École Polytechnique de l'Université de Nantes

Éco-conception de logiciel

Rapport de stage de 5ème année

Lieu de stage : École des Mines de Nantes

Maître de stage :
Thomas Ledoux

Tuteur :
Benoît Parrein

Élève :
Guillaume Delamare

Le 29 Juin 2012

ECOLE DES MINES DE NANTES

POLYTECH NANTES

Remerciements

Je tiens, tout d'abord, à remercier mon tuteur, Thomas Ledoux, pour m'avoir offert l'opportunité de travailler au sein de l'équipe ASCOLA. Je le remercie d'autant plus pour l'apport d'expérience et la richesse des situations dans lesquelles il m'a permis de travailler.

Je souhaite ensuite remercier Rémi Sharrock, pour sa bonne humeur et son aide précieuse tout au long de mon travail, ainsi que l'ensemble des membres de l'équipe ASCOLA et du département d'informatique de l'école des Mines de Nantes pour leur accueil, leur aide et leurs conseils.

Je remercie aussi mon encadrant, Benoit Parrein, pour l'aide et le suivi tout au long de cet exercice qu'est le stage de fin d'étude. Et pour finir je souhaite remercier Polytech Nantes, dans son ensemble, pour les trois années de formation et d'encadrement dont j'ai pu profiter.

Résumé

Dans un futur où il y aura toujours plus d'appareils électroniques, où leurs capacités seront toujours plus grande et où les interconexions seront systématiques, la maîtrise de la consommation d'énergie sera l'un des grands défis. On le sait, il nous faut rationnaliser l'utilisation de l'énergie si l'on ne veut pas aggraver la situation écologique actuel.

Dans ce rapport de stage, je démontre qu'il est possible d'agir sur la conception des logiciels afin de les rendre plus *vert*, c'est-à-dire moins gourmand en ressources énergétiques. Mon objectif était de montrer la variété des champs d'action sur lesquels on peut agir. J'ai aussi mis en valeur une des grandes difficultés de ce travail : Mesurer la consommation énergétique d'un logiciel.

Mon travail durant ce stage n'a pas apporté une solution finie au problème initialement posé. J'ai plutôt oeuvré à poser des bases pour d'autres projets.

Table des matières

Chapitre 1

Introduction

Ce document est le rapport de mon stage de cinquième année à Polytech Nantes. Je l'écris pour rendre compte du travail que j'ai effectué durant mes 5 mois de stage au sein de l'équipe ASCOLA à l'école des Mines de Nantes.

Ce rapport est découpé en 4 chapitres. Dans le premier chapitre, je présente le contexte de mon stage. Plus précisément, je présente le lieu dans lequel se déroule mon stage, puis je présente le sujet sur lequel j'ai été amené à travailler pendant cinq mois.

Dans la deuxième partie je fais un état de l'art de mon sujet de stage dans lequel je tente de définir les notions essentielles à la résolution du problème qui m'est posé.

Ensuite, dans les deux derniers chapitres, je détaille le contenu de mon travail. Le premier est dédié à mes expérimentations sur la mesure de cosommation d'énergie. Puis, je présente dans le dernier chapitre un cadriciel Java dédié à la gestion d'énergie.

Chapitre 2

Présentation du stage

2.1 Présentation du lieu de stage

Mon stage de cinquième année se déroule au sein de l'équipe ASCOLA du laboratoire LINA, dans les locaux de l'école des Mines de Nantes.

2.1.1 L'école des Mines de Nantes

L'école des Mines de Nantes (EMN) est une école d'ingénieur française sous tutelle du ministère de l'industrie. L'école est rattachée à l'institut Mines-Télécom, au Groupe des écoles des Mines et à la Conférence des grandes écoles.

Elle a été créée en 1990 sur le site de la Chantrerie à Nantes et accueille 850 élèves ingénieurs en formation initiale. Le recrutement est effectué par concours après deux années de classe préparatoire. En 2011, l'EMN a décerné 250 diplômes d'ingénieur.

L'école est séparée en 5 départements axés sur les thématiques des équipes de recherche de l'école. On dénombre les départements :
- Informatique
- Automatique et productique (DAP)
- Systèmes énergétiques et environnement (DSEE)
- Physique subatomique et technologies associées (Subatech)
- Sciences sociales et de gestion (DSSG)

2.1.2 L'équipe de Recherche

Mon stage se déroule dans l'équipe ASCOLA. Cette équipe est installée au sein du département informatique de l'EMN au côté des l'équipes TASC et AtlanMod. L'équipe ASCOLA fait partie du LINA (Laboratoire d'Informatique de Nantes Atlantique) ainsi que de l'INRIA (Institut National de Recherche en Informatique et en Automatique).

Cette équipe regroupe une trentaine de personnes qui travaillent sur les langages d'aspects et de composition. Les axes de recherche, tel qu'ils le sont présentés sur la page dédiée à l'équipe ASCOLA sur le site de l'INRIA [1], sont :

- le développement de nouveaux concepts, de support linguistique, et d'outils pour les applications distribuées permettant de gérer notamment les préoccupations transverses comme la distribution elle-même, les comportements transactionnels et la sécurité ;
- la définition d'un modèle qui intègre de manière transparente composants et aspects, en particulier au travers d'une notion d'interface rendant possible le découplage des composants et des aspects concrets, tout en permettant l'analyse et l'application de propriétés de composition dans un contexte hybride composant/aspect ;
- l'investigation des relations entre langages dédiés, langages d'aspects et langages de composition. Nous comptons exploiter les similitudes entre ces classes de langages dans le cadre du développement de techniques de conception et d'implémentation des langages de manière à faciliter un développement par transformations d'applications efficaces et correctes à partir d'abstraction de programmation de haut niveau ;
- l'étude des fondements de la programmation par aspects et de leurs propriétés de composition au moyen de sémantiques formelles pour les aspects (et les composants) ainsi que des techniques d'analyse, de vérification et de validation correspondantes.

Mon stage se déroule sous la tutelle de Thomas Ledoux qui, avec Jean Marc Menaud, Adrien Lèbre, Rémi Sharrock, ainsi que leurs doctorants, travaillent sur des aspects cloud computing, virtualisation et écologique des systèmes d'informations.

2.2 Présentation du sujet de stage

Dans cette partie je vais décrire ce qu'est, dans la pratique, mon sujet de stage. Je joins en annexe le document original de sujet de stage tel qu'il été écrit au moment de la signature de ma convention.

1. www.inria.fr/equipes/ascola

2.2.1 Contexte

Mon stage se positionne dans les travaux GreenIT de l'équipe ASCOLA. Je travaille donc principalement avec la sous-partie de l'équipe qui effectue sa recherche dans ce domaine. L'équipe travaille déjà sur différents projets portant sur le cloud computing, la virtualisation avec en dénominateur commun le GreenIT. On peut citer par exemple des travaux sur la migration à chaud de machine virtuelle qui ont donné lieu à la création de la startup easyVirt.

2.2.2 Sujet

De nos jours, l'informatique est omniprésente. Que ce soit au travail, à la maison, dans nos manières de nous documenter comme dans nos façons de produire, nous utilisons l'outil informatique. Cette informatique est devenue au fil des années de plus en plus gourmande en énergie. La multiplication des réseaux, des centres de données, des terminaux pour se raccorder au système, tout ceci demande toujours plus d'énergie.

Actuellement des efforts sont fait pour optimiser la couche matérielle afin de la rendre toujours plus efficace. Seulement ce n'est pas suffisant. Afin de limiter l'augmentation certaine de la consommation d'énergie des équipements informatiques, il faut aussi s'attaquer à la source de ce besoin d'énergie, le logiciel. C'est lui, en effet qui a besoin d'effectuer des calculs, des entrées/-sorties, des communications réseaux qui font consommer le matériel.

Des travaux ont bien mis en évidence que, les choix fait lors de la conception d'un logiciel, peuvent être déterminants par rapport à la consommation de celui-ci. Ces choix sont divers et variés. Il peut s'agir de la technologie utilisée, mais aussi de l'architecture choisie pour le logiciel ou encore de l'environnement dans lequel ce logiciel fonctionne.

Le sujet de ce stage est donc de proposer, d'étudier et de valider par l'expérimentation un certain nombre de suggestions qui permettraient de réduire l'empreinte énergétique d'un logiciel.

2.2.3 Objectifs

Le Sujet de mon stage est donc *l'éco-conception* de logiciel. Il s'agit d'étudier les différentes pistes qui s'ouvrent à nous pour rendre plus écologique les systèmes d'informations en jouant sur la composante logicielle de ceux-ci. Mon travail est donc de plusieurs ordres.

Je dois tout d'abord rechercher les travaux existants dans ce domaine, qu'ils soient centrés sur l'éco-conception ou bien positionnés sur des domaines an-

nexes comme l'analyse de code ou bien le cycle de vie du logiciel. Suite à ça, je devrais élaborer un certain nombre de propositions de contribution afin de centrer mon travail à venir. Enfin je devrais expérimenter et valider mes résultats afin de pouvoir émettre des recommandations en terme d'éco-conception.

