A Green Programming Model for Cloud Software Efficiency

Ah-Lian Kor, Colin Pattinson
School of Computing, Creative Technologies, and Engineering,
Leeds Beckett University, Leeds, UK
E-mail: {A.Kor, C.Pattinson}@leedsbeckett.ac.uk

***Abstract* — Cloud computing aims to deliver more energy efficient computing provision. The potential advantages are primarily based on the opportunities to achieve economies of scale through resource sharing: in particular, by concentrating data storage and processing within data centres, where energy efficiency and measurement are well-established activities. However, this addresses only a part of the overall energy cost of the totality of the cloud because energy is also required to power the networking connections and the end user systems through which access to the data centre is provided. The impact of application software behaviour on the overall system's energy use within a cloud is less understood. This is of particular concern when one considers the current trend towards "off the shelf" applications accessed from application stores. This mass market for complete applications, or code segments which are included within other applications, creates a very real need for that code to be as efficient as possible, since even small inefficiencies when massively duplicated will result in significant energy loss. This position paper identifies this problem and proposes a supporting tool which will indicate to software developers the energy efficiency of their software as it is developed. Fundamental to the delivery of any workable solution is the measurement and selection of suitable metrics, we propose appropriate metrics and indicate how they may be derived and applied within our proposed system. Addressing the potential cost of application development is fundamental to achieving energy saving within the cloud – particularly as the application store model gains acceptance.**

Keywords—Energy efficiency; green computing; programming model; energy efficient cloud

I. INTRODUCTION

This is a position paper which is concerned with the topical issue of Green Computing, specifically focusing on environmentally aware software development for Open Computing Environments (OCEs). The proposed research looks into novel software energy efficiency methods and development of tools to support software developers in monitoring, minimising the carbon footprint and optimising energy efficiency resulting from developing and deploying software in such environments. This proposed research is timely because it addresses the need for continued development of infrastructure support for OCEs in order to optimise, monitor and reduce carbon footprint and costs for OCE providers and end-users. The major contribution to the carbon footprint of OCE software is energy consumed in its operation, thus the primary aim of this proposed project is to link software design to energy use. Although energy use is of relevance across all software design and implementation, for this project, we will make specific reference to cloud-based services operations: the emergence of cloud computing with its emphasis on shared software components which are likely to be used and reused many times in many different applications make it imperative that developed software be as energy efficient as possible.

A. Background

The primary output of this proposed research is the derivation of explicit measures of energy[*] requirements for inclusion into software design and development process which can be executed on any processing platform. However, the delivery context of this proposed project will be a cloud environment, because cloud computing is a popular paradigm for business computing with a significant potential impact of cloud-based software. Cloud Computing aims to streamline the on-demand provisioning of software, hardware, and data to provide flexibility and agility, and economies of scale in IT resource management. Although building, deploying and operating applications on a cloud can help to achieve speed[1], scalability, and maintain a flexible infrastructure, it brings about a variety of challenges due to its massive scalability, complexity, as well as dynamic and evolving environments. The provision of cloud programming environments for applications and services is currently dominated by several large commercial providers such as Amazon[2], Google[3], Rackspace[4], Microsoft[5], and IBM[6].

II. LITERATURE REVIEW

A. Cloud Application Platforms

A brief survey of various popular application platforms has been conducted. Microsoft has proposed Azure[7] as an application platform for the Cloud based on the Windows Azure Operating System. Common Microsoft programming tools employed to develop Azure basic services such as SQL Azure[8] (to build, host and scale applications in Microsoft data

[*] Note: throughout this document, our focus is on the **energy** requirements of software development.
[1] http://www.ibm.com/developerworks/cloud/
[2] http://aws.amazon.com/what-is-aws/
[3] http://code.google.com/appengine/
[4] http://tools.rackspacecloud.com/category/applications/tools-for-developers/
[5] http://www.microsoft.com/windowsazure/windowsazure/
[6] http://www.internet.com/IBM_Cloud/Door/42153
[7] http://www.microsoft.com/windowsazure/
[8] Ibid.

