



**UNIVERSITY OF PATRAS
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
ELECTRONICS AND COMPUTERS SECTOR
HUMAN-COMPUTER INTERACTION GROUP**

**A NETWORKED USAGE DATA LOGGER
FOR SYMBIAN PHONES**

**DIPLOMA THESIS OF
PATRICIA GÓMEZ TEJEDOR**

STUDENT OF THE UNIVERSITY OF VALLADOLID

SUPERVISORS: PROF. N. AVOURIS

DIPLOMA THESIS NUMBER: /2008

JUNE 2008

First of all, thanks to Mr. A. Stoica and Mr. C. Sintoris because they have made possible the development of this diploma thesis helping in every moment that I needed it and repeating each day the famous “Trabaja, trabaja!!”. Thanks also to the Human-Computer Interaction Group that has welcome us in their working group.

Specially, I have to be grateful for the support of “my Spanish Team”, Manu, Héctor, Jesús, Olga, Berni, Blanca and Raúl, that encourage me in the bad moments and has been there for everything I needed. I have to remember everybody from the Small Estia, the ones that are still there and the ones that already left, for contribute to reach this goal. And, of course, thanks to the nice Greek people, who taught us a different way of living.

Besides, I don't want to forget the grateful to Mr. Erasmus for giving us the opportunity of living this experience and spent the BEST year of our life. Finally thanks to my family and friends for encouraging me from Valladolid and all Spain.

CERTIFICATION

It is certified that the Diploma Thesis with the title:

A Networked Usage Data Logger for Symbian Phones

Of the student of the University of Valladolid, Spain.

GÓMEZ TEJEDOR, PATRICIA

(Επώνυμο)

(Όνομα)

(Πατρώνυμο)

(ΑΜ)

Was presented in public at the Department of Electrical and Computer Engineering of the
University of Patras, Greece on **June 26**.

The Supervisor

The Head of the Electronics and Computers Division

Professor: N. AVOURIS Student: PATRICIA GÓMEZ TEJEDOR

Title:

A Networked Usage Data Logger for Symbian Phones

Abstract

This diploma thesis includes the design and the implementation of an application that allows to a Symbian OS Mobile Phone, to collect data and to store it, in the own device, or sending it through Internet to a server, in order to be used for an analyser to do Usability Evaluation. Before the implementation of this Data Collector tool, the diploma thesis contains a study about the existing technologies related to smarthphones and an overview of the uses where this tool created can contribute to, and an introduction about how data collection can influence in the acceptance of a new application from users, doing previously usability evaluation focusing in the interaction between groups of users and this new tool.

For this reason, data gathering and usability evaluation are introduced, emphasizing their importance in the process of developing a new tool in the field of Human-Computer Interaction, and reviewing the techniques for collecting data and evaluating mobile applications. Besides, in order to evaluate the tool created, it has been tested by a developer, including it in a real application, to log its data and analyse its interaction; and then, giving back its impressions and feedback about the data collector tool.

The resulted application will be included in a larger project of the Human-Computer Interaction group of the University of Patras, concerning about collecting data to do usability evaluation in public places such as museums. This diploma thesis, “**A Networked Usage Data Logger for Symbian Phones**”, includes the design, implementation and evaluation of this application.

INDEX

1	INTRODUCTION	9
1.1	GENERAL IDEA OF THE APPLICATION.....	9
1.2	GOAL OF THE PROJECT	11
1.3	CONTENT OF THE THESIS.....	11
1.3.1	<i>Background</i>	11
1.3.2	<i>Technologies</i>	11
1.3.3	<i>Design</i>	12
1.3.4	<i>Implementation</i>	12
1.3.5	<i>Evaluation</i>	12
1.3.6	<i>Conclusion</i>	13
2	BACKGROUND.....	14
2.1	INTRODUCTION TO HUMAN-COMPUTER INTERACTION	14
2.1.1	<i>What is Human-Computer Interaction?</i>	14
2.1.2	<i>Content of Human- Computer Interaction</i>	15
2.2	DATA COLLECTION.....	16
2.2.1	<i>What is Data Collection?</i>	16
2.2.2	<i>Pros and Cons of Data Collection</i>	17
2.3	USABILITY EVALUATION OF MOBILE APPLICATIONS.....	18
2.3.1	<i>What is Usability Evaluation?</i>	18
2.3.2	<i>Why is useful to do usability evaluation?</i>	19
2.3.3	<i>Why Usability Data is Difficult to Analyze and Communicate?</i>	20
2.3.4	<i>Techniques for Analyzing and Communicating Usability Data</i>	21
3	TECHNOLOGIES.....	26
3.1	SMARTPHONES	26
3.1.1	<i>What is a smartphone?</i>	26
3.1.2	<i>Software in smartphones</i>	27
3.1.3	<i>Operating systems in smartphones</i>	27
3.2	SYMBIAN OS FOR MOBILE DEVICES.....	34
3.2.1	<i>What is Symbian?</i>	34
3.2.2	<i>Symbian OS Phones</i>	34
3.2.3	<i>The device: Nokia N70</i>	36
3.3	THE DEVELOPING TOOL: CARBIDE.C++	37
3.3.1	<i>What is Carbide.C++?</i>	37
3.3.2	<i>Pros and Cons of the development in Carbide.c++</i>	37
3.3.3	<i>Software Development Kit (SDK)</i>	38
3.3.4	<i>The emulator</i>	40
4	DESIGN	42
4.1	OVERVIEW OF THE WHOLE SYSTEM	42
4.2	DATA COLLECTOR TOOL FOR SYMBIAN PHONES.....	44
4.2.1	<i>What is a Dynamic Link Library?</i>	44
4.2.2	<i>Class diagram of the tool</i>	45
4.2.3	<i>Application Programming Interface</i>	47
5	IMPLEMENTATION.....	65
5.1	DATA COLLECTOR TOOL FOR SYMBIAN PHONES.....	65

5.1.1	<i>Prepare the data to be log</i>	65
5.1.2	<i>Log events in the own device</i>	66
5.1.3	<i>Log events in the server</i>	69
5.1.4	<i>Measure the time spent logging events</i>	74
6	EVALUATION	75
6.1	INTRODUCTION.....	75
6.2	EVALUATION MADE BY A DEVELOPER.....	75
6.2.1	<i>Method</i>	75
6.2.2	<i>Data Analysis</i>	77
6.2.3	<i>Video recorded by the developer</i>	80
6.2.4	<i>Feedback from the developer</i>	84
6.3	PERFORMANCE EVALUATION.....	86
6.3.1	<i>Method</i>	86
6.3.2	<i>Data Analysis and presentation of results</i>	86
7	CONCLUSION	89
8	APPENDIXES	91
8.1	APPENDIX A: EVALUATION WITH SLOCCOUNT BASED IN COCOMO MODEL.....	91
	REFERENCES	93

INDEX OF FIGURES

FIGURE 1. GRAPHICAL OVERVIEW OF THE PROJECT WHERE THE DATA COLLECTOR TOOL IS INVOLVED.....	9
FIGURE 2. LISTENING AND LOOKING FOR INFORMATION IN A MUSEUM	10
FIGURE 3. HUMAN-COMPUTER INTERACTION	16
FIGURE 4. TAKING NOTES WRITING ON A PAPER.....	17
FIGURE 5. TAKING NOTES ON THE DEVICE	17
FIGURE 6. GOALS IN EVALUATION	19
FIGURE 7. AFFINITY DIAGRAMMING	22
FIGURE 8. NUMBER OF EVALUATORS VS PROPORTION OF USABILITY PROBLEMS FOUND.....	24
FIGURE 9. NUMBER OF EVALUATORS VS RATIO OF BENEFITS TO COSTS	25
FIGURE 10. A NOKIA N92 WITH SYMBIAN OS	28
FIGURE 11. THE T-MOBILE DASH WITH WINDOWS MOBILE	28
FIGURE 12. THE IPHONE BY APPLE.....	29
FIGURE 13. A BLACKBERRY 8700C.....	30
FIGURE 14. THE LINUX SMARTPHONE.....	30
FIGURE 15. THE TREO 680 SMARTPHONE WITH GARNET OS	31
FIGURE 16. SMARTPHONE SHARES AROUND THE WORLD IN 2007.....	33
FIGURE 17. THE ERICSSON R380.....	34
FIGURE 18. THE NOKIA 9210 COMMUNICATOR.....	34
FIGURE 19. NOKIA N70	36
FIGURE 20. DEVELOPMENT ENVIRONMENT OF CARBIDE.C++	38
FIGURE 21. S60 SOFTWARE.....	39
FIGURE 22. S60 RELEASES.....	40
FIGURE 23. VIEW OF THE EMULATOR	41
FIGURE 24. REPRESENTATION OF THE WHOLE SYSTEM APPLICATION.....	42
FIGURE 25. CLASS DIAGRAM OF THE TOOL.....	46
FIGURE 26. XML FILE CREATED CORRECTLY	68
FIGURE 27. XML FILE WRONG CREATED	69
FIGURE 28. VIEW OF THE REQUESTS RECEIVED BY THE SERVER.....	73
FIGURE 29. VIEW OF THE DATABASE OF PROPERTIES IN THE SERVER	74
FIGURE 30. SCREENSHOT WITH THE APPLICATION IMPORTED TO THE PROGRAM	80
FIGURE 31. SCREENSHOT OF THE LIBRARY REFERRED IN THE HEADER FILE OF THE APPLICATION.....	81
FIGURE 32. SCREENSHOT OF ADDING THE LIBRARY TO THE MMP FILE	81
FIGURE 33. SCREENSHOT OF THE APPLICATION ONCE THE FUNCTIONS ARE INCLUDED.....	82
FIGURE 34. SCREENSHOT OF DEFINING FUNCTIONS IN HEADER FILE	82
FIGURE 35. SCREENSHOT OF ADDING PROPERTIES OF THE EVENT TO THE ARRAY.....	83
FIGURE 36. BUILDING THE APPLICATION.....	83
FIGURE 37. RUNNING THE APPLICATION	83
FIGURE 38. MESSAGE OF EVENT LOGGED SUCCESFULLY.....	84
FIGURE 39. TIME SPENT LOGGING EVENTS LOCALLY	87
FIGURE 40. TIME SPENT LOGGING EVENTS IN THE SERVER.....	88

INDEX OF TABLES

TABLE 1. ROAD MAP FOR SMARTPHONES OPERATING SYSTEMS IN MAY 2008.....	32
TABLE 2. CONTENT OF THE DLL LIBRARY1	47
TABLE 3. CLASSES DEVELOPED INCLUDED IN THE DLL LIBRARY1	48
TABLE 4. CLASSES FROM THE PLATFORM USED IN THE DLL LIBRARY1.....	48
TABLE 5. CONSTRUCTORS OF THE CLASS CLIBRARY1	49
TABLE 6. METHODS OF THE CLASS CLIBRARY1.....	51
TABLE 7. GLOBAL VARIABLES IN CLASS CLIBRARY1	53
TABLE 8. METHODS OF THE CLASS PROPERTY.....	54
TABLE 9. GLOBAL VARIABLES OF THE CLASS PROPERTY	54
TABLE 10. CLASSES IN THE APPLICATION.....	56
TABLE 11. CONSTRUCTORS AND DESTRUCTOR OF THE CLASS CAPPLICATION1ENGINE.....	57
TABLE 12. METHODS OF THE CLASS CAPPLICATION1ENGINE.....	58
TABLE 13. GLOBAL VARIABLES OF THE CLASS CAPPLICATION1ENGINE.....	59
TABLE 14. CONSTRUCTOR OF THE APPLICATION CLASS.....	59
TABLE 15. METHODS OF THE APPLICATION CLASS.....	60
TABLE 16. CONSTRUCTORS AND DESTRUCTOR OF THE DOCUMENT CLASS.....	61
TABLE 17. METHODS IN THE DOCUMENT CLASS	61
TABLE 18. CONSTRUCTOR AND DESTRUCTOR OF THE APPLICATION UI CLASS	62
TABLE 19. METHODS OF THE APPLICATION UI CLASS	62
TABLE 20. VARIABLES OF THE APPLICATION UI CLASS	63
TABLE 21. CONSTRUCTOR AND DESTRUCTOR OF THE CONTAINER CLASS	63
TABLE 22. METHODS OF THE CONTAINER CLASS.....	64

1 INTRODUCTION

1.1 GENERAL IDEA OF THE APPLICATION

Firstly, it is necessary to create to the reader an overview of the project where this application is involved, because it is only a small part of a larger project developed by the Human-Computer Interaction Group of the University of Patras. The goal of this group is to analyse the behaviour of people interacting with mobile devices, in order to test the usability of applications.

