POLITECNICO DI MILANO
Master of Science in Computer Engineering
Dipartimento di Elettronica e Informazione

Study of the correlation between software developer profile and code efficiency

Orientator: Prof.ssa Chiara FRANCALANCI

Master Thesis of:
Emilio Pietro ROSSETO (Student Id. Number: 734273)

Academic Year 2009-2011

POLITECNICO DI MILANO

Laurea Specialistica in Ingegneria Informatica

Dipartimento di Elettronica e Informazione

Study of the correlation between software developer profile and code efficiency

Relatore: Prof.ssa Chiara FRANCALANCI

Tesi di Laurea di:

Emilio Pietro ROSSETO (Student Id. Number: 734273)

Anno Accademico 2009-2011

Acknowledgements

To Eugenio Capra for all the help during this project.

To my parents Maria and Antonio and my brother Vitor, who always believed in my potential and have always provided support.

To my girlfriend Cristina, whom I hold very dear and who have been by my side greatly helping me accomplish my goals.

Abstract

Green IT è un' area di ricerca che studia I mezzi per rendere TI sostenibile ecologicamente, con l'obiettivo di ridurre l'uso di materiali pericolosi, massimizzare l'efficienza energetica nel settore IT e promuovere la riciclabilità o la biodegradabilità dei prodotti. Tra i campi del Green IT, Green Software è quello interessato a comprendere i driver relativi all'efficienza energetica nel software, sviluppando e utilizzando software informatici con un impatto minimo o nullo per l'ambiente. Questo lavoro contribuisce agli studi sul software Green, indagando l'esistenza e la rilevanza della correlazione tra pratiche e competenze degli sviluppatori di software e l'efficienza energetica dei loro software. Il progetto riguarda il tema delle competenze degli sviluppatori software, e crea un sondaggio basato su un framework di riferimento al fine di misurare e confrontare tali competenze. Studia e utilizza metriche software standard e metriche verdi recenti per quantificare e confrontare l'efficienza energetica dei software in un campione. Infine, questo progetto produce come risultato una metodologia che è stata usata qui per verificare la correlazione tra le competenze degli sviluppatori e l'efficienza energetica del software.

Abstract

Green IT is an area of research that studies means to make environmentally sustainable computing or IT, with the goal of reducing the use of hazardous materials, maximize the energy efficiency in IT and promote the recyclability or biodegradability of products. Among the fields of Green IT, Green Software is the one concerned with understanding the drivers related to energy efficiency in software, developing and using computer software with minimal or no impact to the environment. This work contributes to the studies on Green Software by investigating the existence and relevance of correlation between practices and competences of software developers and their software's energy efficiency. The project covers the subject of Software developer Competences, and creates a survey based on a reference framework in order to measure and compare such competences. It also studies and utilizes standard software metrics and recent green metrics to quantify and compare the energy efficiency of software in a sample. Finally, this project produces as its result a methodology that was used here to test the correlation between developer competences and software energy efficiency.

List of Figures

List of Abbreviations

IT	Information Technology
ICT	Information and Communication Technology
SW	Software
r^2	Statistical Coefficient of determination

Contents

1 Introduction

In recent years, environmental issues have gained growing attention. Since the industrial revolution, natural resources use has dramatically increased, and today the rate in which we use them has already surpassed their renewal capability. If the growth continues at this rate, in 2050 we will be consuming more than twice the production capacity of the planet Earth (TOWNSEND; BURKE, 2002). In particular, Green IT, i.e., the study of environmentally sustainable IT, is more and more important, as Information Technology's impact is greater than ever. IT infrastructures are responsible for 2% of the CO^2 world emissions and for the greenhouse effect, which is the first reason of the global warming. Energy costs are increasing and playing a bigger part on the IT infrastructural costs. Nowadays, the cost of powering and cooling servers for one year is almost 60% of the initial purchasing cost (KUMAR, 2007), and energy requirements can be a scalability issue when creating or expanding a data center.

The reasons mentioned above motivate studies in Green IT in the same way environmentally friendly solutions are being developed in many other sectors of the economy. There is already political pressure and regulamentation supporting green solutions and companies are starting to use their green solutions for differentiation in the market.

The most important benefit of a Green ICT strategy is the reduction of costs related to the energy consumption. Some studies says that the costs with power and cooling in data centers can reach up to 20% of the IT cost. In the economic sector, the potential savings for companies could be huge and simple actions can have big impacts in the organization. As David Frampton, VP general manager of Cisco's LAN switching business unit, explained to Reuters (CHESTNEY, 2009), "a bank branch could save nearly US$53,020 just by turning off phones and wireless access points outside business hours". Following the same pattern, last year, Symantec launched a study named "State of the Data Center Report 2008", in which the social responsibility was the least important reason for applying a Green initiative. It states instead that reduction of costs and reduction of power consumption are the most important reasons to invest in such idea.

Another interesting benefit of Green ICT is that companies may use it for differentiation in the market, by a marketing strategy that profits from the growing popularity of environmentally friendly solutions. Vendors have started to put green labels on their products and consumers also started to seek for green products instead of the traditional ones. There are many certification agents, like Energy Star (international standard for energy efficient consumer products), which asserts that a given equipment meets a certain efficiency standard, and labels it with the Energy Star logo. Besides certifications on efficiency, there are examples of local certificates like the Selo Verde da USP (Brazilian University of Sao Paulo's Green Seal) that label IT products which do not contain heavy metals such as lead or cadmium in their composition.

Figure 1.1: Energy Star Logo.

SELO VERDE
CCE
CENTRO DE COMPUTAÇÃO ELETRÔNICA
USP
COLABORANDO COM A SUSTENTABILIDADE

Figure 1.2: USP Green Seal.

When talking about Green ICT, usually what comes to mind is the great advances in hardware performance and efficiency in the last decades. These groundbreaking advances brought IT to a different level, and allowed for the spread of mobile devices due to low consumptions systems and increased battery life. A strong example is the MIPS/W (Millions of instructions per second per watt) of mainframes, that increased by a factor of 28000 in the last thirty years. Nevertheless, research on software efficiency and green software has not been on par. Even today, traditional software development standards (such as the ISO/IEC TR9126:2003, Software engineering - Product quality metrics) do not include energy efficiency in its parameters.

The research hereby presented will refer to the above mentioned Software efficiency issue. It is part of a bigger project in Green ICT from Politecnico di Milano, and will complement previous works while adding a new dimension to the discussion. In the next pages, the results of a research involving software developers and key metrics of their software will be analyzed and discussed. In the next section the motivation for such research will be explained.

1.1 Motivation

As mentioned in the introduction, research on Green ICT, and more specifically in Green Software is of great importance. This importance is clear when we think about the way in which software is the main driver of energy consumption. If, when doing a certain task, software A requires a big number of commutations, then the hardware will require more energy. If software B performs the same task using a much lesser amount of commutations, the hardware will need a lesser amount of energy.

Alberio and Brianza (ALBERIO; BRIANZA, 2009) clarify this concept by means of the following analogy: Imagine a person who has to reach his destination travelling by car. This person wants to accomplish it in an efficient way. He can increase the travel efficiency by choosing a car which has a modern and economic engine, but he can also plan his route carefully in order to minimize the travel distance. While the first approach is similar to using more efficient hardware, the second one is analog to making more efficient software. Finally, the two approaches for efficiency are not mutually exclusive, and it is also the case between Green hardware and software.

Consequently, there is a need to understand how to generate software that is greener, i.e., more energy efficient. In other words, which are the different developing practices and choices that result in greener software?

1.2 Goals

The main goal of this project is to search for a correlation between cause and consequences in green software. The first step on achieving this goal was taken by Galli (GALLI, 2008). In his work, which will be better described in Chapter 2 - State of the Art, Galli studied how energy efficiency and static software metrics are related. But we need to go further to be able to answer some important questions such as:

- If a company wants to produce greener software, which specific skills should it look for when hiring or adding a software developer to a team?
- Which Software Engineering practices should one encourage in a developing team order to achieve greener software?
- Will a certain software developer training program result in the creation of more energy efficient software?

In order to be able to answer the above questions, it is necessary to go beyond software analysis, and study the software developing practices, and the way the software developer works. As mentioned before, the main goal of this project is to search for a correlation between cause and consequences in green software. This can be done by comparing different developing practices with characteristics from the software produced using them.

The project consists of the following steps:

- Study and understanding of the related concepts;
- Selection of a sample from a population of softwares;
- Creation of survey aimed at software developers from the sample;
- Compilation of survey results;
- Compilation of software metrics from the sample;
- Analysis of the correlation between survey results and metrics;

1.3 Overview of the Thesis

This thesis is organized as follows: Chapter 2 presents the fundamental concepts related to the proposed theme. It also presents the state of the art of Green ICT, i.e., an explanation on other studies of the subject, with bigger focus on those closely related to this project.

Chapter 3 explains the methodology used in this project and the steps planned to arrive at the project goals. These steps include the creation of a survey for software developers, the collection of sample software and the measurement of metrics to serve as a proxy for energy efficiency.

Chapter 4 named Implementation, Analysis and results describes the execution of the methodology created in chapter 3. It presents the results of data collection and organization, and exposes it in a meaningful way.

Finally, chapter 5 presents the final conclusions of this project and suggests what can be done to expand on the subject.

2 State of the art

This chapter describes the concepts related to this project, in order to provide a theoretical basis necessary to understand the objective of this project and the obtained results. The next pages will present the state of the art of the research of energy efficiency in IT and of green software.

Section 2.1 will clarify the Green IT concept with detail, explaining its fundamental aspects and its subareas and giving historical examples to illustrate its increasing importance. Section 2.2 will classify IT Technology impacts in two groups: Environmental impacts and competitiveness impacts, and detail them. In section 2.3 the Green Software subject will be presented as a key component of Green IT, and we will see how increasing software efficiency may imply in reduction in power consumption in all components in a computer or data center. Section 2.4 will describe existing software metrics that are used to measure software quality and also metrics for the measuring of energy efficiency. Finally, section 2.5 will be a presentation of the European E-competences Framework, used in this project as a guideline for the elaboration of a survey.

2.1 Green IT

Green IT is presented as a new research field that studies IT related energy and environmental issues. Green IT was born because of the increasing impact and limitations in IT due to energy efficiency, because of the increasing environmental impact is has caused in the last decades, and because of the environmental friendly culture that is establishing itself lately.

