UNIVERSITÉ DE SHERBROOKE

Département d'informatique

INF 749 — Conception de systèmes temps réel

Plan de cours
Hiver 2024

Enseignant :

Patrice Roy
Courriel : Patrice.Roy@USherbrooke.ca
Local : L1-15741
Site : http://h-deb.clg.qc.ca/UdeS/STR/
Disponibilité : avant le cours, mardi soir, ou par courriel

Horaire :

Exposé magistral :	Mardi	18h30 à 21h30	L1-6660

Description officielle de l'activité pédagogique[1]

Cibles de formation :	Connaître et repérer les problèmes inhérents au développement de systèmes temps réel; connaître et appliquer le traitement du temps au niveau des systèmes informatiques; spécifier, concevoir, programmer et vérifier des systèmes temps réel.
Contenu :	Types de systèmes temps réel. Représentation du temps, contraintes de temps, horloge, synchronisation d'horloges. Formalismes utilisés dans la spécification de systèmes temps réel : machines à états, *statecharts*, réseaux de Petri, Grafcet. Approche axiomatique de spécification de contraintes temporelles. Architecture des systèmes temps réel. Acquisition et traitement de l'information en temps réel. Modèles utilisés dans la conception de systèmes temps réel : modèles basés sur les évènements, modèles basés sur les graphes, modèles des tâches, modèles des processus, modèles du contrôle. Programmation d'applications
Crédits	3
Organisation	3 heures d'exposé magistral par semaine 6 heures de travail personnel par semaine
Particularités	Aucune

[1] https://www.usherbrooke.ca/admission/fiches-cours/inf749/

1 Présentation

En vous inscrivant au cours INF749, vous prenez un risque, au sens où ce cours porte une signature atypique. Il porte sur un créneau méconnu et mal compris du monde de la programmation, et mêle l'abstraction de haut niveau aux détails très concrets, très près de la machine, le tout en visant l'atteinte d'un objectif que plusieurs considéreront secondaire… et que d'autres, moins nombreux mais pas moins importants, considéreront essentiel.

Ce cours présume chez l'étudiant(e) une connaissance de la programmation impérative / procédurale, de la programmation orientée objet (OO), de même qu'une solide base avec des langages tels que C, C++ et Java, quand bien même ce ne serait que pour faciliter l'accès à la syntaxe des exemples qui seront proposés.

Aussi, bien que ce cours ne soit porte pas strictement sur les systèmes « à haute performance », cette perspective ne nous sera pas étrangère.

Enfin, le cours présume chez l'étudiant(e) des aptitudes analytiques se prêtant à la spécialisation et à l'optimisation de processus. Par nature, ce cours sera régulièrement porté vers la définition de contraintes de performance et vers la mesure du respect de ces contraintes.

Il est hautement probable que ces différents a priori soient acquis par chacune et chacun d'entre vous, mais personne n'est parfait. Pour les travaux, envisagez fortement vous adjoindre des coéquipières et des coéquipiers ayant des atouts qui seront complémentaires aux vôtres. Cela activera la discussion.

Oui, mais j'ai déjà vu cela…

Il y aura inévitablement des éléments de votre formation, dans ce cours comme ailleurs, qui recouperont votre vécu académique (ou professionnel, dû à des stages ou à une expérience de travail antérieure).

De même, dû à des variations selon les parcours académiques et professionnels de chacun(e), certain(e)s parmi vous trouveront évident ce que d'autres estimeront lourd (ou cauchemardesque).

Sachant cela, mettons de l'eau dans notre vin (pas trop, quand même, pour qu'il garde son cachet) et soyons tolérant(e)s les un(e)s envers les autres.

1.1 Mise en contexte

Les systèmes temps réel (STR) constituent un sujet à la fois important et méconnu. Assimilés, souvent à tort, à l'idée de systèmes très rapides, ou à celle d'interface-personne/machine répondant instantanément aux demandes d'un usager, les STR sont perçus par le grand public à travers une lorgnette floue, inexacte.