Nowadays, mobiles devices are widespread between people around the world, every common person has become in a potential user, and mobile phones are now a basic part of the usual life. In this situation, it is really important to design and develop new applications easily understanding by everyone, even if they are not used to the management of this devices; this is the reason why new programs are focused to get the acceptance of users and become in a well-known tool, and to raise this goal developers are focused on testing the usability of their applications in the first steps of the design.

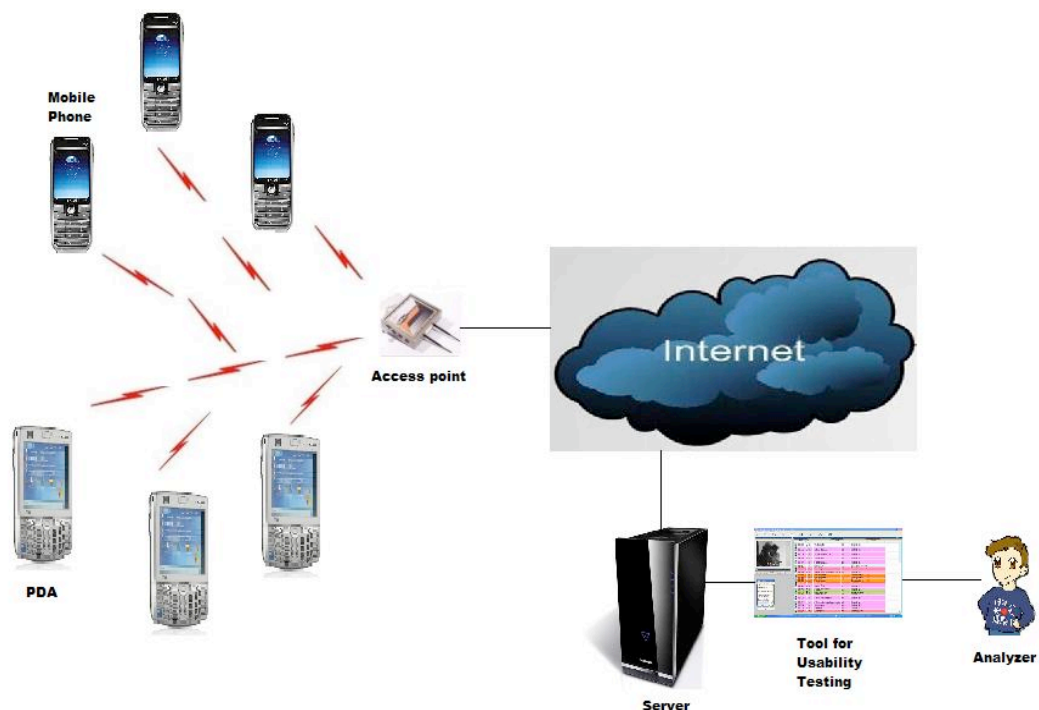


Figure 1. Graphical overview of the project where the Data Collector tool is involved

The project where the data logger is involved, it is composed by three parts pretty different between them, but all together accomplish a common goal (Figure 1). In one side, there are the mobile devices that gather data from the field and send this information to the server, these devices are mobile phones and PDAs, which work in different operating systems, and therefore, has different tools with a similar schema of operation but programmed in different languages. The information collected by this devices is sent to a server, that represents the second part of this system; its main goal is to manage the data received and store it in a database. After that, the third part of the project, the analyser, can start doing its job that consists on collect the data from the database and evaluate its information thanks to a usability tool.

The scenario of this project could be a museum, another collaborative institution, one laboratory of usability testing, one laboratory of a developer, etc. Taking the example of the museum, all the monuments have an identifier and visitors can play with mobile phones or PDAs, in order to know more information about the history of the pieces, even children can play some games during the tour, and at the end, the analyzer can check what they have been doing. It means, in what moment and with who they were playing, in which sculpture people spend more time, if the games or the summarize explaining the sculpture were attractive for them, and with all this information the applications can be improve increasing the attention and the quality of the museum. [37]



Figure 2. Listening and looking for information in a museum

1.2 GOAL OF THE PROJECT

The goal of this part of the project is to design and implement an application for Symbian mobile phones to collect data and send it to a server for storage or management, for a future use of an analyser. This tool consists in a library programmed on Symbian language developed with Carbide.c++ and it will support Symbian OS for the mobile phone Nokia N70.

The tool created is really useful to developers, usability testers, etcetera, in order to do Data Gathering. This data collected is applicable to many uses as, for example, with the data collected it is possible to do Usability Evaluation of the behavior of interactive groups of people. Therefore, it is very important in the field of Human-Computer Interaction (HCI) to know how people react to a particular situation and how the application tested can act in each moment or how it can fix the problems with the changing behavior of people; and also for testing one application for developers, in order to understand better how their tool works and how they can solve its fails.

1.3 CONTENT OF THE THESIS

1.3.1 Background

Firstly, it will be introduced the concept of Human-Computer Interaction, the field of work that the department when we are working is focused in. That is the reason why it is very important for the development of this report to know a basic definition about it.

Secondly, one of the most important topics to start talking about this project is to understand the meaning of one of its principal implementation: Data Collection. To know why it is important, why it is used for and why it has been more widespread lately, will be the goal to be focused in during this chapter.

Besides, the topic of Usability Evaluation will be introduced and explained, because it is one of the main uses that analyzers, users and developers could give to the data collected, in order to test applications and evaluate them.

1.3.2 Technologies

In this chapter, there is explained what are the technologies used in the development of this project, such as which are the devices that we are working with, why they were designed, and what kind of tasks can be done with them. Also, the software and operating systems that smartphones use, including a brief description and some characteristics of each one of them.

Then, we will focus in Symbian OS, the operating system that our device in the project uses, with an explanation about its most important properties. Besides, there is a brief comment about the kind of mobiles phones that use this operating system and then, a description about the phone used in the experimentation, the Nokia N-70.

And finally, the program Carbide.c++ where the tool has been programmed is introduced and explained its basic characteristics of use.

1.3.3 Design

In this section, it is explained the design of the application programmed to gather data using a mobile phone, from the library to a example application that it is going to use it, understanding the different functions and their task, to know exactly what it is done in each moment, and all the possibilities of use.

Moreover, a class diagram is shown to present how the parts of the project are related between them, and how is the communication between these parts.

1.3.4 Implementation

This is an explanation about how the tasks that the tool carries out are done, and how it is the interaction with the server, besides the establishment of a connection and the way to log events in the own device.

Also, there are presented some examples of the result of the communication between the server and the device, as how the data arrives to the server, and another practical experiments.

1.3.5 Evaluation

In this chapter the usability of the designed tool is evaluated with different techniques and it can give us a global idea about how worthy is it.

One of these methods is the evaluation done from a developer that with a group of instructions add this data collector tool to an application already working and give back feedback about its impression of the usability of the system.

The other technique is interpreting the capability and speed of the tool carrying out some test with different parameters, and then, obtaining some conclusions from this results.

1.3.6 Conclusion

To sum up, some comments about the environment where the tool has been created, its importance; besides some impressions about the work done during the development of this thesis to understand it better and to show what has been achieved at the end of this process, in order to give to the reader an overview of the conclusion of the project.

Moreover, some ideas about the future work to do with this application and in its scenario, are proposed.

2 BACKGROUND

2.1 INTRODUCTION TO HUMAN-COMPUTER INTERACTION

The concept of Human-Computer Interaction is of vital importance for the development of this project, because it is involved in a larger project focused on this topic. So, firstly, it is necessary to have an idea about the meaning of this subject.

2.1.1 What is Human-Computer Interaction?

Human-computer interaction (HCI) is the study of interaction between people (the users) and computers to make easier to work with these machines in the software side (interfaces) and in the hardware.

The following definition is given by the Association for Computing Machinery [2]:

"Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."

From a computer science point of view, HCI is focused on the interaction between one or more humans and one or more computational machines. The first idea someone can imagine is a person working on his personal computer, but the "Human-Computer Interaction" field includes many situations. Instead of computers, it could be also parts of computational machines, such as parts of a spacecraft or a microwave. Another situation to consider, which is described in this report, is a group of humans or an organization working all together in the same project and using each one a mobile device.

But Human-Computer Interaction is not only focused on the computer science field, HCI in the large is an interdisciplinary area. It includes several disciplines, each one with different goals: computer science (application design and engineering of human interfaces), psychology (the application of theories of cognitive processes and the empirical analysis of user behavior), sociology and anthropology (interactions between technology, work, and organization), and industrial design (interactive products).

As HCI studies a human and a machine in communication, it must support knowledge about these both fields, the machine and the human side. On the machine side, techniques in computer graphics, operating systems, programming languages and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology and human performance are relevant. And, of course, engineering and design methods are also relevant.

2.1.2 Content of Human- Computer Interaction

The main topic of Human-Computer Interaction can be divided in five interrelated aspects, according to ACM SIGCHI (*Association for Computing Machinery Special Interest Group on Computer-Human Interaction*) [2]:

- *The nature of human-computer interaction*, that gives an overview about the different points of view, objectives, history and intellectual root of HCI.
- *The use and context of computers*, that explains the different applications that users can give to the machines in the computer world, giving us an idea of the variety of necessary interfaces; the general social, work and business context; and the problem to fit and adapt to each other computers, human users and context, all together considering them as a whole.
- *Human characteristics*, that includes a basic knowledge about human information-processing characteristics, how human action is structured, the nature of human communication including the different aspects of language, and human physical and physiological requirements, studying their relationship with the workspace and the environment.
- *Computer system and interface architecture*, that explains the specialized components that machines have to interact with humans, like input and output devices and their characteristics, dialogue techniques for interacting with humans, basic concepts for computer graphics, and software architecture and standards for user interfaces.
- *The development process*, that includes the construction of human interfaces as designing and engineering process, taking in consideration the rest of the system; also making tools for implementation and methods of evaluation, as case studies and classic designs to use it as extended examples of human interface design.