Before Green IT, the advances in IT were focused in performance. The great evolution seen in the last decades brought us faster and smaller processors (and devices in general), but coupled with this advanced there was an increase in the energy consumed by such devices. While an Intel 80486 processor from 1989 would dissipate 10W, a Pentium IV from 2000 dissipates 120W. Nowadays, a blade server dissipates about 1KW, as much as a domestic refrigerator, and a commonly utilized blade server rack may need 40KW. A medium sized data center can use 250KW, while big data servers may use up to 10MW. Such high energy consumption numbers have alerted business and scientists into start developing more efficient IT.

As mentioned before, Green IT investigates IT related energy and environmental issues. Although new, this is a broad area of research, and contains very different subfields of study. The three main divisions of green IT are:

2.1.1 Eco-compatible management of the lifecycle of IT

This research area proposes to study new methodologies and technologies for eco-compatible manufacturing of IT components, to optimize packaging, and to minimize the environmental impacts of the whole lifecycle of IT. This includes eco-labeling and eco-compatible management and storage of waste and dismissed IT components. The motivation for this research comes from the need of recycling components and mainly from the existence of hazardous materials in the composition of electronic devices, such as heavy metals. It is known that 70% of the landfills of lead, cadmium and mercury derives from the IT industry (MINES et al., 2007). In many countries and in the European Union, current legislation delegates the responsibility of giving e-waste an eco-friendly end to the component maker (The European Parliament, 2003). And since this responsibility implies in an additional cost, there is a big interest in developing new and better methodologies for accomplishing the task.

2.1.2 IT as an enabler of green governance

This area is focused in the use of IT technologies to implement Green Governance policies. Green governance is a systematic approach adopted by a company or organization with the objective of driving it towards overall sustainability. Green IT in this case means using IT in tasks such as measuring and monitoring the green parameters. This includes the design of monitoring devices as well as decisional support systems and dashboards to store, analyze, and compare green KPIs. Examples of green parameters are energy consumption, temperature, generation of hazardous material, age of devices, and device usage among others. Understanding where each parameter and its origins enables the company to intelligently prioritize green actions like renewing devices or changing a product life-cycle. Green governance is not restricted to IT, but encompasses general areas in the business, such as:

- Product design;
- Strategy;
- Project management;
- Performance management;
- Infrastructure management;
- Human resources.

2.1.3 Energy efficiency of IT

Finally, this research area is related to energy efficiency in IT systems and IT hardware themselves. In IT energy efficiency the IT systems and IT infrastructure are the object of study and optimization, differently from the preceding research area on which IT technology was a mean for achieving greener solutions and optimization in other areas.

The objective of IT energy efficiency initiatives is to design energy-efficient IT architectures and data centers, covering also all the effects that utilizations practices have on energy consumption. Energy consumption impact on operating cost has grown in the last years and is now very significant.

It is pertinent to mention that IT energy efficiency, or more precisely the ratio between performance and energy usage has actually grown in the last years. This is because processor performance has had a great increase, while energy consumption has had considerable but smaller increase. For example, we already mentioned that the processor power went from 10W to 120W when comparing the Intel 80486 with the Pentium 4. But while the energy consumption ratio from the two processors in of the order of magnitude of 10, the ratio between their processing power (measured in MIPS - Millions of instructions per second) is of the order of magnitude of 10^3, representing a much bigger growth.

But although we have seen an increase in energy efficiency, the growth of the absolute value of energy consumption is still a problem that must be dealt with. Today, energy consumptions are by far higher than what they could be because current IT systems introduce a number of inefficiencies related to many different layers, which go from logical

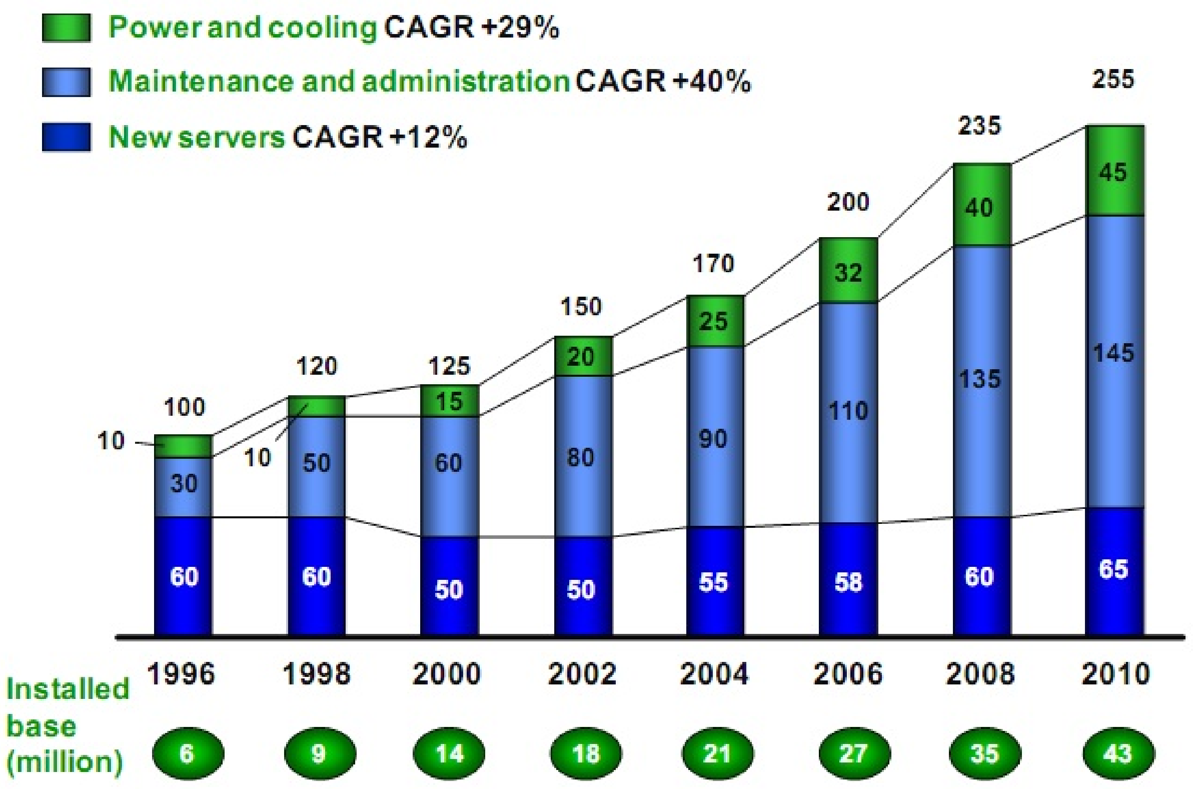

Figure 2.1: Global spending for server (Billion dollar, Source: Josselyin et al., 2006).

gates level to server's architecture and data centers infrastructure level. From another standpoint, this means that there are many possibilities for increasing the energy efficiency of IT, and this is a great challenge that the scientific community needs to face.

2.2 Impacts of IT Technology

Green IT is attracting more and more attention both in the scientific and business communities. In the past decades research and innovation have focused on increasing clock frequency and on miniaturization (SCHALLER, 1997), with only a marginal focus on power consumption, mainly associated with battery autonomy of laptop devices. This has resulted in extremely fast IT systems, but which consume a lot of energy that is very often inefficiently employed. Energy consumption has devastating effects on:

- Equivalent CO2 emissions;
- Operating costs;
- Scalability.

As consumptions rise, the attention on Green IT gains momentum, and initiatives such as new legislation work as an incentive on solutions that focus on minimizing the harming effects listed above. These effects can be separated in two areas:

2.2.1 Environmental impacts of IT

Environmental Impacts of IT involves from the lifecycle (manufacturing, use and disposal) of computers to the equivalent CO2 emissions caused directly and indirectly by it systems, such as powering and cooling. Both the inefficient use of energy and the manufacturing and disposal of computer systems leads to the generation and release of toxic compounds into the environment. Examples of toxic components are lead, barium and chromium (BAUL, 2002). Such components are currently inherent characteristics of computers, thus making the ecological production, use and disposal difficult and costly.

Aside from hazardous wastes, the production and use of computers consumes vast amounts of energy, which in most cases originate from nonrenewable sources such as fossil fuels, contributing then to climate change and global warming.

2.2.2 Competitiveness impact of IT

Green IT can make IT more competitive by dealing with the impacts of inefficiency and non optimal energy usage. In fact most of these impacts are directly related to high operational costs. Some of these costs are:

Power costs

An IT system which is inefficient in performing its tasks will require more energy than another more optimized system will. In other words, there is a greater cost per task (energy wise) caused by IT inefficiencies, be it originated by hardware or software;

Cooling costs

Cooling costs are a very relevant component of IT operational. Together with power costs, it represents 60% of the total operational spending for new IT infrastructures (JOSSELYIN et al., 2006). The interesting fact about cooling costs is that they accompany power costs. This is because cooling is more needed the more energy (and consequently, heat) is dissipated. As mentioned above, Green IT solutions can reduce powering costs by reducing inefficiency, thus making dissipating less energy per task. Consequently, cooling costs can also diminish;

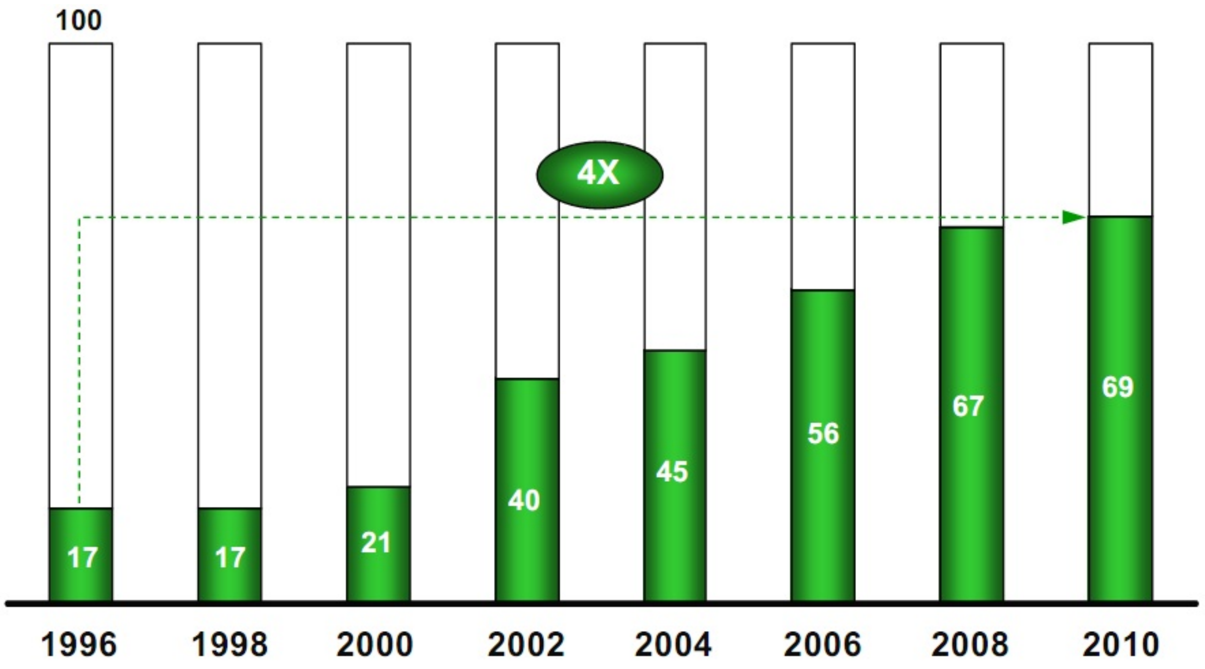

Figure 2.2: Spending for energy and cooling/spending for new servers (Percent, Source: Josselyin et al., 2006)

