UFR 919 Sciences de l'ingénieur
Master Informatique – Parcours STL
Unité d'enseignement ALASCA

Gestion autonome d'énergie dans des quartiers « intelligents »

Jacques Malenfant
professeur des universités

version 1.0 du 18/09/2024

Résumé

L'objectif de ce projet est de comprendre l'utilisation des concepts, des approches et des techniques de modélisation, de simulation, d'implantation et de génie logiciel pour le développement des systèmes de contrôle cyber-physiques et des systèmes autonomiques. Cet objectif sera poursuivi par la réalisation d'une application visant à gérer automatiquement la production et la consommation d'énergie électrique d'un logement autonome dotée d'équipements numérisés (appareils électriques, capteurs, panneaux solaires, éoliennes, batteries de stockage, ... équipés comme objets de l'internet). Ce projet se place dans le contexte de la transition énergétique par une évolution vers des logements et des quartiers écologiques utilisant essentiellement des énergies renouvelables en auto-consommation.

Rappel des dates et informations importantes (détails à la fin du document)

	Le projet ***doit*** être réalisé en ***Java SE 8***.
27/09/2024	Date limite pour la formation des équipes (2 personnes).
9/10/2024	Audit 1 (10% de la note finale).
10/11/2024	Rendu de code préalable à la soutenance de mi-semestre (format `tgz` ou `zip` *uniquement*).
13/11/2024	Soutenance de mi-semestre (semaine des ER1, 30% de la note finale).
9/01/2025	Audit 2 (10% de la note finale).
2/02/2025	Rendu de code préalable à la soutenance finale (format `tgz` ou `zip` *uniquement*).
5/02/2025	Soutenance finale (semaine des ER2, 50% de la note finale).
1-5/09/2025	Soutenance de seconde session (créneau à définir, note remplaçant entièrement celle de 1re session).

1 Problématique générale

Face à la pollution, aux changements climatiques et aux dangers non encore maîtrisés de l'énergie nucléaire (déchets, accidents, consommation d'eau, ...), l'Union européenne ainsi que de nombreux autres acteurs publics et privés dans le monde ont engagé une double conversion de leurs réseaux de production et distribution d'électricité. Il s'agit :

— de réduire progressivement la part d'électricité produite par des grands équipements polluants (centrales au pétrole, au charbon ou au gaz, centrales nucléaires, ...) pour les remplacer par des unités de production fonctionnant à partir d'énergies renouvelables (soleil, vent, rivières/fleuves, marée, ...) ;
— de rendre « intelligents » (numériser) à la fois les appareils électriques ainsi que les équipements du réseau de production et distribution puis de les relier par réseau informatique

de manière à contrôler de manière beaucoup plus fine, précise et réactive l'équilibre entre production et consommation d'énergie.

Ces changements permettent de parler d'un passage aux *grilles d'énergie* dont le fonctionnement sera bien différent des réseaux de production et distribution d'aujourd'hui. En effet, un réseau électrique ne dispose pas (généralement) de stockage ; il doit donc absolument éviter la surcharge voire la disjonction du réseau en maintenant un niveau instantané de production de puissance électrique [1] supérieur à la consommation. Or, la production d'électricité via les énergies renouvelables est bien plus fluctuante et répartie sur le réseau que celle via les énergies fossiles ou nucléaires. Aujourd'hui, la production est très centralisée et les décisions pour ajouter (démarrage d'une centrale au gaz par exemple) ou retirer de la puissance produite se font sur des prévisions à plusieurs heures voire quelques jours, correspondant aux délais de mise en route et d'arrêt de tels équipements (facilement entre 8 et 24 heures pour des centrales au gaz, par exemple, plus encore pour les centrales au fioul ou au charbon). Ces décisions prises très en amont rendent la fluctuation dans la production invisible et pratiquement sans conséquence pour le consommateur.

Dans le cas des énergies renouvelables, des fluctuations importantes de la puissance produite apparaissent naturellement et ce, à des échelles de temps non plus de plusieurs heures mais de quelques *secondes* (vent fluctuant, temps partiellement nuageux, *etc.*). Ces fluctuations peuvent être amorties à court terme à l'aide d'unités de stockage d'énergie, mais l'équilibre entre puissance consommée et puissance produite doit absolument être retrouvé avant d'épuiser ces unités de stockage. Pour palier ces variations plus fortes et plus rapides dans la production, les grilles d'énergie prévoient que les consommateurs jouent un rôle central dans le maintien de l'équilibre entre production et consommation. Les trois principaux moyens envisagés côté consommateur pour assurer l'équilibre lorsque la consommation dépasse la production sur le réseau national sont :

1. Repousser certaines consommations à plus tard (resp. les avancer), par exemple en empêchant un appareil (chauffage, réfrigérateur, chauffe-eau, ...) de démarrer si la puissance produite est insuffisante, ou en reportant (resp. avançant) le démarrage d'appareils comme le lave-linge ou le four, surtout s'ils sont préprogrammés.
2. Utiliser des ressources locales, soit des unités de production (éoliennes, panneaux solaires, ...), soit des unités de stockage (batteries fixes du logement, batterie d'un véhicule électrique, ...) en auto-consommation pour compenser partiellement et/ou temporairement une production insuffisante du réseau.
3. L'échange d'énergie directement entre consommateurs, certains possédant des unités de production ou de stockage dont il peuvent temporairement céder une partie de la capacité de production.

Bien sûr, il n'est pas réellement envisageable que les consommateurs eux-mêmes prennent des décisions manuellement toute la journée, y compris la nuit, pour gérer leur consommation et leur potentielle production. L'automatisation de ces fonctions sera la clé d'une mise en œuvre réaliste de cette vision du futur. Une solution possible serait de confier à l'opérateur du réseau de distribution le soin de contrôler les appareils des logements via des compteurs « intelligents ».

Une alternative décentralisée consiste plutôt à doter chaque logement d'un contrôleur local, indépendant du compteur électrique, mais capable de « dialoguer » avec le réseau de distribution, de prendre puis d'appliquer des décisions au nom du consommateur à partir de règles de gestion décidées par ce dernier de manière opaque au réseau (pour protéger autant que possible les données personnelles des consommateurs). Notons dans les deux cas que la coopération des équipements est nécessaire, d'où le besoin d'évoluer vers des équipements « intelligents » ou numérisés, connectés souvent inclus dans ce que l'on désigne par « internet des objets ».

2 Gestion d'énergie « intelligente »

Mise en garde préalable : L'objectif du projet n'est pas le réalisme absolu dans la matérialisation des phénomènes cyber-physiques (puissance électrique, évolution de tempéra-

1. La puissance électrique, exprimée en watts ou en ampères (watts = ampères × voltage), est une mesure instantanée. Elle exprime soit une capacité produite par des générateurs soit une capacité consommée par des appareils électriques à un instant donné. L'énergie électrique est le produit d'une puissance par le temps pendant lequel cette puissance est fournie ou consommée. Elle s'exprime souvent en kilowatts∗heures.

ture ambiante du logement ou interne au réfrigérateur, *etc.*) ou des contrôles exercés. Il s'agit principalement de comprendre globalement ces phénomènes et leurs implications. Par contre, une attention importante sera accordée à la bonne conception des composants et de l'architecture logicielle et à la qualité du développement, comme il se doit en développement logiciel, un des principaux métiers auxquels forme STL.