In the Figure 3, some of the interrelations between these elements are represented:

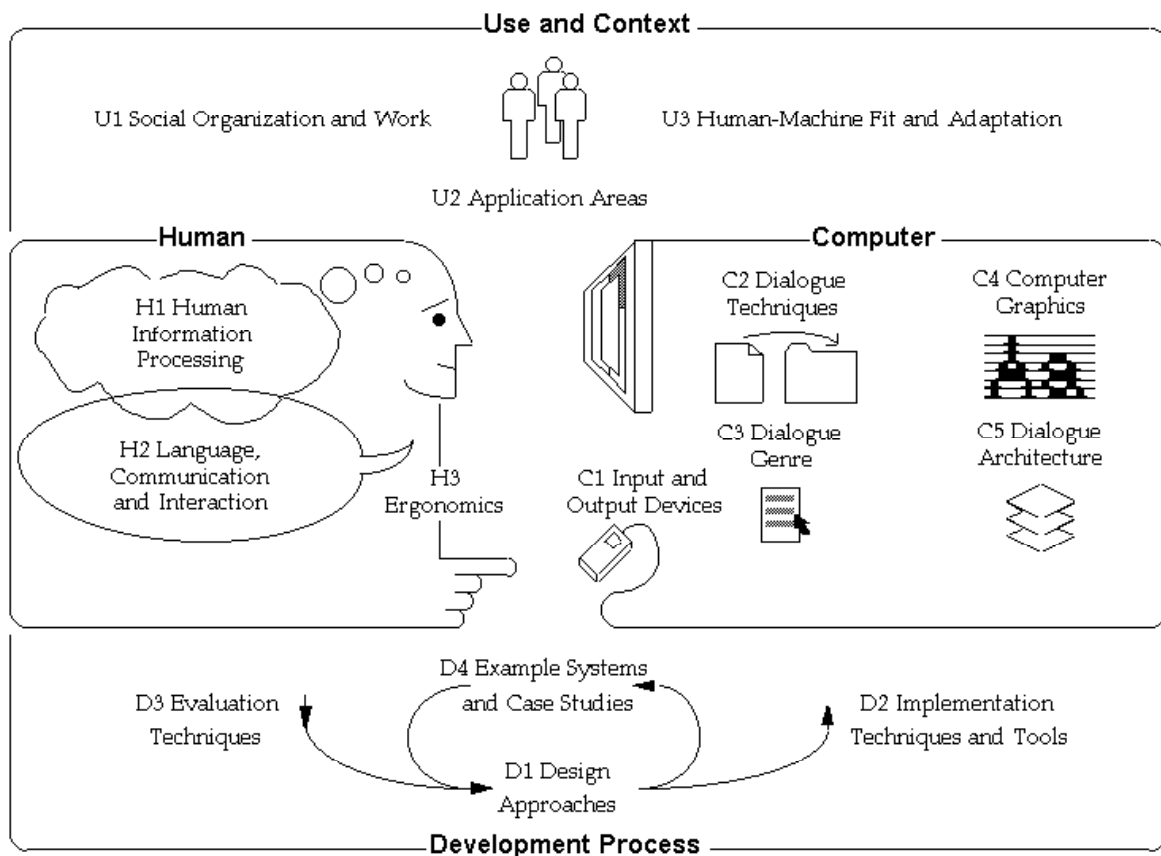


Figure 3. Human-Computer Interaction

2.2 DATA COLLECTION

2.2.1 What is Data Collection?

Data collection is the process of recording data using a handheld or mobile computer. Usually, this process takes place while the user is, for example, away from an office, where the access to a main system or database is limited. In that moment, instead of taking notes writing on a paper (Figure 4), the user can enter the information that he wants to collect, directly into the device (Figure 5). This information is saved on the mobile device for later use, and it can be transferred to a computer, shared with others or added to a database.



Figure 4. Taking notes writing on a paper



Figure 5. Taking notes on the device

If you imagine this concept of gathering data some years ago, it could seem almost impossible, even when the mobile devices, as PDAs and handhelds, have been in the market for several years. But recently their use has been very widespread because the price of these devices is not very expensive nowadays. So, this technology is not only for big businesses and companies, currently is affordable for everyone, including students, researchers and small business owners.

2.2.2 Pros and Cons of Data Collection

To have a general idea about how data collection with a handheld device is useful in our life, next there is an explained list of advantages of data gathering:

- With data collection **time is saved** because you can enter data in mobile devices quickly with a few clicks, and at the same time you are saving the time you would spend if, after gathering data handwritten in the field, you would need to copy this data to a computer. Using a mobile device you just need to type it once directly from the scenario when you are working, and later you can send it easily to a personal computer without writing it again.

- Other important advantage is to **reduce clutter** in your daily life, because all data is collected and stored in a small portable device and it not necessary to carry around, keep together and organize tons of papers, pens and clipboard with the information.

- One of the most important sources in an experiment can be saved with the correct collection of the data in a mobile device: **money**. Mistakes and wasted time due to illegible handwriting and lost papers are reduced, so it is possible to collect data faster and the tester can be focused on analysis and can make quicker decisions, using less time and, for that reason, saving money.

Therefore, not all are advantages using handhelds, there are things that are not improved with this technique compared to traditional paper-based methods, and so it is necessary to explain also the disadvantages of it:

- Even with the correct use of the mobile device during the research, money is saved; it is needed to count with the expense of the **initial cost** of devices. However, prices for smartphones are been reduced each day, so there are not very expensive as when they appeared in the market, and nowadays are becoming more widespread.

- Also, it is needed to take in consideration possible **problems during use** of the device. It can make you loose your time and data already collected, especially when you are working on the field. On the other hand, data security measures such as external memory cards are really useful in these situations and their price is not very expensive nowadays, besides, devices are continuously tested and their developers try to fix every possible fail.

- And, as any new technology, it is necessary to include the **initial learning** of users that could be a handicap during the implementation, because maybe they are not familiarized with the usage of a handheld device or uncomfortable with the visibility of displays or bottoms. Successfully, these devices included carefully explained information and user's manual for the inexperience users, as well as it could be provided an appropriate training for them.

2.3 USABILITY EVALUATION OF MOBILE APPLICATIONS

In mobile applications, usability evaluation is a really important tool nowadays, in order to discover the problems that users can find while they are working in different fields. To find out these difficulties is really important when an application is developed, because it could be improved after the evaluation thanks to the results obtained, and more important, thanks to this improvement it could better considered in the market between costumers and it could get more acceptance between final users.

2.3.1 What is Usability Evaluation?

Usability evaluation is an analysis or study of the usability of a prototype, application or system. Its main goal is to provide feedback in development in order to follow an iterative development process to improve the created product. To carry out this, it is needed to follow some basic steps as recognize problems, understand the causes of these problems and plan changes to improve it.

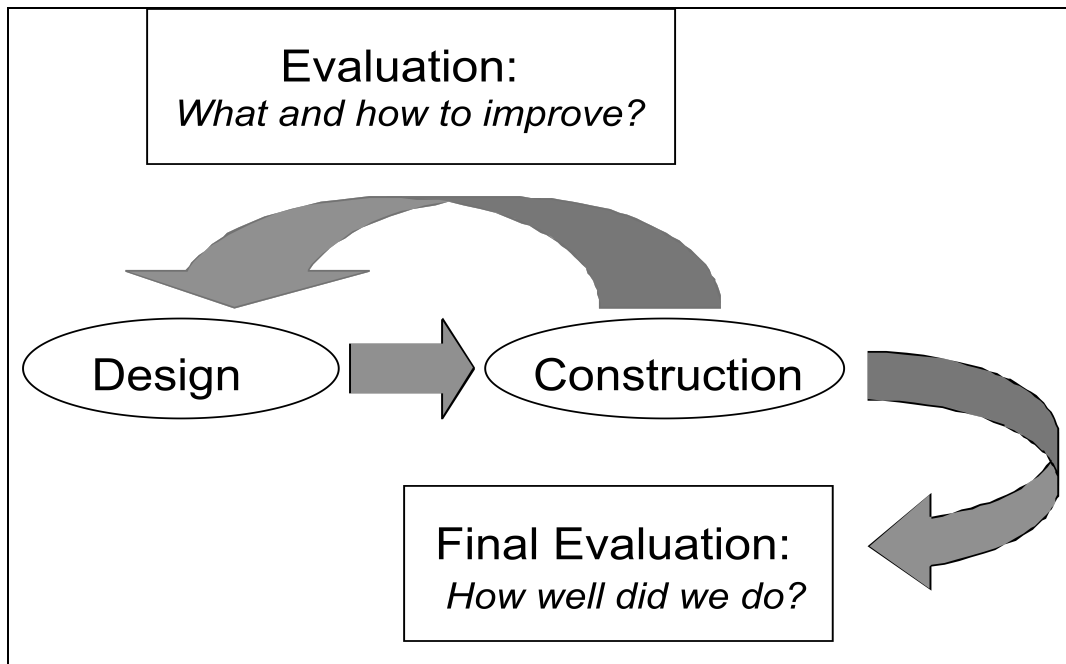


Figure 6. Goals in evaluation

2.3.2 Why is useful to do usability evaluation?

As it was said before, the usability evaluation is a tool to provide feedback in development and to improve a product created. In the field of Human-Computer Interaction, this feedback will be focused on the behaviour of groups of people that interact with this product developed; looking for its satisfaction, effectiveness and efficiency.

The usability Evaluation has several benefits, below are explained some of the most important ones in our field of work [14]:

1. Making mobile computer systems aware of their user's contextual setting, designers can use this information to present only information and functionality relevant in specific situations. This way, the user interface can be simplified and the demand for user interaction can be reduced. Consequently, it gets some savings of time for the user, getting better the feeling that this application cause in this user.
2. Tailoring the interface to its context may also facilitate partial automation of repetitive and trivial tasks.
3. Also, making the system react to contextual changes may be used to increase security of data and users.

In spite of all of these good reasons to begin to use the usability evaluation, the context-awareness for mobile information systems has not been yet fully carried out in practice.

It is clearly seen that, applying usability evaluation is really useful. However, applying the methods do usability, it has several troubles or problems due to their difficulty to analyze and communicate, these problems will be explained in the next section.

2.3.3 Why Usability Data is Difficult to Analyze and Communicate?

One of the problems to use and to analyse usability data is the nature of the data: usability data is often observation-based rather than measurement-based. Often these behaviour observations are not very meaningful at the beginning and have to be interpreted by the usability evaluator. The usability evaluator often has to extrapolate the usability observations based on “*hunch*” or “*gut feelings*”. This makes the design recommendations made by the usability evaluator questionable as "subjective" or "biased".

Another problem referring with this usability is to decide between field or laboratory study. When testing a user interface of a mobile device application, field testing may not necessarily be the best place; mostly because it is more time consuming than the lab test, and usually more expensive, because you have to move all this equipment and people who has to work with it. Besides, when testing in the field, be prepared that things will not go as planned: there may be interruptions and unexpected events more than in lab.

Other difficult part of doing this is what to communicate. One way around is to provide the chain of reasoning that the usability evaluator used to derive the design recommendations from the raw observational data. This allows the recipient of the data to understand how the interpretations and design recommendations were arrived at. So the challenge is to provide meaningful information with enough rationale that lends the information sufficient credibility.

Seeing more problems of using usability, appear one related with, who receives the data. Often the users for the usability testing data are not homogeneous: it could be a mix of design engineers, technical writers, and different levels of management and so on. Some portions of this audience may want every last detail of the data while others may want only a high level synopsis. Fitting the content of the communication to the appropriate audience without having to write several flavours of the report is a challenge.

Another trouble is concerning about how to communicate. Even if the time limitations associated with collecting data was not an issue, usability practitioners need to understand that the reporting of data needs to be in formats and use several things in order to get understandable the information by the development team members. For example, software engineers work and address problems in terms of modules. However, usability testing can reveal “systemic” or general issues that cross module boundaries creating communication challenges with the software engineers.

The right time to spend your usability budget is one of the worst problems related with usability evaluation, because it involves the budget, the money to spend in the project.

Other typical trouble is timeliness. Late usability results and change recommendations are often received as “bad news”. However, the concern that bugs may be introduced by making changes late in the product lifecycle must be considered. You have to

be carefully with when you do the usability recommendations, because it is better to do in at the beginning in order to can change all the defects expending, the less, as time as money, possible.

Besides, to all of these problems, who collects the data is a really difficult trouble too. As usability data still tends to be collected by specialists without too much participation from development team members, there is still certain hostility about the methods used and the conclusions and recommendations obtained from testing. To overcome this, usability practitioners have to spend their time resources to “make a case” for the usability data. This activity takes additional time in already late breaking results making it a vicious circle.

Finally, another difficult and one of the most important problems, it is concerning about when designing a usability evaluation: Key decisions must be made regarding methods and techniques for data collection and analysis. There are some techniques and tools that are more suited for basic research and have to be adapted significantly to fit into a software development environment, these techniques are described in the section “Techniques for Analyzing and Communicating Usability Data” [15].