Idle processor time costs

When a Data Center or any data processing system is in project, it is usually dimensioned to be able to handle expected workload peaks. This means that when not in workload peaks, the processors will work less than they could, having idle time. The problem is that the higher cost of buying a better performing system is not being justified during all times when it is not performing close to or at maximum capability. Additionally, there is another issue. If a processor, for example, is under 50% of workload, it actually consumes more than 50% of the energy consumed when it is with 100% of workload. This is because other than the proportional consumption related to the calculations, there is a fixed component in the processor's energy consumption. Techniques such as virtualization can greatly reduce the idle time of processors thus optimizing the energy consumptions. Virtualization works by distributing tasks between a number of processors, thus reducing the workload variability and allowing for more cost-efficient system dimensioning.

Scalability limitations

Energy consumption is also a limit to the scalability of data centers. New IT equipment requires an extremely high quantity of energy per square meter (e.g., a rack with 5 blade servers of 8 units consume more than 20KW, as much as an apartment complex) and also the energy required by personal computers rises at a rate of 8-10% per year. When data centers are located in areas with high population density, as it often happens in Europe, it may be difficult for power distributors to bring the required energy in the same building. As power infrastructure modifications are difficult and expensive, data centers that are not energy efficient cannot expand their capabilities. According to Forrester Research (MINES et al., 2007), in the next few years 60% of data centers will be limited by power consumption, cooling, and space issues.

2.3 Green Software

Software plays a crucial role in this scenario, being a relevant driver of energy consumption. Although software does not directly consume energy, it deeply affects the consumption of hardware equipment. Software applications, ranging from operating systems and drivers for hardware devices to decision support systems and ERP suites, indicate how information should be elaborated and to some extent guide the functioning of hardware. Consequently, they are indirectly responsible of energy consumption. In fact, not only software, but the many different logical layers of an IT system are responsible for parts of the total amount of energy consumption. As an example, the energy consumed by a data center is absorbed by different components. Figure 2.3 (based on data from (RENZI, 2007), and (STANFORD, 2008)) shows that approximately 40% of the power entering a data center is used for cooling, distribution devices and batteries, and that an additional significant amount is used by auxiliary components of the servers (e.g., AD/DC converters, fans), whereas only 18% reaches the processor.

The processor may stay in idle for some time. Consequently only a minor part of energy is used for real computation (on average 3%). In addition to that, it is not yet clear what is the energy efficiency of the operations performed by the processor with respect with the final business operations, which are the goal of the whole data center working.

Even though most of the power absorbed by a data center is absorbed by the infrastructural layer, it is also important to investigate the role of software, which is the

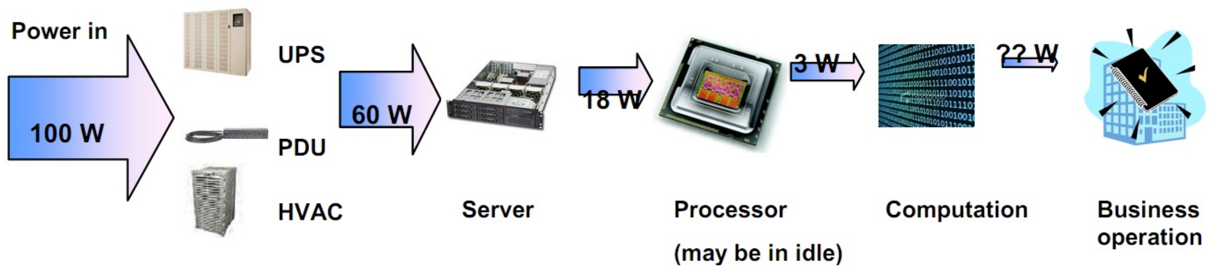

Figure 2.3: Power consumption break down in a data center.

first cause of consumption, as it guides the operations performed by the processor and thus influences the consumption of all the layers above. But contrarily from all the other layers, energy efficiency of the software layer of a data center remains largely unexplored. Researchers have not yet even agreed on a common methodology to measure and assess the energy efficiency of end-user applications, and no code-based predictive energy consumption metrics are so effective to be usable at this level.

Some works (e.g. (CHATZIGEORGIOU; STEPHANIDES, 2002)) propose methodologies to estimate software energy efficiency. Fornaciari (FORNACIARI et al., 1998) investigates low power embedded systems and introduce accurate and efficient power metrics to drive the hardware/software co-design. However, all these works are limited to embedded systems and cannot be extended to more complex software, such as business applications. The flow of operations performed by an embedded system is by far more predictable and less subject to change than in larger systems. In addition to that, in embedded systems software is tightly coupled with hardware and its power consumption can be more easily modeled. Depending on its complexity, business applications can have multiple layers (hardware, operating systems, middleware, database management system, end-users applications), with multiple and concurrent operations. Accordingly, the software developer has a lot of different choices and usually cannot modify the lower architectural layers.

Addressing this difficulty in evaluating software energy efficiency, Galli (GALLI, 2008) created a Green Metric for energy efficiency, which will be discussed later.

To accomplish the goal of this project, a better understanding of software metrics is needed, since they are the ones which will be later analyzed and put side by side with the software developer‘s survey responses. In the following section we will describe both classic Software quality metrics and Green Software Metrics.

2.4 Software Metrics

Software metric can be defined as the quantification of a generic property (any quantitative or dimensional attribute for example) of a software product or of the process of developing a software product (MILLS, 1988). Usually the measurement of software properties are motivated by needs such as estimating the production costs, improving the management of project activities, measuring software productivity or software quality etc. Good software metrics should not only describe a product or process characteristic but also help the estimation of related parameters. (CONTE; DUNSMORE; SHEN, 1986) describes the ideal metric as having the following characteristics:

- Simple and accurate definition;
- Objectivity;
- Relevance - The metric should focus on what it should really measure;
- Flexibility - the metric should be able to remain effective even with small changes in the product or process it measures.

2.4.1 Software Quality Metrics

Software quality is the measure of the success of a software product in fulfilling a number of expectations regarding the way it works or how its is structured.

Most research in the field of software engineering is devoted, directly or indirectly, to quality issues. In particular, research has been made in objectively establishing what software quality means, and in defining relevant quality metrics. Later research has focused on finding techniques for the measurement of these parameters and the development of technologies and methodologies that facilitate the creation of quality software.

Especially in the field of object-oriented languages, there are a number of measures of quality of software design ((MERLO, 2005), (CHIDAMBER; KEMERER, 1994), (HENRY; KAFURA; HARRIS, 1981)) that uses the source code as input. And although these metrics were not created with software energy efficiency in mind, one cannot discard the possibility of correlation.

Metric	**Acronym**	**Definition**
Coupling Between Objects	COB	Number of distinct classes with which a given clas is coupled, excluding inheritance relationships
Cyclomatic Complexity	CC	Number of linearly independent paths
Data Abstraction Couplings	DAC	Number of attributes that have a different class as their type
Depth of Inheritance Tree	DIT	Maximum depth of inheritance tree
Information flow metrics	IFM	Measure of the transference of information between two variables
Lack of Cohesion in Methods	LCM	Difference between the number of couples of methods without any fields in common and the number of couples that share at least one field
Number of Children	NCH	Number of direct subclasses
Number of Constructors	NCO	Number of constructors
Number of Methods	NM	Number of methods
Response For a Class	RFC	Cardinality of the set M containing the class methods and the sets of methods directly invoked by the elements of M
Weighted Method Per Class	WMPC	Sum of the weights of the methods of a class

Table 2.1: Classic software design quality metrics calculated over a single class or object.

Metric	Acronym	Definition
Attribute Hiding Factor	AHF	Ratio between the sum of the invisibility of all attributes (% of classes in which the attribute is not visible) and the total number of attributes
Attribute Inheritance Factor	AIF	Percentage of attributes that are inherited
Coupling Factor	COF	Ratio between the number of couplings between objects and the total number of possible couplings
Method Hiding Factor	MHF	Ratio between the sum of the invisibility of all methods (% of classes in which the methods is not visible) and the total number of methods
Method Inheritance Factor	MIF	Percentage of methods that are inherited
Polymorphism Factor	POF	Ratio between the number of polymorphic situations and the total number of possible polymorphisms
Cyclomatic Complexity	CC	Number of linearly independent paths

Table 2.2: Classic software design quality metrics calculated over a a group of objects.

(GALLI, 2008) compared the classic metrics for software quality found in Table 2.1 and Table 2.2 with data on energy consumption, and discovered that none of them had any correlation with energy consumption, meaning that they cannot be used as a green metric. Consequently, in order to evaluate energy efficiency in IT systems, one must use a new metric created specifically for this purpose, i.e., a Green Metric.

2.4.2 Expressivity Entropy of a language

The Expressivity entropy expresses the level of utilization of all functionalities a programming language has in a certain code fragment. In this context, functionality is understood as every option a developer has when writing software, be it the programming language's constructs, or even external libraries and frameworks. In other words it is an indicator of how much of a programming language's potential is being utilized by a given code fragment, where a big entropy means a more extensive use of the diversity of symbols available.

Symbols are any sequence of characters recognizable by a language's parser. They can be of many different categories depending on their function. In particular we have the following categories:

- Keywords: The set of all language keywords, including decision instructions (if, else, switch, etc.), iteration (for, while, etc.), variable types (int, char, void) and other language particular keywords (for example, in java: package, class, interface);
- Variables: The set of all variables created by the user;
- Internally defined functions: The set of all methods, objects and functions created by the user;
- Externally created constants and function: The set of all components written by others that are available to a developer. Usually represented by frameworks and external libraries.
- Mathematical symbols: The set of operators representing mathematical operation, both logical and algebraic, and also language specific mathematical operators (such as "+" when used in the concatenation of strings);
- Custom words: The set of all character sequences that are not classifiable as symbols of any other type. A typical example are strings in quotes;
- Parenthesis: All types of brackets;
- Other symbols: Mostly made of separators and punctuation (",", ":", ";" etc.)