2.1 Équipements intelligents

Pour arriver à automatiser la gestion d'énergie d'un logement, il faudra des équipements électriques se prêtant à un contrôle à distance via réseau. [2] Un appareil de chauffage par exemple pourra offrir un contrôle beaucoup plus fin de sa consommation (en modulant sa puissance courante) que simplement être allumé ou forcé à être éteint en lui coupant le courant par la prise à laquelle il est connecté en permettant par exemple de fixer à distance sa puissance à 500, 1000, 1500 ou 2000 watts.

Dans l'esprit de l'internet des objets, projetons-nous plutôt dans un avenir pas si lointain où les équipements seront pilotables à distance par réseau et ce, en offrant une interface permettant de récupérer des informations sur ses fonctionnalités, sur son état courant ainsi que des opérations permettant de modifier son mode et ses paramètres de fonctionnement. Par exemple, un refrigérateur devrait pouvoir annoncer qu'il possède deux compartiments (froid et congélation), que chacun a une température cible pilotable à distance (généralement 4° pour la partie froid et -18° pour la partie congélation) ainsi que deux températures internes courantes pouvant être lues à distance et un ou deux compresseurs pouvant être mis en attente individuellement. Un autre exemple serait un chauffe-eau dont la puissance consommée peut être modulée à distance.

Dans le contexte économique d'aujourd'hui, les nombreux fabricants d'appareils électriques offriraient vraisemblablement une myriade d'interfaces spécifiques à chacun de leurs appareils. Cela pose un défi majeur pour concevoir et réaliser un gestionnaire d'énergie *générique*, capable de piloter des appareils de différents fabricants. Une solution serait de normaliser les interfaces, mais il n'est pas toujours simple, même en accord avec tous les fabricants, de définir des normes uniques satisfaisantes. D'autant que le pilotage de la consommation d'énergie ne sera pas la seule application de la numérisation des appareils.

Pour mettre en œuvre un gestionnaire générique, il faut extraire les fonctionnalités essentielles de chaque appareil pour en piloter la consommation d'énergie. Dans ce but, nous allons classer les appareils électriques en deux catégories [3] :

1. Les appareils à consommation ajourd'hui *incontrôlable* et qui le resteront vraisemblablement vu leur faible coût, dont on ne peut que subir les effets (sèche-cheveux, ventilateur, ...) selon la volonté de l'utilisateur.
2. Les appareils à consommation *modulables* qui sont souvent en opération permanente mais à consommation intermittente pouvant être suspendue ou modulée pour un temps plus ou moins long (refrigérateur, chauffe-eau, ...). Cette modulation de leur puissance consommée doit respecter des règles selon leurs conséquences (prévisibles) comme des règles de sécurité (par exemple, la température dans le réfrigérateur ne doit pas monter trop haut pour une trop longue durée sinon on prendrait des risques sanitaires ou de perte sur les aliments).

Du point de vue du gestionnaire d'énergie, la première catégorie n'est prise en compte que pour le total de la puissance que ces appareils consomment à tout instant puisqu'ils ne sont pas contrôlables. Ils ne seront donc pas individuellement connectés au gestionnaire ; ce dernier ne connaîtra leur consommation qu'indirectement, par les mesures globales du compteur électrique. Pour contrôler les appareils de la seconde catégorie, nous allons définir une interface *générique* permettant au gestionnaire de piloter n'importe quel appareil modulable, *sans en connaître la nature exacte*, via des opérations génériques comme « suspendre » ou « reprendre » le fonctionnement

2. Il serait possible de se contenter d'équiper le réseau électrique interne du logement de prises contrôlables à distance (par exemple, des prises reliées en BlueTooth permettant de couper le courant à l'équipement qui y est connecté), mais cela ne donnerait qu'un contrôle très grossier et pas toujours correct pour certains équipements qui ne doivent pas être complètement débranchés.

3. Une troisième catégorie d'appareils *planifiables* regroupe ceux qui disposent d'une fonctionnalité de programmation comme le départ différé sur les lave-linges et les lave-vaisselles. Pour simplifier le projet, nous ne traitons pas cette catégorie.

nominal ou « changer de mode » pour passer d'un mode plus consommateur de puissance à un autre moins consommateur ou *vice versa*.

Bien sûr, pour chaque appareil, il faudra qu'une « traduction » s'opère entre ces opérations génériques appelées par le gestionnaire d'énergie et les opérations des interfaces propriétaires offertes par les appareils qui vont exécuter les changements de modalité de fonctionnement correspondantes. Par exemple, l'opération `downMode` faisant passer un appareil d'un mode plus consommateur à un mode moins consommateur pourra se traduire pour un réfrigérateur par l'interdiction de fonctionner pour le compresseur de la partie à 4°, par exemple, alors que cette même opération pour un chauffe-eau pourra se traduire par une réduction de la puissance de chauffage de 50%, par exemple. Nous revenons sur ce point un peu plus loin.

De manière similaire, les unités locales de production d'énergie peuvent être classées dans deux catégories :

1. Les unités de production *aléatoires* (panneaux solaires, éoliennes, ...) qui peuvent produire en permanence mais dont on subit la fluctuation dans le niveau de production en fonction des aléas de l'environnement (ensoleillement, vent, ...) ; ces unités offrent des possibilités d'intervenir uniquement pour des raisons de sécurité ou d'entretien (arrêt des éoliennes pour leur mise en sécurité lorsque le vent est trop fort, par exemple).
2. Les unités de production *intermittentes* (turbine hydro-électrique, génératrices à essence ou à gaz) qui ne produisent pas en permanence mais qui peuvent « à coup sûr » (sauf pannes, ce qu'on va exclure) produire une puissance électrique, dont le niveau sera possiblement contrôlable, pour une durée bornée et prévisible (par exemple, selon le niveau des réservoirs d'eau, de gaz ou d'essence).

Dans le cadre du projet, vous devrez inclure [4] :

1. ***Au moins deux appareils incontrôlables***.
2. ***Au moins deux appareils modulables***.
3. ***Au moins une unité de production aléatoire***.
4. ***Au moins une unité de production intermittente***.

De plus, il sera nécessaire de pouvoir obtenir les niveaux totaux instantanés de puissances électriques consommée et produite du logement. Pour plus de réalisme, cela sera assuré par un ***compteur électrique*** numérique. Une version initiale du composant compteur vous sera fournie dès le début du projet. Aux étapes 2 et 3, il faudra y inclure la simulation et le code nécessaire pour mesurer les puissances instantanées produites et consommées dans le logement et rendre ces mesures accessibles au gestionnaire d'énergie.

Cas particulier : unités de stockage

Les unités de stockage se distinguent par deux vues complémentaires : une vue « fournisseur d'énergie » (déchargement des batteries pour fournir de l'énergie, par exemple), qui se comporte comme une unité de production intermittente (selon le niveau d'énergie restant dans les batteries, par exemple), et une vue « consommateur d'énergie » (chargement des batteries, par exemple) qui se comporte comme un appareil consommant de la puissance électrique.