2.3.4 Techniques for Analyzing and Communicating Usability Data

The following section describes several techniques for Analyzing Usability Data, as in the field as in a laboratory. In general, the techniques involved affinity diagramming, data visualization, prioritizing problems, and scenarios of use. Each group of techniques is described below.

- Affinity Diagramming -

In general, the technique involves writing observations, comments, issues, and concerns on index cards or adhesive sheets (). These cards or sheets are then laid out and sorted to derive a pattern or structure observations. The technique was often used to help sort and interpret the hundreds of comments generated during contextual inquiry that relate to the user's work, environment, hardware, software, and organizational issues. The comments can be structured by creating an affinity diagram or grouping along common themes. By performing this grouping, patterns or trends can emerge from the derived structure.



Figure 7. Affinity Diagramming

There are several versions of this basic technique:

1. Affinity diagramming is used in analyzing observations from a usability test. For example, in one application of the technique, team members entered observations on 3x5 cards which are collected by the usability practitioner and later the entire design team gathered to categorize the cards into groups.
2. Another approach to affinity diagramming is to group pictures collected from observations rather than textual descriptions (“picture-based affinity”).

In general, affinity diagramming is a simple and cost effective technique for asking ideas from a group and obtaining results on how information should be structured.

- Data Visualization -

Several techniques emphasized the need to visualize user behaviour and performance to assist analysis and presentation of findings. For example, “*GUI morphing*” is used to highlight the impact of redesign of windows. Snapshots of an interface are taken before, during, and after redesign. An a morphing application is used to display the transformation from the old to the new interface, essentially, the windows change between a misaligned, anaesthetic collection of controls to a thing of beauty. This presentation of data is considered useful for showing management what user interface design efforts could do for a product.

Video snippets or screenshots are a popular way to make the data more “real”. Typically these snippets and screenshots would show the difficulties users have had with a product.

Another visualization technique, it is involved combining several approaches into a multimedia presentation or report. For example, clicking on an image of an interface design could pop up a video segment showing the difficulties users had with the selected control or send some reports to the analyzer, in order to solve these problems or to get better the application.

This technique is used in the main project where the data collector tool is involved. The mobile devices take information in order to send it to the server and contribute to the usability evaluation.

- Ordering Problems by Priority -

Evaluation, design and testing for usability evaluation, usually leads to the discovery of many problems or concerns. Short product lifecycles and limited resources require that problems be prioritized because it is unlikely that every problem will be fixed. Reporting and rating the severity of “*usability bugs*” appeared to be a popular means of communicating usability problems found in testing. Usability bugs are typically interpreted in the same language as other types of bugs found in the system and can take advantage of the same defect tracking mechanisms in place in most product development environments. Use of the same defect tracking systems has the additional benefit of putting usability defects on equality with others with the same defined priority. Another advantage of putting usability bugs into the corporate bug list is that you can track how many were actually fixed. However, the estimation of severity should be based on well-established guidelines and frameworks, not in “gut feelings”. Defects could be logged by other members of the design team such as documentation writers, product developers, quality engineers or other team members who have been trained in estimating and categorizing problems using a common framework.

- Creating Use-Scenarios -

Use-scenarios typically involve describing the people who will use the designed product, the tasks and the contexts of the tasks. The scenarios are usually concrete. The creation of use scenarios appeared to be motivated by several factors. One was an interest in aggregating individual data to describe trends or to describe potential market segments.

Observational data from contextual inquiry could be used to create “*character maps*”. This character maps are relevant attributes put together to build composite characters, that illustrate a stereotypical user and stories of usage scenarios built for each user. These characters and scenarios could be used rather like a coat hanger, to hang new design ideas on, and see if they fit.

- Think-aloud protocol -

Think-aloud protocol is a method to gather data in usability testing that involves participants thinking aloud as they are performing a set of specified tasks. Users are asked to say whatever they are looking at, thinking, doing, and feeling, as they go about their task. This enables observers to see first-hand the process of task completion (rather than only its final product). Observers at such a test are asked to objectively take notes of everything that users say, without attempting to interpret their actions and words. Test sessions are often audio and video taped so that developers can go back and refer to what participants did, and how they reacted, and study about that behaviour. The purpose of this method is to make

explicit what is implicitly present in subjects who are able to perform a specific task. But this is not easily seen for developers, due to is important to say that such people are taking part of a test, and their reactions are not always the same that in the real world.

- Heuristic Evaluation -

Heuristic Evaluation is another usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design. It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles (the “heuristics”) [18].

In general, heuristic evaluation is difficult to do for only a person because he will never be able to find all the usability problems in an interface. Luckily, experience from many different projects has shown that different people find different usability problems. Therefore, it is possible to improve the effectiveness of the method significantly by involving multiple evaluators.

But how can the number of evaluators be decided? Figure 8 shows the proportion of usability problems found as more evaluators are added and it also shows the ratio of benefits to cost per number of evaluators. The figure clearly shows that there is a nice payoff from using more than one evaluator. It would seem reasonable to recommend the use of about five evaluators, but certainly at least three. The exact number of evaluators to use would depend on a cost-benefit analysis. More evaluators should obviously be used in cases where usability is critical or when large payoffs can be expected due to extensive or mission-critical use of a system.

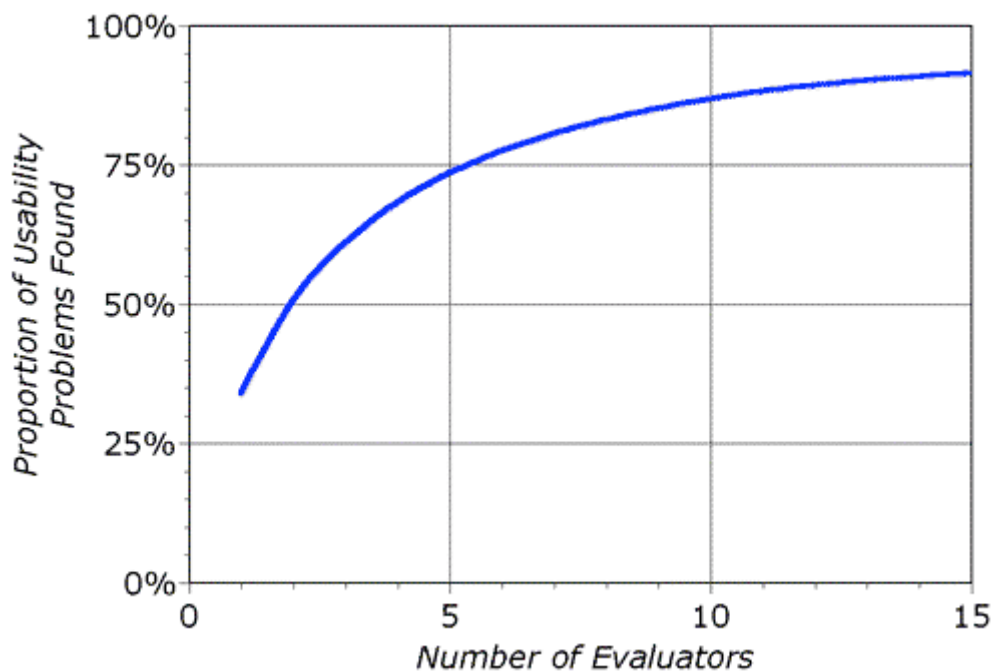


Figure 8. Number of evaluators vs proportion of usability problems found

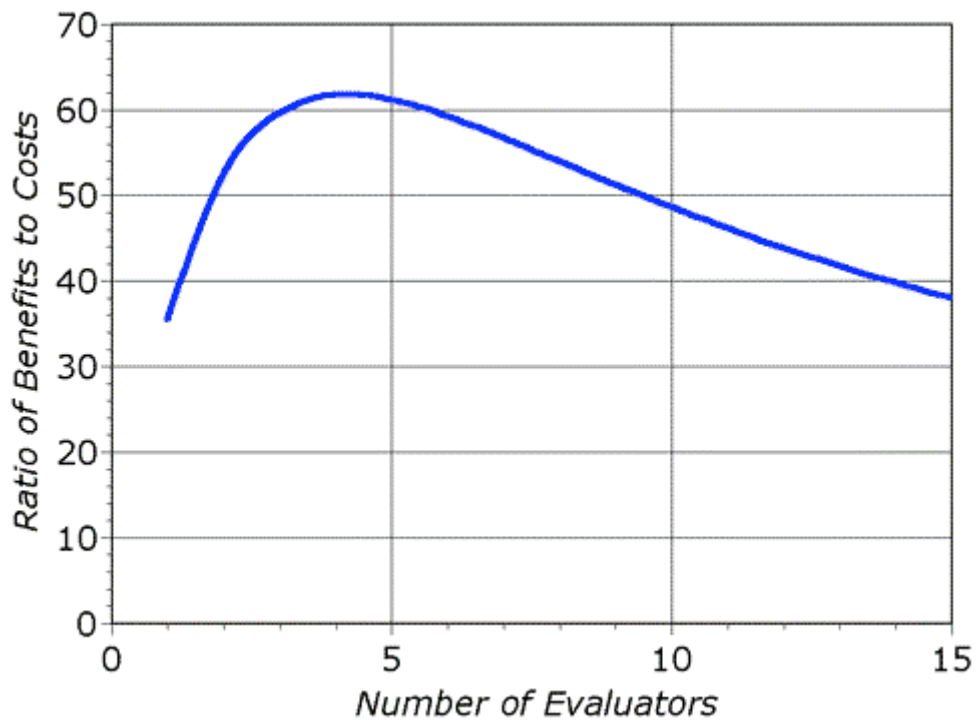


Figure 9. Number of evaluators vs Ratio of benefits to costs

- Instant Data Analysis (IDA) -

A technique for increasing the speed of data analysis and improving its focus is IDA. Used in combination with the think-aloud protocol for user-based evaluations, this technique makes it possible to conduct a complete usability evaluation in a day; that is, a technique for reducing the efforts spent on analyzing data from usability evaluations. The aim of applying this technique is to make it possible to conduct an entire usability evaluation in one single day. IDA technique is not the best technique for analyzing usability data, but is effective in supporting fast identification of the most critical usability problems of a software system [15].

The IDA technique does not aim at producing an elaborated theoretical understanding of the use of the system being evaluated. Rather it aims directly at producing a ranked list of usability problems for the design team to take into consideration when redesigning the system.

- Expert analysis -

Expert analysers are used to get the collective observations and opinions of the “best of the breed”. They are particularly useful when there is not one correct solution or procedure.

3 TECHNOLOGIES

3.1 SMARTPHONES

3.1.1 What is a smartphone?

A smartphone is a mobile phone that offers advanced capabilities beyond a typical mobile phone, often with functionality similar to a PC. Nowadays, there is not a standard definition of a smartphone because there is no agreement in the industry. For some, a smartphone is a phone that runs complete operating system software providing a standardized interface and platform for application developers. For others, a smartphone is simply a phone with advanced features.

How it can be possible? These days, you can combine in a small device that fits in your pocket, a mobile phone with the technology of your computer. That would not be possible without the advancements realized during the last years in integrated circuits, microprocessors, semiconductor miniaturization, battery technology and, of course, the invention of telephone and radio. Computers have progressed from the old centralized mainframes to the current personal computers, especially since the creation and extension of Internet, World Wide Web and email around the population that made computers an important part of everyday life as productivity, entertainment and communication device. That is reason why laptops were introduced into the market to allow personal computers to be portable and then, the size of the PDA made that even easier. In that moment, the creation of smartphones, allow the combination of mobile phone and PDA in single and small device saving space in your bag.