centres) or Windows Live[9] do not offer specific programming models. Although Microsoft .NET[10] claims to offer a comprehensive and consistent programming model, it has not addressed the issue of energy efficiency. Manjrasoft Aneka[11] platform is oriented on enabled .NET-based enterprise Grid and Cloud platform. It provides services for authentication/authorisation, dynamic resource allocation, accounting, etc. Aneka considers multiple programming models: Thread Programming Model (to adopt multi-threaded application on a distributed system); Task Programming Model (to implement independent bag of tasks applications); and MapReduce Programming Model[12] (proposed by Google as a programming model for developing distributed data intensive applications in data centres). Hadoop[13] is an open source software platform that permits the processing of vast amounts of data. Hadoop MapReduce is a programming model and software framework for writing applications that facilitates parallel processing of vast amounts of data on large clusters of compute nodes. MapReduce divides applications into many small blocks of work. The Hadoop Distributed File System (HDFS) creates multiple replicas of data blocks for reliability, placing them on compute nodes around the cluster so that MapReduce can then process the data where it is located. The Google App Engine[14] provides facility to develop and run web applications on proven Google's infrastructure. The current supported languages are Python and Java. Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. The Amazon Elastic Compute Cloud (Amazon EC2[15]) offers a web service that allows businesses to run their application programs in the Amazon.com web-based and virtual computing environment. Here, EC2 practically serves as an unlimited set of virtual machines. On the other hand, OASIS provides an open standard executable language, Web Services Business Process Execution Language (WS-BPEL[16,17]) for the formal specification of business processes (based on Web Services) and business interaction protocols. The interactions included in the standard are of two different types: executable business processes, and abstract business processes. Executable business processes model actual behaviour of a participant in a business interaction while abstract business processes are partially specified processes that are not intended to be executed (they have a descriptive role).

In this proposed research, the main program will only have invocations to services, nothing is related to the middleware that is responsible for running services and composition of applications will be based on general Web Services. The programming model itself is an innovation in the field of Cloud Computing. Thus far, only MapReduce has been proposed as programming models in this environment. Though BPEL (Business process Execution Language) has a more general approach and allows description of service workflows, it does not focus on the Cloud as the execution infrastructure. Our proposed Programming Model covers a broader set of applications than MapReduce, since MapReduce applications require the programmer to open and close parallel regions in the code. Our other proposed innovation is the co-existence of specific application code parts and published services that will be orchestrated by the proposed Programming Model run-time (known as core elements of the application in this proposed research). Some service compositions may require the following actions: performing some appropriate calculations or checking some data in order to inform decision making on how to proceed. These calculations are not intended to be a service but rather, a piece of code to support the overall process. The proposed Programming model run-time will be able to dynamically schedule the core elements of an application and allocate the resources from infrastructure providers where the core elements are to be executed, according to energy efficiency terms (more discussion in the remaining part of this paper). The Programming Model run-time will exchange messages with decision modules that will consider many different parameters of the Cloud (e.g. Quality of Service, QoS) in order to decide the best location for scheduling the Programming Model core element.

B. *Cloud and Energy Efficiency*

The outcomes of this proposed work are equally applicable to any of the "as a service" options (Software, Platform, and Infrastructure), all three are affected by the performance of the software which is run which is the energy requirements of software (i.e. the focus of this work). According to Gartner Inc.[18], the adoption of cloud applications and services by enterprises is rapidly increasing. Consequently, experts warn of a dramatic increase in energy consumption for cloud computing. A Greenpeace report[19] predicts that the global energy consumption for cloud computing will increase from 632 billion kWh in 2007 to 1,963 billion kWh by 2020 and the associated CO_2 equivalent emissions will reach 1,034 mega tonnes. In order for cloud providers to implement a "green cloud", it is essential to promote energy efficiency awareness among all its stakeholders, and produce metrics to demonstrate energy gains. Though some work has been undertaken on the development of energy metrics, most published measurements can be characterised as "hardware-related", and do not address the impact of software performance on the overall energy efficiency of an IT system. Currently, available metrics focus on relative measures of data centre operations (e.g. Power Usage Effectiveness (PUE) and Data Centre Infrastructure Efficiency (DCiE))[20]. Energy efficiency metrics for a programming environment are yet to be developed.