Votre projet supposera que le logement sera doté de batteries. En tant qu'unité de production, vous poserez comme hypothèse qu'elles sont en permanence connectées au compteur électrique et *automatiquement* sollicitées *dès* que le niveau de puissance produite passe sous le niveau de la puissance consommée, sans qu'il soit nécessaire de les déclencher explicitement. Lorsque les batteries seront en mode production d'énergie, l'objectif du gestionnaire d'énergie sera de réduire au plus vite la puissance totale consommée dans le logement pour éviter d'épuiser complètement les batteries avant de pouvoir les recharger.

4. Tout au long du projet, des exemples vous serons fournis pour vous aider à mieux comprendre ce qu'il faut faire et ce qui est attendu. Dès le départ, vous aurez donc des exemples d'appareils et un début de programmation du projet. Vous devrez inclure ces éléments fournis dans votre projet, mais les éléments demandés ici *s'ajoutent* aux éléments fournis que vous choisirez d'inclure dans votre projet.

En tant qu'appareil consommant de l'énergie lors de la charge, il faudra traiter les batteries comme un *appareil modulable*, c'est à dire que le chargement des batteries est activé en permanence mais ne se fait que de manière intermittente lorsque nécessaire [5], comme le compresseur d'un réfrigérateur est activé en permanence mais fonctionne (et consomme de la puissance) uniquement pendant que la température interne du réfrigérateur demeure au-dessus d'une certaine limite. Le gestionnaire d'énergie peut suspendre le chargement des batteries si la situation exige plutôt de faire fonctionner d'autres appareils modulables et que la puissance produite par les unités de production est insuffisante. Toutefois, en tant qu'appareil modulable, les batteries ne seront pas gérées *via* l'interface générique discutée plus haut car le gestionnaire d'énergie doit prendre des décisions spécifiques qui imposent de savoir qu'il opère sur des batteries et non sur un appareil modulable quelconque.

Un objectif important d'un projet comme celui décrit dans ce cahier des charges serait de *dimensionner* la capacité totale des batteries et des unités de production à installer dans le logement pour éviter avec une forte probabilité (à fixer selon le client) de se voir priver d'électricité à un moment ou un autre étant donnée son profil de consommation prévisible. Pour cela, il faudrait procéder à des simulations sous conditions météorologiques (ensoleillement, vent, *etc.*) réalistes pour déterminer les bonnes capacités de production et de stockage, celles nécessaires et suffisantes pour éviter le surcoût d'équipement du logement (le coût des batteries et des uités de production est lourd, de fait).

2.2 Gestion de l'énergie et contrôle

Si nous pouvions déployer une telle application dans un logement *réel*, nous pourrions connaître la puissance électrique consommée par les différents appareils en plaçant des capteurs de mesure sur le réseau ou grâce à des capteurs intégrés aux appareils et interrogeables à distance. Malheureusement, nous ne disposons pas d'une plate-forme expérimentale pour cela, et ce serait de toutes façons actuellement un effort trop important dans le cadre d'un cours de 6 ECTS. Nous allons donc nous contenter d'un développement logiciel *sans matériel* où la *simulation remplacera les mesures réelles.* [6]

Côté logiciel, un appareil électrique numérique est doté d'un logiciel embarqué et il est connecté via réseau, ce qui permet d'échanger des informations et de piloter ses fonctions à distance. Dans le projet, chaque appareil, chaque unité de production et unité de stockage sera représenté par un composant logiciel dont les services et les méthodes internes formeront le logiciel qui serait embarqué (déployé) sur l'appareil correspondant. Les connexions (via ports et connecteurs) entre composants vont abstraire (et remplacer ici) le fait que celles-ci puissent être établies via réseau sans fil de type WiFi ou Bluetooth dans la réalité.

Pour développer du logiciel assez réaliste, nous allons également nous tourner vers la simulation numérique. Chaque appareil, chaque unité de production et chaque unité de stockage devra être à même de simuler sa puissance électrique consommée ou produite (en ampères ou en watts). Pour les appareils simples et incontrôlables, il suffit souvent d'avoir une consommation constante : soit l'appareil est allumé et il consomme une quantité d'électricité fixe (par exemple, un sèche-cheveux consommant 1200 watts), soit il est éteint et il ne consomme bien sûr rien. Un cas à peine plus complexe est un appareil incontrôlable ou modulable qui a différents modes de fonctionnement (par exemple, un ventilateur à trois niveaux de puissance). Dans ces cas, le simulateur produira une consommation électrique constante par morceaux et le basculement entre les niveaux se fera par des actions représentées par des événements dans la simulation : allumage, extinction, passage de mode de bas à haut et vice versa, *etc.*

Pour les appareils incontrôlables, ces actions seront déclenchés par les utilisateurs des appareils, à simuler comme des événements. Pour les appareils modulables, en plus des actions de l'utilisa-

5. Le chargement des batteries n'est bien sûr activé que lorsque la puissance produite par les unités de production, hors batteries, dépasse la puissance consommée, puisqu'on ne saurait utiliser la puissance produite par les batteries pour les charger elles-mêmes.

6. En fait, nous allons voir en cours que de toutes façons, le développement d'une version co-simulée avec le logiciel, dîte en simulation « *software-in-the-loop* », précède généralement celui d'une version simulée pour l'environnement uniquement mais incluant le matériel dîte « *hardware-in-the-loop* », avant le passage en déploiement réel.

teur (allumer et éteindre, par exemple), des actions pourront être déclenchées par le gestionnaire d'énergie (suspendre et reprendre, par exemple), ce qui demandera de déclencher des événements dans la simulation pour provoquer les changement de modes de consommation. Grâce aux simulateurs qui incluront la simulation de la puissance consommée ou produite, le compteur électrique pourra « mesurer » le niveau total de puissance consommée et produite dans le logement. Le composant compteur électrique offrira des services capteurs permettant au gestionnaire d'énergie de récupérer ces valeurs pour prendre ses décisions et les actions en fonction de la situation (écart entre production et consommation, niveau des batteries, *etc.*).

D'autres appareils, plus complexes, en plus des actions de l'utilisateur ou de pilotage à distance, passent d'un mode à l'autre de manière *autonome*. Ces changements de modes peuvent se produire à cause d'une limite physique. Par exemple, la capacité des batteries étant limitée, elles passent du mode « en charge » à « chargé » lorsque cette limite est atteinte. Pour d'autres appareils, ces changements de mode se produisent par une *action de contrôle* interne. Voici quelques exemples d'appareils de tous les jours qui intègrent une fonction de contrôle simple :

- Chauffage : à partir d'une température cible et d'une lecture de la température ambiante dans le logement, le thermostat du radiateur allume et éteint les éléments chauffants de telle manière à maintenir la température ambiante proche de la température cible.
- Chauffe-eau : de manière similaire à celle d'un appareil de chauffage, le chauffe-eau maintient une température cible de l'eau dans son réservoir en agissant sur des éléments de chauffage électriques placés dans le réservoir.
- Four : similaire au chauffe-eau, avec le plus souvent une minuterie qui arrête le chauffage au bout d'un temps déterminé par l'utilisateur.
- Réfrigérateur : il agit de même pour maintenir la température cible de ses compartiments, à ceci près qu'il utilise un compresseur pour refroidir et non des éléments de chauffage pour augmenter la température ; un réfrigérateur à un compresseur refroidit ses deux compartiments (s'il y en a deux) en même temps alors qu'avec deux compresseurs, les deux compartiments sont gérés indépendamment.