Chapitre 3

État de l'art

3.1 Introduction

Dans ce troisième chapitre, je vais tenter de faire un état de l'art, le plus complet possible, des sujets abordés durant mon stage. J'ai découpé cet état de l'art en quatre parties. Une introduction dans laquelle je détaillerai des généralités sur le développement durable en informatique. J'entrerai ensuite dans le coeur de mon sujet, en commençant par les méthodes de mesure de la consommation d'énergie d'un logiciel. Puis j'aborderai les notions et concepts que j'ai utilisé durant mon stage pour faire des expérimentations sur l'éco-conception de logiciel. Enfin je terminerai cet état de l'art en présentant une liste non exhaustive de projets en rapport avec le développement durable du coté logiciel.

3.1.1 Pourquoi économiser de l'énergie ?

D'un premier abord, la réponse à cette question parait assez évidente. Cela peut être pour une raison économique, pour une question d'éthique ou encore pour répondre à une contrainte de consommation. Cela dit, si je souhaite commencer cet état de l'art par cette question, c'est que je pense qu'il faut bien réfléchir au besoin d'économie d'énergie. Ici, ce que l'on souhaite faire c'est réduire l'empreinte énergétique d'un logiciel et non pas optimiser un programme. Bien sur, si on améliore un algorithme en terme de complexité ou bien même en rapidité, on réduit le coût énergétique de son éxécution. Mais ceci n'est qu'un aspect de ce que l'on souhaite effectuer dans ce projet.

Cet aspect ne sera pas abordé dans mes travaux car il fait partie d'autres sujets comme l'optimisation d'algorithmes ou l'amélioration de compilateur et ces sujets font l'objet de travaux depuis plusieurs années. De plus, ce que

l'on souhaite dans ce projet, c'est placer le critère de consommation d'énergie au premier plan et pourquoi pas, si cela est nécessaire, dégrader la qualité de service de l'application[6].

On doit donc étudier l'éco-conception de logiciel sous la forme d'un compromis entre la qualité de service et l'économie d'énergie[2].

3.1.2 L'éco-conception logiciel : définition

L'éco-conception n'est pas une notion propre à l'informatique. C'est un terme qui désigne une volonté de concevoir un produit respectueux de l'environement. Actuellement, on parle facilement de l'éco-conception d'un batiment ou bien de celle d'une baie de serveur. Mais parler de l'éco-conception d'un logiciel est assez rare. Certaines entreprises commencent à y réfléchir, à l'image de Facebook qui, devant sa facture d'électricité, a décidé d'abandonner le langage PHP pour le C++, en créant au passage le compilateur HipHop[8]. Cette initiative a permis à la société de diviser sa consommation d'électricité par deux.

Ce genre d'exemple reste pour l'instant assez rare. Il doit ammener les géants de l'informatique à mieux réfléchir au choix de conception de leurs produits. Cependant, il ne faut pas limiter l'éco-conception de logiciel à la maitrise de la consommation à l'exécution. La réduction de l'empreinte énergétique peut être vue de plusieurs façons, qui sont autant de sujet de reflexion pour notre projet :

- La baisse de la consommation d'énergie à l'exécution/Augmentation de l'autonomie des appareils mobiles ;
- La baisse des coûts énergétiques de développement et de maintenance ;
- L'allongement de la durée de vie du matériel sur lequel est exécuté le logiciel ;
- L'allongement de la durée de vie du logiciel.

L'analyse du cycle de vie

L'analyse du cycle de vie (ACV) est l'un des outils principal utilisé par l'industrie dans l'éco-conception d'un produit. Il se base sur le cycle de vie de celui-ci de sa conception à sa mise en déchet en passant par sa fabrication et son utilisation (figure 3.1). On utilise ensuite un modèle pour transposer ces étapes en un impact environnemental.

La particularité du produit qu'est le logiciel par rapport aux produits standards de l'industrie (la dématérialisation en particulier) fait qu'il est difficile d'appliquer un modèle existant pour effectuer une ACV d'un logiciel. Ce n'était pas l'objet de mon stage, je n'ai donc pas travaillé à l'élaboration

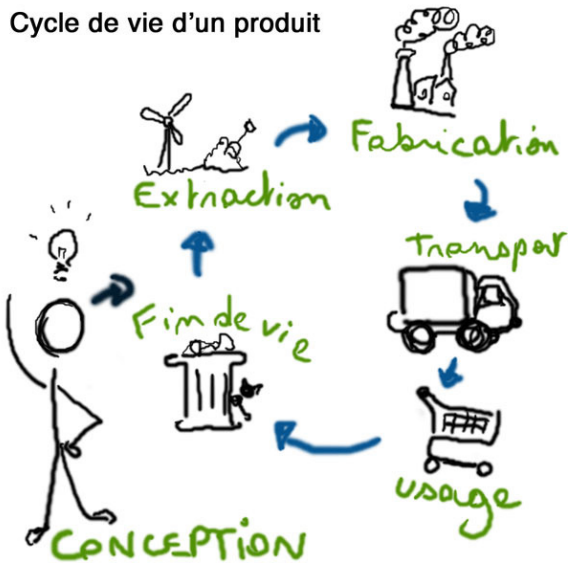


FIGURE 3.1 – Cycle de vie (source : wikipedia, auteur : DamarisBasileJudith)

d'un modèle. En revanche, j'ai tenté de toujours raisonner en terme de cycle de vie et non en consommation à l'exécution.

3.2 La mesure de consommation d'énergie

Dans cette partie de l'état de l'art je vais parler des différentes techniques pour extraire et utiliser une information sur la consommation d'un logiciel. J'ai pu identifier trois méthodes différentes pour mesurer l'énergie consommée. Elles ont chacunes leurs avantages et leurs défauts.

3.2.1 La mesure par un appareil externe

La première méthode, la plus commune pour mesurer les resources consommées par un appareil électrique, est de placer un wattmètre en série sur l'alimentation de cet appareil. Avec un wattmètre récent on peut mesurer un ensemble de paramètre tels que les watts, les watts/heure, les volts et les ampères. De plus, la plupart des outils récents proposent un système d'enregistrement et/ou de transfert des données vers un ordinateur.

Par exemple, les *plugwizes* sont des modules qui se placent sur la prise sont capables de couper l'alimentation de l'appareil branché dessus et de mesu-

FIGURE 3.2 – Photo d'un watts up pro .net

rer la consommation électrique de celui ci. Ils sont capable de transmettre ces informations sans fil jusqu'à un ordinateur. Un autre exemple de ce type de matériel, les wattmètres de la marque *watts up ?*. Ils permettent, suivant leur gamme, de collecter de l'information et de la transmettre. Le *watts up ? pro .net* (figure 3.2) dispose d'un système d'enregitrement de, au maximum, 120000 mesures de watts et de deux interface de communication, USB et Ethernet.

Cette technique de mesure n'offre qu'une vue globale de notre système. Il est impossible par cette méthode d'identifier de manière certaine la consommation d'un logiciel. On peut par contre corréler un changement de consommation avec un événement survenu[5] (le démarrage d'une application, le lancement d'une tâche gourmande en traitement...).

3.2.2 La mesure par un logiciel interne

La deuxième méthode consiste à analyser les ressources utilisées par un programme et, à partir d'un modèle de consommation, d'en déduire la consommation énergétique de l'application[3]. Différentes ressources peuvent être ciblées. On peut les classer en deux catégories : les unités de calcul (CPU, GPU...) et les dispositifs d'entrées/sorties (disque dur, interface réseau...).

Dans cet article[7], les auteurs détaillent le modèle de consommation qu'ils ont mis au point pour leur application de mesure *PowerAPI*. On peut y voir qu'ils obtiennent des résultats intéressants par rapport à la mesure faite en parallèle sur un wattmètre. L'inconvénient de ce type de méthode est qu'il faut calibrer le modèle. Cela est nécessaire car chaque processeur fonctionne différemment et donc ne consomme pas la même chose. De plus ce type de méthodes est intrusive et peut perturber le fonctionnement du logiciel.

En revanche cette technique permet facilement d'isoler un logiciel en particulier (ou plutôt un processus). C'est donc une technique intéressante pour mon projet.

3.2.3 Vers une mesure par un appareil interne

Enfin une troisième technique est à envisager dans un futur plus ou moins proche. Comme l'explique l'article *Looking back on the language and hardware revolutions : measured power, performance, and scaling*[4], les fabricants de matériel intègrent, de plus en plus, dans leurs composants, des

mécanismes de mesure afin d'adapter leurs fonctionnements à leurs consommations. À la fin de l'article, les auteurs recommandent l'ouverture de ces mécanismes au travers d'API pour que les développeurs puissent y accéder.

Si les constructeurs acceptent de faire cet effort, on pourra facilement savoir ce que consomme réellement une application en temps réel. Ce type de mécanisme serait idéal dans mon projet car il permettrait d'avoir une consommation réelle de l'application mesurée. On pourrait même concevoir, très facilement, des applications qui s'adaptent en fonction de leur consommation.

3.3 Fabriquer un éco-logiciel

Cette partie ne va pas expliquer comment fabriquer à coup sûr un éco-logiciel. Je souhaite seulement y présenter certains concepts importants pour appliquer le développement durable au logiciel.

3.3.1 De la Conception...

La conception d'un éco-logiciel doit être particulièrement bien pensée. C'est lors la conception que l'on prend la majorité des décisions qui entraineront une sur-consomation par la suite. Il m'est difficile de lister l'ensemble des questions à se poser lors de cette phase. Je ne citerais que quelques points importants.