Pourtant, un STR n'est pas un jouet : on attend d'un STR à ce qu'il respecte certaines contraintes de « performance », la « performance » étant exprimée en temps d'exécution de tâches précises, et ce même dans le pire cas. Un STR n'est pas tant un système rapide globalement qu'un système en tout temps assez rapide (ou jamais trop lent, ce qui revient au même).

Parfois, la contrainte temps réel (TR) à respecter sera simple à exprimer : une pulsation au pire chaque nième quantum de temps; une écriture sur disque d'une quantité donnée d'information à chaque seconde dans le pire cas; au minimum six lectures de touches au clavier par seconde; un rafraîchissement complet d'écran au moins chaque soixantième de seconde; etc. Évidemment, exprimer et garantir sont deux verbes de nature bien différente.

La conception des STR constitue un sujet plus complexe qu'il n'y paraît sur les plateformes contemporaines, du fait que la majorité d'entre elles sont multiprogrammées. Il est en général impossible d'offrir des garanties assez fortes pour qu'un système puisse être qualifié de STR sur une plateforme comme Microsoft Windows ou Linux, par exemple, ce qui ne signifie pas que les plateformes ne permettent pas l'écriture de code « performant » dans l'ensemble. Ceci met en relief la distinction entre TR souple, qui rejoint l'idée que se fait la majorité des gens des STR (quelque chose d'instantané en apparence), et TR strict, qui est d'un autre ordre : le non-respect de contraintes TR souples entraîne un désagrément, mais le non-respect de contraintes TR strictes peut entraîner des bris matériels ou des blessures.

Le mode des STR est un monde de praticiens, même chez les théoriciens du domaine. Règle générale, un STR correspond à un besoin réel, défini de manière précise et mesurable. Un STR, surtout s'il est strict, a pour particularité que si ses contraintes ne sont pas rencontrées, le fruit de son travail n'est pas valide. Les résultats de l'exécution d'un STR dépendent autant de la validité de ses calculs que du respect de ses contraintes de temps. C'est d'ailleurs la définition canonique, en quelque sorte, des STR; nous explorerons aussi d'autres définitions, d'autres visions du sens à accorder au terme STR.

Enfin, les STR ont ceci de particulier qu'ils s'inscrivent systématiquement dans une démarche architecturale : une modification à un endroit, par exemple pour fins d'optimisation, peut entraîner un non-respect des contraintes ailleurs dans le même système.

Optiques retenues

La tradition des STR est associée de près à celle des systèmes à haute performance, même s'il n'existe pas d'adéquation stricte entre les deux mondes. Certaines tâches TR (souples) peuvent être réalisées sur des plateformes grand public (Microsoft Windows, une distribution conventionnelle de Linux, peu importe), mais d'autres tâches TR (strictes) requièrent un véritable SETR. Nous partagerons notre temps entre ces deux types de plateformes.

Il est presque impossible de discuter de systèmes contemporains complexes sans adresser la question de la multiprogrammation. Nous devrons donc lui accorder une place de choix.

Le monde de l'informatique contemporaine accorde une place importante aux systèmes pris en charge, qui sont intrinsèquement indéterministes sur le plan « performance » (ces systèmes utilisent habituellement des moteurs de collecte automatique d'ordures et d'autres mécanismes qui appauvrissent le contrôle que nous avons sur leur profil d'exécution). Nous prendrons soin d'examiner comment une plateforme comme Java peut prétendre se décliner sous une forme TR, et à quel prix.