Pour ces contrôleurs intégrées aux composants représentant les appareils, le plus simple sera d'utiliser des formes de contrôle basiques venant de l'automatique comme les contrôleurs à seuils avec *hystérésis* que nous verrons en cours, mais il vous sera possible d'utiliser des contrôleurs plus complexes si vous le souhaitez.

Au-dessus de ces actions et contrôles internes, le gestionnaire d'énergie du logement va exercer lui aussi une forme de contrôle à partir de la collecte de données sur les puissances totales instantanées produites et consommées ainsi que de l'état courant des appareils. Il conservera une liste des appareils dont la consommation a été modulée (réfrigérateur, chauffe-eau, chauffage, ...). À partir de ces informations, il pourra décider des actions de gestion d'énergie appropriées. Ce type de contrôle pourra utiliser les techniques présentées en cours dans le contexte de l'informatique autonomique ou de la robotique autonome.

2.3 Influence de l'environnement

L'environnement inclut tous les facteurs extérieurs influençant la gestion d'énergie en dehors des équipements (hormis les actions des utilisateurs). Par exemple, on va intégrer, le cas échéant :

- l'intensité d'ensoleillement courant qui influe sur la puissance produite par des panneaux solaires ;
- la force du vent qui influe sur la puissance produite par des éoliennes ;
- la température extérieure qui influence via la porosité thermique des murs du logement la manière dont décroît la température intérieure ;
- la température de l'eau froide provenant du réseau de distribution d'eau qui influe sur la température de l'eau du réservoir d'eau chaude lorsque la consommation d'eau chaude nécessite son remplissage.

Les éléments entrant en ligne de compte pour la gestion de l'énergie dans votre projet, selon vos choix d'appareils et d'unités de production, devront être intégrés dans l'architecture logicielle par des capteurs « connectés » permettant de connaître les valeurs courantes de ces facteurs environnementaux (thermomètre extérieur, thermomètre sur le compteur d'eau, anémomètre, ...). Pour être en mesure de mettre au point, de tester systématiquement et de dimensionner le système,

il faudra aussi simuler ces phénomènes pour produire des données d'entrée réalistes.

2.4 Impact des utilisateurs

Dans un logement, les activités des utilisateurs sont à l'origine de la demande d'énergie des appareils. Par exemple, les phénomènes suivants pourront être intégrés :

— les choix des températures cibles pour les appareils concernés ;
— les moments et durée de consommation d'eau chaude (aux heures de repas pour laver la vaiselle, matin pour les douches, soir pour les bains, ...) ;
— les accès au refrigérateur (ouvertures et fermetures des portes) qui influent sur la rapidité de l'augmentation de la température interne des compartiments ;
— allumage et extinction des appareils à consommation incontrôlable.

Encore ici, pour mettre au point, tester le système et le dimensionner, il faudra simuler ces comportements, selon les appareils qui seront inclus dans le système. Toutefois, le comportement des utilisateurs peut différer passablement. Il sera bon de prévoir des paramètres de simulation permettant de faire varier les scénarios utilisateurs afin de dimensionner en fonction de ceux-ci (nombre de personnes vivant dans le logement, par exemple).

3 Mise en œuvre du projet

Plan général

Le projet va se dérouler en quatre étapes correspondant aux quatre évaluations prévues :

Étape 1 (3 semaines, du 18/09 au 9/10) : Développement en BCM4Java des composants représentants les appareils, les unités de production et l'unité de stockage à batteries, sans toutefois tenir compte de la puissance électrique qui sera intégrée aux étapes suivantes via la simulation.

Étape 2 (5 semaines, du 9/10 au 13/11) : Développement des simulateurs modulaires permettant de simuler les puissances électriques consommées et produites ainsi que les autres éléments influençant celles-ci : températures internes, facteurs environnementaux, actions des utilisateurs, *etc.*

Étape 3 (8 semaines, du 13/11 au 9/01/2025) : Intégration des simulateurs modulaires développés à l'étape 2 au sein des composants développés à l'étape 1 pour former des composants cyber-physiquesde la co-simulation « *software-in-the-loop* ».

Étape 4 (4 semaines, du 9/01 au 5/02) : Développement du contrôle exercé par le gestionnaire d'énergie ainsi que des scénarios de tests d'intégration et de dimensionnement du système.

Étape 1 — Composants équipements et leurs interconnexions

***Nota bene* :** À cette étape, nous nous intéressons uniquement à la partie composants logiciels du projet, sans simulation, et donc en ne tenant aucun compte de l'électricité. [7] Ce n'est qu'à l'étape 2 que nous aborderons la partie purement électrique, c'est à dire la production et la consommation de puissance électrique en termes quantitatifs, par la simulation. De même, les divers contrôles (internes aux appareils, comme la température du four, ou sur la consommation globale de la maison dans le gestionnaire d'énergie) ne seront pour leur part programmés qu'à l'étape 3, lorsque les simulateurs seront intégrés aux composants. Toutefois, les méthoeds nécessaires devront être incluses dans les interfaces et des « bouchons » pour le test inclus dans le code.

7. Au sens des simulateurs DEVS vus en cours ; pour les besoins des tests unitaires ou des tests d'intégration, il sera possible de simuler certains éléments comme le temps courant pour s'assurer du bon fonctionnement du code sans avoir à attendre des heures pour le déroulement d'un test en temps réel. Voir les exemples fournis pour ce genre de « simulations » pour le test logiciel.

Pour cette première étape, chaque équipement sera représenté par un composant avec au moins une interface offerte, que vous devrez concevoir, explicitant les opérations permises sur celui-ci selon les entités tierces les utilisant (gestionnaire d'énergie, actions des utilisateurs, actions du contrôleur local selon les appareils, *etc.*). Le compteur électrique et le gestionnaire d'énergie seront également représentés par des composants. [8]

Tous les appareils électriques modulables, les unités de production et l'unité de stockage seront connectés (par ports et connecteurs BCM4Java) au gestionnaire d'énergie pour être pilotés à distance par ce dernier. Le rôle du composant représentant le compteur électrique ne sera effectif qu'à l'étape 3 quand des données de puissances électriques consommées et produites seront obtenues par la simulation et que les simulateurs auront été intégrés aux composants. À cette étape 3, les mesures des puissances totale produite et consommée faites par le composant compteur depuis son simulateur pourront être récupérées par le gestionnaire via des méthodes dont les signatures vous sont déjà fournies mais qu'il reste à implanter.