What can I do with my smartphone? In order to make an idea of smartphone's capabilities, here it is a list of tasks that can be possible to carry out with it:

- Send and receive mobile phone calls.
- Manage personal information as notes, calendar and to-do list.
- Communication with laptop or desktop computers.
- Establish a WiFi connection.
- E-mail.
- Instant messaging.
- Applications such as word processing programs or video games.
- Play audio and video.
- Data synchronization with applications like Microsoft Outlook and Apple's iCal calendar programs.

And the promised future applications for smartphones are even more impressive. For example, the Nokia 6131 NFC is a phone that could allow the phone act as a wireless credit card using a two-way communication system to transfer payment information thanks to near-field communication technology.

3.1.2 Software in smartphones

We can visualize the software for smartphones as a software stack consisted of the following layers:

- Kernel: management systems for processes and drivers for hardware.
- Middleware: software libraries that enable smartphone applications (such as security, web browsing, messaging, etc).
- Application execution environment (AEE): application-programming interfaces, which allow developers to create their own programs.
- User interface framework: the graphics and layouts seen on the screen.
- Application suite: the basic applications users access regularly such as menu screens, calendars and message inboxes.

3.1.3 Operating systems in smartphones

The most important software in any smartphone is its operating system because it is who manages their hardware and software resources. Some OS platforms cover the whole range of the software stack; some others may only include the lower levels (typically the kernel and middleware layers) and rely on additional software platforms to provide a user interface framework. The most common smartphone operating systems are:

- Symbian OS from Symbian Ltd.

Symbian OS is used nowadays for more than 100 different models of phones. The operating system consists of the kernel and middleware layers of the software stack, while the upper layers are supplied by application platforms like S60, UIQ, and MOAP.

This operating system is dominant in most markets worldwide right now, with an estimated market share of sales of 65 percent (according to Market Share Sales Q4 2007), but it is estimated that the operating systems Linux and Microsoft will hold more market share than Symbian in 2010. But, until now, the success of Symbian is due to its shareholder and customer Nokia. Although, it is used by all the main handset manufactures as BenQ, LG, Motorola, Samsung and Sony Ericsson.



Figure 10. A Nokia N92 with Symbian OS

- Windows Mobile from Microsoft

Windows Mobile OS covers all layers from the software stack, from the kernel to the application interface. The operating system is based on Windows CE along with Windows Mobile middleware. The strongest characteristic of this OS is its compatibility with the Microsoft Office suite of programs.

Microsoft had the 12 percent of the market share sales (according to Market Share Sales Q4 2007). And at the beginning of 2007, Microsoft unveiled the latest version of the software platform: Windows Mobile 6 Professional for touch screen devices, and in the first half of 2008 was improved with Windows Mobile 6 Standard.



Figure 11. The T-Mobile Dash with Windows Mobile

- iPhone OS from Apple Inc.

The iPhone OS is the operating system developed by Apple Inc. for the iPhone (Figure 12) and iPod Touch. This operating system is derived from Mac OS X, and is based on the same kernel and Darwin core as Mac OS X; besides, it has three abstraction layers: a Core Services layer, a Media layer and a Cocoa Touch layer. The native application support was recently announced in March 2008 and nowadays requires a beta version of the iPhone OS which is available for developers and corporations for testing. On June 2008, the new release of iPhone OS 2.0 should be available and it will be free, but until last year it had the 7 percent of the market share sales (according to Market Share Sales Q4 2007).



Figure 12. The iPhone by Apple

- RIM BlackBerry operating system

Blackberry OS is the proprietary software platform made by Research In Motion for their Blackberry line of handhelds. This operating system was created for business but, lately, it started to be used for a lot of people, especially in USA, and it has an 11 percent of the market share (according to Market Share Sales Q4 2007).

It is focused on easy operation and the current OS allows complete wireless activation and synchronization with Exchange's e-mail, web browsing, install messaging and personal information management software. Recently it has seen a surge in third-party applications for the Blackberry, like games and productivity applications, and it has been improved to offer full multimedia support.



Figure 13. A BlackBerry 8700C



Figure 14. The Linux smartphone

- **Linux operating system**

Linux has an important characteristic that differs from the other operating systems, and it is that its development is done by a community of developers and several costumers rather than by a central company, that is the reason why it is called an organic OS, because developers are constantly changing and updating the operating system, even in the kernel layer, and platforms based on Linux can be very different from one to another, for example Motorola and TrollTech that are very incompatible.

Linux is right now the operating system that supports more processors than any other one in the market but Symbian is still the leader on sales, with the 5 percent of market share sales (according to Market Share Sales Q4 2007) for Linux. This is because smartphone companies find risky to invest in Linux due to the differences between platforms. To avoid this situation, a standardized platform based in Linux is being developed by the LiMo foundation, an organization created by Motorola, NEC, NTT DoCoMo, Panasonic, Orange, Samsung, Vodafone and the newest member Texas Instrument. The same goal has the Open Handset Alliance, a group of more than 30 technology and mobile companies created mainly by Google, which is developing Android: the first complete, open, and free mobile platform based on Linux that will be available by the second half of 2008.

- Garnet OS

Garnet is an operating system, formerly known as Palm OS. Palm OS was initially developed by U.S. Robotics' owned Palm Computing, Inc. for personal digital assistants (PDAs), but currently is owned by the company ACCESS.

The Garnet operating system combines a Linux platform with applications written for the old Palm OS that was mainly used in PDAs and the Treo line of smartphones. Since the end of 2007, the new mobile phones with Garnet OS are available in the market.



Figure 15. The Treo 680 smartphone with Garnet OS

3.1.3.1 Comparative between operating systems and market sales

As it was explained in the last pages, there are many different operating systems for mobile phones, each one of them has its advantages and disadvantages, and of course, its field of devices where are used in. But, to have a general idea of the main characteristics of these operating systems and a visual comparative of all them together, it is necessary to take a look in the following table (Table 1), where it is represented a summary about the OS explained, including the latest version of its platform and the next expected releases, the weakest and strongest points of each one of them, and the main properties of each operating system.







Current release	 Symbian 9.5	 Linux LiMo Platform Release 1	 Palm OS 5.4 Garnet and Palm OS Cobalt	 Windows Mobile 6.1 "Crossbow"	 BlackBerry 4.5	 iPhone OS
Next release	Unknown	LiMo Platform Release 2, expected in early 2009. Android by Google in Summer 2008.	Garnet OS 5.5	Windows Mobile 7 "Photon" is planned for release in 2nd half of 2009	Unknown	iPhone OS 2.0 is expected for end of June 2008
The trend	Symbian 9.5 includes components of the IP Multimedia Subsystem in anticipation of IMS-based services such as videoconferencing over IP and the ability to automatically switch between cell and Wi-Fi networks. There's also a location API for GPS-enabled applications. In development: Bluetooth kits for plugging into car audio systems.	The LiMo foundation continues developing its own Linux OS to unify Linux development for mobile phones. The same goal has Android, and the Open Handset Alliance created by Google last year.	The Access Linux Platform will include an emulation layer that supports applications originally developed for Palm devices. Separately, Palm Inc. is developing another Linux-based successor to Palm OS Garnet.	Crossbow is strongly linked to Windows Live and Exchange 2007 products. In development: Microsoft is not talking, but industry observers believe Photon will eliminate the need for separate software development kits for Windows Mobile's Smartphone and Pocket PC editions.	RIM jumped into consumer-class smartphones in September 2007 with the BlackBerry Pearl running version 4.2 of the BlackBerry OS. The upgraded version BlackBerry 4.5 has new features like digital camera support, HTML support for mail, upgrade Over The Air and streaming support.	Tiger supports HTML e-mail, Web browsing, and apps found on Mac computers like widgets, Safari, calendar, text messaging, notes, and an address book.
Strength	Widely available, global market leader	Army of developers	Ease of use	Windows integration	Designed for business	Sleek consumer design
Weakness	Small USA presence	Incompatible OS variants	Declining market share	Unstable	Few non-RIM devices supported	In compatible OS variants

Table 1. Road Map for SmartPhones Operating Systems in May 2008

Due to the variety of features in the different operating systems, its situation on the market is also very different. The following graph (Figure 16) explains how was divided the sales of smartphones per operating system, during 2007.

The world market is dominated by Symbian, being the most sold in each region, less in North America, where the market is very different from the rest of the world, with strong companies like Microsoft and RIM BlackBerry and Garnet (Access), followed by Apple that will increase its sales with the world launch of its iPhone, but until 2007 it was sold only in the USA.

In Europe, Middle East and Asia (EMEA) the market is dominated by Symbian, with a small Microsoft pocket and an even smaller RIM and Linux market share. It's also interesting to see the large Linux share in Japan and China (PRC).

The column dedicated to the rest of the world (ROW) represents that in the unspecified countries the leader is also Symbian. Microsoft and Linux has a small percentage of sales and Acces and RIM even smaller.

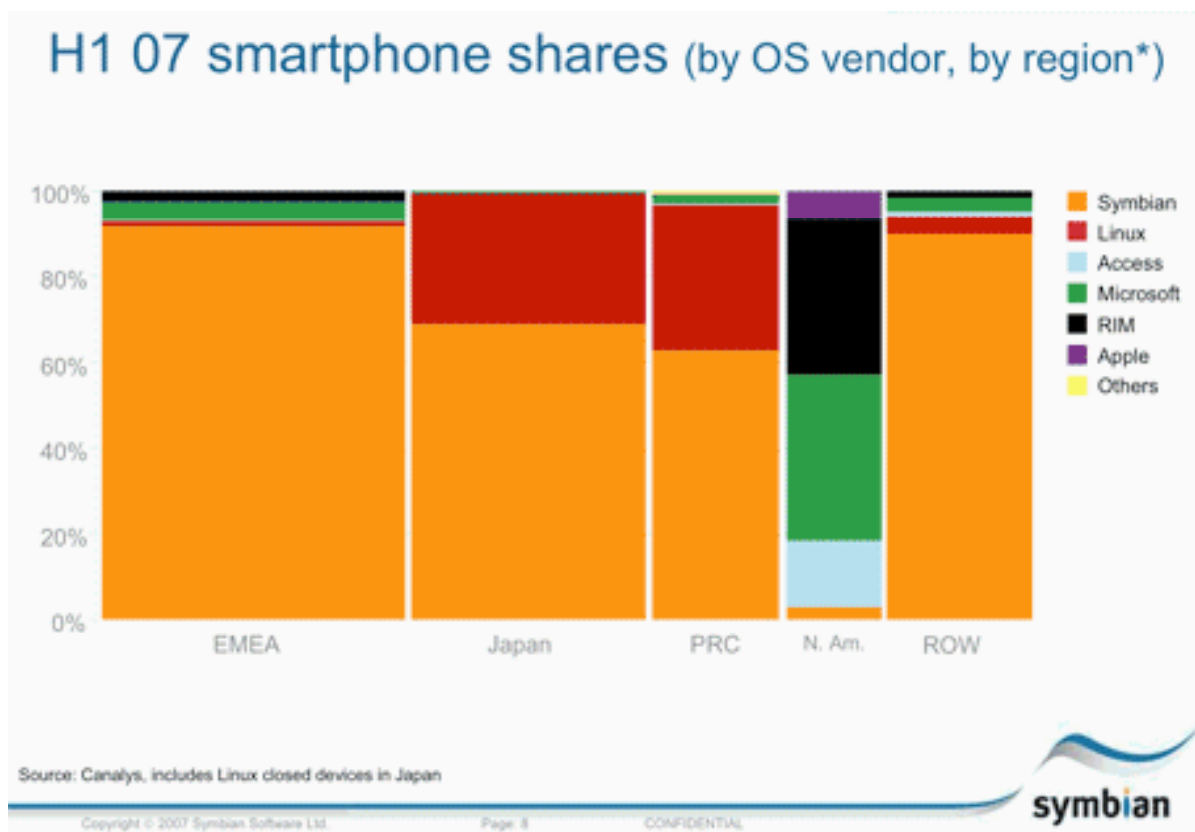


Figure 16. Smartphone shares around the world in 2007

3.2 SYMBIAN OS FOR MOBILE DEVICES

3.2.1 What is Symbian?

Symbian OS is an open standard operating system, designed for data-enabled mobile devices (smartphones) produced by Symbian Ltd. This operating system has associated libraries, user interface frameworks, data services enablers, application engines and reference implementations of common tools that make the work easier to developers. Also, it offers a high level of integration with communication and personal information management. It supports Java, PC synchronization, Bluetooth local wireless access and GPRS packet-switched data. Besides, it is open for third-party development by independent software costumers, enterprise IT departments, network operators and Symbian OS licensees.