1.2 Cibles de formation spécifiques

À la fin de cette activité pédagogique, l'étudiante ou l'étudiant sera capable :

1. De distinguer un STR d'un système qui ne l'est pas
2. D'expliquer les caractéristiques inhérentes à un STR
3. De définir la spécification de contraintes TR
4. De mesurer le respect de contraintes TR
5. De développer un module d'un STR
6. D'utiliser des primitives d'un système d'exploitation lors de la programmation de STR

Plusieurs objectifs plus contextuels seront rencontrés en chemin : penser un programme dans un contexte de ressources limitées (les systèmes embarqués, par exemple, qui sont sujets à être soumis à des contraintes TR); modifier un moteur de collecte d'ordures (typique des STR sur plateforme prise en charge); appliquer des techniques d'optimisation locales et systémiques; développer des systèmes dans une optique de résilience et de tolérance aux pannes; etc.

1.3 Contenu détaillé

Le cours sera exigeant du point de vue de la charge de travail (certaines idées s'assimilent mieux par la pratique que par osmose). Des livrables seront exigés en chemin. Le cours comprenant un volet technique important, l'étudiant(e) devra mettre la main à la pâte et y aller d'un effort pratique.

Nous aborderons les STR sous plusieurs angles (concepts, façons de faire, comparatifs, technologiques), ce qui pourra donner une impression de désorganisation par moment (ne vous en faites pas, cependant) mais permettra un tour d'horizon plus complet.

Thème	Contenu	Heures	Objectifs	Travaux[2]
1	**Première approche** Exploration des divers sens de : temps, temps réel Historique, problèmes et définitions Clarification de ce qu'est (ou non) un STR STR et déterminisme STR et sécurité	3	1-2	
2	**Complexité et mesure** Brève exploration : des conteneurs standards; des itérateurs ; des algorithmes standards; des garanties de complexité. Complexité vs temps. Complexité en taille ou en espace Conteneurs maison ; itérateurs maison ; adaptateurs Outils de mesure portables ; outils de mesure non portables ; approches RAII Risques et bénéfices Quoi mesurer	3	2-5	
3	**Contraintes** Contraintes strictes ou souples Contraintes locales ou systémiques Conséquences d'un non-respect Respect préventif (externe) Respect disciplinaire (interne) Scrutation Interruptions forcées Respect et matériel Respect local vs contraintes d'ensemble	3	1-4	

2 Normalement, cette colonne devrait indiquer les sujets qui devraient obligatoirement faire l'objet d'un travail pratique ou d'un projet; cependant, dans ce cours, les règles du travail de session et des livrables qui le constituent sont décrites dans un document annexe nommé « STR – Consignes quant au projet »

Thème	Contenu	Heures	Objectifs	Travaux[2]
4	**Multiprogrammation** Fils d'exécution vs processus (bref) Implémentations *Win32* et POSIX std::thread et ses variantes Fonctions résumables et enjeux `TR` Synchronisation et prévisibilité Mémoire partagée Algorithmes sans verrous et progression Vision générale ou disciplinaire Systèmes multi cœurs Déterminisme et multiprogrammation	6	1-6	
5	**Enjeux propres aux SETR** Environnement de développement Outils de synchronisation Mécanismes et métaphores de communication Processus et états Ordonnanceur et commutation Tests d'ordonnancement Gestionnaire d'interruption Variantes : autres SETR; exécutifs TR; solutions maison	6	1-6	
6	**Entrées/ sorties** Contraintes TR et E/S Traitement de signal TR Approches non bloquantes Approches multiprogrammées Impact systémique des E/S E/S très rapides Vitesse locale ou équilibre systémique Équilibre et espace	3	1-6	
7	**Idiomes et schémas de conception** Relation et implémentation d'idiomes et de schémas de conception aux STR : singleton; observateur; pImpl; *proxy*; ordonnanceur; *etc*.	3	1;5	
8	**Gestion de la mémoire** Allouer ou construire Déclinaisons de new et de delete Surcharge de new et delete Allocation positionnelle Enjeux d'alignement Allocation assistée Arénas Allocateurs	6	3-4	

Thème	Contenu	Heures	Objectifs	Travaux[2]
9	**Exécutifs** Contexte (disciplinaire) Avantages et risques Concevoir un exécutif Droits et obligations Démonstration détaillée	3	1;3-4	
10	**Systèmes pris en charge** Déterminisme « Performance » Java TR Collecte automatique d'ordures État de la recherche	6	1-6	
11	Systèmes embarqués (si le temps le permet)	3		

2 Organisation

L'organisation du cours est décrite dans les sections qui suivent.