In chapter 3 a methodology to measure energy efficiency using the Expressivity entropy will be shown. The methodology, empirically studied in (GALLI, 2008) groups software in two categories based on software classic metrics, and then uses the entropy as a proxy for Energy efficiency.

2.4.3 Green Metrics

The research for metrics to measure energy consumption is a complex issue, and largely unexplored. Some effort has been made in making metrics that look into the machine code (or java bytecode) in order to understand the efficiency of a program based on its microinstructions. More details on such metrics can be found on (GALLI, 2008).

In this project instead, software efficiency will be measured using the above mentioned expressivity entropy methodology.

2.4.4 Java Metrics Analyzer

The metrics described above will be calculated by means of a tool called JMA (Java Metrics Analyzer). It was developed by Galli (GALLI, 2008) in Java and is able to calculate metrics over machine code, java class files or java source code. For this project, an adaptation of this tool was needed. This adaptation will be described on chapter 3.3.2 Tool adaptation and metric gathering.

2.5 The European E-competences framework

As seen in the previous pages, an important part of this project is to gather metrics from the software sample to evaluate how they correlate with their respective software developers‘ profiles. Ultimately, the objective is to determine a correlation between developer profile and software energy efficiency. But while gathering the metrics is important, there is another relevant part, which is to somehow gather developer profile data in a way that makes it objectively comparable. In order to accomplish this task, a survey was created based on the European e-Competences Framework (http://www.ecompetences.eu/). More details on the survey can be found on Chapter 3.2. The European e-Competences Framework (e-CF) is a reference framework of information and communication technologies (ICT) competences that can be used and understood by ICT user and supply companies, ICT practitioners, managers and HR departments, the public sector, educational and social partners across Europe. It has been developed by a large number of European ICT and HR experts in the context of the CEN (European Committee for Standardization) Workshop on ICT Skills with the objective of standardizing the reference and the classification of job proficiency in ICT. The framework enumerates the different proficiencies and provides a description for each one, including additional information that helps classifying an ICT professional in one of the proficiency levels.

3 Methodology

This Chapter will describe the roadmap of the development of this project. After studying the state of the art in Green IT (see chapter 2), chapter 3 focuses on more specific points of study which were explored in this project.

Chapter 3.1 defines the project steps. Then, on Chapter 3.2 an important subject of this project is explained, that is, the creation of a complete but objective survey that covers and evaluates the competences of a software development professional. Chapter 3.3 Metric gathering and analysis covers another core part of the project, the methodology used to retrieve sample software and calculate their metrics.

3.1 Project Roadmap

This section aims at listing the activities involving this project in the order they were performed. The project had the following phases:

1. Study of related material on Green ICT, Green Software, and the e-Competences Framework;
2. Particularization of the e-Competences Framework for the Software Developer competences;
3. Software developer survey creation and validation;
4. Sending of the survey by e-mail to a sample of software developers;
5. Survey data collection and organization;
6. Download of software corresponding to answered surveys;
7. Filtering and validation of downloaded software;
8. Gathering of software metrics;

9. Analysis of the correlation between survey answers and Software metrics;

10. Conclusion.

3.2 Creation of a Survey to Evaluate Software Developer Competences

In order to gather software developer‘s data for this project‘s analysis, a questionnaire has been created and sent to a set of developers who use Sourceforge. Sourceforge (http://sourceforge.net) is a leading community dedicated to build open-source software projects. The choice for Sourceforge as the source for this project‘s developer database was motivated by its numerous members, and because the software and the developer contact is open to all members.

The creation of the questionnaire has been based on the European e-Competences Framework (citation needed). The European e-Competences Framework is a reference framework of information and communication technologies (ICT) competences. It takes into consideration a broad area of professions in ICT. More information about the framework can be found in Chapter 2.5.

This project studies the impact of the developer’s good practices and skills in the energy-efficiency of software. Therefore, the questionnaire has been created based on a subset of the Framework’s competences that are present in the context of the software developer profession. For instance, it makes no sense to consider the competence *Business Change Management* for our study, but it makes sense to consider *Testing*.

3.2.1 Software Developer Competences

The software developer profession is a particular subset in the set of ICT professions. For this reason, not all, but a subset of the competences listed on the e-Competences Framework have been taken into consideration. Also, further personalization has been done by means of rewriting the chosen competences so they refer more directly to the software developer. This differs from their original description that is not specific, but applicable to all ICT professions.

Accordingly, the following personalized competences have been taken into consideration for this case:

Specification Creation

Analyses and defines current and target status. Estimates cost effectiveness and designs decision templates. Develops structure plans, timescales and milestone descriptions. Maintains a project diary and manages status reporting and change requests. Documents system acceptance and completion reports. Defines delivery quantity and provides an overview of additional documentation requirements. Specifies correct handling of products and identifies adverse consequences of poor handling and treatment.

Systems Architecture

Specifies, refines, updates and makes available a formal approach to implement ICT technology, necessary to develop and operate ICT systems in line with business requirements. Identifies the components required, hardware, software and technology platforms that need to be integrated to meet current and future business needs. Ensures that all technical aspects take account of interoperability, scalability and usability.

Application Design

Plans and specifies the conceptual design of an application in accordance with ICT policy and user or customer needs. Estimates development costs, installation and maintenance of application. Selects appropriate technical options for building the solution. Validates the models with representative users.

Sustainable Development

Is able to estimate the impact of IT solutions in terms of energy consumption. Is able to sensitize various stakeholders (business and IT) on alternatives related to sustainable development, while remaining within the business strategy. Is able to establish a policy of purchasing IT eco responsible.

Design and Development

Designs and engineers software programs/modules and/or hardware components to meet required specifications. Follows a systematic methodology to analyze and build the required components and interfaces. Performs unit and system testing to ensure functional and performance criteria are met.

Testing

Constructs and executes systematic test procedures for IT systems or customer usability requirements to establish compliance with design specifications. Ensures that new or revised components or systems perform to expectation. Ensures meeting internal, external, national and international standards including health and safety for either usability, performance, reliability or compatibility. Produces documents and reports to evidence certification requirements.

3.2.2 Survey Aspects

Before creating a survey of this type, some considerations were taken to make it the most effective. First of all, it will be sent via e-mail to many developers. So, it is expected that the number of answers may be considerably smaller than the number of e-mails sent. To increase the response rate, the questionnaire will have the following two characteristics:

- Conciseness: A long questionnaire will discourage developers from answering it. A large number of questions may also make the analysis results less clear. Therefore the questionnaire must not be long;
- Multiple-choice answers: Several reasons render multiple-choice as the best option in this situation. They are less demanding to answer, increasing the response rate. They also make for easier analysis, reduce subjectivity by dividing answers into groups and reduce misinterpretation of the answers by the analysts, when compared to text answers.

The European e-Competence Framework suits well with the above aspects. It explains each competence, and describes different levels of proficiency. These levels can be adapted for this particular case and used as the multiple-choice answers of the question related to that competence. The only adjustments will be of rewriting the competence description and proficiency levels. Originally they were written in a general way, to make sense for any ICT profession. In this case, they will be rewritten to better suit the Software developer profession.

3.2.3 Survey Questions

Based on the personalized competences from the e-Competences framework, a survey with the following questions was built:

1. Regarding project documentation, how well do you analyze and define current and target project status, develop timescales and milestone descriptions?
2. How well do you maintain a project diary and manage status reporting and change requests?
3. Do you document system acceptance and completion reports?
4. Do you take responsibility for the strategic direction of product, technical architecture or technology development?
5. Do you take responsibility for complete project specification, supporting other developers in your team if needed?
6. Regarding the System Architecture, how well do you specify a formal approach for software implementation, identifying the components required and technology platforms that need to be integrated?
7. How well do you ensure that all technical aspects take account of interoperability, scalability and usability?
8. Are you able to define strategies to implement technology compliant with the project needs, and account for the current technology platform and latest technological innovations?
9. Do you plan and specify the conceptual design of an application in accordance with ICT policy and user needs, ensuring that the application is correctly integrated within an environment, and selecting appropriate technical options for building the solution?
10. How well are you able to estimate development, installation and maintenance effort for an application?
11. Do you ensure meeting standards for usability, performance, reliability or compatibility, producing documents and reports to evidence certification requirements?

12. How well are you able to estimate the impact of IT solutions in terms of energy consumption, and to establish a policy of purchasing IT with eco responsibility?
13. Are you able to sensitize various stakeholders on alternatives related to sustainable development, while remaining within the project strategy?
14. Do you construct and execute systematic test procedures to establish compliance with design specifications by organizing test programs or scripts to stress likely vulnerabilities, recording and reporting outcomes and providing analysis of results?
15. Do you provide software support by analyzing user feedback, and answering user questions?

For all the questions above, the developer was invited to choose between one of four choices:

- I don't do it;
- Poorly;
- Moderately;
- Perfectly.

The alternatives were made to be interpreted as of increasing intensity, quality or frequency. As a final step before being sent, the survey has been validated by 4 people with experience in the software development field. This validation resulted in some adjustments to better clarify the survey questions. For an exact model of the survey e-mail sent to the software developer sample, see Appendix A.

3.2.4 Question Grouping and Score

To be able to effectively measure, quantify and compare answers, a score was created. The score is simply a quantitative way to express the increasing intensity, quality or frequency of the multiple choice answers.

Survey answer Score

- I don't do it;

- Poorly;
- Moderately;
- Perfectly.

Additionally, the fifteen questions were grouped into semantic groups, depending on which subject they cover. The combined score of these semantic groups of questions were later compared in a correlation analysis with the retrieved software metrics. This grouping was done for two reasons. Firstly, the objective of the project is to understand how certain software development practices relate to energy efficiency (or a proxy of energy efficiency, such as expressivity entropy, in this project). Secondly, in order to attenuate the effects of outliers. If, for instance, a combine score of 4 questions is used for a correlation analysis, the impact of a question answered erroneously would be smaller than if the comparison were to be done with every single question.

The questions were grouped in the following way:

- Documentation or Specification Creation - Questions 1, 2, 3;
- System Architecture - Questions 4, 6, 9;
- Project Management - Questions 5,7,8,10;
- Application design and development - Questions 7,8,9,11;
- Sustainable Development or Green IT - Questions 12,13;
- Testing - Question 14;
- Handling feedback - Question 15;

3.2.5 About the Download, filtering and classification of Software sample

In this project, it was necessary to choose a Software Sample that allowed for the analysis to be done. This section will explain the conditions considered and the choice that has been made.