La première question à se poser est celle de l'architecture et de la localisation des traitements[8]. Des traitements effectués dans un Centre de données auront un coup plus faible car un centre de données dispose souvent de mécanisme d'optimisation énergétique. On peut citer en exemple le *Green Challenge 2010*[1] ou les équipes devaient travailler sur une application de décodage de QRcode et de géolocalisation pour la rendre la moins consommatrice possible. Les gagnants ont choisi de déplacer au maximum les traitements du coté serveur.

Une autre question importante à se poser, est celle de la configuration par défaut[1]. En effet, une majorité d'utilisateurs laisseront l'application avec son mode de fonctionnement par défaut. Il peut donc être intéressant que l'application conçue effectu des traitements de basse qualité par défaut.

On pourait dresser une liste importante de ce qu'il faut ou ce qu'il ne faut pas faire. En résumé, il faut réfléchir en terme d'écologie chaque choix fait durant la conception.

1. https ://sites.google.com/a/octo.com/green-challenge/

3.3.2 ... À la programmation

La programmation d'application « verte »est encore plus dur à définir que la conception. Il n'y a pas de méthode miracle. Il faut avoir une bonne connaissance du langage, bien choisir les mécanismes utilisés (boucle, saut conditionnel...). Le plus simple ici est de s'assister d'outils d'aide à la programmation. Il n'en existe pas encore de suffisament évolué dans l'éco-conception, mais beaucoup d'optimisation proposée par des outils comme Sonar [2] réduisent l'empreinte énergétique d'un code.

Durant mon stage j'ai pu réaliser plusieurs petits programmes pour mettre en évidence ce que peut être l'impact d'un mauvais choix fait à la programmation. Par exemple en C, le choix fait sur l'ordre de déclaration des élements d'un structure à un impact sur la place en mémoire de cette structure. Voici un exemple :

```
struct bad_struct{
    char c;
    double d;
    short s;
    int i;
};
```

```
struct good_struct{
    double d;
    int i;
    short s;
    char c;
};
```

La structure de gauche occupe en mémoire 24 octets, celle de droite en occupe seulement 16. Cette différence est du a l'alignement que tente de faire le compilateur. Il rajoute donc des bit de bourrage entre certaine variable. Dans notre exemple la structure *bad_struct* est composé de : 1 octet (char) + 7 octets de bourrage + 8 octets (double) + 2 octets (short) + 2 octets de bourrage + 4 octets (int) = 24 octets. La structure *good_struct* est, elle, composé de : 8 octets (double) + 4 octets (int) + 2 octets (short) + 1 octet (char) + 1 octet de bourrage = 16 octets.

Voici un autre exemple, cette fois en java :

```
String s = "";
for(i=0; i<time; i++) {
    s+="";
}
```

```
String s = "";
StringBuffer buf = s;
for (i=0; i<time; i++) {
    buf.append("");
}
s = buf.toString();
```

2. http ://www.sonarsource.org/

Ici encore, le fonctionnement du langage est en cause. La version de gauche fonctionne sans problème mais la version de droite est approximativement 10 fois plus rapide [3]. La différence est simple. Dans la première version, la concaténation sur la chaine est remplacé par la création d'un objet *StringBuffer*, l'appel à la méthode *append()* et la re-transformation en *String*.

D'autres exemples existe comme le dépliage de boucle[8] qui permet au mécanisme d'anticipation des processeurs de mieux fonctionner.

3.3.3 La programmation Modulaire

Intéret de la programmation modulaire dans l'économie d'énergie

La programmation modulaire présente beaucoup d'avantage pour l'éco-conception de logiciel. Dans l'absolu, un programme modulaire est plus facile à maintenir et a une durée vie potentiellement plus longue qu'un programme standard. Comme ses modules sont indépendant les uns des autres, il suffit de remplacer la brique deffectueuse pour réparer un problème. De même, l'évolution du programme se fait par l'ajout de nouvelles briques.

D'autres aspects intéressants pour l'éco-conception sont à prendre en compte. Si dans une application modulaire l'utilisateur ne se sert pas de toutes les fonctionnalités, on doit pouvoir retirer les modules qui ne servent à rien. Cela peut potentiellement économiser de l'énergie à l'exécution mais surtout alléger considérablement le logiciel et donc allonge la durée de vie de l'ordinateur.

OSGi : du java modulaire

OSGi [4] est une cadriciel java élaboré par l'OSGi Alliance. C'est un consortum formé en 1999 composé entre autre d'IBM, Motorola,Oracle ou encore Samsung. Le but de ce cadriciel est de faciliter la création d'application modulaire en Java.

Différentes implémentations de ce framework ont été développés à partir de la spécification produite par l'OSGi Alliance. Les plus connu sont Equinox développé par l'équipe du projet Eclipse, Felix issu du projet universitaire Oscar et maintenu par Apache ou encore Knopflerfish élaboré par MakeWave.

3. Comparaison faites sur plusieurs essais chronométré et avec time = 99999999
4. Open Services Gateway initiative

3.4 Travaux connexes

Dans cette section je vais présenter les projets en rapport avec mon sujet de stage et dont j'ai soit entendu parlé soit pu rencontrer les acteurs.

3.4.1 Le Green code Lab

Le green code lab est un groupement de personne souhaitant promouvoir le développement durable dans le logiciel. C'est une communauté virtuelle qui se charge de publier des articles d'informations, d'organiser des évenements et des formations autour de l'éco-conception de logiciel.

Leur site internet (www.greencodelab.fr) sert de plateforme de publication de leurs articles. Ils ont aussi mis en place un GitHub qui sert de laboratoire pour effectuer des expérimentations. Enfin le green code lab a publié un livre intitulé *Green Pattern - Manuel d'éco-conception des logiciels* et disponible à la vente au format papier ou numérique.

Durant mon stage, j'ai eu l'occasion de me rendre à l'un de leurs évenements sur le thème de la mesure de la consommation électrique d'un logiciel. Là-bas, Olivier Philippot, un membre actif du green code lab, nous a présenté une méthodologie basée sur des *plugwizes* en affichant les mesures au moyen du logiciel KST [5]kst-plot.kde.org/.

3.4.2 Le projet code vert

Code Vert est un projet labélisé par le pôle de compétitivité Images et Réseaux qui a débuté en Février 2012 (en même temps que mon stage) et durera 2 ans. Il regroupe 4 acteurs : l'école d'ingénieur ICAM, les sociétés KaliTerre et TOCEA, ainsi que la ssii SIGMA Informatique.

L'objectif de ce projet est de mettre en place un référentiel pour juger de l'éco-conception du code d'un logiciel. Cela passera par une analyse statique du code qui sera noté. On pourra ressortir de cette analyse des recommandations qui permettront de l'améliorer.

3.5 Conclusion

Voici que se termine mon état de l'art. Avec les connaissances présentées dans ce chapitre j'ai pu réaliser un certain nombre de choses que je vais maintenant présenter.

5. http ://kst-plot.kde.org/

Chapitre 4

Comment peut-on mesurer la consommation d'un logiciel ?

Dans ce chapitre je ne vais m'intéresser qu'à la consommation du logiciel lors de l'exécution. Je ne prendrai pas en compte les différentes étapes du cycle de vie. Voici l'ensemble des logiciels que j'ai pu expérimenter pour mesurer la consommation énergétique d'un programme. Pour chacun, je note mes observations sur son fonctionnement, son installation ou toute autre remarque pouvant servir. Afin de mieux organiser mes notes, je sépare c'est programme en deux parties. D'abord ceux inapropriés à mon projet mais que j'ai quand même essayé. Puis ceux pouvant apporter une solution à la problématique d'identifier la consomation d'un logiciel.

4.1 Logiciels inapropriés

4.1.1 Energy Checker

Le premier que sur lequel je me suis renseigné est *Energy Checker*[1]. Le projet *Energy Checker* développé par Intel, est un ensemble d'outils destiné à faciliter la remontée et l'affichage des mesures faites sur la consommation d'un programme. Il se compose principalement de l'API *Energy checker* qui permet d'utiliser les mécanismes de remonter d'information (PL - Productivity Link).

Je n'ai pas approfondi mes recherches sur ce logiciel car il est difficile à mettre en place et n'apporte pas de solution sur la mesure de consommation

1. software.intel.com/en-us/articles/intel-energy-checker-sdk/

d'énergie. Ne voulant pas perdre trop de temps (la mesure de consommation d'énergie n'était pas le coeur de mon sujet), je suis passé à d'autre solution.

4.1.2 PTop

PTop [2] est un programme de collecte d'information sur l'exécution de processus sous linux. Il est censé récupérer les données sur l'utilisation du CPU, de la RAM et des échange réseaux. Il doit ensuite être capable, une fois le modèle paramétré, sortir des informations sur la consommation énergétique.

Malheureusement, il s'est avéré que pTop ne fonctionne pas entièrement. Après plusieurs essais infructueux, j'ai supposé que le logiciel avait été abandonné sans être fini.