2.1 Méthode pédagogique

Pour favoriser l'intégration des nombreux concepts au menu, nous suivrons essentiellement le modèle suivant :

- Exposés magistraux en classe, où les étudiant(e)s sont fortement encouragés à contribuer par leurs questions et commentaires
- Travaux pratiques et exercices à teneur formative, qui permettront aux étudiant(e)s de mesurer concrètement leur compréhension de la matière explorée, et qui pourront être corrigés par le professeur ou autocorrigés à l'aide d'une grille de vérification, selon le cas
- Travaux pratiques à teneur sommative, évalués par le professeur en fonction des mêmes critères que ceux appliqués dans le cadre des activités formatives. Ces travaux seront soumis à un échéancier de livrables (voir le document Consignes quant au projet, ci-joint)
- Des questions de réflexion (et parfois à saveur technique) sur une base quasi hebdomadaire, et
- Un contrôle théorique récapitulatif, en toute fin de parcours, vérifiant formellement l'atteinte des objectifs

Certains aspects plus généraux du développement informatique seront abordés au passage : par exemple, nous chercherons non seulement à rédiger des modules réalisant le travail demandé, mais aussi du code à la fois performant et portable.

La portabilité du code peut sembler être un peu « hors champ » pour un cours de STR, mais il faut comprendre que les programmes TR tendent à être déployés sur des plateformes spécialisées. Pour cette raison, maximiser la portabilité et circonscrire ce qui est spécifique à une plateforme ou une autre est un atout.

Les questions et contrôles présumeront que chaque membre d'une équipe a contribué activement à la réalisation de chacun des travaux pratiques, et a bien compris les implications philosophiques et techniques de ces travaux.

Le SETR que nous utiliserons, QNX Neutrino, est un système essentiellement conforme à la norme POSIX ce qui suggère qu'une forte familiarité avec les langages C et C++ soit un atout.

2.2 Calendrier approximatif du cours

La planification hebdomadaire ci-dessous se veut un aperçu plus précis du cours tel que planifié. Cette planification se veut souple : selon le rythme que nous parviendrons à maintenir, des débordements seront possibles d'un cours à l'autre, et l'ordre proposé pour les thématiques se veut plus indicatif que contraignant. Il est possible aussi que les questions en classe nous amènent à changer nos plans à l'occasion.

La colonne Séance indique le numéro de la séance. Ces numéros vont de S00 à S14 inclusivement, pour un total (dans un monde idéal) de 15 séances, ce qui représente une approximation du déroulement de la session. Un résumé visuel du contenu prévu pour la séance apparaît aussi à cet endroit, incluant une mention indiquant l'intention d'aller ou non au laboratoire en cours de séance.

La colonne Contenu décrit brièvement le contenu prévu pour cette séance.

La colonne Objectifs indique auxquels des objectifs spécifiques du cours, présentés plus haut, la séance sera principalement arrimée. La notation suit la convention selon laquelle le trait d'union indique un intervalle et le point-virgule indique une énumération. Ainsi :

- L'écriture 1-3 signifie de 1 à 3 inclusivement; et
- L'écriture 4-7;9 signifie de 4 à 7 inclusivement de même que 9.

Plusieurs séances couvriront, à leur façon, l'ensemble des objectifs généraux du cours. Ce n'est pas un accident.

Consultez le document décrivant les consignes quant au projet de ce cours, qui est joint au plan de cours, pour bien organiser votre temps.