Une fois que vous aurez sélectionné les appareils et unités de production que vous allez inclure dans votre projet, il faudra faire une conception logicielle (composant) de chacun, en incluant l'unité de stockage, puis les intégrer avec les composants compteur électrique et gestionnaire d'énergie fournis. Pour ce faire, il faudra déterminer les opérations qui seront disponibles sur ces derniers et pour quels types de composants tiers de manière à spécifier les interfaces qu'ils vont offrir. Vous devrez ensuite implanter les différents composants puis produire un assemblage de composants exécutable avec des scénarios de test.

Connexion BCM4Java générique des équipements à consommation contrôlable

***Nota bene* :** La problématique de connexion *générique* des appareils au gestionnaire d'énergie attaquée dans cette sous-section concerne en réalité tous les appareils, unités de production et unités de stockage. Toutefois, pour ne pas surcharger inutilement le projet, vous n'aurez à traiter de manière générique que les appareils modulables. [9] Les appareils incontrôlables ne sont pas concernés, puisqu'ils ne seront pas connectés au gestionnaire d'énergie. Pour les autres équipements, ils seront connectés au gestionnaire d'énergie par les interfaces de composants spécifiques que vous aurez créées.

Dans la description générale en introduction, nous avons évoqué la problématique pragmatique de la connexion (informatique) des équipements avec le gestionnaire d'énergie. En BCM4Java, il est possible de connecter deux composants (ici le composant client gestionnaire d'énergie et un composant serveur représentant un équipement électrique, comme un réfrigérateur) le premier requérant une interface A et l'autre offrant une interface B différente de A. Il faut alors implanter un connecteur qui est appelé selon l'interface A par le client mais qui appelle le serveur selon l'interface B, faisant ainsi la « médiation » ou la « colle » entre les deux.

Dans une économie de marché libre, il est difficile d'imaginer que tous les équipements électriques proposés par tous les industriels au monde se conforment aux mêmes interfaces standards. Mais si chaque équipement électrique produit dans le monde qui devait pouvoir être connecté à notre gestionnaire d'énergie offrait une interface spécifique propriétaire, il faudrait prévoir à l'avance tous les connecteurs pour tous ces équipements, une solution ingérable en pratique, d'autant que de nouveaux appareils nécessitant de nouveaux connecteurs apparaîtraient tout le temps pendant la durée de vie du gestionnaire d'énergie. Cette problématique très courante dans l'industrie va nous permettre d'explorer les possibilités offertes par la *génération de code à l'exécution* à l'aide de l'outil Javassist pour les appareils contrôlables.

Dans le cadre du projet nous allons supposer que le gestionnaire d'énergie requiert (au sens de BCM4Java) une interface de contrôle *générique* via laquelle il va piloter les appareils contrôlables appelée `AdjustableCI` (qui vous est fournie). Pour connecter un appareil modulable offrant une interface de composant spécifique, `RefrigeratorCI` par exemple, il faut un connecteur qui fait la

8. Une version préliminaire des composants compteur et gestionnaire vous sera fournie. Il faudra les modifier pour inclure le code nécessaire pour compléter votre projet.

9. Dans les exemples qui vous sont fournis comme point de départ du projet, les appareils de ce type (appareil de chauffage) sont connectés via des connecteurs programmés *manuellement*. Vous aurez à faire en sorte dans votre livraison de modifier cela pour remplacer la connexion par connecteur généré automatiquement.

médiation entre l'interface requise `AdjustableCI` et l'interface offerte `RefrigeratorCI`.

Dans le code fourni comme point de départ du projet, ces connecteurs sont programmés manuellement et vous les trouverez dans le répertoire du composant gestionnaire. Vous pourrez procéder de la même manière dans un premier temps pour vos propres appareils. Mais cette solution est insatisfaisante, comme expliqué ci-haut. Dans un second temps, vous devrez passer à de la génération de code des connecteurs nécessaires *à la volée*. Pour savoir comment implanter les méthodes définies dans l'interface requise (par exemple, `AdjustableCI`) en termes d'appels à l'interface offerte (par exemple, `RefrigeratorCI`), on va utiliser un descripteur en XML fourni par l'équipement. [10] Cette approche vous permettra d'expérimenter des architectures logicielles complexes, où on cherche à connecter tardivement (*pendant l'exécution*) des parties de logiciels développés avec différentes interfaces voire dans différents langages de programmation, ce qui devient de plus en plus courant dans l'industrie avec ce qu'on appelle les *systèmes de systèmes*.

Maintenant, comment un appareil va fournir au composant gestionnaire d'énergie son descripteur XML ? Dans le cadre du projet, le composant gestionnaire d'énergie va offrir une interface d'enregistrement et de connexion pour les équipements contrôlables appelée `RegistrationCI`. Selon cette approche, lors de leur enregistrement, les appareils vont fournir au gestionnaire leur document XML. Ce document XML décrit par du code source Java comment chaque méthode de l'interface requise générique va être implantée pour appeler les méthodes de l'interface spécifique offerte par l'appareil. Ce document XML suit un schéma Relax NG fourni (`control-adapter.rnc`). Il doit être utilisé pour générer à la volée le code de la classe de connecteur à utiliser pour connecter (au sens de BCM4Java) le composant gestionnaire au composant représentant l'appareil qui souhaite s'enregistrer.

Voici en résumé le scénario qui doit être implanté. Tous les appareils connaissent l'URI du port du gestionnaire d'énergie offrant l'interface d'enregistrement `RegistrationCI`. Pour être géré par celui-ci, un composant appareil ajustable qui vient d'être allumé s'y connecte et appelle la méthode `register` pour s'enregistrer. Cette méthode prend un identifiant unique de l'équipement (son numéro de série, par exemple), l'URI de son port entrant offrant son interface de contrôle et le document XML décrivant son interface de contrôle sous la forme d'une chaîne de caractères. Le gestionnaire va alors générer dynamiquement une classe de connecteur avec Javassist puis l'utiliser pour se connecter à l'appareil. [11] Lorsque l'équipement est éteint, il se désenregistre du gestionnaire en appelant la méthode `unregister`.

Modalité de réalisation

À titre de suggestion, la réalisation de cette étape peut elle-même se décomposer en tâches :

— identification des équipements à inclure dans votre projet (appareils et unités de production) ;
— recherches personnelles et sur Internet pour trouver des informations réalistes sur les fonctionnalités des équipements sélectionnés pour bien modéliser leurs services ;
— définition des interfaces de composants puis développement des composants (les puissances électriques n'étant pas modélisées à cette étape, contentez-vous de vous assurer que vos équipements passent bien d'un mode à l'autre quand les opérations sont appelées et que les opérations permettant de récupérer des données fonctionnent correctement) ;
— programmation manuelle des connecteurs pour les équipements qui ne sont pas concernés par la connexion générique (appareils incontrôlables, unités de production, batteries) ;
— programmation manuelle des connecteurs pour les composants représentant vos appareils modulables (cela vous facilitera ensuite la production du descripteur en XML) ;
— programmation du générateur de connecteurs en utilisant l'outil Javassist et développement des documants XML pour vos appareils modulables à partir du code Java des connecteurs développé manuellement à l'étape précédente ;
— assemblage des composants pour former une application exécutable ;

10. Un exemple complet vous sera fourni.
11. Dans votre développement, il est vivement conseillé de programmer d'abord manuellement les connecteurs de vos appareils et d'utiliser ces connecteurs manuels pour mettre au point votre code. Seulement ensuite, vous vous attaquerez à la génération automatique, pour bien séparer les préoccupations en vous concentrant alors sur la génération de code.