Procédure d'installation

- récupérer les sources sur http ://mist.cs.wayne.edu/ptop.html
- installation du nécessaire pour compiler les sources (paquets debian) :
  - ◇ build-essential
  - ◇ libmysqlclient-dev
  - ◇ libncurses-dev
- Installation de la base de données Mysql :
  - ◇ création d'une base de donées ptop
  - ◇ exécution du script db_schema.sql
  - ◇ remplacement du login et password pour accéder à la base de données dans le fichier database.c
- Compilation :

```
make
chmod +x ptop
```

Exécution

Lancer le programme au moyen de la commande :

```
./ptop
```

Le programme va s'initialiser et enregistrer les mesures dans les bases de données. Il faut ensuite interrompre les mesures au moyen du signal SIGINT.

2. mist.cs.wayne.edu/ptop.html

Données de sortie

Il n'existe pas, à ma connaissance, de documentation précise sur les données de sortie du programme pTop. Le programme stocke les données dans une base de données Mysql nommé ptop. Elle est composée de quatre tables :
- device_energy
- process_energy
- process_info
- sys_info

Seules les deux dernières tables sont remplies par le programme (à cause soit d'un bug, soit d'une version non fini du programme).

4.2 Logiciels appropriés

4.2.1 ClassMexer

ClassMexer [3] est un API java permettant d'estimer la taille en mémoire d'un objet java. On peut télécharger cet API sous la forme d'un fichier JAR sur son site internet. ClassMexer est un *Instrumentation agent* comme spécifié dans le pattern Instrument [4]. Cela implique de travailler avec, au minimum, la version 5 de java.

procédure d'installation

- Télécharger l'API
- Importer le fichier JAR dans le projet.
- Insérer des appels sur l'objet que l'on souhaite mesurer. Par exemple :

```
public class Main {
    public static void main(String[] args) {
        String str="Hello world !";

        long noBytes=MemoryUtil.deepMemoryUsageOf(str);

        System.out.println(noBytes);
    }
}
```

- Exécuter le programme avec l'option -javaagent :<fichier JAR>

3. www.javamex.com/classmexer/
4. docs.oracle.com/javase/6/docs/api/java/lang/instrument/package-summary.html

4.2.2 JouleMeter

JouleMeter[5] est un outil développé par Microsoft et qui sert à mesurer la consommation d'énergie sur un ordinateur. Il ne fonctionne que sous Windows 7.

Procédure d'installation

- Télécharger le logiciel sur son site.
- Exécuter le programme.

Fonctionnement

JouleMeter est un logiciel de mesure qui se base sur un modèle de consommation. La première étape est donc de générer ce modèle. Deux méthodes existent.

La première n'est utilisable que sur un ordinateur portable muni d'une batterie. Il suffit de débrancher l'alimentation et de lancer la calibration (15 minuntes sans utiliser le PC). Avant de lancer la calibration, il faut faire attention à arrêter toutes les applications afin que le logiciel puisse déterminer une consommation minimum en mode "Idle".

Le deuxième mode nécessite un Wattmètre WattsUp pour générer le modèle. Il suffit de le brancher en USB, de couper toutes les applications possibles et de lancer la calibration (3 minutes sans utiliser le PC).

Expérience

Afin de tester la fiabilité du modele créé, j'ai effectué différentes expériences en comparant les résultats obtenu par *JouleMeter* avec ceux lu sur un wattmètre. J'ai pu mettre en évidence de grosses faiblesses qui rendent peu fiable les résultats obtenu. La figure 4.1 met en évidence ces faiblesses. J'ai obtenu ces mesures en laissant fonctionner *JouleMeter* sur un ordinateur, démaré mais inutilisé, durant toute une nuit. En parallèle, j'ai récupéré les mesures que prenait un wattmètre *WattsUp pro.*

Les deux coubes affichent un échantillon de 5000 mesures (durant approximativement 5000 secondes). Le modèle de consommation sur lequel se basait *JouleMeter* a été généré juste avant le début de l'expérience. D'un point de vue général, on peut déjà remarquer que la courbe de *JouleMeter* est très irrégulière et majoritairement en dessous de celle du watttmètre. Cela est

5. research.microsoft.com/en-us/downloads/fe9e10c5-5c5b-450c-a674-daf55565f794/

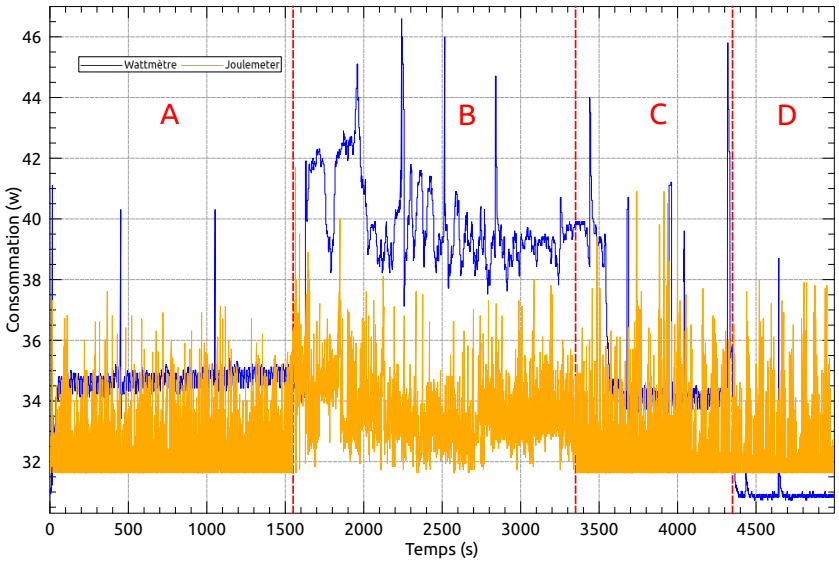


FIGURE 4.1 – Mesures faites par Joulemeter comparé à celles d'un wattmètre

principalement du, au fait que l'impact du processeur est largement sous-estimé dans le modèle.

Ensuite on peut noter un seuil en dessous du quel *JouleMeter* ne descend pas (ici, 31,6 watts). On constate même que les mesures du wattmètre passe en dessous de celle de *JouleMeter* dans la partie *D* du graphique. Cet effet de seuil est engendré par le modèle qui considère que le système consomme toujours une constante quand il est allumé. Donc, quelle que soit la charge du système, *JouleMeter* considère qu'il consomme au minimum 31,6 watts (valeur qu'il a mesuré lors de l'établissement du modèle).

Enfin, et tout n'est pas négatif, on peut remarquer que dans la zone B *JouleMeter* réussi à mesurer une hausse de la consommation qui peut être noté sur la courbe du wattmètre (avec un certain décalage cependant).

4.2.3 PowerAPI

PowerAPI est un outil sur lequel travail l'équipe ADAM de l'université de Lille. Les premiers résultats de cet outil ont été présentés dans un papier[7] publié à l'ICSE [6] 2012 à Zurich. Cet outil parait prometteur et s'il continue à

6. International Conference on Software Engineering

être développé, pourrait bien apporter une solution intéressante au problème de la mesure de consommation énergétique de logiciel.

Selon la présentation qui en a été faite, il fonctionne actuellement pour la consommation CPU. Cet outil collecte le temps CPU d'un processus et la fréquence de celui-ci en temps réel. Puis il détermine l'énergie consommée grâce à des tables de consommations par rapport à la fréquence fournis par les fabriquant de processeur.

Je n'ai malheureusement pas essayé *PowerAPI* par manque de temps. Je n'ai eu accès au programme que tardivement durant mon stage et à ce moment je travaillais sur d'autres aspects.

4.3 Conclusion

Pour conclure ce chapitre, je dirais qu'il est difficile pour l'instant de mesurer éfficacement la consommation d'un logiciel. Le seul outil que j'ai pu réélement tester sur un logiciel est *JouleMeter*. J'ai montré dans mes expériences que, même si on peut identifier certains évenement sur la consommation, il est impossible de ce fier aux relevés obtenu.

Je pense que *PowerAPI* apportera une solution intéressante. Il faudra suivre son développement avec intérêt.

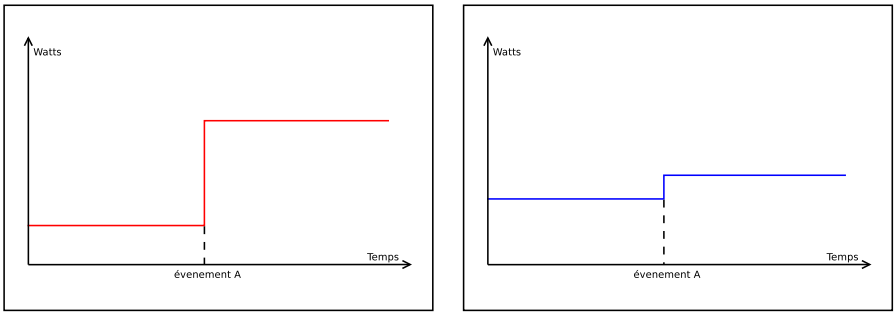
Chapitre 5

La programmation modulaire au service de l'économie d'énergie

Afin de mettre en valeur l'importance de l'architecture logicielle dans la consommation, j'ai proposé de développer une application basée sur la technologie java OSGi [1]. L'objectif de ce projet est de pouvoir réaliser facilement des démonstrations d'application modulaire qui s'adapte en fonction de contrainte énergétique. J'ai donc choisi d'élaborer un cadriciel qui permettrait d'échanger une partie de l'application (dans notre context un « bundle »OSGi) contre une autre afin d'optimiser la consommation. Afin d'expliquer plus clairement le but du projet, je vais détailler un cas d'utilisation du cadriciel.