Semaine[3]	Date	**Thème**	**Contenu**
1	10 janvier	1 – Première approche 2 – Complexité et mesure	Où nous définirons les termes et concepts qui nous occuperont le reste de la session, en particulier celui du temps.
2	17 janvier	2 – Complexité et mesure	Nous mettrons en place quelques brèves bases de programmation efficace à partir d'outils standards et nous discuterons de stratégies de mesure du temps écoulé. *Ces séances seront fortement axées sur la programmation en C++*.
3	24 janvier	3 – Contraintes	Qui dit STR dit respect de contraintes de performance. Nous prendrons donc soin d'examiner la question des contraintes.
4	31 février	4 – Multiprogrammation	Les systèmes multiprogrammés peuvent être moins déterministes que leurs contreparties monoprogrammées. On ne pourrait discuter de STR sans aborder cette question.
5	7 février	5 – Enjeux propres aux SETR	Nous examinerons plus en détail un SETR particulier : métaphores de programmation et de multiprogrammation, mécanismes de synchronisation et de communication, outils de mesure, approches synchrones et asynchrones, *etc*.
6	14 février	5 – Enjeux propres aux SETR	
7	21 février	6 – Entrées/sorties	Les entrées/ sorties, comme la multiprogrammation et le matériel, compliquent la question du respect des contraintes TR. Nous nous pencherons donc sur cette question.
8	28 février	7 –Idiomes et schémas de conception	Les schémas de conception et les idiomes de programmation sont des outils tout à fait désirables et souhaitables. Cependant, la présence de contraintes TR influencera la gamme de choix disponibles et les conséquences de ces choix.
9	7 mars	8 – Gestion de la mémoire	Plusieurs STR interdisent le recours à l'allocation dynamique de mémoire, mais il est possible de contrôler finement cette mécanique. Nous verrons comment.
10	14 mars	8 – Gestion de la mémoire	
11	21 mars	9 – Exécutifs	

3 Notez que les semaines sont numérotées à partir de 1 dans le plan de cours pour respecter le format départemental, mais tous les documents du cours (de même que le site Web) suivront un format où les numéros de séance débutent à zéro.

Semaine[3]	Date	**Thème**	**Contenu**
12	28 mars	9 – Exécutifs	Nous examinerons une approche pour réaliser un STR malgré un ensemble important de contraintes de complexité systémique.
13	4 avril	10 – Systèmes pris en charge	Les plateformes et les langages pris en charge ont la cote aujourd'hui mais se prêtent mal, *a priori*, au développement d'un STR. Beaucoup de recherche se fait pour faciliter le pont entre les deux.
14	11 avril	5 – Enjeux propres aux SETR	Examen d'enjeux propres aux ordonnanceurs et à la commutation de tâches, de même que de tests d'ordonnancement
15	18 avril	Finaux	Examen final

Certains aménagements seront possibles en cours de session, par exemple pour explorer du côté des systèmes embarqués[4].

4 Si le temps le permet.

2.3 Évaluation

Les évaluations sommatives seront réparties et pondérées comme suit.

De petits tests, souvent d'une seule question, auront lieu sur une base relativement régulière, soit une par semaine, et ce, la plupart des semaines. Seuls les huit mieux réussis pour chaque étudiant(e) seront comptabilisés. Chaque question portera sur le thème de la semaine précédente, parfois des deux semaines précédentes. Les questions seront surtout orientées sur la réflexion, mais la technique à proprement dit s'y glissera à l'occasion. Le poids de chacun de ces petits tests sera 5% de la note finale. En général, le temps alloué pour y répondre sera d'environ 15 minutes, au début du cours.	Minitests (40%)
Un projet, dont le format peut vous sembler quelque peu inhabituel, sera composé de travaux pratiques (voir document joint pour l'échéancier, les outils, la constitution des équipes de travail, la philosophie selon laquelle les notes seront attribuées, etc.) et devra être réalisé en cours de session. Dû à la nature du cours, les efforts à consentir pour réussir cette activité seront principalement axés vers la conception, la réalisation concrète et la validation du respect des contraintes TR dans un système. Les modalités d'évaluation et les dates de livraison des livrables sont indiquées dans le document décrivant les consignes quant au projet. Notez que le professeur peut accorder un ajustement de ± 20% en fonction de la qualité du travail.	Projet (30%)
Un examen final récapitulatif valant 30% de la note finale aura lieu lors de la dernière séance de la session.	Examen (30%)