Work at the CLOUDS Laboratory (in University of Melbourne [2]) focuses on the deployment of virtual machines and their interconnection and migration, with a Service Level

[9] http://explore.live.com/
[10] http://www.microsoft.com/net/overview.aspx
[11] http://www.manjrasoft.com/products.html
[12] http://code.google.com/edu/parallel/mapreduce-tutorial.html
[13] http://hadoop.apache.org/
[14] http://code.google.com/appengine/
[15] http://aws.amazon.com/ec2/
[16] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
[17] http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm
[18] http://www.gartner.com/it/page.jsp?id=1389313
[19] http://www.greenpeace.org/international/Global/international/planet-2/report/2010/3/make-it-green-cloud-computing.pdf
[20] http://www.microsoft.com/environment/our_commitment/articles/green_grid.aspx

Agreement process to allow the customer to specify their energy demands. Srikantaiah et al. [5] have also approached the issue of energy-efficient cloud computing through (virtualised) resource consolidation. Mittinen and Nurminen [4] have addressed the use of mobile clients to access cloud provision, and described the split of energy use as a client-server (or local-remote) relationship indicating that energy efficiency is highly sensitive to workload, data communication and technology in use. The SPECS "Power and Performance Benchmark Methodology" (SPECS, 2010)[21] uses an incremental series of benchmark loads to determine the power/performance behaviour of a server under load, expressing the result in terms of "performance per Watt". However, in both these cases, the "performance" or "throughput" is derived from benchmark-type processing activity, which may not be easily related to specific tasks.

The relevant aspect of this proposed project is the "Green Tracker" being developed by Amsel et al. (2010) [8] which assesses the energy consumption at the CPU level of complete software applications, and aims at allowing users to select between alternatives at run-time. Whilst an interesting development (and evidence of the timeliness and suitability of this proposed project, the focus on CPU use; on extant software and on a user-level presentation mean that it does not address the totality of software energy; and does not lend itself to a tool which can directly assist software developers in building applications. PAS 2050[22] provides a benchmark and standard assessment method to determine the carbon footprint across the entire lifecycle of a product. In the context of our research, we shall adapt the principles of PAS 2050, where appropriate for calculating the energy requirements of software. For scoping purposes, our focus will only be on the calculation of energy costs relating to the developing and running of code (as illustrated in Fig. 1). While the projects noted in the previous paragraph represent significant contributions to the area, we believe that the energy requirements of the software applications which run on hardware units (virtual or real) must be incorporated into the overall development and deployment process. The total characterisation of *software energy* with respect to the impact of the software structure on energy use is not incorporated into any current models. It is this gap which this project will address. Determining the relationship between software structure and its energy use will allow us to define a set of software energy metrics similar in concept to those for hardware. By associating those metrics (via tags) with existing component libraries, and by creating the tags during the production of new software components, we will be able to populate a software development toolkit with information to predict the energy requirements of applications, thereby allowing alternative selections of both existing (reused) and newly-created code components to be made, using energy as a selection criterion.

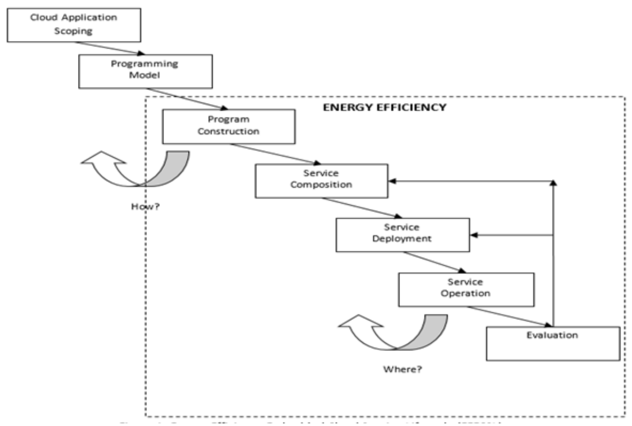

Fig. 1. Energy Efficiency Embedded Cloud Service Lifecycle (EEECSL).

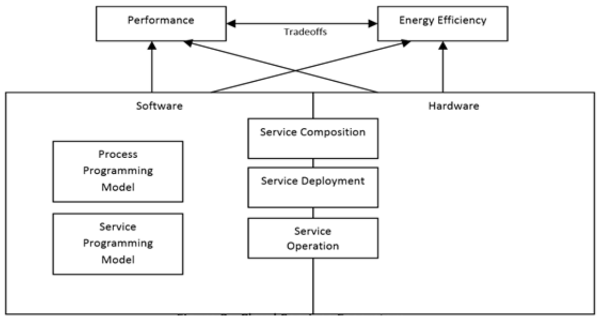

Fig. 2. Cloud services ecosystem.