Un programme doit réaliser une tache. Le développeur connait deux algorithmes pour coder cette tache. Ils ont chacun leurs forces et leurs faiblesses. La version A du code consomme peu d'énergie en temps normal (figure 5.1a). Mais, dans certaines conditions, ici quand un évènement A se produit, l'algorithme se met à surconsommer. La version B du code, elle, consomme plus d'énergie (figure 5.1b) que la version A. Par contre il est moins sensible à l'arrivé de l'évenement A. Donc pour réaliser cette tâche en économisant le plus d'énergie il faudrait exécuter le code A en temps normal, et le code B quand l'évenement A se produit. La solution de mettre les deux algorithmes dans le même programme n'est pas une bonne solution. En faisant de la sorte, on alourdit le programme, ce le qui rend plus exigent en ressource et donc avance l'obsolescence des machines. De plus, cela rend la maintenance du code plus difficile.

1. www.osgi.org



(a) Consommation du module A (b) Consommation du module B

FIGURE 5.1 – Illustration de l'optimisation de la consommation par l'échange de module

Le cadriciel que j'ai développé a pour but de remplacer à chaud le code A contenu dans un *bundle OSGi* par le code B contenu dans un autre *bundle* lorsque l'évènement se produit. Il y a d'autres cas d'utilisation que l'on pourrait imaginer telle que la mise en place d'une politique énergétique sur une machine (par exemple réduire la consommation d'une machine utilisant l'énergie solaire pendant la nuit).

5.1 Présentation de l'infrastructure développée

Cette infrastructure est entièrement basée sur OSGi. Elle a donc été conçue sous la forme de modules indépendants les uns des autres. On peut voir sur la figure 5.2 les modules ainsi que les liaisons entre ces modules. Une flèche d'un module A vers un module B signifie que le module A à besoin de B pour pouvoir fonctionner. Dans cette section je vais détailler l'utilité, voir le fonctionnement de chacun des modules.

5.1.1 Le coeur du système

La partie centrale du cadriciel se trouve dans le package *org.oep.core*. Il se découpe en deux classes qui gèrent l'application que l'on souhaite faire fonctionner. La première est la classe *BundleManager*. C'est elle qui s'occupe de gérer les modules qui forment l'application. Elle permet d'installer deux types de modules. Ceux définissant un service et ceux rendant un service. C'est ensuite par cette classe que l'on démarre ou arrête un module offrant un service. Le *BundleManager* garantit qu'il n'y ait qu'un et un seul module démarré par service.

La deuxième classe, *ServiceManager*, s'occupe de référencer les services dé-

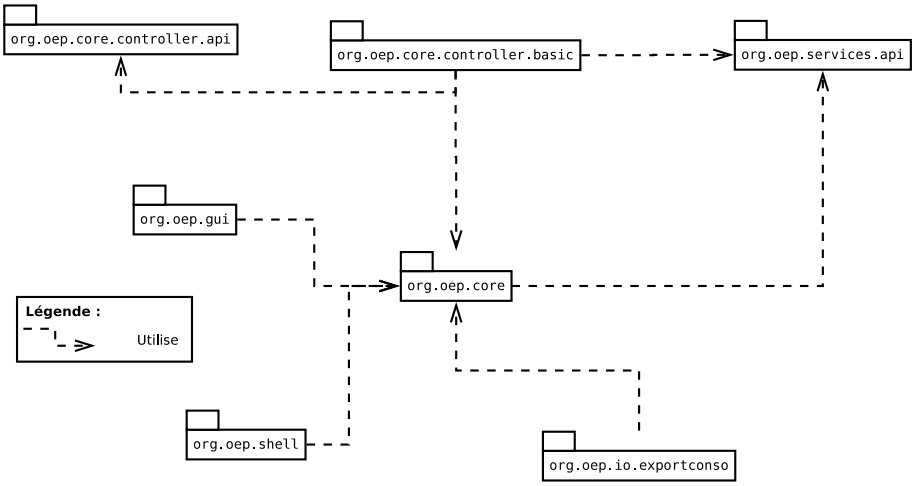


FIGURE 5.2 – OSGiEcoFramework : diagramme de liaison des modules

marrés. Pour cela elle implémente l'interface *ServiceListener*. Elle permet ensuite de consulter leur consomation au travers de l'interface *EcoService*.

5.1.2 Les interfaces

J'ai développé deux types d'interfaces pour ce framework. Une première graphique, basé sur la technologie *Swing*, l'autre en ligne de commande. L'interface graphique est découpée en deux vues. Une pour installer des services (figure 5.3) et l'autre pour démarrer ou arrêter les services ainsi qu'afficher la consommation (figure 5.4).

L'interface en ligne de commande est basé sur le shell felix[2]. J'ai défini seulement trois commandes :
- une pour installer une famille de module
- une pour installer un module
- une pour obtenir la consommation global

Pour obtenir un shell complet, il faudrait ajouter à cela une commande pour démarrer, une commande pour arrêter et une commande pour obtenir la consommation d'un module. Je n'ai pas écrit ces commandes car j'ai choisi de privilégier l'interface graphique qui me paraissait plus pertinente pour faire des démonstrations.

En plus des interfaces utilisatrices, j'ai conçu un module qui s'occupe d'exporter les données de consommation vers un fichier *CSV*. Cela permet de faire facilement des sauvegardes d'historique de consommation ou bien d'afficher en temps réél une courbe au moyen du logiciel KST.

2. http ://felix.apache.org/site/apache-felix-shell.html

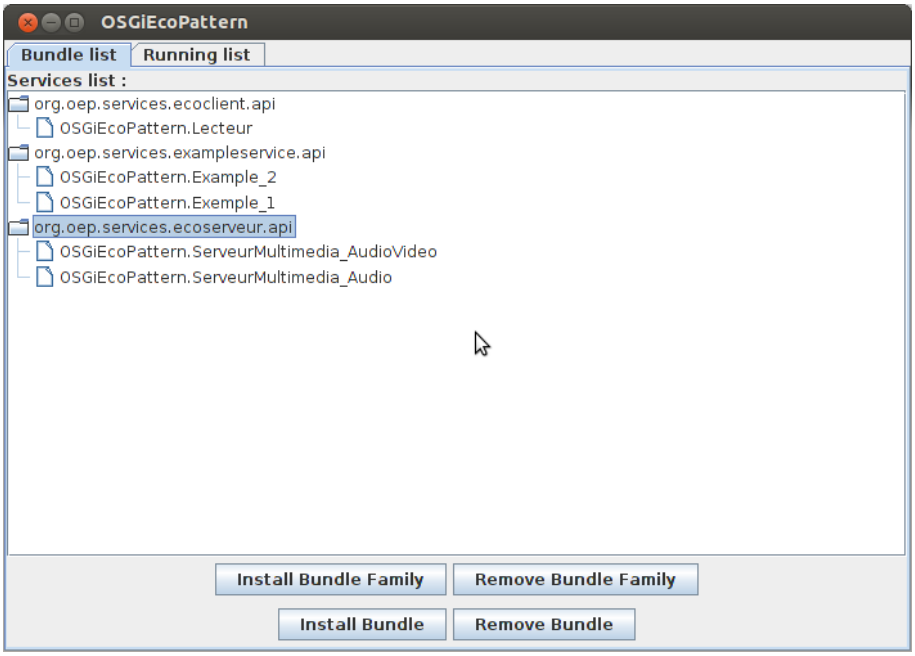


FIGURE 5.3 – Interface d'installation des services

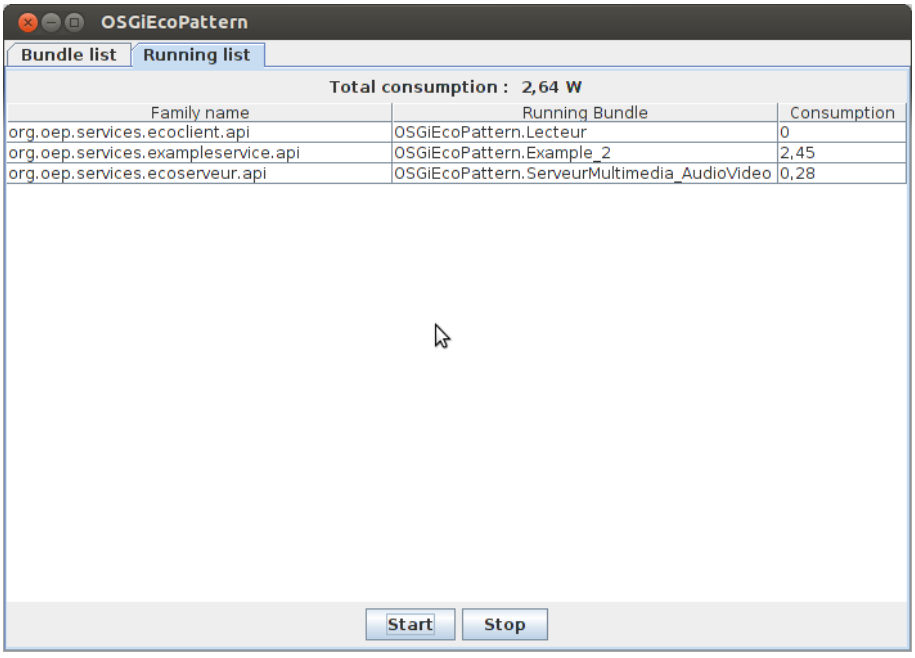


FIGURE 5.4 – Interface de gestion des services démarrés

5.1.3 Le controleur

Le controleur est la partie sur laquelle j'ai pu le moins travailler. Il est charger de détecter des changements dans la consommation des modules et de déclencher une action de reconfiguration si besoin. Je souhaitais utiliser pour cela l'API *Wildcat*[3]. C'est un projet de l'OW2 consortium qui a pour but de fournir un cadriciel permettant la mise en place de sondes et de déclencher des évènements au moyen de requêtes écrites dans un langage proche du SQL.