Symbian OS is an evolution of EPOC operating system from the company Psion Software, but in 1998 Ericsson, Nokia, Motorola and Psion created a company to support the EPOC OS as an independent entity. Nowadays, Symbian is owned by Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) and Samsung (4.5%).

3.2.2 Symbian OS Phones

Symbian OS phones have evolved so much since the first one shipped, the Ericsson R380 in November 2000 (Figure 17), and the first open Symbian phone available, the Nokia 9210 Communicator in 2001 (Figure 18).



Figure 17. The Ericsson R380



Figure 18. The Nokia 9210 Communicator

In 2007, out of more than a hundred million smartphones sold worldwide, three quarters of them ran Symbian OS. So, we can say that Symbian OS is the world-leading open operating system that powers the most popular and advanced smartphones today from the world's leading handset manufacturers. As example of this, we have the following devices that have used Symbian OS as operating system and this list grows almost every week [26]:

- **Ericsson R380** (Figure 17) was the first commercially available phone based on Symbian OS in 2000. This device was closed and the user could not install new C++ applications. However, the R380 could not even run Java applications, and for this reason, some have questioned whether it can properly be called a 'smartphone'.

- Using the **Nokia Series 80 interface**: Nokia 9210 Communicator smartphone (Figure 18) shipped in 2001, 9300 Communicator (2004) and 9500 Communicator (2004).

- **UIQ interface**: Used for PDAs such as Sony Ericsson P800 (2002), P900 (2003), P910 (2004), P990 (2005), W950 (2006), M600 (2006), P1 (2007), W960 (2007), Motorola A920, A925, A1000, RIZR Z8, RIZR Z10, DoCoMo M1000, BenQ P30, P31 and Nokia 6708 using this interface.

- **Nokia S60 interface** (2002): Is used in various phones, the first being the Nokia 7650, then the Nokia 3650, followed by the Nokia 3620/3660, Nokia 6600, Nokia 7610, Nokia 6670 and Nokia 3230. The Nokia N-Gage and Nokia N-Gage QD gaming/smartphone combos are also S60 platform devices. It was also used on other manufacturers' phones such as the Siemens SX1, Sendo X, Panasonic X700, Panasonic X800, Samsung SGH-D730, SGH-D720 and the Samsung SGH-Z600. Recent, more advanced devices using S60 include the Nokia 6620, Nokia 6630, the Nokia 6680, Nokia 6681 and Nokia 6682, a next generation Nseries, including the Nokia N70, Nokia N72, Nokia N73, Nokia N75, Nokia N80, Nokia N81, Nokia N82, Nokia N90, Nokia N91, Nokia N92, Nokia N93 and Nokia N95, and the Enterprise model Eseries, including the Nokia E50, Nokia E51 Nokia E60, Nokia E61, Nokia E62, Nokia E65, and Nokia E70.

- **Nokia Series 90 interface**: Nokia 7710 (2004).

- Nokia 6120 classic, Nokia 6121 classic.

- Fujitsu, Mitsubishi, Sony Ericsson and Sharp phones for NTT DoCoMo in Japan, using an interface developed specifically for DoCoMo's FOMA "Freedom of Mobile Access" network brand. This UI platform is called MOAP "Mobile Orientated Applications Platform" and is based on the UI from earlier Fujitsu FOMA models.

3.2.3 The device: Nokia N70

The model of Symbian OS smartphone used in this project is the Nokia N70-1 (Figure 19), one of the handsets in the Nokia Nseries line of smartphones. It is a multimedia 3G smartphone made by Nokia and launched in the third quarter of 2005.

The operating system used by the Nokia N70 is the Symbian OS v8.1a, and the S60 user interface, with the developer platform S60 2nd Edition, Feature Pack 3.

The Nokia N70 is a smart and elegant tool for real time visual sharing, also it has MMS, e-mail, Bluetooth, WAP 2.0 browsing and Java technology. The device is equipped with two integrated cameras, one of 2.0 megapixel in the back with flash and another VGA in the front with two-way video call and real time video sharing application; it has also FM radio, Bluetooth, digital music player and support for 3D Symbian, Java games and other S60 2nd Edition software.

At the time of its launch, the Nokia N70 had the most built-in memory alongside its system memory and was the penultimate (before the N72) Symbian OS 8.x device released by Nokia, since the introduction of their new OS9 platform released in 2003 which offers more flexibility than the original that was made in 1998.



Figure 19. Nokia N70

3.3 THE DEVELOPING TOOL: CARBIDE.C++

3.3.1 What is Carbide.C++?

Carbide.c++ is a software development tool for C++ development on Symbian OS. It is used to develop phones that use this operating system, as well as applications that run on those phones. It is based on the open Eclipse IDE (Integrate Development Environment) platform enhanced with additional plug-ins which make Eclipse understand how to handle Symbian C++ source files as well as build Symbian projects and, in that way, it makes possible to support the Symbian OS development on the S60 platform, the Series 80 platform, UIQ and MOAP.

Carbide.c++ belongs to the new generation of mobile development tools family created by Nokia, the Carbide development tools. It was created to replace the program CodeWarrior for Symbian OS as the principal development environment for Symbian OS, but being a new line of products, Carbide.c++ has not displaced yet CodeWarrior within the Symbian OS development community.

3.3.2 Pros and Cons of the development in Carbide.c++

The reason because Carbide.c++ has not displaced CodeWarrior yet, it is due to several complains that it has from developers, as, for example, lack of Symbian OS style coding, difficulty to find elements in files, not good speed of import Symbian OS build files (*mmp* files) and difficulties to do debugging on devices.

In addition, the Managed Build System doesn't work properly yet; its method of work is to delete everything and start again rather than make an incremental built. Otherwise, however, the reception of this tool has been warm between developers because of its advantages, such as, a better development environment than CodeWarrior. The IDE (“Integrated Development Environment”) is based on Java so there are some concerns about speed and memory; the IDE is often slow and has a pretty big memory use.

Also, another good point that Symbian C++ developers can find with this tool, it is that for those who already have various Symbian projects on their computer and have been either using CodeWarrior or the command line to build them, will find that Carbide.c++ provides a facility to import easily their projects and migrate to Carbide.c++ without having to create a new project and then manually insert the source files.

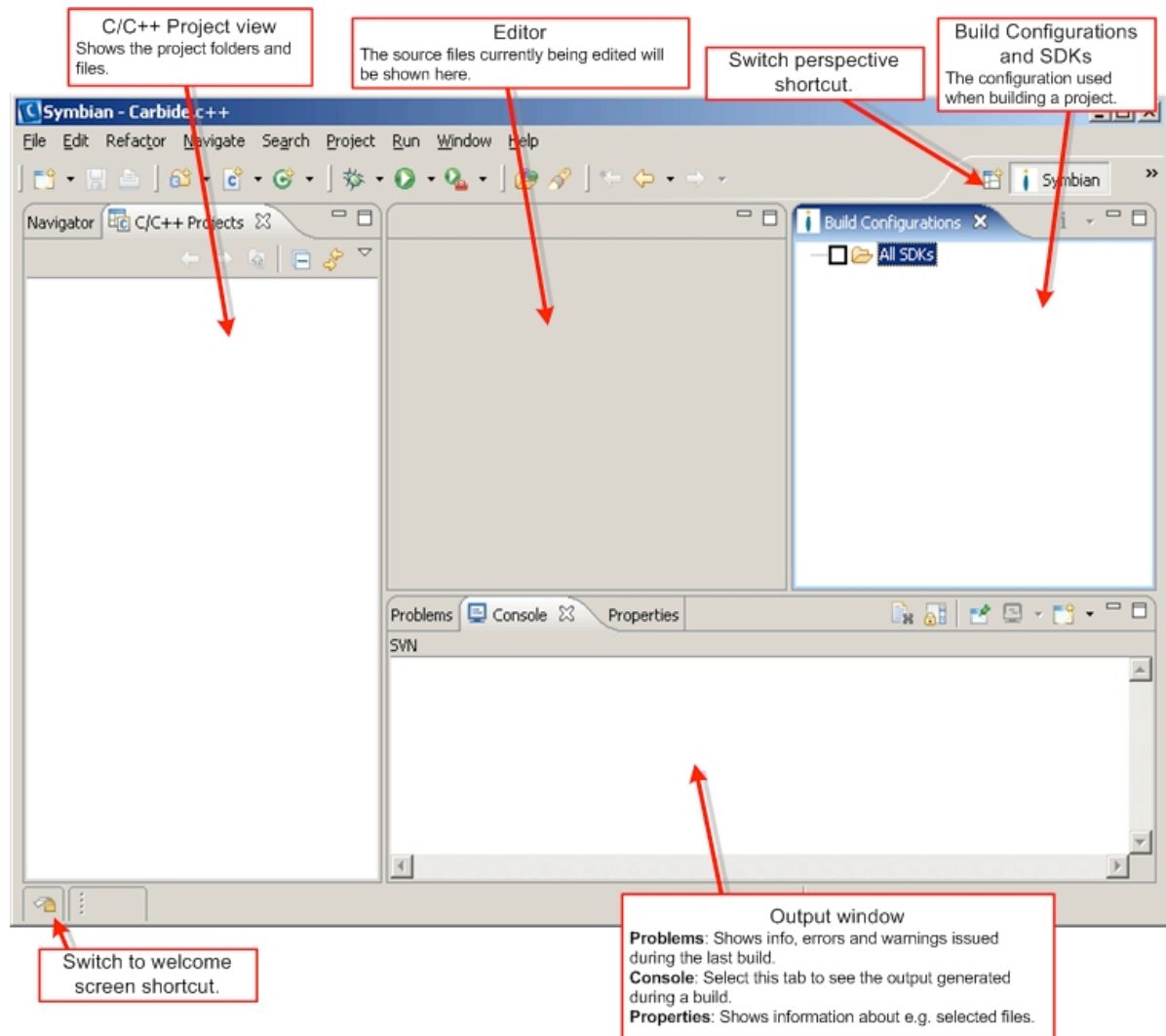


Figure 20. Development environment of Carbide.c++

3.3.3 Software Development Kit (SDK)

In addition to Carbide.c++, to do Symbian OS C++ development you will need to obtain also a Symbian OS Software Development Kit (SDK). A SDK is typically a set of development tools that allows a software engineer to create applications for a certain software package, software framework, hardware platform, computer system, video game console or operating system, as it is done in this project with Symbian OS.

Usually, this SDK contains the emulator and also the libraries and header files required for Symbian OS development. Once installed, the own Carbide.C++ offers the opportunity to download the necessary SDK but it can be also download independently from, for example, the Nokia web site.

In addition to the common user interface components and standard application suite, the used SDK, contains all the interfaces to the dynamic link libraries, executables and device drivers for controlling the keyboard, display and Bluetooth. While it is optimized for the requirements of mobile devices, including economic use of power CPU and memory resources, the Symbian OS is extremely robust. The Symbian operating system is based on a client-server architecture, where many applications are clients that use the resources of servers.

For the development of this project, it has been used the version of SDK for Nokia Series S60 platform created for the mobile phone Nokia N70, the device we are working with, and it is the S60 SDK 2nd Edition Feature Pack 3.