Conformément au règlement facultaire d'évaluation des apprentissages[5], l'enseignant peut retourner à l'étudiante ou à l'étudiant tout travail non conforme aux exigences quant à la qualité de la langue et aux normes de présentation.

Le plagiat consiste à utiliser des résultats obtenus par d'autres personnes afin de les faire passer pour sien et dans le dessein de tromper l'enseignant. Si une preuve de plagiat est attestée, elle sera traitée en conformité, entre autres, avec l'article 9.4.1 du Règlement des études[6] de l'Université de Sherbrooke. L'étudiant ou l'étudiante peut s'exposer à de graves sanctions, dont automatiquement une note de zéro (0) au devoir ou à l'examen en question.
Ceci n'indique pas que vous n'ayez pas le droit de coopérer entre deux équipes tant que la rédaction finale des documents et la création du programme restent le fait de votre équipe. En cas de doute de plagiat, l'enseignant peut demander à l'équipe d'expliquer les notions ou le fonctionnement du code qu'il considère comme étant plagié. En cas de doute, ne pas hésiter à demander conseil et assistance à l'enseignant afin d'éviter toute situation délicate par la suite.

5 https://www.usherbrooke.ca/sciences/fileadmin/sites/sciences/Etudiants_actuels/Informations_academiques_et_reglements/2017-10-27_Reglement_facultaire_-_evaluation_des_apprentissages.pdf

6 https://www.usherbrooke.ca/registraire/droits-et-responsabilites/reglement-des-etudes/

2.4 Échéancier des travaux

L'échéancier des livrables du travail de session suit. Notez que le thème de ce travail est indiqué dès la séance initiale, soit S00, et qu'il en va de même pour les explications de chacun des livrables, ce qui explique que seules les dates de remise soient répétées ci-dessous :

Livrable	Remise
L00	S03
L01	S08
L02	S12

Les consignes détaillées de chaque livrables sont indiquées dans le document Consignes quant au projet, qui accompagne ce plan de cours.

2.5 Utilisation d'appareils électroniques et du courriel

Selon le règlement complémentaire des études, section 4.2.3[7], l'utilisation d'ordinateurs, de cellulaires ou de tablettes pendant une prestation est interdite à condition que leur usage soit explicitement permise dans le plan de cours.

Dans ce cours le règlement 4.2.3 s'applique à moins d'avoir obtenu personnellement l'autorisation du professeur. Cette permission peut être retirée en tout temps, si l'appareil n'est pas utilisé uniquement à des fins d'apprentissage.

Tel qu'indiqué dans le règlement universitaire des études, section 4.2.3[8], toute utilisation d'appareils de captation de la voix ou de l'image exige la permission du professeur.

Note : L'utilisation du courrier électronique est recommandée pour poser vos questions.

7 https://www.usherbrooke.ca/sciences/fileadmin/sites/sciences/documents/Intranet/Informations_academiques/Sciences_Reglement_complementaire_2017-05-09.pdf

8 https://www.usherbrooke.ca/registraire/droits-et-responsabilites/reglement-des-etudes/

3 Matériel pour le cours

Des notes de cours seront mises à votre disposition, et devraient être votre référence principale pour la session qui s'annonce.

Aucun manuel n'est obligatoire. La littérature sur les STR est surtout composée d'articles et de guides accompagnant certains SETR; nous l'utiliserons sous sa forme en ligne, ce qui sera plus pratique pour nous à bien des égards que ne le seraient des imprimés.

