

Oui / Non.

(Si oui, transmettre les accords et/ou les conventions dans le dossier papier de la formation).

LETTRE D'INTENTION TYPE

(En cas de master coparrainé par un autre établissement universitaire)

(Papier officiel à l'entête de l'établissement universitaire concerné)

Objet : Approbation du coparrainage du master intitulé :

Par la présente, l'université (ou le centre universitaire) déclare coparrainer le master ci-dessus mentionné durant toute la période d'habilitation de ce master.

A cet effet, l'université (ou le centre universitaire) assistera ce projet en :

- Donnant son point de vue dans l'élaboration et à la mise à jour des programmes d'enseignement,
- Participant à des séminaires organisés à cet effet,
- En participant aux jurys de soutenance,
- En œuvrant à la mutualisation des moyens humains et matériels.

SIGNATURE de la personne légalement autorisée :

FONCTION :

Date :

LETTRE D'INTENTION TYPE

(En cas de master en collaboration avec une entreprise du secteur utilisateur)

(Papier officiel à l'entête de l'entreprise)

OBJET : Approbation du projet de lancement d'une formation de master intitulé :

Dispensé à :

Par la présente, l'entreprise déclare sa volonté de manifester son accompagnement à cette formation en qualité d'utilisateur potentiel du produit.

A cet effet, nous confirmons notre adhésion à ce projet et notre rôle consistera à :

- Donner notre point de vue dans l'élaboration et à la mise à jour des programmes d'enseignement,
- Participer à des séminaires organisés à cet effet,
- Participer aux jurys de soutenance,
- Faciliter autant que possible l'accueil de stagiaires soit dans le cadre de mémoires de fin d'études, soit dans le cadre de projets tuteurés.

Les moyens nécessaires à l'exécution des tâches qui nous incombent pour la réalisation de ces objectifs seront mis en œuvre sur le plan matériel et humain.

Monsieur (ou Madame)…….est désigné(e) comme coordonnateur externe de ce projet.

SIGNATURE de la personne légalement autorisée :

FONCTION :

Date :

CACHET OFFICIEL ou SCEAU DE L'ENTREPRISE

V- Avis et Visa des organes Administratifs et Consultatifs.

Intitulé du Master : Génie Logiciel.

Chef de Département	Responsable de l'Equipe de Domaine
Date et visa	Date et visa
Comité Scientifique de Département	
Date et visa	
Conseil Scientifique de la Faculté	
Date et visa	
Doyen de la Faculté	
Date et visa	
Chef d'Etablissement Universitaire	
Date et visa	

VI- Avis et Visa de la Conférence Régionale.
(Uniquement dans la version définitive transmise au MESRS)

VII- Avis et Visa du Comité Pédagogique National de Domaine.
(Uniquement dans la version définitive transmise au MESRS).