First of all, the software sample needed to be of a source that allowed its free download and its use for academic studies. The software preferably had to be written in Java. This is because parsing characteristics from the Java language make it easier to measure their metrics. This was also the main reason for the JMA tool, used in this project, to be developed to measure Java software. Another requirement was that the full availability of software source, since some of the metrics are based on the source code of the software.

These requirements directed the choice of the software sample to the Sourceforge repository (http://sourceforge.net). Sourceforge, in addition to meeting the above criteria, is the leading resource for open source software. Additionally, software developers who use sourceforge have their contacts available to all sourceforge users, which proved very useful in this project, since it was necessary to match software data with the respective developer data. A problem with sourceforge was that it has little standardization and control of its users' uploads. It was then expected that if a certain amount of software was initially selected, the final number of software matching all of the requirement would be considerably smaller.

As an additional note, the sending of the survey and the compilation of its results would be impossible without a survey tool. For this project, a free web survey tool was used. It was chosen because it was free, and also because it had no limitation on the number of questions asked, neither on the number of survey responses. The tool used can be found on www.esurveyspro.com.

Details can be found on Chapter 4 on the obtained results of the software developer survey, on technical aspects of the software download, and also on its filtering and classification.

3.3 Metric gathering and analysis

This section explains the methodology used for the download, filtering, metric gathering (including the adaptation of the JMA tool), and later classification and analysis of software.

3.3.1 Download and Filtering

As explained in Section 3.2, the adapted e-competences survey would be sent to numerous sourceforge.net users. The following step would be then to wait for the targets to send their responses.

After the compilation of the results from the survey, the usernames of the targets had to be used to search and download their correspondent software on sourceforge.net.

As mentioned before in this document, sourceforge.net has little standardization of the content uploaded by its users. Consequently, the process of software downloading also included the filtering of software. Only those with complete source code and written in java would be useful for this project. Because of this prerequisite, it was expected that the number of downloaded software after this step would be reduced compared to the total amount of survey answers.

3.3.2 Tool Adaptation and Metric Gathering

Following the download and filtering of software, the JMA (java metrics Analyzer) tool was used to calculate both the quality metrics and the expressivity entropy metric. But at this point an adaptation of the tool was needed.

The JMA tool was developed mainly to measure classic quality metrics form single files and whole programs, to measure the expressivity entropy metric from single files and to measure green metrics. In this project, it was used to measure classic quality metrics from whole programs, and to measure the average expressivity entropy from whole programs. Thus, some changes were made to adapt JMA for this project's objectives.

The first change was in the classic quality metric measurement. The tool initially measured both program metrics and single file metrics. The change here was to make the tool measure and calculate the weighted average values for all single file metrics in the same time it measures program metrics. In this way, it was possible to run the tool in a multi-file jar package and measure both single file metrics (returning the average) and program metrics.

The second change was to make it possible to calculate the expressivity entropy of software. The program initially was able to measure it from single java files. For this project, it was needed to have a value for the entropy of the whole program. The adaptation, then, was made so that the entropy measurement would recursively work on all program files, and the final result would be its weighted average.

With the tool adapted for the needs of this project, the measurement took place. Details on the number of measure software and on results can be read on Chapter 4.

3.3.3 Classifying and Analysis

Having both compiled data from the survey responses and collected the metrics from their respective software, one last step was done before the correlation analysis was made. It was the classification of data into groups. This grouping was done to software in the following way.

The software was classified into two groups based on their classic quality metrics. As already mencioned in Chapter 2.4.2, according to empirical data collected and analyzed by Galli in (GALLI, 2008), a good proxy for measuring energy efficiency is the Expressivity Entropy of a program. But this entropy is directly proportional to energy efficiency in a certain group of programs, and inversely proportional in another group. These groups are characterized by their program's classic metric values.

Red Group

High complexity programs, with more intense use of inheritance, higher cyclomatic complexity and bigger binary length. For this group, a higher value of Expressivity entropy relates to bigger energy efficiency, i.e., decreasing energy consumption.

$$Energy\ Efficiency \propto \frac{Expressivity\,Entropy}{Binary\,Length} \tag{3.1}$$

Green Group

The opposite of the red group, with less complex programs. For this group, a high Expressivity entropy value points to inefficiency, i.e., more energy consumption.

$$Energy\ Efficiency \propto \frac{1}{\frac{Expressivity\,Entropy}{Binary\,Length}} \tag{3.2}$$

Metric	**Red Cluster avg**	**Green Cluster avg**	**Total Avg**
AHF	0,665	0,563	0,621
AIF	0,154	0,097	0,130
COF	0,097	0,093	0,095
MHF	0,226	0,213	0,220
MIF	0,242	0,175	0,213
CC	4356	1028	2938
CBO	1,106	1,110	1,108
DIT	0,143	0,076	0,114
LCOM	1,631	1,533	1,589
RFC	3,760	4,498	4,074
WMC	1,546	1,607	1,572
NOM	2378	690	1659
BL	97766	20814	64787

Table 3.1: Average values for the red and green groups according to empirical observation.

4 Implementation, Analysis and Results

The objectives of this chapter are to describe the execution of the data collection and organization, to expose the data in a meaningful way and to perform analysis of it. The results of the analysis will indicate which correlations are relevant. Ultimately, the results should indicate if and how software developer's competences and software metrics are related.

4.1 Implementation

4.1.1 Software download and filtering

The methodology for software download and filtering can be found at section 3.3.1. In this chapter the practical aspects of software downloading and filtering will be explained.

The process of software download from Sourceforge (see Chapter 3.2.5) started after collecting the developer's survey answers. The total number of answers obtained was 103. The 103 sourceforge developer pages were visited, but due to the lack of standardization of uploads from sourceforge, many of the respective programs were simply not available. In some cases, only the executable program was available but not the source code. After this step the number of downloaded programs which were complete and allowed for correct metric measuring was of only 20. This was most unfortunate, since such a small number of data points make any statistical results weaker.

4.1.2 Metric Gathering

From the 20 obtained programs were extracted the values of classic metrics and of Expressivity entropy. The measurement was made using the modified JMA program (Java Metrics Analyzer) (see chapter 3.3.2). The individual program names and developer names cannot be disclosed in this document, since anonymity was guaranteed to those who answered the survey.

4.1.3 Software Grouping

As seen in chapter 3.3.3, using the Expressivity entropy as a proxy for energy efficiency requires that the target programs are classified in two groups. The grouping was done to the 20 program sample, resulting in a group with 11 programs and a group with 9 programs.

The Red Group, corresponding to the programs of higher complexity and with Energy Efficiency directly proportional to the Entropy divided by the programs binary length, ended up with 9 programs. The Green Group, corresponding to the programs with lower complexity and with Energy Efficiency inversely proportional to the Entropy divided by the programs binary length, ended up with 11 programs.

In more detail, for every program, its classic metrics were compared to the average values in table 3.1 (see chapter 3.3.3). After the comparison, the program was allocated at the group it matched better.

4.2 Analysis

This section will present the empirical results of the comparison between the score obtained by a developer in the survey and the energy efficiency (measured by a proxy, see section 3.3.3) of the respective program. Each developer competence was made into a group, and the questions pertinent to that group were used to create a score for each competence (see section 3.2.4). Then, each developer competence score has been compared with the programt's energy efficiency individually, and the graphical representation will be shown in the following sections. It is important to mention that, since only 20 data points were obtained, the correlation test is statistically not as strong as it could be, and it is possible that if the same process is applied to another sample, results may differ.

4.2.1 Classic Results

As seen on section 3.3.3, the measurement of classic metrics in this project was used to group programs in two categories. This grouping was guided by the averages on table 3.1. The average value of classic metrics from the 20 program sample was slightly different from the reference average.

Metric	Reference avg	Sample avg
AIF	0,130	0,157
MIF	0,213	0,220
CC	2938	1654
DIT	0,114	0,717
BL	64787	39680

Table 4.1: Average values for sample of 20 programs.

4.2.2 Lower Complexity Software

The data obtained for the Green group of software with lower complexity is shown below. Each graphic compares one of the developer competences with its Expressivity entropy. For this group, a higher value of Entropy means less efficient software.

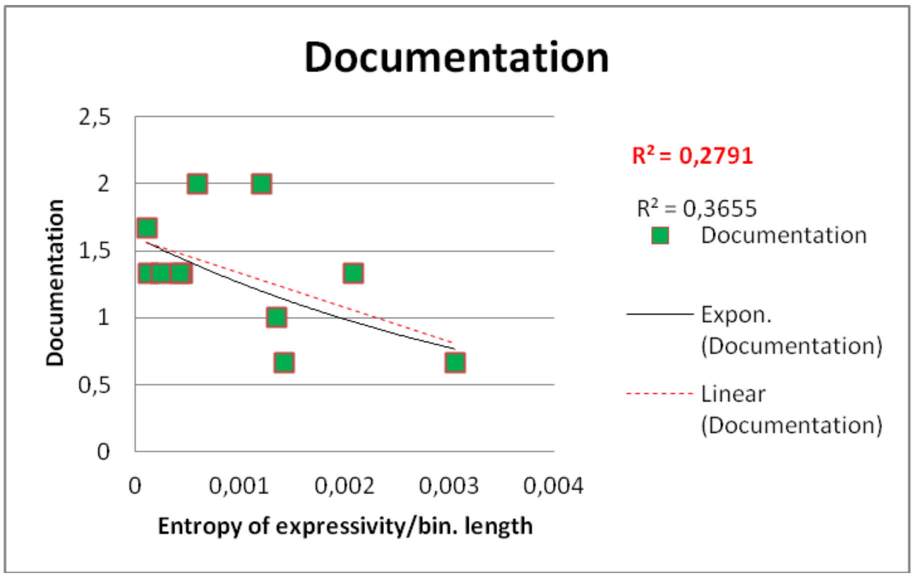

Figure 4.1: Green group - Documentation versus Entropy/binary length.