III. PROGRAMME

A. Research Aim and Objectives

The proposed project's goal is to understand and characterise the factors which affect energy efficiency in software development, deployment and operations. Our main novel contribution is the incorporation of a holistic approach to references to both hardware and software energy efficiency in the lifecycle (as shown in Fig. 1 and 2). We will demonstrate the outputs of this characterisation through the development of an integrated development environment (IDE) for the deployment of services, the target users being programmers. We envisage this taking the form of a dashboard with drag and drop components including identified energy efficiency and performance measures. Users can program and choose from alternative service compositions through a user friendly interface. In particular, the research objectives for this project are to:

1) Elicit requirements for the Programming IDE and Runtime Environment with the following primary components: programming models; service composition; service deployment; service operations; energy efficiency;

2) Develop and evaluate the description of the services ecosystem components (depicted in Fig. 2);

3) Develop and evaluate a framework with identified energy efficiency parameters and metrics for services;

4) Develop, verify, and validate programming models for particular applications and services;

[21] http://www.spec.org/power/docs/SPECpower-Power_and_Performance_Methodology.pdf
[22] BSI (2008). Guide to PAS 2050 How to assess the carbon footprint of goods and services http://www.footprintexpert.com/PCFKB/Lists/kbdocuments/Guide%20to%20PAS%202050.pdf

5) Develop methods for measuring, analysing, and evaluating energy use for software development;

6) Integrate energy efficiency (measures, analysis, and evaluation) into service composition, deployment, and operations;

7) Develop, implement, and evaluate the Programming IDE and Runtime Environment in Research Objective 1.

IV. Methodology

Typical components in existing Service Oriented Architecture (SOA) service lifecycles are: identification, modelling, composition, provisioning, deployment, management, and evaluation (ORACLE)[23]; or modelling, assembly, deployment, and management (IBM)[24]. These lifecycles primarily focus on the software aspect. However, IBM recently extended such a typical service lifecycle which is referred to as Service Lifecycle Management[25]. IBM's Service Lifecycle Management is socio-technical centric and includes novel as well as existing integrations of software. The integrations afforded in the extended model are as follows: enhanced hardware and software connection to streamline the resolution of infrastructure-related problems; service desk and development connection to effect efficient cooperation between service desk and operation tools; operations and test/development connection to simplify and automate information handoffs between the two faculties by: streamlining dataflow between them; minimizing response time to data requests; facilitate efficient data analysis. In this project, we have adapted IBM's SOA Service Lifecycle and its extended model (Service Lifecycle Management), calling it the Energy Efficiency Embedded Service Lifecycle (acronym, EEESL; see Fig. 1) which contains functional blocks and addresses questions (shown as arrows in Fig. 1). The novelty of our model is that it has energy efficiency as its pivotal anchor. The main goal of this model is to create a service ecosystem whose components (see Fig. 2) work efficiently and seamlessly together.

V. Underlying Components for the Proposed Cloud Service Ecosystem

A. *Energy Software*

Intuitively, there is a relationship between software design and energy consumed by that software: at a very basic level, code fragments (in whatever programming language is used) convert into a sequence of machine code instructions, and since each instruction can be associated with a number of CPU operations, the total number of such operations can be determined, and, if we know the energy use per cycle, the overall energy "cost" can be identified. It is not difficult to envisage a formalisation of this intuitive relationship. However, in practice, outside the specialist implementations where resources are at a premium, such considerations do not seem to designers' priority. Further complexity is caused by overall energy use being affected by factors such as the sequence in which a test-branch operation is carried out, the order in which incoming data is presented and the final output requirements of any process. Furthermore, energy is not consumed only by CPU cycles. However, other parts of the system (such as those involved in data storage and transfer) are of increasing significance as the data and processing environment becomes more disparate. Extrapolating this to take account of a large piece of software, potentially implemented across a number of different platforms and comprising software components from a range of sources, indicates the possible complexity involved, and suggests that an automated approach is required. Such a solution is possible, making use of a combination of theoretical analysis of software code, measurement of existing components, modelling tools for a complex situation (e.g. AHP–Analytic Hierarchical Process[26]), and simulation. We will associate energy with processor activity and other resource use, and use that association to derive the energy requirements of sequences of operations, and hence of identifiable program segments. Our analysis will permit us to create a knowledge base of the energy use of specific code segment. Through a combination of measurement (of existing code) and simulation (of newly developed code), we will extend this knowledge base to support a generic energy estimation tool. We envisage this tool being used by program designers and developers to guide selection of new and reused code fragments, to make deployment choices between alternative platforms, and to allow informed negotiations around service level agreements. Information–referred to from here on as metrics-derived from this knowledge base will be associated (via tags) with software components, allowing users to understand the effect of design choices on overall energy requirements.