— programmation de scénarios de démonstration en utilisant les méthodes `start` et `execute` des composants qui va enregistrer les appareils auprès du gestionnaire puis tester les fonctionnalités sur les appareils via ce composant gestionnaire (comme démontré dans le logiciel fourni comme point de départ).

Pour la génération du code des connecteurs à partir d'un document XML, vous pouvez vous référer à la classe `fr.sorbonne_u.components.cvm.config.ConfigurationFileParser` de BCM4Java pour un exemple de lecture et traitement de fichiers XML en Java.

Objectifs à atteindre

À l'issue de cette étape, lors de l'audit 1, vos réalisations devraient permettre :
— d'expliquer votre conception, les choix que vous aurez faits et l'état d'avancement que vous aurez atteint ;
— d'avoir les composants appareils, unités de production et unité de stockage intégrés avec les versions fournies mais complétées des composants compteur et gestionnaire d'énergie dans un assemblage exécutable ;
— de réaliser la connexion des appareils modulables via le protocole d'enregistrement auprès du gestionnaire d'énergie incluant la génération du code des connecteurs au moment de l'enregistrement avec Javassist ;
— d'exécuter des scénarios de tests d'intégration vérifiant le bon fonctionnement du code des appareils en suivant les changements de mode de fonctionnement ordonnés par le gestionnaire et prévus dans les scénarios ;
— de faire cette démonstration en mono-JVM.

Étape 2 — Simulation MIL

Nota bene* :** À cette étape, nous nous intéressons *uniquement* à la simulation « *model-in-the-loop* » de l'énergie électrique et des autres aspects physiques, c'est à dire sans le logiciel, ***laissant donc de côté les composants qui ne seront repris qu'à l'étape 3.

Dans la seconde étape, en se basant sur les connaissances et compétences développées en cours sur la simulation « *model-in-the-loop* » (MIL) des systèmes cyber-physiques, l'objectif sera d'implanter une première simulation MIL de la puissance électrique consommée par les appareils, produite par les unités de production, stockées par les batteries et mesurées par le compteur électrique. Par simulation MIL, nous entendons un lgiciel de simulation contenant uniquement des modèles DEVS, *sans* interaction avec le logiciel. À cette étape, nous nous limitons aussi aux différents modes de fonctionnement des appareils de manière à produire correctement leur consommation/production de puissance électrique. Les basculements entre modes sont provoqués par des événements. Pour les basculements correspondant à des changements dans l'environnement ou à des actions des utilisateurs, des modèles testeurs seront inclus dès cette étape pour émettre les différents événements afin de vérifier le bon fonctionnement des modèles. Pour les basculements qui exigeront une décision (ou un contrôle, comme couper le chauffage lorsque la température souhaitée est atteinte), des modèles testeurs émettront *arbitrairement* ces événements car la partie décision ne sera intégrée qu'à l'étape 3 sous forme de code dans des composants contrôleurs et dans le composant gestionnaire d'énergie. [12]

Pour la modélisation de la puissance électrique consommée ou produite, chaque équipement va être doté d'un simulateur DEVS qui va le modéliser et le simuler en fonction de son mode de fonctionnement, des paramètres de la simulation (par exemple, températures cibles, température extérieure, *etc.*) et les effets de l'environnement et des utilisateurs (voir ci-haut). Les connaissances et compétences nécessaires pour ce faire sont développées principalement aux cours 3 à 5 (vus avant le début de cette étape 2). À cette étape, on met de côté les contrôles qui ne seront abordés qu'à l'étape 3. En pratique, si un appareil a plusieurs modes de fonctionnement, que ce soit suite à des décisions de l'utilisateur (allumer, éteindre, ...), à des contrôles (suspendre, reprendre, ...) ou

12. Il n'est pas nécessaire de produire des modèles de simulation de l'environnement et des utilisateurs très complexes dès cette étape ; des modèles assez simples suffiront dans un premier temps puis des modèles plus complexes seront développés à l'étape 4.

à des effets de l'environnement (variation de la température extérieure, ...), on se contentera à cette étape de modéliser ces basculements par des événements à traiter par le modèle de puissance électrique consommée ou produite. Les décisions de contrôle seront développés comme du code dans des composants aux étapes 3 et 4. [13]

Modalité de réalisation

À titre de suggestion, la réalisation de cette étape peut elle-même se décomposer en tâches :
- recherches personnelles et sur Internet pour trouver des informations réalistes sur les puissances électriques consommées et produites des équipements sélectionnés (commencez par choses simples, en prenant un niveau de puissance fixe par mode de fonctionnement puis, quand vos simulateurs fonctionneront bien, vous pourrez développer des modèles plus complexes) ;
- développement des simulateurs de chaque appareil en DEVS à l'aide de la bibliothèque NeoSim4Java fournie avec quelques tests unitaires pour s'assurer de leur bon fonctionnement ;
- développement des modèles de simulation de l'environnement et des actions des utilisateurs ;
- création d'un modèle couplé complet capable de réaliser une simulation de l'ensemble des appareils (incluant les composants fournis) selon un scénario que vous aurez prévu et programmé.

Objectifs à atteindre

À l'issue de cette étape, vos réalisations devraient permettre :
- d'expliquer votre conception des simulateurs, les choix que vous aurez faits et l'état d'avancement que vous aurez atteint ;
- de faire une démonstration individuelle (sortes de tests unitaires) du comportement simulé des différents appareils avec des scénarios illustrant l'ensemble des fonctionnalités de chacun ;
- de faire une démonstration de simulation complète (sortes de tests d'intégration) permettant de suivre les puissances totales consommées et produites via le modèle du compteur électrique (qui fait la somme de ces puissances).

Lors de la soutenance de mi-semestre, vous devrez présenter les résultats de l'étape 2 et vous pourrez aussi revenir sur les résultats de l'étape 1 si des compléments ont été apportés depuis l'audit 1. La notation du premier examen réparti incluera la notation du code rendu des étapes 1 et 2, leur exécution correcte et la qualité de la soutenance elle-même (explications et réponses aux questions).

Dans le cas où vous n'auriez pas le temps de finir tous les simulateurs pour tous les appareils, il sera préférable de terminer entièrement les simulateurs pour une partie des appareils plutôt que de ne faire qu'une partie des simulateurs pour tous les appareils. De même, il sera préférable d'avoir un test d'intégration complet que seulement une partie des tests unitaires sans test d'intégration.

Étape 3 — Intégration en simulation SIL et contrôles locaux

Dans cette troisième étape, vous allez faire l'adaptation et l'intégration des modèles de simulation de l'étape 2 dans les composants développés à l'étape 1 puis modifier ces derniers de manière à pouvoir mener une simulation SIL (simulation intégrée avec l'exécution du code des composants).