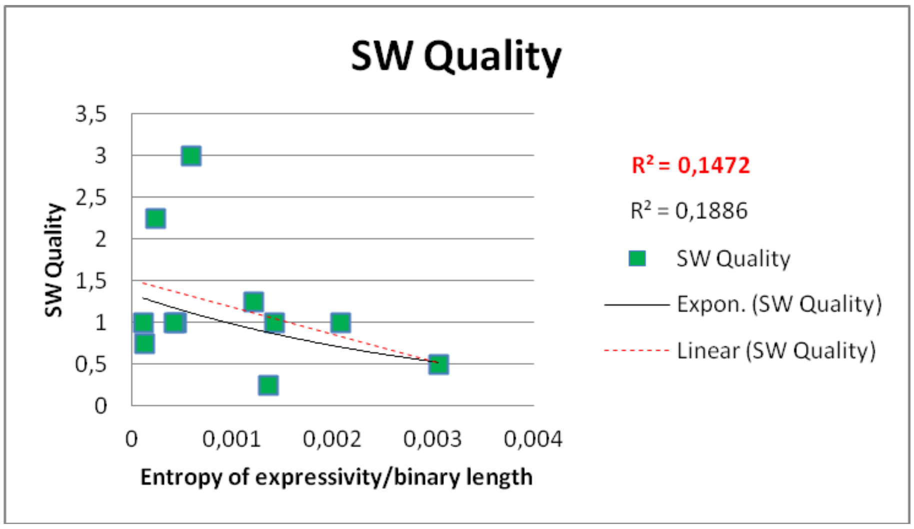

Figure 4.2: Green group - Software Quality versus Entropy/binary length.

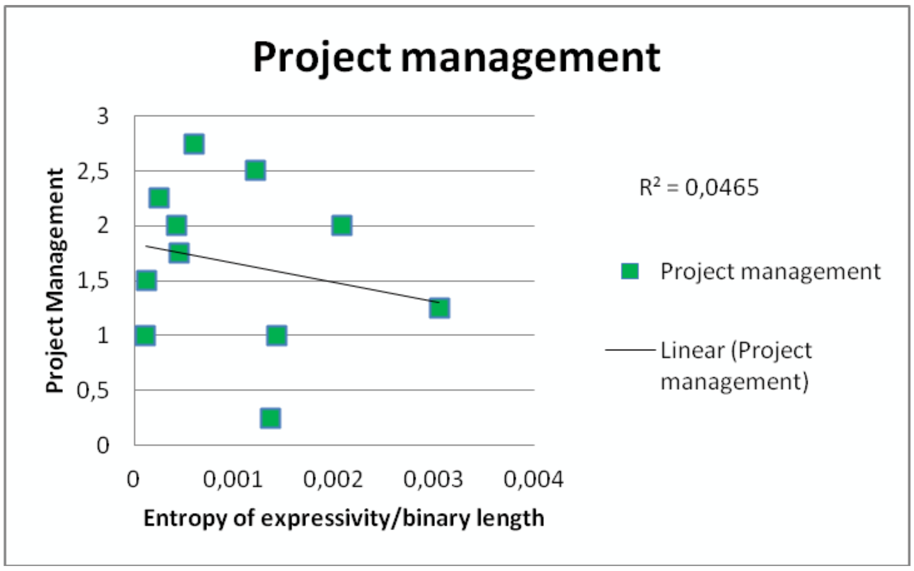

Figure 4.3: Green group - Project management versus Entropy/binary length.

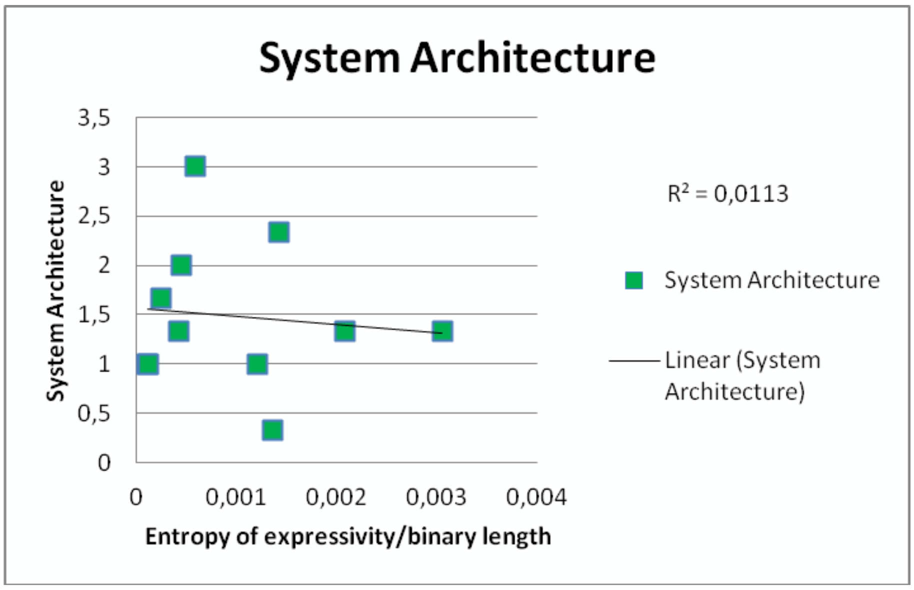

Figure 4.4: Green group - System Architecture versus Entropy/binary length.

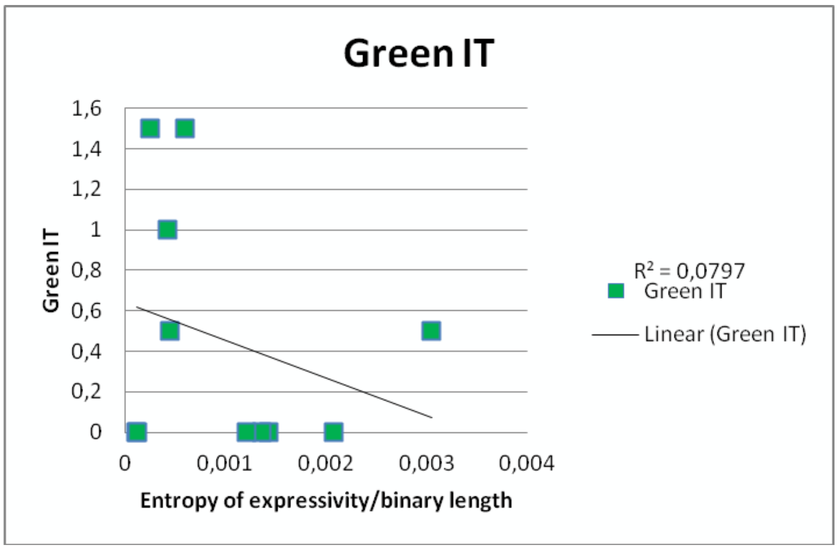

Figure 4.5: Green group - Green IT versus Entropy/binary length.

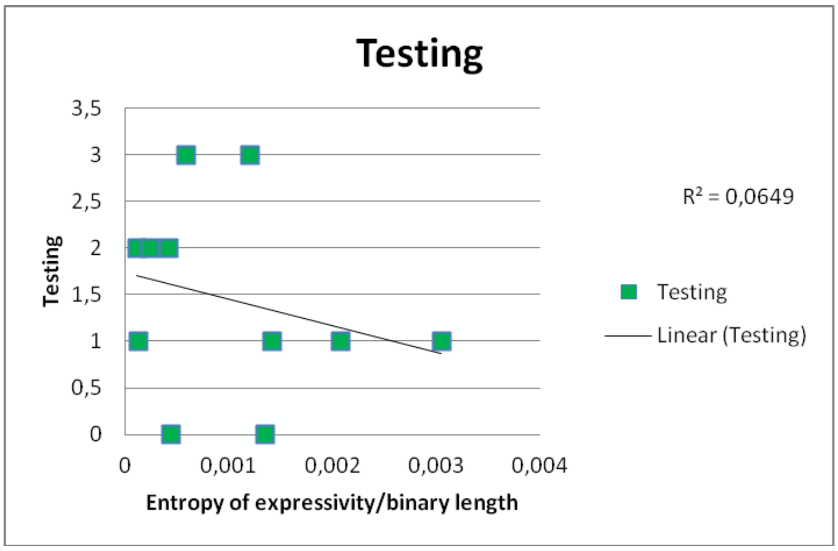

Figure 4.6: Green group - Testing versus Entropy/binary length.

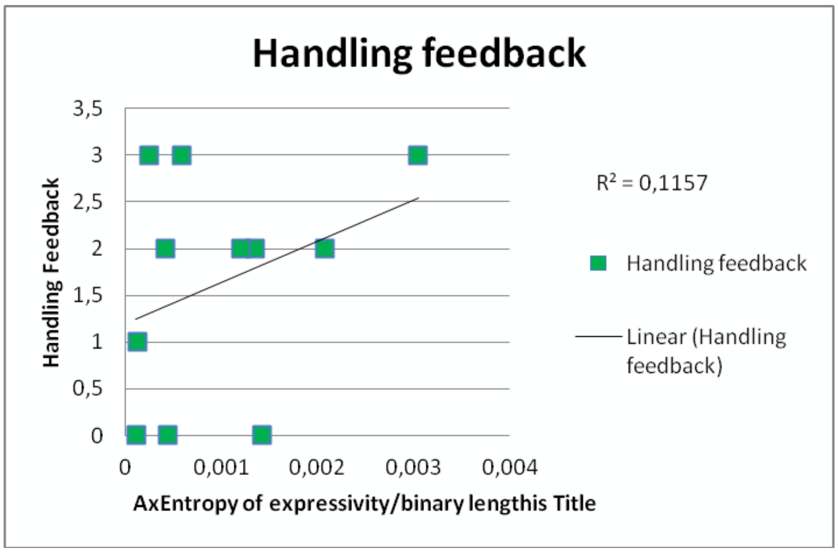

Figure 4.7: Green group - Handling Feedback versus Entropy/binary length.

4.2.3 Lower Complexity Software Conclusions

As expected, such small number of data points makes it hard to draw any conclusion regarding the correlation. We may consider then, that the results shown above only argument to a probable correlation or not between the analyzed factors, but in order to have stronger conclusions, more data would have to be analyzed.

Nevertheless, looking at the obtained results it can be observed that, for the green group, the competence that is more related to the energy efficiency is the practice of systematic software documentation. With a linear r^2 of $0,28$ ($0,37$ exponential r^2), we can say that for our sample, developers who more intensively practice systematic documentation of software result in having software that is more energy efficient.

For the competence of Design and Development (Software Quality), a linear correlation index of $0,15$ with entropy was found. Although the r^2 is small, it strengthens the intuitive notion that following methodologies of design and development will result in software that is better suited in doing specific tasks. It is reasonable to expect that software developed without such preoccupation with design and development may perform its task inefficiently.

For the other competences the observed r^2 was small, showing little to no correlation in the sample.

4.2.4 Higher Complexity Software

The data obtained for the Red group of software with higher complexity is shown below. Each graphic compares one of the developer competences with its Expressivity entropy. For this group, a higher value of Entropy means more efficient software.