A. *Programming Model, Program Construction, and Incorporation of Energy Metrics*

Software will be constructed based on a programming model (shown in Fig. 1) specifically developed for open computing environments including but not restricted to the cloud. Our project confers equal importance to the development of new software, the adaptation and combination of legacy software, and the composition of software services within a larger context. As noted earlier, we base our proposal on earlier work by IBM[27]. The two types of programming models introduced by IBM are: Service Programming Model and Process Programming Model. IBM (2000, 2004)[28] views the service programming model as one that defines/describes what a service is and how it is developed while a process programming model defines/describes what a process is and how it is developed. Description or semantics annotation of services is facilitated by service modelling languages (e.g. WSDL, Web Services Description Language). The dimensions in a service programming model are as follows: data; service interface (a mechanism for logical grouping of business operations); and service implementations (description of how services are implemented and executed by an endpoint such as a business application system). The development of a service

[23]http://download.oracle.com/docs/cd/E13159_01/osb/docs10gr3/concepts/introduction.html

[24] http://www.ibm.com/developerworks/rational/library/mar07/mcbride/

[25] http://ptaknoel.com/wp-content/uploads/2009/12/IBM-Service-Lifecycle-Management-FINAL.pdf

[26] IBM(2009). How much energy do your IT devices use? A guide to comparing their efficiency and cutting their carbon footprint, supported by the UK Government's Green CIO, http://www-935.ibm.com/services/uk/cio/pdf/howmuchenergy_lr.pdf

[27] http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel/

[28] Ibid.

programming model entails the following steps: the creation of service definition based on an existing implementation (e.g. a Java class, or IS function, etc.) or creation of a service implementation from an existing service interface (e.g. a Java class skeleton); generating deploy code for the services to be deployed so that services of an integrated application are accessible via different protocols and easily tested within a test environment (e.g. IDE); creating service proxy which involves the creation of the client side proxy for a service[29] offered by an integration application. As for the Process Programming Model, it defines/describes what a process is and how it is developed[30]. The development of a process could either be a top-down or bottom-up approach encompassing process interfaces that are synchronous or asynchronous and process implementation. In our project, we envisage the adoption and adaptation of IBM's notion programming models. Our novel contribution to the development of programming models for an open computing environment (e.g. cloud) is the incorporation of energy efficiency metrics into the service and process programming models as described in the previous section.