3.3.3.1 What is S60?

The Nokia S60 platform is software for smartphones with advanced data capabilities and many built-in features for devices, like messaging, calendar and a wide variety of personalization options and advanced web browsing. It is used in the majority of Symbian OS smartphones shipped to date. Although owned by Nokia, it is also licensed to other handset manufacturers such as Panasonic, Samsung and Siemens.

Based on the Symbian OS, it includes an application suite, user interface framework, and middleware components (Figure 21). The flexibility of S60 allows a variety of hardware designs and software configurations.

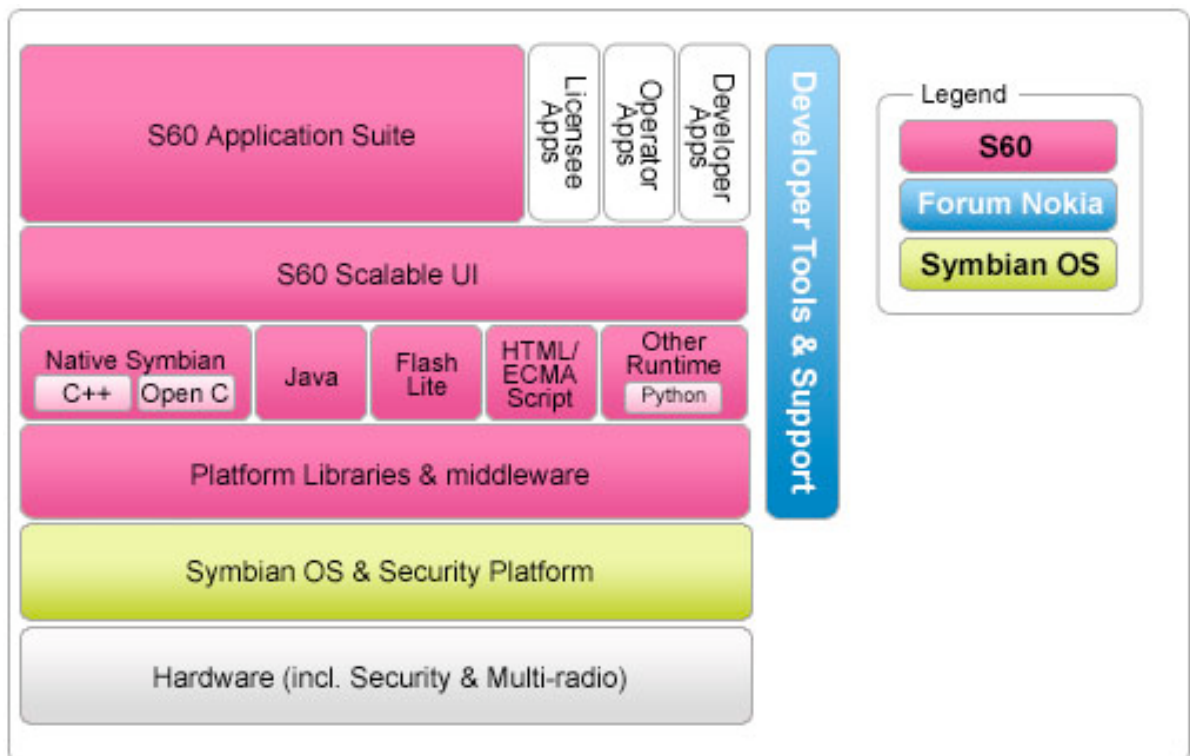


Figure 21. S60 Software

3.3.3.2 S60 Software versions and editions

In order to introduce new features and functionality, new releases of S60 based on Symbian OS are periodically brought to market. While introducing new S60 releases, Edition level releases contain more improvements and architectural changes. Feature Packs, on the other hand, are used to introduce new features on top of this main edition architecture. Each feature pack also includes all the functionality of the previous feature packs before that edition.

As it was explained previously, the device used to develop this project works with S60 2nd Edition Feature Pack 3, that is the latest release of this editions before the creation of the 3rd edition of this series (Figure 22).



Figure 22. S60 releases

3.3.4 The emulator

The emulator is a Windows application that simulates the phone hardware on the PC. This enables the development of phone software to be substantially PC-based, with only the final development stages focused on the hardware. The use of the emulator saves time at the beginning of the development, since you can use the Carbide.c++ development environment to debug the code easily and to resolve most of the initial bugs and design problems without copying each time the executable file on the phone. For example, if a panic occurs in the code and you are working on the emulator, the debugger can provide information to diagnose the error that caused it.

Besides, the use of the emulator reduces the number of times it is needed to create an installation package, transfer it to the mobile phone and install it, which otherwise would be wasted time in the early phases of development.

The Figure 23 represents the view of the emulator used in the development of this project. It has the appearance of a common Nokia mobile phone, as the real one where the tool designed will be used.

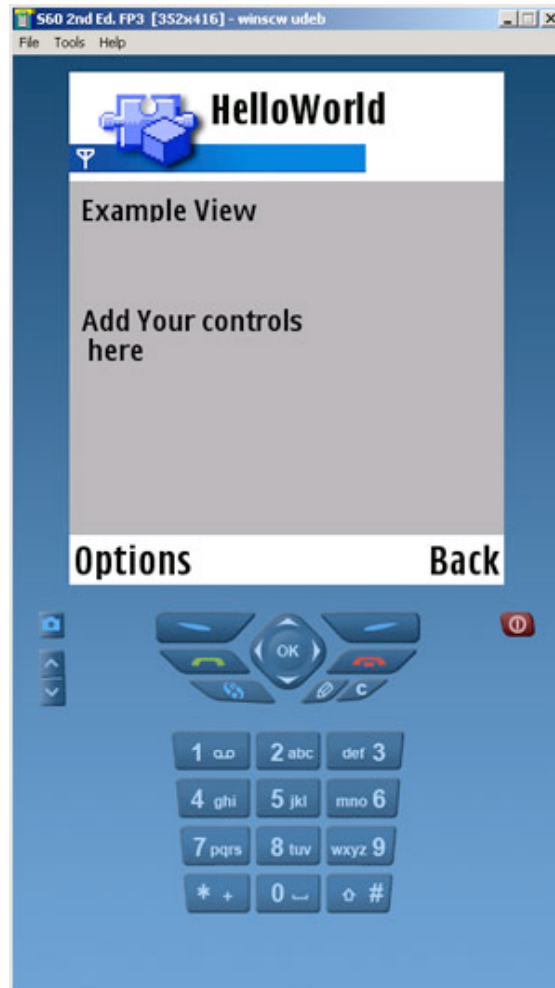


Figure 23. View of the Emulator

4 DESIGN

In this chapter, the design of the whole scenario, in which the data collection tool has been implemented, will be described. Besides, as it was explained before, in order to use the advantages of data collection into practice, an application to do this work has been designed. This application has different parts, each of them with different specific aims; the design of these parts and their different functions, will be explained in this section.

4.1 OVERVIEW OF THE WHOLE SYSTEM

Firstly, the scenario where the tool designed has to work, it is shown in the Figure 24, in order to get a complete idea of the whole project in which its tool is only a small part of a complete system.

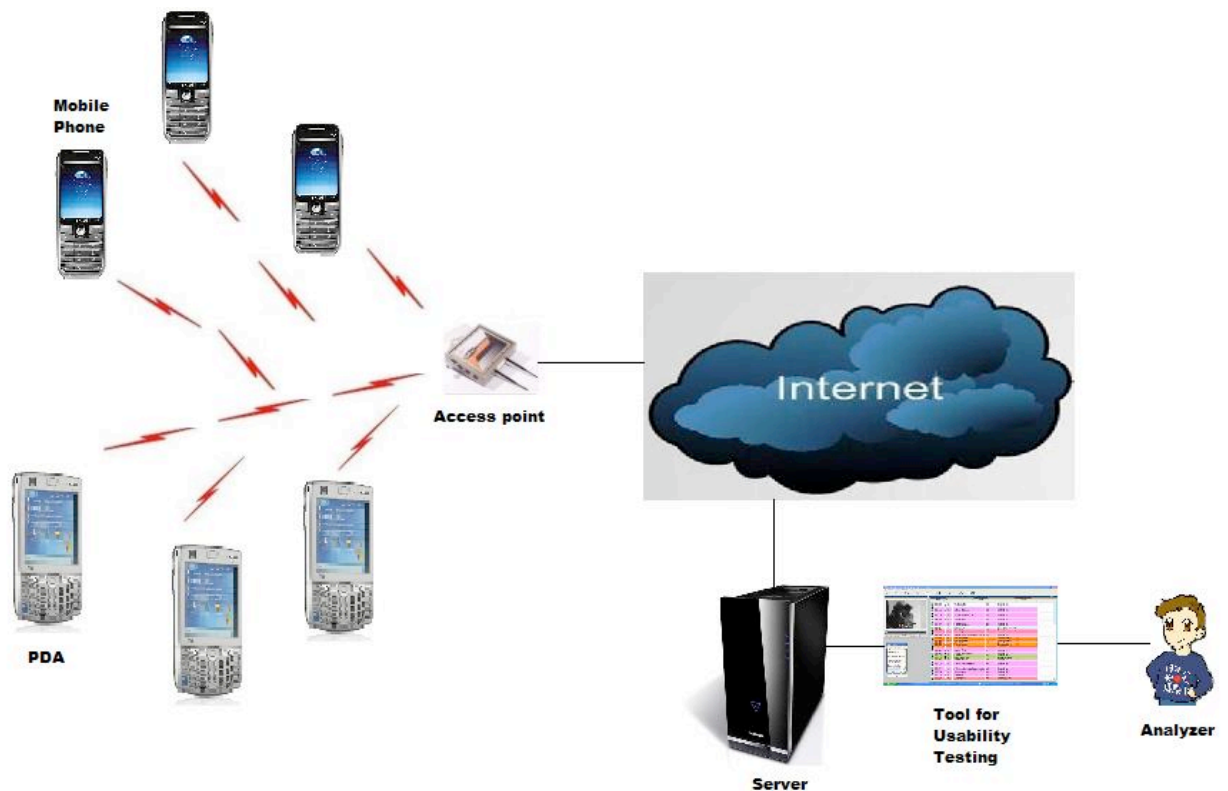


Figure 24. Representation of the whole system application

In this figure, we have represented the entire project developed in the HCI laboratory, where the tool studied in this report, has taken part. On the left, there are the mobile phones where our tool has to be implemented; under them there are represented some PDAs that thanks to another tool designed by Raúl Esgueva, they are doing the same task of gathering data and sending files to the server that our tool, but in different devices, so the programming of these tools is different to our, although there are some points in common.

As it was explained before, our tool and the one designed for the PDAs, besides the collection of the data, they will send files to a server through Internet, using the network that it is on their range. This server has a tool, designed by Héctor Martín, to collect the data received from the mobile devices and PDAs and to store it correctly.

Once the data is stored in the server, this information will be used by another application designed by George, with the objective to analyze it, doing different statistics and usability tests, and facilitating the work of an analyzer who will have to study the data collected.

Then, it is needed to take in consideration the scenario where this complete system will be used. This project was created with the goal to use it in a museum, in which it is necessary to communicate each mobile device with the others and with the server; the objective is to show information to the visitors of the museum, which will carry one of the devices. This data given to the visitors concerns not only about a guide of the sculptures, paintings and more elements exhibited, but also about games created to entertain children during the visit and more applications.

This overview of the complete project that is been developed by the group of Human Computer Interaction of the University of Patras, could show us how it is needed to do a good job in the collection of the data, in the mobile phones as in the PDAs. In this situation is where the tool described in this report appear trying to give a solution to gather data in the smartphones and then use this data to do the usability evaluation in the following steps of the project.

As it is imagined, always there are problems in the development of a new application, and, in this case, the main problems are related to how the data is stored and how it is sent to the server. Moreover, it has to be taken in consideration the necessity of logging the data in the different devices in the same way, even if for ones of them the tool is programmed in Symbian C++ language and for the others in .NET Framework C#; because in order to process and manage this data in the server side with the same infrastructure for all the devices, the designed tools need a similar structure to construct and send the information.