J'ai abandonné cette solution après avoir passé plusieurs jours à tenter d'intégrer Wildcat dans un module OSGi sans y arriver (ceci à cause de problème de dépendance à d'autres bibliothèques). J'ai donc finalement opté (un peu dans l'urgence) pour une solution plus basique qui vient tester tour à tour chaque module pour vérifier qu'il ne dépasse pas une limite de consommation. Cette version du controleur ne permet pas de mettre en place une politique de consommation élaborée, mais cela m'a permis de faire fonctionner le cadriciel dans le temps imparti pour mon stage. Étant donné qu'il est totalement indépendant du reste du cadriciel, on peut facilement le remplacer par un autre plus complexe.

5.1.4 L'application

L'application que l'on souhaite faire fonctionner au sein de ce cadriciel doit être basé sur la notion services. Afin d'être pris en compte, les modules de cette application doivent implémenter l'interface *EcoService* présent dans le module *org.oep.services.api*. Cette interface ne définit qu'une seule méthode :

```
public double getConsumption();
```

qui sert à récupérer la consommation d'un module. L'interface sert aussi à catégoriser les modules comme étant intégré au cadritiel.

5.2 Résultats obtenus

Afin d'illustrer les résultats obtenu par ce framework, j'ai développé une application exemple. Elle est composée de deux modules qui sont interchangeable. Ces modules ne font rien à part augmenter de façon artificielle leurs consommations en partant de 2 watts. Durant l'expérience, le controleur est programmé pour remplacer un module dès que celui ci dépasse une consommation de 3 watts.

3. http ://wildcat.ow2.org/

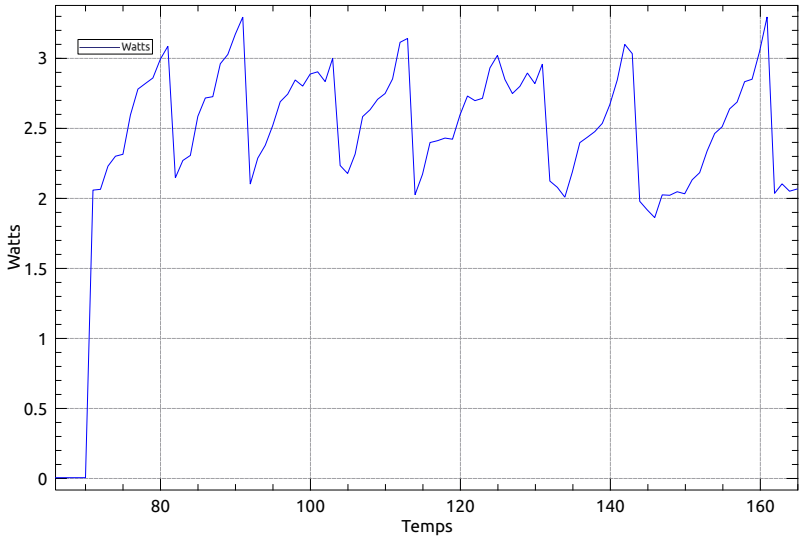


FIGURE 5.5 – Courbe de la consommation du programme Exemple

Ce prototype d'application provoque donc l'échange régulier des deux modules entrainant par la même une baisse de la consommation. On peut voir le résultat sur cette courbe de consommation (figure 5.5) ou l'on constate bien que régulièrement la courbe chute brutalement. Ces chutes correspondent à l'échange de module.

5.3 Développer une application pour ce framework

Développer une application pour ce cadridiel est avant tout développer une application sous OSGi. Je ne détaille pas ici les principes de cette technologie (je l'ai déjà abordé dans mon état de l'art). On doit donc développer des modules ou *bundle*. Deux types de modules doivent être fait pour que le cadriciel fonctionne :

- ceux définissant une famille de modules
- ceux appartenant à une famille de modules

L'application Exemple développé dans la section précédente permet d'illustrer simplement l'utilité de ces deux type de modules. On peut voir sur la figure 5.6 les modules qui composent cette application. Comme son nom l'indique, le module *Famille_Exemple* défininit la famille de modules auquelle appartiennent les deux autres modules.

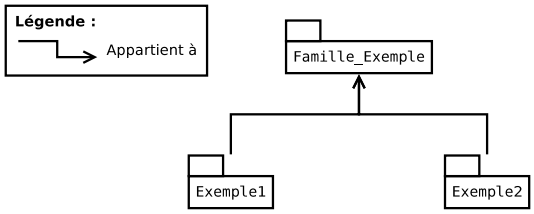


FIGURE 5.6 – Diagramme des modules de l'application Exemple

Pour voir plus en détail comment faire, le module *Famille_ Exemple* contient une interface héritant de *EcoService*. Les deux autres modules contiennent sensiblement la même chose, à savoir une classe implémentant l'interface définit dans le module *Famille_ Exemple*.

5.4 Un exemple d'application : lecteur audio

Afin d'illustrer les possibilités de ce framework, j'ai développé un exemple concret d'application. Cette application est un simple lecteur multimedia. Sa particularité est que quand on a besoin d'économiser l'énergie il n'affiche plus l'image et ne diffuse que le son du fichier numérique que l'on souhaite lire. Ce lecteur est basé sur la bibliothèque Java Xuggle [4].

L'architecture de cette application (figure 5.7) est divisé en deux familles de modules. La première, définit par le module *Famille_ Lecteur*, fournit l'interface graphique du lecteur multimedia. Pour cet exemple, je n'ai développé qu'une version contenu dans le module *Lecteur*. La deuxième famille est celle définit dans le module *Famille_ ServeurMultimedia*. Elle sert à charger, décoder et lire le fichier multimédia. Pour cette famille j'ai développé deux modules. Le module *LecteurMultimedia_ AudioVideo* qui décode et affiche les flux audio et vidéo du fichier. L'autre, le module *LecteurMultimedia_ Audio*, ne s'occupe que de la partie Audio du fichier.

Au final, nous avons donc un module qui rend une qualité de service supérieur mais qui consomme plus (11,5 watts mesuré avec un wattmètre) et l'autre qui dégrade la qualité de service, en supprimant le traitement de la vidéo, pour consommer moins (4,2 watts mesuré avec un wattmètre).

5.5 Conclusion

Pour conclure ce chapitre, le cadriciel que je souhaitais développer fonctionne. Il reste cependant basique est peut être amélioré sur beaucoup de

4. www.xuggle.com

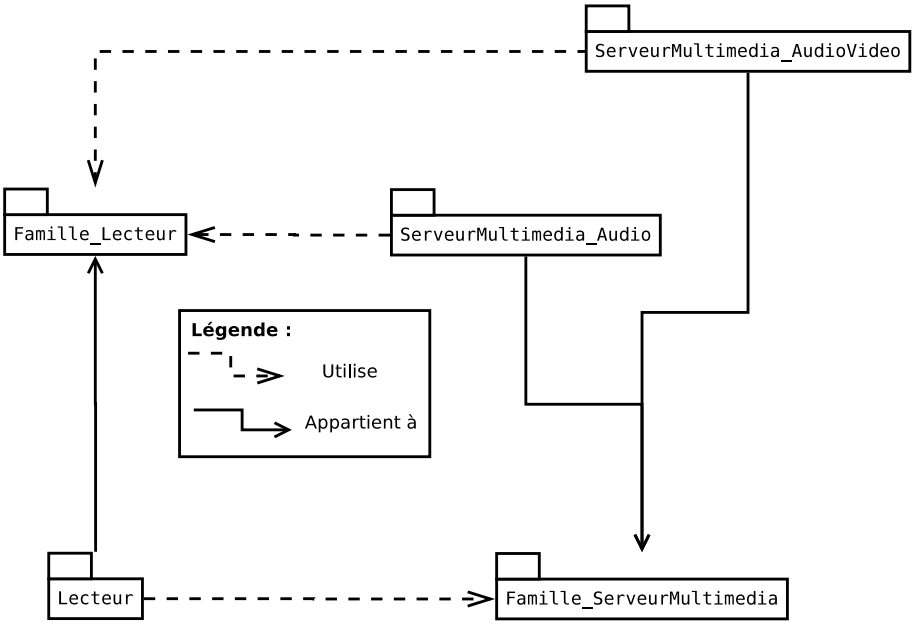


FIGURE 5.7 – Lecteur multimedia : Diagrammame de liaison des modules

points. En premier plan, le controleur développé reste très primitif. Il est possible de l'améliorer de bien des façons et pourquoi pas d'en créer plusieurs applicant différentes stratégies. L'utilisation de *Wildcat*, comme il était initialement prévu, donnerait de larges possibilités au cadriciel.