C. *Service Composition, Deployment, Operation, and Evaluation*

As previously mentioned, a service in this project is formed by the integration of new code, existing services (not necessary traditional Web Services), and the adaptation/combination of legacy software. We envisage the building/reuse of these platform-independent software components and we shall call them service components. An application could be assembled from a set of appropriate service components and this process is called a service composition. A hybrid mechanism (manual, automated or semi-automated) will be developed for assembling the services. Each service component will be described by either a semantic annotation (of what it does) and by a functional annotation (of how it behaves) [6]. The novelty of our contribution is to incorporate an associated energy efficiency measure into the functional annotation of each service component (the energy metrics "tags" referred to above) so as to promote the development of energy efficiency aware service compositions (note: users who are software developers will be able to view the calculated total possible energy efficiency for each type of chosen service composition). One of the critical issues to address in service provisioning is a Service Level Agreement (SLA) where service consumers and providers effectively achieve agreements on non-functional aspects such as the Quality of Service (QoS) [7]. As a result of the association of energy requirements with created software, our service compositions are QoS aware (in respect of the likely resource demands of a service instance) and thus, resource allocation can be consistent with the SLA [3] description of service quality related content according to SLA parameters; enabling the service provider to apply QoS values for invoked services. In order to facilitate wide adoption and reach the critical mass of software services required to support open computing environments such as clouds, it is imperative to avoid ad-hoc and manual processes in the remaining steps of the service lifecycle. This project will offer a set of tools to automate and standardize service deployment and operation, with energy efficiency as the basis for the decisions taken at each stage. In the deployment phase, a service is placed on an Infrastructure Provider (IP) for operation. Some of the activities in this phase are[31]: testing and debugging of a service (to test whether it works properly)[32]; modification of a service according to an environment; selecting a suitable IP to host the service; deploy a service package to a deployment slot within a new hosted service. Other activities include the negotiation of SLA terms between Service Providers (SP) and Internet providers (IP), and the propagation of contextual information necessary for instantiating the service once when it has been deployed. Service deployment tools will facilitate the packaging of services and their complete software stacks as well as the addition of security capabilities to images prior to deployment and execution. However, services testing tools will provoke a reflection on service integration, autonomy, stability, performance, etc…[33]. This project aims to provide an integrated approach for deploying services, including functionality for a careful evaluation of resource providers in respect of their energy requirements. To achieve this, the energy consumption requirements of the software components are specified during the programming construction phase and the integration is carried out during service operation (or execution). The former is the result of our development of software energy metrics, as described above. The latter is achieved through dynamic allocation of resources according to the energy consumption requirements of each element within the context of the service. The result is that, without the intervention of a software developer, the execution of the service evolves according to the infrastructure requirements of each core element and the status of the available infrastructure. As previously mentioned, the AHP (Analytical Hierarchical process) [34]modelling technique would also be implemented in the service lifecycle because it takes into account the different parameters and metrics for energy efficiency for each phase, sets out the relevant information clearly and succinctly so as to support the software developer's decision making with regard to the choice of software components for a particular service. The three main aspects that will be considered in the operations phase are: result, energy efficiency and performance. The project will aim to improve the execution of services by utilising a governance process to define energy efficiency policies for harmonising all management activities throughout the service lifecycle. These policies will integrate disparate software management requirements, from high-level Business Level Objectives (BLOs) to resource requirements, into a unifying view to verify that the services are executing as expected.

[29] http://msdn.microsoft.com/en-us/library/dd815336(VS.85).aspx
[30] http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel/

[31] http://msdn.microsoft.com/en-us/library/ff683668.aspx
[32] http://www.infoworld.com/t/architecture/soa-services-deployment-putting-theory-practice-232
[33] http://www.infoworld.com/t/architecture/soa-services-deployment-putting-theory-practice-232
[34] IBM(2009). How much energy do your IT devices use? A guide to comparing their efficiency and cutting their carbon footprint, supported by the UK Government's Green CIO, http://www-935.ibm.com/services/uk/cio/pdf/howmuchenergy_lr.pdf

D. Integrated Development Environment (IDE) and Runtime Environment

In this proposed project, an integrated environment for the programming, composition, deployment and execution of services and applications will be developed. The IDE will have available a friendly GUI with several sets of tools: creation tools for creating services and processes; composition tools for integrating services and services, services and processes, processes and processes; tools for testing, debugging, and running the services and processes; deployment tools for deploying the services (to put the service into an application server); invocation tools for clients to invoke deployed services. This interface will include also tools for easily publishing and un-publishing services to make them available to third parties. The tools developed for this phase will be operating system independent and codes generated for the services and processes will conform to industry standards and open-source specifications (e.g. Web Service Invocation Framework, WSIF[35]). Many traditional approaches for Web Services Composition rely on applying static planning techniques to deploy and execute the compositions [1]. In contrast, the Programming Model runtime developed in this project will be able to dynamically schedule the core elements of an application and dynamically allocate resources from infrastructure providers where the core elements are to be executed, according to energy efficiency considerations. Messages will be exchanged with decision modules regarding different parameters of the potential deployment environment (e.g. QoS) in order to decide the best location to schedule the Programming Model core element.