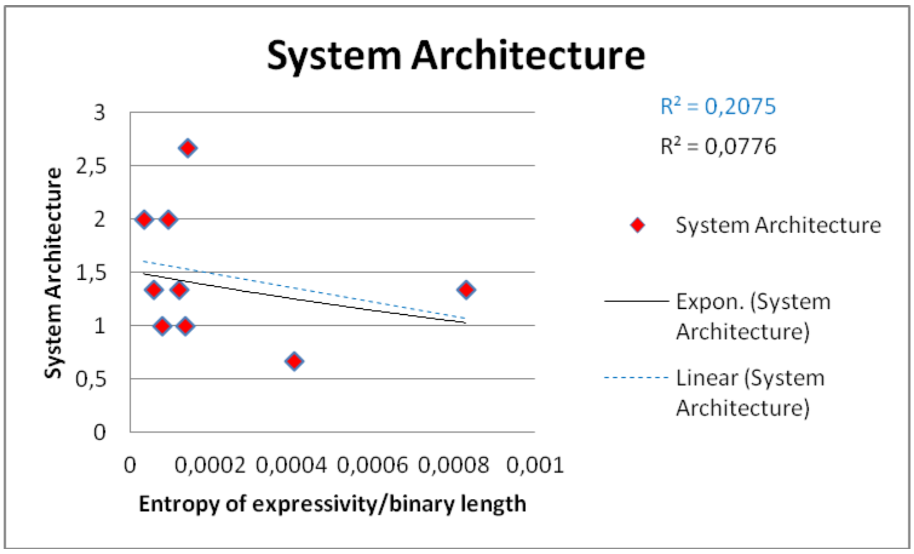

Figure 4.8: Red group - System Architecture versus Entropy/binary length.

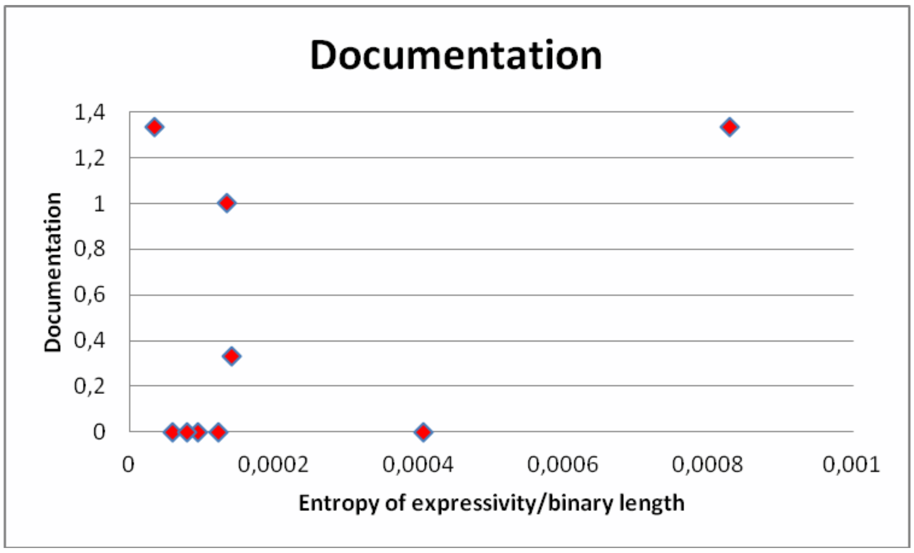

Figure 4.9: Red group - Documentation versus Entropy/binary length.

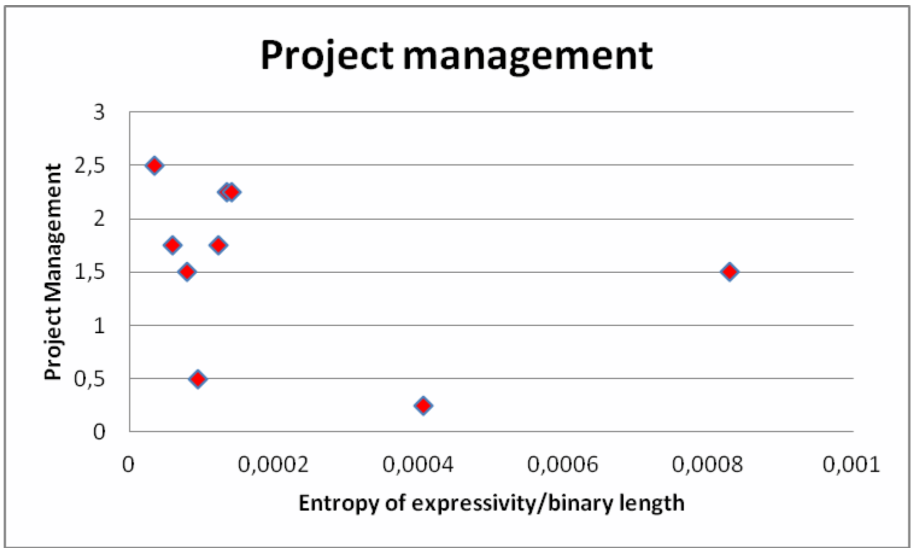

Figure 4.10: Red group - Project Management versus Entropy/binary length.

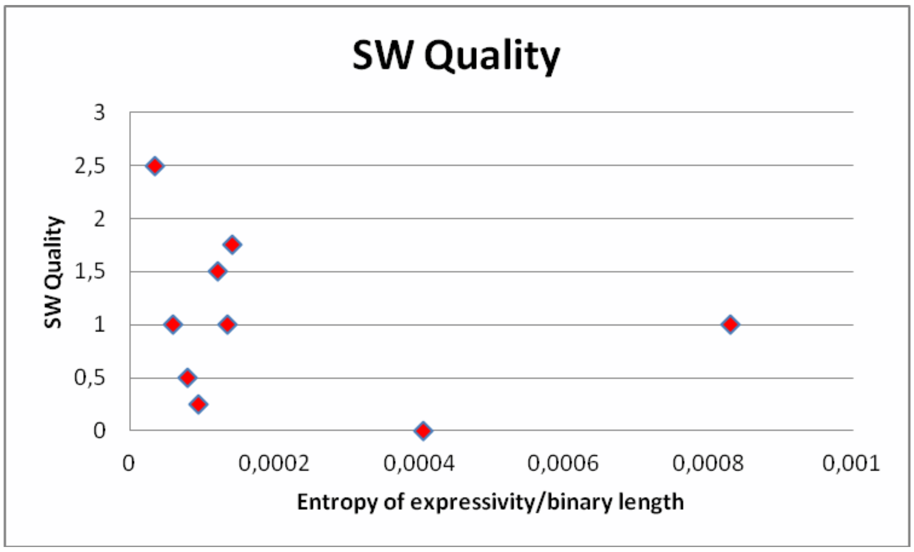

Figure 4.11: Red group - Software Quality versus Entropy/binary length.

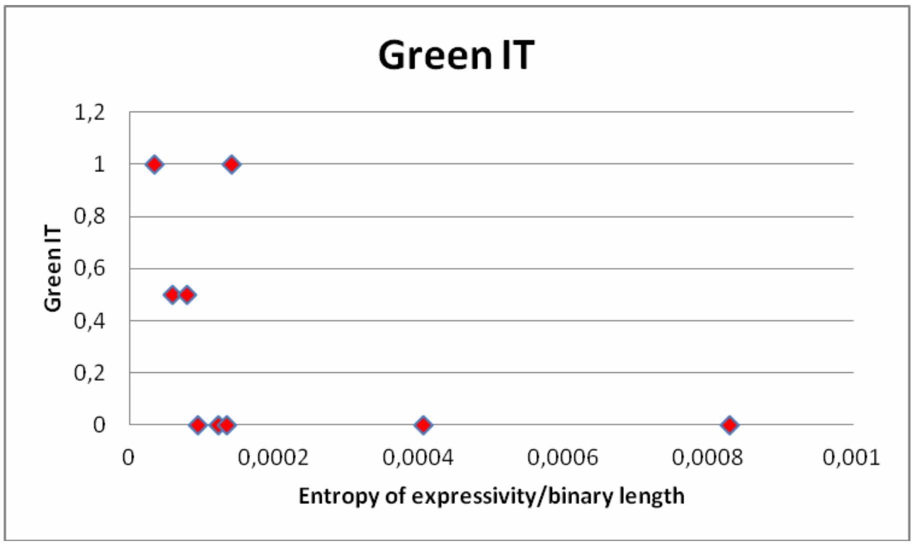

Figure 4.12: Red group - Green IT versus Entropy/binary length.

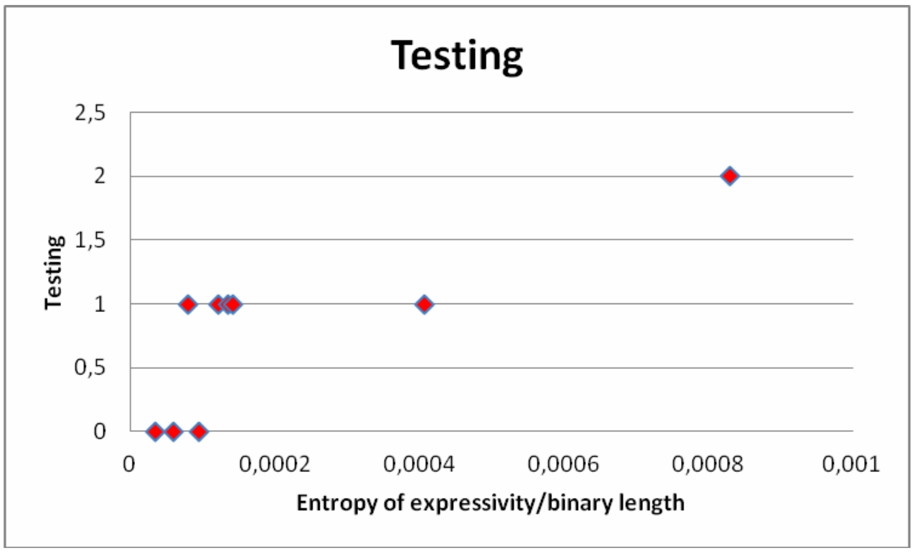

Figure 4.13: Red group - Testing versus Entropy/binary length.