Un autre axe d'amélioration serai de travailler sur la mesure d'énergie d'un bundle. Ma solution reste très statique et n'apporte pas la finnesse souhaité dans un tel type de projet.

Chapitre 6

Conclusion

Voici donc l'ensemble du travail que j'ai pu effectuer durant mon stage de fin d'étude. Il reste encore beaucoup de travail pour pouvoir obtenir des résultats concrets et enfin pouvoir dire que l'on sait éco-concevoir un logiciel. Cinq mois furent trop courts pour pouvoir développer l'ensemble des idées que je souhaitais aborder. J'ai du faire des choix et cibler mon étude.

Pour moi, ce fut un travail passionnant. Je suis heureux d'avoir eu l'opportunité de travailler dans de telle condition. J'ai pu apprendre beaucoup de nouvelles technologies à commencer par OSGi ou Wildcat. J'ai aussi beaucoup appris sur des mécanismes de systèmes et sur le fonctionnement de certains langages. J'ai débuté mon stage en étant convaincu de l'utilité de ce type de démarche, j'en ressors en pensant que c'est une nécéssité.

Bibliographie

[1] *Green Patterns - Manuel d'éco-conception des logiciels.* Green Code Lab, 2012.

[2] Woongki Baek and Trishul M. Chilimbi. Green : a framework for supporting energy-conscious programming using controlled approximation. *SIGPLAN Not.*, 45(6) :198–209, June 2010.

[3] G. Da Costa and H. Hlavacs. Methodology of measurement for energy consumption of applications. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 290–297. IEEE, 2010.

[4] Hadi Esmaeilzadeh, Ting Cao, Yang Xi, Stephen M. Blackburn, and Kathryn S. McKinley. Looking back on the language and hardware revolutions : measured power, performance, and scaling. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ASPLOS '11, pages 319–332, New York, NY, USA, 2011. ACM.

[5] Abram Hindle. Green mining : A methodology of relating software change to power consumption. In *ICSE 2012 - New Ideas and Emerging Results*, Zurich, Suisse, 2012.

[6] Alexandre Mello Ferreira, Kyriakos Kritikos, and Barbara Pernici. Energy-aware design of service-based applications. In *Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, ICSOC-ServiceWave '09, pages 99–114, Berlin, Heidelberg, 2009. Springer-Verlag.

[7] Adel Noureddine, Aurélien Bourdon, Romain Rouvoy, and Lionel Seinturier. A Preliminary Study of the Impact of Software Engineering on GreenIT. In *First International Workshop on Green and Sustainable Software*, Zurich, Suisse, 2012.

[8] Olivier Philippot. Eco-conceptions des logiciels et green patterns. *Programmez !*, 2012.

Table des figures

Annexe A

Sujet de stage

Sujet de Master M2 recherche

Développement logiciel : *the green touch*

Encadrant : Thomas Ledoux (Thomas.Ledoux@mines-nantes.fr), tel. : 02 51 85 82 19
Lieu : Ecole des Mines de Nantes
Stage gratifié ou rémunéré, possibilité de poursuivre en thèse Cifre

Mots clés : énergie, langages, green patterns, architectures logicielles, systèmes autonomes, Cloud computing

Contexte et problématique

En quelques années, le problème de la maîtrise énergétique est devenu un enjeu majeur de société. En informatique, les principaux travaux se sont concentrés sur des mécanismes permettant de maitriser l'énergie au niveau du matériel [1] et des infrastructures serveurs (centre de données) [2]. Le renforcement du rôle de l'informatique dans notre société - de plus en plus numérique - conduit à traiter ces problèmes également au niveau du logiciel.

L'exemple de la société Facebook rapidement confrontée à ce trait d'union entre le logiciel, le matériel et l'énergie est révélateur de l'importance de traiter l'efficacité énergétique directement au sein du logiciel. Une estimation passée montrait que les 30.000 serveurs de Facebook consommaient 100 millions de kWh par an [3]. Facebook utilise maintenant le pré-compilateur HipHop for PHP qui transforme du code PHP en code C++ qui est ensuite compilé à l'aide de g++. 90% des pages de Facebook reposent aujourd'hui sur HipHop avec un gain moyen de 50%. Il faut donc 2 fois moins de serveurs à Facebook pour fonctionner [4] !

L'anecdote précédente montre combien il est important de s'intéresser à l'*éco-conception du logiciel*. L'écriture de programme, la conception de l'architecture logicielle peuvent avoir une incidence sur l'environnement. Le logiciel est aujourd'hui le principal facteur d'obsolescence des équipements informatiques qui implique alors toute une chaine de remplacement très énergivore (la fabrication du matériel électronique concentre les principales nuisances environnementales) [5]. Il est donc important d'analyser les différentes pistes et de déterminer les divers leviers possibles pour participer à la génération de « code vert », à l'efficacité énergétique du logiciel.

Objectifs

La maitrise énergétique du logiciel tout au long de son cycle de vie (de sa conception à son exécution) constitue l'objectif principal de ce stage. Pour faire suite à un *position paper* de l'équipe en 2010 [6], il s'agit de faire de réelles propositions et de les mettre en œuvre. Plusieurs axes de recherche sont possibles : (i) au niveau langage et compilation [7] [8] ; (ii) au niveau règles/bonnes

pratiques de programmation et *patterns* [9][10][11] ; (iii) au niveau infrastructures logicielles (client/serveur, SOA, Cloud computing) [12] [13].[1]

Plan de travail

En fonction de l'état d'avancement et des affinités du candidat, le travail à réaliser s'effectuera autour deux parties indépendantes et complémentaires.

Partie 1

- Etat de l'art multi-axes

- Tester les outils de mesure qui permettent de mesurer les consommations des ressources (e.g. Energy Checker d'Intel [14]). Faire un bilan.
- Identifier et valider un certain nombre de règles d'éco-conception (recensées sur différentes architectures, dans différents langages).

- Conception/réalisation de « composant vert »

- Proposer un certain nombre de green patterns pour Java, pour le développement d'applications Web.
- Proposer des outils d'introspection de code, voire de refactoring de « code gourmand ». Par exemple, on voudrait savoir si telle portion de programme ne fait pas trop d'échanges avec le serveur ou n'appelle pas de bibliothèques énergivores (e.g. Flash).
- Définir des métriques permettant de noter l'efficacité énergétique d'une application.

Partie 2

Reconfiguration dynamique d'application « SaaS verte »

Le passage à l'échelle et la disponibilité d'une application logicielle est souvent assurée aujourd'hui par son exécution dans un Cloud. Or, notre équipe a récemment montré, grâce à un simulateur, qu'une application de type SaaS peut-être reconfigurée à la volée pour être plus *eco-compliant* en fonction du contexte d'exécution tout en garantissant une qualité de service (QoS) correcte [13]. L'objectif de cette partie du stage est de réaliser un *proof-of-concept* pour le Cloud montrant une application SaaS se reconfigurant dynamiquement pour réduire son empreinte énergétique et d'évaluer notre modèle théorique.

Positionnement

Ce stage s'inscrit au cœur des activités Green IT de l'équipe Ascola. Le stage est complémentaire aux travaux de l'équipe réalisés dans l'ANR MyCloud (http://mycloud.inrialpes.fr) et se place également au carrefour des sensibilités Langage et Systèmes distribués de l'équipe. **Possibilité de poursuite en thèse Cifre[2] avec une société nantaise en septembre 2012.**

Profil du candidat

Le candidat possèdera une certaine expérience dans le domaine du génie logiciel et les systèmes répartis. Un « appétit » pour les problématiques d'environnement durable sera nécessaire. Une

[1] Les axes d'étude systèmes d'exploitation, réseaux et télécoms seront exclus de ce stage.
[2] http://www.anrt.asso.fr/fr/espace_cifre/accueil.jsp?r=3&p=1

connaissance approfondie dans un ou plusieurs des domaines suivants sera appréciée : langages dédiés, systèmes autonomes, architectures orientées services, Cloud computing.

Références

[1] Susanne Albers. Energy-efficient algorithms. Communications of the ACM Vol 53, n°5 (May 2010)

[2] Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, New York, NY, USA, ACM (2009)

[3] http://www.greenit.fr/article/energie/facebook-consomme-autant-quun-tgv

[4] http://www.zdnet.fr/blogs/greenit/hiphop-facebook-divise-par-deux-le-temps-d-execution-de-php-39712771.htm

[5] http://www.greenit.fr/article/logiciels/logiciel-la-cle-de-l-obsolescence-programmee-du-materiel-informatique-2748

[6] Jean-Marc Menaud, Adrien Lèbre, Thomas Ledoux, Jacques Noyé, Pierre Cointe, Rémi Douence and Mario Südholt. Vers une réification de l'énergie dans le domaine du logiciel. In Journées du GDR Génie de la Programmation et du Logiciel (GDR GPL 2010), Pau, mars 2010.

[7] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. AI Magazine, 17(3), 1996.