E. Environmental Impact

The proposed project will focus on a particular type of OCE - cloud computing - to develop and demonstrate the research outcomes. Cloud computing gives users seamless access to a wide range of computational, storage and network resources, enabling them to execute tasks far beyond the capabilities of their own resources/infrastructures. Usage of OCEs in general was initially driven by the e-Science community (in particular their use of Grid computing). However, developments of recent years, especially those addressing key issues such as transparency, security, data transfer, resource brokering, workflow and risk management[36] have driven larger scale commercial uptake of the technology and seen the emergence of cloud computing as a way of offering mass data processing and storage without the cost of purchase and maintenance. Many organisations which stand to benefit from such environments, especially those within the commercial sector have policies relating to energy and sustainability, including the consumption of ICT resources and services. These policies are being increasingly shaped by a social awareness of green issues and a wish to conduct business with the least possible impact on the environment. Therefore, not only does this research seek to reduce the environmental impact of such infrastructures; it has the potential to drive commercial uptake further by bridging the gap between organizational green policy and practice, and drawing attention to green issues within OCEs. Increasing discussion of environmentally friendly ICT at both government and European Commission level are further evidence of the importance with which the subject is viewed. At UK level, the government has published an 18 step carbon emissions reduction plan[37] with a focus on reducing the carbon impact of government ICT operations throughout its entire operation. Most recently, an EU-supported code of conduct[38] for data centers has further emphasised the significance of the ICT sector in the delivery of climate-saving policies.

VI. CONCLUSIONS

Our novel EEESL incorporates aspects of software efficiency as a fundamental component in all steps of the service lifecycle. In order to deliver this vision, appropriate measures of efficiency for this context will be identified: measures which must be accurate, usable and repeatable. With these definitions of software efficiency, we will be able to construct a model to estimate the relative performance of different design approaches. An important element of our work will be that the methods derived from it will be applicable both to new and pre-existing software, thus a measurement methodology will be developed to assess energy efficiency at a number of activities in the service lifecycle: design; coding; compilation; machine readable (executable). To reiterate, no workable and comprehensive methodology for the measurement of energy use of software processes currently exists, so it is envisaged that the creation of such a methodology, and its verification and validation, will itself offer a major contribution to the field. Decision support will be provided for the deployment phase of the resulting software, since the platform(s) on which the resulting code is run will also have a major impact on the overall efficiency. Therefore, a list of the following will be investigated on: the relationship between energy use and hardware and network design, deriving a model of the efficiency structures. Once again, this work is novel, we are not aware of any existing models which effectively describe the relationships among all the components in a Services Ecosystem depicted in Fig. 2 with the main emphasis on the trade-offs between performance and energy efficiency.

REFERENCES

[1] Agarwal, V., Chafle, G., Mittal, S., and Srivastava, B. (2008). Understanding approaches for web service composition and execution, In Proceedings of the 1st Bangalore Annual Compute Conference (Bangalore, India, January 18 - 20, 2008); COMPUTE '08. ACM, New York, NY.

[2] Beloglazov, A., and Buyya, R. (2010). Energy Efficient Allocation of Virtual Machines in Cloud Data Centers. Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010), Melbourne, Australia, May 17-20, 2010.

[3] Dong, W. L., and Jiao, L. (2008). QoS-Aware Web Service Composition Based on SLA, Proceedings of the Fourth International Conference on Natural Computation,Vol.5,pp.247-251.

[4] Mittinen, A. P., and Nurminen, J. K. (2010.). Energy efficiency of mobile clients in cloud computing, Proceedings of HotCloud'10, USENIX Association Berkeley, CA, USA.

[35] http://ws.apache.org/wsif/ ; http://ws.apache.org/wsif/ , http://www.ibm.com/developerworks/webservices/library/ws-appwsif.html
[36] http://www.ggf.org

[37] http://www.gartner.com/DisplayDocument?id=730607.
[38] *Commission Joint Research Centre Code of Conduct on Data Centres Energy Efficiency*. 2008, European Commission: Brussels.

[5] Srikantaiah, S., Kansal, A., and Zhao, F. (2008). Energy Aware Consolidation for Cloud Computing, Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08). Berkeley, CA, USA.

[6] Srivastava, B., and Koehler, J. (2003). Web Service Composition - Current Solutions and Open Problems, ICAPS 2003 Workshop on Planning for Web Services, pp. 28-35.

[7] Yan, J., et. al (2007). Autonomous service level agreement negotiation for service composition provision, Future Generation Computer Systems, Volume 23, Issue 6, pp.748-759.

[8] N. Amsel, B. Tomlinson. 2010. "Green Tracker: A Tool for Estimating the Energy Consumption of Software." In: ACM Conference On Human Factors In Computing Systems (CHI 2010), Work in Progress. Atlanta, GA.