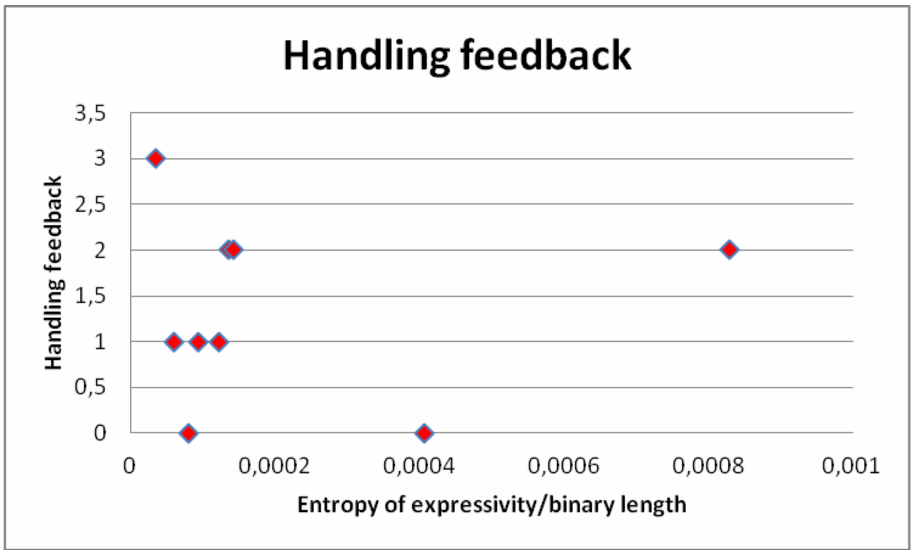

Figure 4.14: Red group - Handling Feedback versus Entropy/binary length.

4.2.5 Higher Complexity Software Conclusions

For the red group, we can see from the graphic representation of the data shown above that the small number of data point makes it difficult to have any conclusion. Another problem is the fact that data is not spread, on the contrary, many points are almost together.

Even if outliers were to be taken out, the conclusion for this group would remain the same, that there is not enough data to prove or disprove correlation between the level of developer competences and energy efficiency of the programs.

5 Conclusions and Future Work

This project is one more step taken in studying an area that has only recently started being studied, and that certainly still can promote much discussion. The study of the State of the Art showed that while Green IT is lately focusing attention, the Green Software subject in particular has been largely unexplored. To arrive at the intended goals of understanding the correlation between software developer profiles and software energy efficiency, a complete methodology was created. This very methodology can be considered one of the end results of this project, since it can be replicated with different software developer samples to test the same correlation tested in this work. The survey part of the methodology is specially interesting. As it is explained in the project, the survey is based is the result of a particularization of the European e-competences framework. While the framework generally describes competences for all ICT professions, the survey covers only competences pertinent to Software developing. In other words, it is a survey focused on the developer profession, but that is still an adaptation of a framework created by a respectable institution, the European Committee for Standardization. It is also relevant that the survey by itself is not restricted to the Green IT context, but covers software developing competences in general. Regarding the final analysis of the software metrics data and survey answers, it would not reasonable to draw any strong conclusions. The number of data points obtained is not large enough, and may give misleading results. Considering this, for the studied sample there was a considerable correlation between the practice of structured documentation and software energy-efficiency in the subset of lower complexity software. It cannot be considered a strong statistical result, but it advocates in favor of structured development even for simpler programs. The smaller number of resulting data points in comparison to the initial expectations was mainly caused by the wrong assumption that most programs uploaded to sourceforge.net were complete, including source code. Indeed, were this the case, the number of data points would be close to one hundred. Instead, due to the lack of standardization, there was a reduction of 80% of the number of data points, arriving at 20. Suggestions of future work in this project's subject include using the same methodology created here, but considering not only classical metrics and Expressivity Entropy, but also other green metrics, such as those used by Galli (GALLI, 2008). An additional possibility would be to consider a larger and more heterogeneous data sample. A possibility is to include not only sourceforge is the sample, but also developers and software that can be reached through the university's network, such as students and software companies.

Bibliography

ALBERIO, A.; BRIANZA, B. Un'analisi esplorativa sull'ottimizzazione del consumo energetico del software. *Tesi di Laurea*, Politecnico di Milano, 2009.

BAUL, H. Error: dumping does not compute. *Alternatives Journal. 28:2*, 2002.

CHATZIGEORGIOU, A.; STEPHANIDES, h. G. Energy metric for software systems. *Software Quality Control*, Kluwer Academic Publishers, Hingham, MA, USA, v. 10, p. 355–371, December 2002. ISSN 0963-9314.

CHESTNEY, N. It industry joins energy efficiency push, http://www.reuters.com/article/technologynews/idustre50r4al20090129. Jan 2009.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design, http://portal.acm.org/citation.cfm?id=630808.631131. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 20, p. 476–493, June 1994. ISSN 0098-5589.

CONTE, S. D.; DUNSMORE, H. E.; SHEN, V. Y. *Software engineering metrics and models*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1986. ISBN 0-8053-2162-4.

FORNACIARI, W. et al. Power estimation of embedded systems: a hardware/software codesign approach, http://dx.doi.org/10.1109/92.678887. *IEEE Trans. Very Large Scale Integr. Syst.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 6, p. 266–275, June 1998. ISSN 1063-8210.

GALLI, G. Un approccio alla valutazione dei consumi energetici del software basato sull'analisi statica del codice. *Tesi di Laurea*, Politecnico di Milano, 2008.

HENRY, S.; KAFURA, D.; HARRIS, K. On the relationships among three software metrics, http://doi.acm.org/10.1145/800003.807911. In: *Proceedings of the 1981 ACM workshop/symposium on Measurement and evaluation of software quality*. New York, NY, USA: ACM, 1981. p. 81–88. ISBN 0-89791-038-9.

JOSSELYIN, S. et al. Worldwide and regional server 2006-2010 forecast. *Technical report, IDC*, Nov 2006.

KUMAR, R. Important power, cooling and green it concerns. *Gartner report*, p. 4, 2007.

MERLO, F. Un modello orientato ai costi per l'ottimizzazione delle politiche di riutilizzo del software. Politecnico di Milano, 2005.

MILLS, E. Software metrics, curriculum module sei-cm-12-1.1. *Software Engineering Institute*, Carnegie-Mellon University, 1988.

MINES, C. et al. Topic overview: Green it. *Forrester Research*, Nov 2007.

RENZI, F. Evolution scenarios for data centers technologies and architectures and impacts on energy consumptions. *IBM, Presentation held at the conference "Green ICT"*, Milan, Italy, Nov 2007.

SCHALLER, R. R. Moore's law: past, present, and future, http://portal.acm.org/citation.cfm?id=254613.254618. *IEEE Spectr.*, IEEE Press, Piscataway, NJ, USA, v. 34, p. 52–59, June 1997. ISSN 0018-9235.

STANFORD, E. Environmental trends and opportunity for computer system power delivery. *20th Int'l Symposium on Power Semiconductor Devices and IC's*, 2008.

The European Parliament. Directive 2002/96/ec of the european parliament and of the council of 27 january 2003 on waste electrical and electronic equipment,weee, http://eur-lex.europa.eu/lexuriserv/lexuriserv.do?uri=oj:l:2003:037:0024:0038:en:pdf. 2003.

TOWNSEND, M.; BURKE, J. Earth will expire by 2050, http://www.guardian.co.uk/uk/2002/jul/07/research.waste. Jul 2002.

APPENDIX A -- Survey e-mail

The following is a model of the actual survey e-mail that will be sent to the set of developers: Subject: Survey about Software Developer competences Dear <Name>, We are a research group from Politecnico di Milano University, and we are currently studying the software developer characteristics and the impacts caused by different developer competences. It would be very helpful if you could take a few minutes to answer to some questions. Thank you very much! Please answer these questions considering the competences and skills you had during the Project: <Project Name>, but not those you acquired after that. The answers are in a scale from the lowest to the highest skill level. Keep in mind that your answers will be used only for academic analysis of tendencies. None of your personal data will be mentioned or used at any time, nor it will be related to your answers in any way. Yours, sincerely, Emilio Pietro Rosseto

1. Regarding project documentation, how well do you analyze and define current and target project status, develop timescales and milestone descriptions?
   - I don't do it;
   - Poorly;
   - Moderately;
   - Perfectly.
2. How well do you maintain a project diary and manage status reporting and change
   - I don't do it;
   - Poorly;
   - Moderately;
   - Perfectly.
3. Do you document system acceptance and completion reports?

•I don't do it;

•Poorly;

•Moderately;

•Perfectly.

4.Do you take responsibility for the strategic direction of product, technical architecture or technology development?

•I don't do it;

•Poorly;

•Moderately;

•Perfectly.

5.Do you take responsibility for complete project specification, supporting other developers in your team if needed?

•I don't do it;

•Poorly;

•Moderately;

•Perfectly.

6.Regarding the System Architecture, how well do you specify a formal approach for software implementation, identifying the components required and technology platforms that need to be integrated?

•I don't do it;

•Poorly;

•Moderately;

•Perfectly.

7.How well do you ensure that all technical aspects take account of interoperability, scalability and usability?

•I don't do it;

•Poorly;

•Moderately;

•Perfectly.

8. Are you able to define strategies to implement technology compliant with the project needs, and account for the current technology platform and latest technological innovations?

   •I don't do it;

   •Poorly;

   •Moderately;

   •Perfectly.

9. Do you plan and specify the conceptual design of an application in accordance with ICT policy and user needs, ensuring that the application is correctly integrated within an environment, and selecting appropriate technical options for building the solution?

   •I don't do it;

   •Poorly;

   •Moderately;

   •Perfectly.

10. How well are you able to estimate development, installation and maintenance effort for an application?

   •I don't do it;

   •Poorly;

   •Moderately;

   •Perfectly.

11. Do you ensure meeting standards for usability, performance, reliability or compatibility, producing documents and reports to evidence certification requirements?

   •I don't do it;

   •Poorly;

   •Moderately;

   •Perfectly.

12. How well are you able to estimate the impact of IT solutions in terms of energy consumption, and to establish a policy of purchasing IT with eco responsibility?

    •I don't do it;

    •Poorly;

    •Moderately;

    •Perfectly.

13. Are you able to sensitize various stakeholders on alternatives related to sustainable development, while remaining within the project strategy?

    •I don't do it;

    •Poorly;

    •Moderately;

    •Perfectly.

14. Do you construct and execute systematic test procedures to establish compliance with design specifications by organizing test programs or scripts to stress likely vulnerabilities, recording and reporting outcomes and providing analysis of results?

    •I don't do it;

    •Poorly;

    •Moderately;

    •Perfectly.

15. Do you provide software support by analyzing user feedback, and answering user questions?

    •I don't do it;

    •Poorly;

    •Moderately;

    •Perfectly.