La transformation des modèles de simulation MIL en modèles de simulation SIL requiert d'abord d'inclure dans les composants le code permettant de créer et d'interconnecter les simulateurs au moment de l'assemblage des composants. Ensuite, il faudra ajouter le code nécessaire pour assurer les interactions entre le code des composants et les simulateurs embarqués. Par exemple, dans le composant compteur électrique, les puissances totales consommée et produite sont calculées dans son simulateur, créé et interconnecté avec les simulateurs ds autres équipements, puis des méthodes dans le composant permettent d'en obtenir les valeurs instantanées en allant les

13. Des exemples vous seront fournis.

chercher dans le simulateur embarqué dans ce composant. Il faudra donc ajouter le code dans le simulateur permettant de lire ces valeurs puis le code dans les méthodes du composant appelant le code dans le simulateur pour aller les chercher. De même, les méthodes des composants provoquant des changements de mode, comme une suspension, vont devoir mettre à jour l'état de l'appareil dans sa simulation, ce qui sera fait en ajoutant dans le code des méthodes des appels au simulateur pour lui envoyer des événements qui le feront basculer de mode.

Le second point important du passage de MIL à SIL est le passage d'une simulation en temps logique à une simulation en temps réel (accéléré).

Sur la base de ce que vous aurez réalisé à l'étape 2, l'étape 3 doit d'abord intégrer les simulateurs DEVS MIL aux composants, les transformer en simulateurs SIL puis développer le code d'interaction entre composants et simulateurs. Les compétences nécessaires sont abordées principalement au cours 8. Un point d'étape intéressant serait alors de réussir à faire s'exécuter les simulations MIL de l'étape précédente mais cette fois-ci à travers les composants avant de les transformer en simulateurs SIL interagissant avec le code des composants.

Ensuite, vous allez développer les contrôles locaux des équipements concernés en commençant par les appareils, les unités de production et l'unité de stockage. Pour cela, vous allez utiliser les moyens de communication entre modèles DEVS et composants. [14] Par exemple, pour un réfrigérateur, le contrôleur interne implantera une boucle de contrôle qui va lire à intervalle régulier sa température interne, la comparer à la température cible puis déclencher au besoin les événements basculant entre les modes « en refroidissement » et « compresseur en pause ». Les connaissances et compétences nécessaires sont développées principalement au cours 6 (vu avant le début de cette étape 3).

Bien que cela alourdisse un peu les choses, il est important dans le cadre du cours de respecter un bon découpage en composants de ce projet (par exemple, maintenir des composants de contrôle séparés des composants de base pour les appareils comme un réfrigérateur ou un chauffe-eau). En effet, l'objectif du cours demeure de produire de bonnes architectures logicielles, sans pour autant vous forcer à développer une application énorme qui justifierait certes ces découpages mais vous noierait sous la charge de travail nécessaire pour en arriver à bout.

Modalité de réalisation

À titre de suggestion, la réalisation de cette étape peut elle-même se décomposer en tâches :
- construction de l'architecture de simulation jointe à l'architecture à base de composant en intégrant les simulateurs MIL en temps logique de l'étape 2 dans les composants représentant les appareils développés à l'étape 1 ;
- mise au point pour pouvoir exécuter un scénario de test d'intégration où les tests de l'étape 1 et les simulations de l'étape 2 s'exécutent en parallèle mais indépendamment ;
- passage des simulateurs MIL en temps logique intégrés précédemment à des simulateurs SIL en temps réel ;
- validation informelle de cette intégration en faisant s'exécuter les scénarios de simulation de l'étape 2 sur cette architecture combinée simulateurs/composants (c'est à dire sans exécution du code des composants en tant que tel) ;
- fusion de la partie simulateurs et de la partie code des composants en ajoutant les interactions entre le code des composants et les simulateurs ;
- développement de scénarios de simulation « software-in-the-loop » plus élaborés et leur exécution.

Objectifs à atteindre

À l'issue de cette étape, vos réalisations devraient permettre :
- d'expliquer l'ensemble de votre conception, des choix que vous aurez faits et l'état d'avancement que vous aurez atteint ;

14. Des exemples vous sont fournis.

— d'illustrer les contrôles locaux ajoutées à vos équipements grâce à l'interaction entre le code et la simulation ;
— de faire une première démonstration d'ensemble en mono-JVM.

Lors de l'audit 3, c'est l'état d'avancement de cette intégration entre composants et simulation SIL qui sera au cœur des critères d'évaluation.

Étape 4 — Gestionnaire d'énergie et scénarios d'exécution

La quatrième et dernière étape vise à parfaire l'intégration de l'ensemble des éléments développés jusque-là en développement progressivement les règles de contrôle utilisées par le gestionnaire d'énergie pour assurer l'équilibre entre les puissances électriques consommées et produites. Vous implanterez des règles de contrôle capables de moduler (suspendre) la consommation de certains appareils ou d'utiliser une unité de production intermittente pour palier temporairement la faiblesse de production de l'unité de production aléatoire. L'idée générale est que le gestionnaire d'énergie réagit lorsqu'il détecte que la puissance totale produite passe en-dessus de la puissance totale consommée en choisissant des appareils dont la consommation peut être modulée à la baisse. S'il n'y a pas de tel appareil, il pourra décider de lancer l'unité de production intermittente. De même, quand il détecte que la puissance produite dépasse la puissance consommée, il peut arrêter l'unité de production intermittente, faire reprendre les appareils suspendus ou recharger les batteries si nécessaires. De bonnes règles doivent permettre de bien gérer le fonctionnement des appareils modulables en fonction de leurs contraintes, la charge des batteries et les capacités de production intermittente en les maintenant le plus longtemps possible.

Pour illustrer de manière plus marquante le bon fonctionnement du gestionnaire d'énergie, il faut qu'il y ait suffisamment de variation et de contraintes de production et suffisamment de possibilités de modulation sur les appareils contrôlables. Dans les étapes précédentes, il vous a été demandé d'implanter un nombre limité d'appareils. Pour obtenir des possibilités suffisantes dans les scénarios de cette étape, il sera bon d'augmenter le nombre d'appareils sous contrôle du gestionnaire. La manière la plus simple pour ce faire sera de créer de mutiples instances des composants que vous avez déjà programmés. Par exemple, plutôt qu'un seul réfrigérateur, créez-en 10, chacun pouvant être modulé individuellement.

Conjointement au développement progressif du gestionnaire d'énergie, vous devrez améliorer vos scénarios de tests et de dimensionnement. Pour cela, il faudra s'assurer de bien déterminer les paramètres des simulateurs et en particulier les simulateurs des actions des utilisateurs pour produire des variations dans la consommation et la production qui permettent d'obtenir des résultats intéressants *in fine*.

Modalité de réalisation

Comme expliqué ci-haut, cette étape cherche principalement à consolider les développements des trois étapes précédentes. Dans cette optique, et comme expliqué, les deux tâches princpales sont :

— le développement des règles de gestion de l'énergie dans le composant gestionnaire autour d'une boucle qui récupère les puissances totales instantanées produites et consommées, décide d'une action pour réagir et applique cette action sur le ou les appareils ou unités concernés ;
— le développement des scénarios de tests et de dimensionnement par l'introduction d'actions d'utilisateurs et l'ajustement des paramètres des simulateurs.