Le site Web du cours devrait être, avec les notes de cours en tant que telles, votre référence principale :

http://h-deb.clg.qc.ca/UdeS/STR/

Un ensemble toujours croissant de textes et de liens portant sur les STR peut être consulté aux adresses suivantes :

http://h-deb.clg.qc.ca/Sujets/TempsReel/index.html

http://h-deb.clg.qc.ca/Liens/Temps-Reel--Liens.html

Pour des liens sur le temps en tant que tel, voir :

http://h-deb.clg.qc.ca/Liens/Temps--Liens.html

Sujets connexes :

http://h-deb.clg.qc.ca/Liens/Multiprogrammation--Liens.html

http://h-deb.clg.qc.ca/Liens/Optimisation--Liens.html

Je vous tiendrai au courant, en cours de route, des travaux de SG14, le groupe d'études sur les systèmes à basse latence pour l'évolution du langage C++.

4 Bibliographie

La bibliographie complète pour le cours est disponible sur le site Web.

UNIVERSITÉ DE SHERBROOKE

L'intégrité intellectuelle passe, notamment, par la reconnaissance des sources utilisées.
À l'Université de Sherbrooke, on y veille !

Extrait du *Règlement des études* (Règlement 2575-009)

9.4.1 DÉLITS RELATIFS AUX ÉTUDES

Un délit relatif aux études désigne tout acte trompeur ou toute tentative de commettre un tel acte, quant au rendement scolaire ou une exigence relative à une activité pédagogique, à un programme ou à un parcours libre.

Sont notamment considérés comme un délit relatif aux études les faits suivants :

a) commettre un plagiat, soit faire passer ou tenter de faire passer pour sien, dans une production évaluée, le travail d'une autre personne ou des passages ou des idées tirés de l'œuvre d'autrui (ce qui inclut notamment le fait de ne pas indiquer la source d'une production, d'un passage ou d'une idée tirée de l'œuvre d'autrui) ;
b) commettre un autoplagiat, soit soumettre, sans autorisation préalable, une même production, en tout ou en partie, à plus d'une activité pédagogique ou dans une même activité pédagogique (notamment en cas de reprise) ;
c) usurper l'identité d'une autre personne ou procéder à une substitution de personne lors d'une production évaluée ou de toute autre prestation obligatoire ;
d) fournir ou obtenir toute aide non autorisée, qu'elle soit collective ou individuelle, pour une production faisant l'objet d'une évaluation ;
e) obtenir par vol ou toute autre manœuvre frauduleuse, posséder ou utiliser du matériel de toute forme (incluant le numérique) non autorisé avant ou pendant une production faisant l'objet d'une évaluation ;
f) copier, contrefaire ou falsifier un document pour l'évaluation d'une activité pédagogique ;

[…]

Par plagiat, on entend notamment :

- Copier intégralement une phrase ou un passage d'un livre, d'un article de journal ou de revue, d'une page Web ou de tout autre document en omettant d'en mentionner la source ou de le mettre entre guillemets ;
- reproduire des présentations, des dessins, des photographies, des graphiques, des données… sans en préciser la provenance et, dans certains cas, sans en avoir obtenu la permission de reproduire ;
- utiliser, en tout ou en partie, du matériel sonore, graphique ou visuel, des pages Internet, du code de programme informatique ou des éléments de logiciel, des données ou résultats d'expérimentation ou toute autre information en provenance d'autrui en le faisant passer pour sien ou sans en citer les sources ;
- résumer ou paraphraser l'idée d'un auteur sans en indiquer la source ;
- traduire en partie ou en totalité un texte en omettant d'en mentionner la source ou de le mettre entre guillemets ;
- utiliser le travail d'un autre et le présenter comme sien (et ce, même si cette personne a donné son accord) ;
- acheter un travail sur le Web ou ailleurs et le faire passer pour sien ;
- utiliser sans autorisation le même travail pour deux activités différentes (autoplagiat).

Autrement dit : mentionnez vos sources

Document informatif V.3 (août 2017)