[8] Seo, C., Malek, S., Medvidovic, N.: Component-Level Energy Consumption Estimation for Distributed Java-Based Software Systems. In: Component-Based Software Engineering, Springer Berlin Heidelberg (2008)

[9] https://sites.google.com/a/octo.com/green-challenge

[10] http://greencodelab.fr/

[11] Gregor Hohpe, Bobby Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional, 2004

[12] Mello Ferreira, A., Kritikos, K., Pernici, B.: Energy-aware design of service-based applications. In: ICSOC-ServiceWave '09: Proceedings of the 7th International Joint Conference on Service-Oriented Computing, Berlin, Heidelberg, Springer-Verlag (2009)

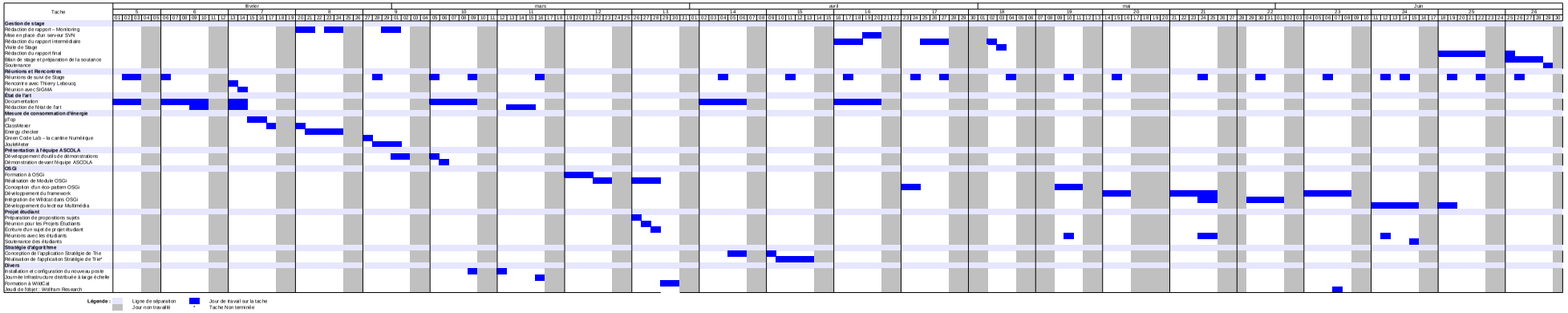
[13] Frederico Alvares De Oliveira Junior and Thomas Ledoux. Self-management of applications QoS for energy optimization in datacenters, 2nd International Workshop on Green Computing Middleware, In conjunction with ACM/IFIP/USENIX 12th International Middleware Conference, 2011

[14] http://software.intel.com/en-us/articles/intel-energy-checker-sdk/

Annexe B

Plannification

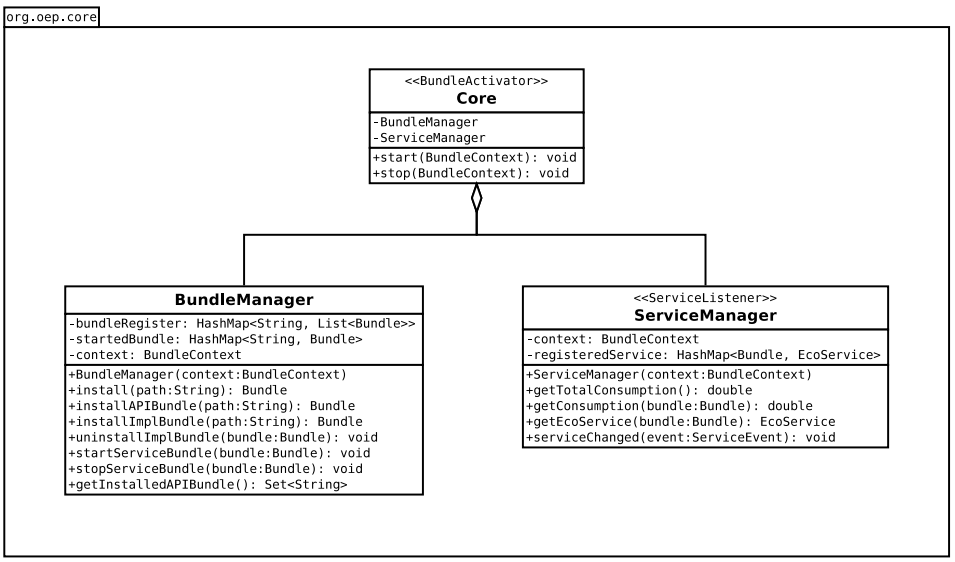
Planning Réalisé



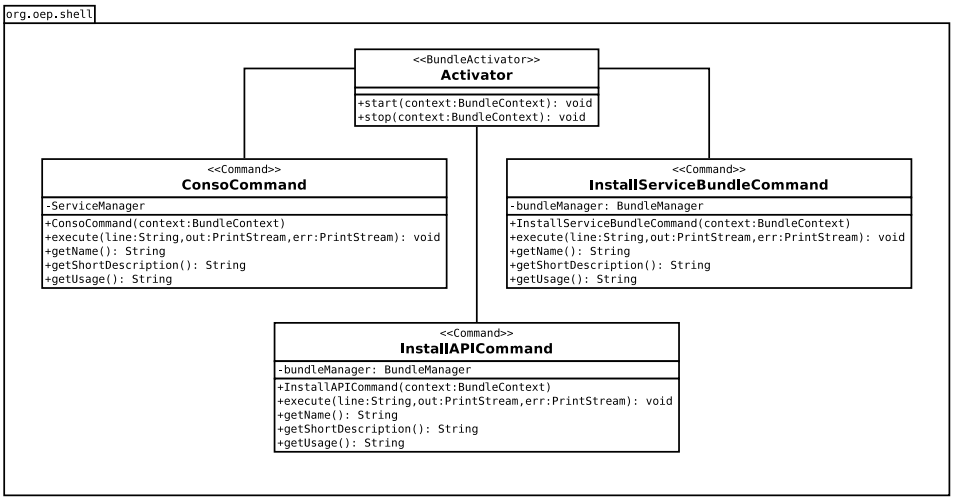
Annexe C

Diagrammes de Classes d'EcoFramework

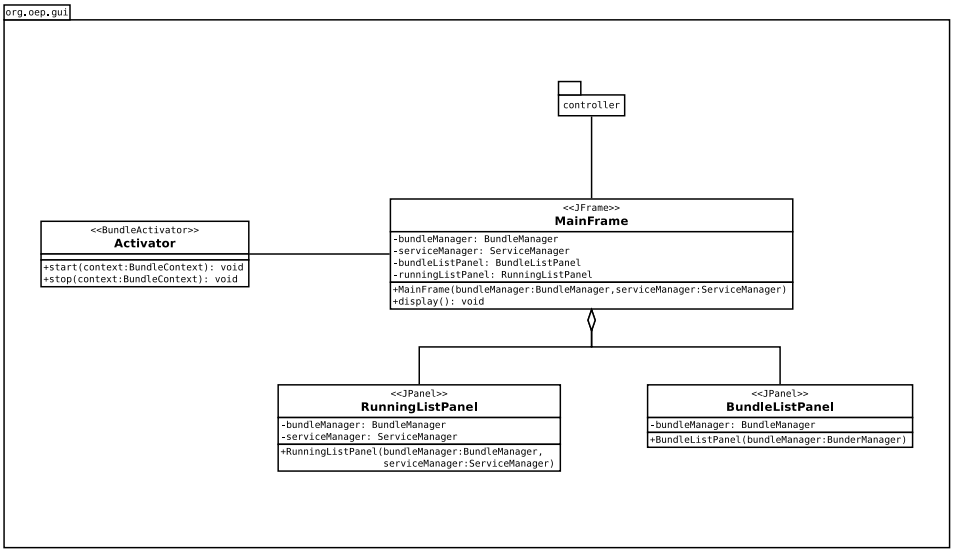
C.1 Package org.oep.core



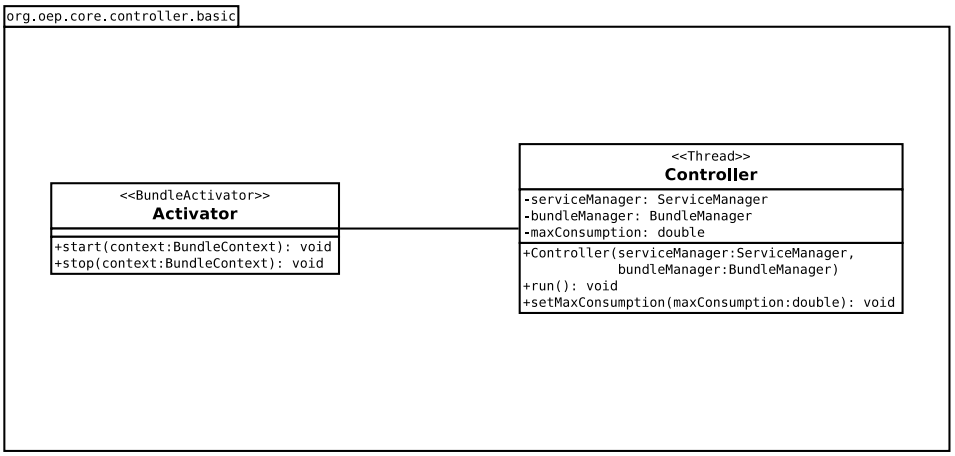
C.2 Package org.oep.shell



C.3 Package org.oep.gui



C.4 Package org.oep.core.controller.basic



C.5 Package org.oep.service.api