Pour arriver à un résultat final opérationnel et le plus satisfaisant possible, il est vivement suggéré d'avancer en parallèle ces deux tâches pour construire une démonstration la plus étoffée possible à la fin tout en s'assurant d'en avoir une qui fonctionne à présenter si vous n'avez pas le temps de finir entièrement les scénarios pour tous les appareils.

Objectifs à atteindre

À l'issue de cette étape, vos réalisations devraient permettre :

— d'expliquer l'ensemble de votre conception, des choix que vous aurez faits et l'état final que vous aurez atteint ;
— d'illustrer les contrôles et les fonctions de planification ajoutées à vos appareils et à votre gestionnaire par des scénarios pertinents ;
— de faire une démonstration en mono-JVM.

Une exécution en multi-JVM donnera lieu à un bonus ; elle ne sera donc pas exigée pour avoir une note finale maximale.

4 Modalités de remise et d'évaluation des projets :

— Le projet se fait **obligatoirement** en **équipe de deux étudiant(e)s**. Tous les fichiers sources du projet doivent comporter les noms (balise `author`) de tous les auteurs en Javadoc. Lors de sa formation, chaque équipe devra se donner un nom et me le transmettre avec les noms des étudiant(e)s la formant au plus tard le **27 septembre 2024** à minuit.
— Le projet doit **impérativement** être réalisé avec Java SE 8 ! Attention, peu importe le système d'exploitation sur lequel vous travaillez, il faudra que votre projet s'exécute correctement sous Eclipse en Mac Os X (que j'utilise).
— Votre rendu *final* de projet devra inclure une *documentation Javadoc* des différents paquetages de votre projet. La documentation Javadoc sera générée et incluse dans votre livraison dans un répertoire `doc` au même niveau que votre répertoire `src`.
— Votre code final doit aussi être commenté, être lisible et correctement présenté (indentation, choix pertinents des identifiants, ...).
— L'évaluation comportera quatre épreuves : deux audits intermédiaires, une soutenance à mi-semestre accompagnée d'un rendu de code et une soutenance finale accompagnée du rendu final du code du projet et de la documentation associée. Les deux audits intermédiaires dureront 15 minutes chacun. La soutenance à mi-parcours durera 20 minutes. La soutenance finale durera 30 minutes. Le tout se déroulera selon les modalités suivantes :
    1. Les **deux audits intermédiaires** auront lieu pendant les séances du cours, *a priori* le **9 octobre 2024** (séance 4) puis le **9 janvier 2023** (séance 13 avancée). Ils se dérouleront de manière informelle devant votre ordinateur où vous pourrez me présenter l'état d'avancement de votre projet et répondre à mes questions. Ils donneront lieu à une note sur 20 valant chacune 10% de la note finale de l'UE. L'audit 1 visera à vérifier que vous avez atteint les objectifs de l'étape 1. L'audit 2 portera sur l'atteinte des objectifs de l'étape 3 tel que décrits précédemment.
    2. La **soutenance à mi-parcours** portera sur l'atteinte des objectifs des deux premières étapes du projet et elle aura lieu dans la semaine des premiers examens répartis, *a priori* le **13 novembre 2024**. Elle donnera lieu à une note sur 20 comptant pour 30% de la note finale de l'UE. Elle comportera une discussion des réalisations (code) pendant une quinzaine de minutes (devant l'écran sous Eclipse) et une courte démonstration de cinq minutes. Les rendus à mi-parcours se feront le **dimanche 10 novembre 2024 à minuit** au plus tard.
    3. La **soutenance finale** portant sur l'ensemble du projet mais avec un accent sur les troisième et quatrième étapes aura lieu dans la semaine des seconds examens répartis, *a priori* le **5 février 2025**. Elle donnera lieu à une note sur 20 comptant pour 50% de la note finale de l'UE. Elle comportera une présentation d'une douzaine de minutes (en utilisant des transparents), une discussion d'une dizaine de minutes également devant écran sur les réalisations suivie d'une démonstration (scénarios de tests d'intégraiton et scénarios de domensionnement). Les rendus finaux se feront le **dimanche 2 février 2025 à minuit** au plus tard.
— Bien que les audits et les soutenances se font par équipe, l'évaluation reste à chaque fois **individuelle**. Lors des audits et des soutenances, *chaque étudiant(e)* devra se montrer capable d'expliquer différentes parties du projet, et selon la qualité de ses explications et de ses réponses, sa note peut être supérieure, égale ou inférieure à celle de l'autre membre de son équipe.
— Les audits et les soutenances auront lieu selon un horaire de passage qui sera publié le moment venu sur le site de l'UE. **Tout retard** de l'équipe de plus de la moitié de la

durée prévue de ces audits ou soutenances entraînera son **annulation** et une note de 0 sera attribuée aux deux membres de l'équipe pour l'épreuve concernée. Si un(e) seul(e) des membres d'une équipe arrive en retard, il(elle) sera exclu(e) de l'épreuve et une note de 0 sera attribuée. Dans ce cas, si l'autre membre de l'équipe est à l'heure, il(elle) passera l'épreuve seul(e).

— Le rendu à mi-parcours et le rendu final se font sous la forme d'une archive contenant un répertoire directement utilisable sous Eclipse pour créer un nouveau projet. Le répertoire de projet et votre archive ***doivent*** avoir pour nom celui de votre équipe (ex. : équipe LionDeBelfort, répertoire de projet `LionDeBelfort` et archive `LionDeBelfort.tgz`). Cette archive doit se présenter au format `tgz` si vous travaillez sous Unix ou `zip` si vous travaillez sous Windows et ce ***à l'exclusion de tout autre format !*** Vous m'enverrez le fichier à `Jacques.Malenfant@lip6.fr` comme attachement fait proprement avec votre programme de gestion de courrier préféré ou encore par téléchargement depuis un site tiers avec un lien envoyé par courrier électronique (en lieu et place du fichier).

  Pour éviter de transmettre des archives trop volumineuses qui seront rejetées par le système de courrier électronique, ne transmettez que le strict minimum, essentiellement les sources Java. Évitez d'inclure les binaires et les jars qui vous sont fournis, par exemple. Par contre, si vous utilisez des jars autres que ceux fournis, alors il faudra les inclure dans la livraison.

— **Tout manquement à ces règles élémentaires entraînera une pénalité dans la note des épreuves concernées !**

— Pour l'épreuve de deuxième session, si elle s'avérait nécessaire, elle consiste à poursuivre le développement du projet pour résoudre ses insuffisances constatées à la première session. Elle donnera lieu à un rendu du code puis à une soutenance dont les dates seront déterminées en fonction du calendrier du master. Quand un seul membre de l'équipe doit passer la seconde session, il(elle) fait ce travail seul(e) ; l'évaluation tient alors compte du fait que le rendu de seconde session a été fait seul(e) ou à deux. Un membre peut également choisir de passer la seconde session seul(e) même si les deux membres de l'équipe doivent passser l'épreuve ; dans ce cas, les deux membres travaillent et passent l'épreuve en solitaire selon les modalités précédentes.

Fin du document.