Also, it is necessary to take in considerations the problems that could appear because of the network used to communicate all the elements of the project. One of then is the synchronization of all the equipments to share the information; other could be the possible bottleneck created if the server where all the mobile devices send its data is down and cannot see the information until the problem is fixed.

4.2 DATA COLLECTOR TOOL FOR SYMBIAN PHONES

The development of the tool for Symbian phones to collect data and send it to the server is based in the creation of a Dynamic Link Library that includes the classes and functions necessary for the correct implementation of an application for a user that wants to do these tasks.

4.2.1 What is a Dynamic Link Library?

A Dynamic Link Library (DLL) is a library of compiled C++ code that is loaded in memory when is needed and its functions are available to all running programs. Only a single copy of each loaded DLL exists in memory at a time. This is more efficient than the traditional static library, where each executable that uses the library's functions links to a separate copy of its code.

On Symbian OS there are two main types of DLL:

- **Shared library DLL**, also known as static interface DLL, are the traditional style of libraries, containing a collection of classes and functions that are made available to call programs.

- **Polymorphic DLLs** are used as plug-ins, as instead of only providing classes and functions as in shared library DLLs. They provide a concrete implementation for some abstracted interface.

For this project, it is used a shared library DLL that implements library code that will be used by multiple components of the user's application. The filename extension of the shared libraries is always *.dll*, as in the tool developed, where the library is called *Library1.dll*.

This shared library exports API functions according to a module definition file, currently known as *.def* file. It has any number of exported functions from the library, and each one of them is an entry point into the DLL. It releases a header file (*.h*) for other components that use the library functions to compile against, and an import library (*.lib*) to link against in the *.mmp* file of the applications that use the library, in order to resolve the exported functions.

Then, when the executable code that uses the library runs, the Symbian OS loader has the task of loading any shared DLLs that it links to and any further DLLs that those DLLs require, doing this recursively until all shared code needed is loaded.

4.2.2 Class diagram of the tool

The class diagram of this tool was designed to be easy to understand, and the communication represented between the different classes is very simple and it can be followed in the Figure 25.

Firstly, the Application class is the first one created, by framework, when the application that is using the data collector tool wants to be used, and it will represent the properties of this application and some specific information. Then, this Application class will create an object of the Document class that takes care of the data related to the application and creates the interface between user and application. The Application UI class implements this interface, and controls the views of the application, besides it creates an object of the Container class that will handle the components of this interface.

Then, from the Application User Interface class, one of the methods will create an object of the Application1Engine class that implements the events that has to be log, and interact to the library creating an object of it and calling its functions. This Application1Engine class contains some object of the class Property necessary to insert the name and value of the properties from the event to collect. When all the properties are stored in elements of the class Property, there are included in to an array called PropertyArray that acts as a list of properties that will be pass as parameter to the library methods.

Once the Library received the array of properties, this information will be extract and treated to convert this data to a format understand by the server or to write it in a local log file. Besides, the library will create a connection, using some elements of the RHTTPSession and RHTTPTransaction auxiliary classes; and through it connection will communicate with the server to send and receive data.

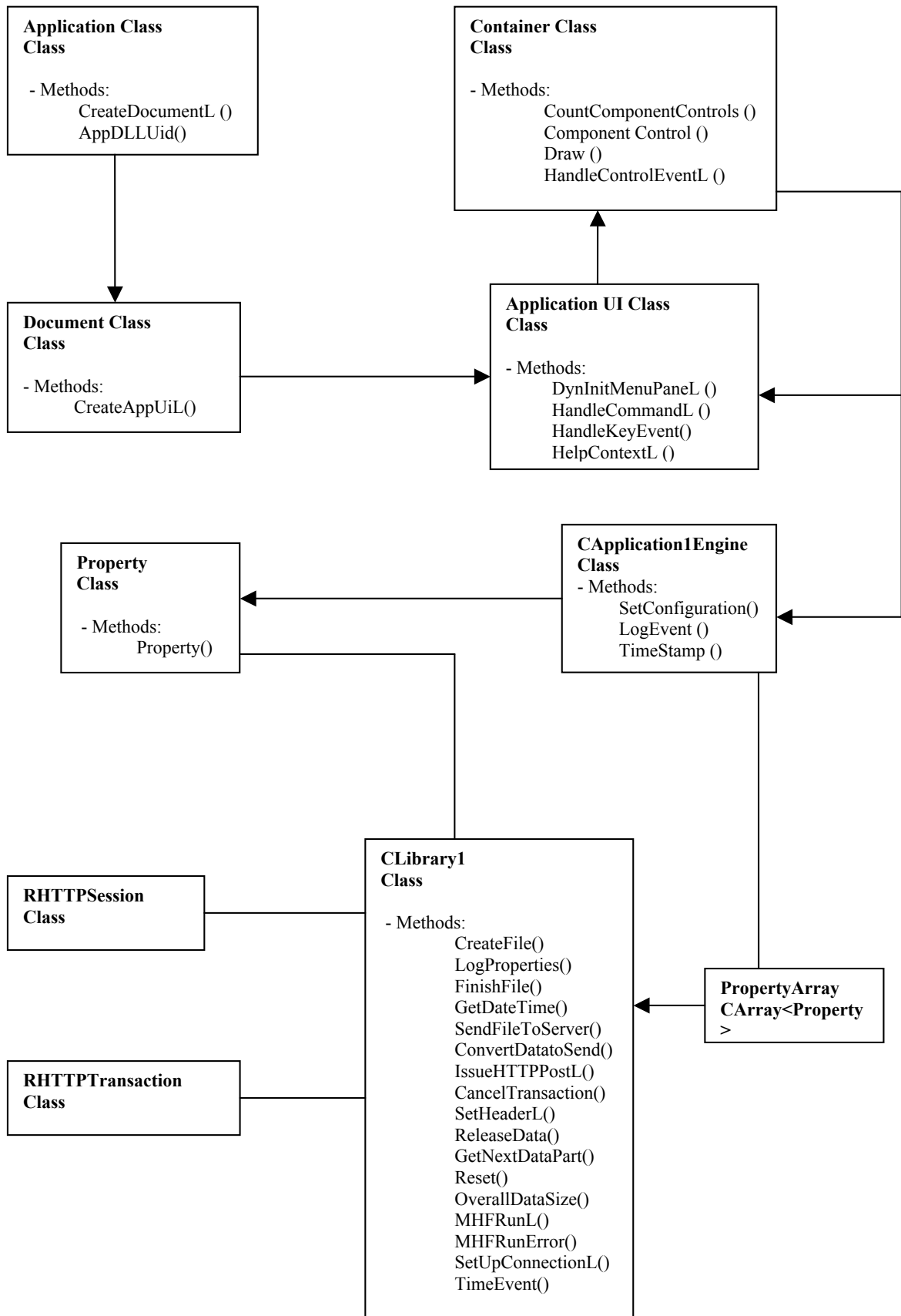


Figure 25. Class Diagram of the tool

4.2.3 Application Programming Interface

The application programming interface (API) is a set of declarations of the methods that an operating system, library or service provides to support requests made by computer programs. [13]

For the tool developed, next there are explained the library created, the classes used, the methods developed and the variables included, in order to do understand better the code and the different tasks done for it.

- The library -

As it is said before, in this tool it is used a Dynamic Link Library (DLL) that contains the necessary methods to carry out the tasks that the user asks to the application linked to it.

Name	Description
Library1	This is the DLL that contents all the classes and functions to collect data locally and to send data to a server. The user application links against it to use their methods and to do the necessary tasks with the data.

Table 2. Content of the DLL Library1

- **Classes developed included in the library –**

The library includes two classes developed by the programmer during the elaboration of this tool, one of them is the main one in the project, the class CLibrary1 that contains most of the methods implemented. The other one, the class Property is used to set the name and value of the properties in each event, as it will be explained next.

Class	Description
CLibrary1	This class manages the functions includes in the library, includes all the necessary methods to log an event and sent data to the server. An object of this class is created in the user's application.

Property	This class is used to give a name and a variable to every property that the user wants to log in an event.
----------	--

Table 3. Classes developed included in the DLL Library1

- **Classes used in the library provided by the platform -**

Besides the previous classes, the library makes use of other classes that are not developed by the programmed, but are provided by the platform that includes a group of references to this kind of classes, that are very common in applications, to make easier the work for the developer. And in this library, these two classes are very important to create an object of them because it is necessary to establish a connection with the server where the data wants to be sent, so they must be declared in the header file of the library.

However, several classes more provided by the platform are used in the methods of the tool designed, but not all of them are explained in this document, only the most important ones for its development that must be declared in the header files for a correct operation creating objects of these classes.

Class	Description
RHTTPSession	This class handles sessions. This session is a set of HTTP transactions using the same connection settings and the same set of filters.
RHTTPTransaction	This class manages a HTTP transaction. It encapsulates one HTTP request and response. Each transaction must be associated with a session.

Table 4. Classes from the platform used in the DLL Library1

- Class CLibrary1 –

This class of the DLL library implements the most of the necessary tasks that the application tool wants to do to log data locally and to send data to the server. It means that the code of this class is the most complex and important of the project and it contains many functions and variables to do this job.

1. Syntax

This is the syntax of this class `CLibrary1`, and it is shown that this class inherits from the class `CBase`, the class `MHTTPDataSupplier`, used to deliver the response data to the client and to supply request body data to HTTP in POST transactions; and the class `MHTTPTransactionCallback`, used to receive events during the transaction.

```
class CLibrary1: public CBase, public MHTTPDataSupplier, public
MHTTPTransactionCallback
```

2. Constructors and Destructor

This class has a two-phase construction, so it implements the following methods to create an object of this class in several steps and one more to destroy the created object after its use.

Access	Type	Name	Description
Public	static	NewL	Create a <code>CLibrary1</code> object and returns a pointer to the created instance of <code>CLibrary1</code> . It is called by the user's application and receives the parameter <code>Configuration</code> , that is send to the function <code>NewLC</code> that calls to continue the creation of the object.
Public	static	NewLC	Create a <code>CLibrary1</code> object and returns a pointer to the created instance of <code>CLibrary1</code> . It is called by the method <code>NewL</code> and receives the parameter <code>Configuration</code> , that is send to the function <code>ConstructL</code> that calls to continue the construction of the object.
Private		<code>CLibrary1</code>	Perform the first phase of two-phase construction of a <code>CLibrary1</code> object.
Public		<code>~CLibrary1</code>	Destroy the object created.
Public	void	<code>ConstructL</code>	Perform the second phase construction of a <code>CLibrary1</code> object. It receives the parameter <code>Configuration</code> as a variable to know if the user wants to log locally or in the server side.

Table 5. Constructors of the class `CLibrary1`

3. Methods

The following methods are implemented in the class CLibrary1 to log events in a file and to send data to the server. It is necessary to take in consideration that some of these functions are inherit from the class MHTTPDataSupplier as GetNextDataPart, OverallDataSize, ReleaseData and Reset; and also there are methods inherit from MHTTPTransactionCallback as MHFRunL and MHFRunError.

Access	Type	Method	Description
public	void	CreateFile()	Connect to the file server to create a new file to log events.
public	void	LogProperties()	Receive the array PropertyArray with the properties of the event and write this information in the file. After that, deletes the array.
public	void	FinishFile()	Finish the file where the events are logged and close it. Also, disconnect from the server file.
public	TPtrC	GetDateTime()	Get the date and time when is called and returns it to the method CreateFile to set the name of the log file depending on the moment when is created.
public	void	SendFileToServer()	Open a new HTTP session.
public	void	IssueHTTPPostL()	Starts a new HTTP POST transaction.
public	void	ConvertDatatoSend()	Set the data to send to the server.
public	void	CancelTransaction()	Closes currently running transaction and frees resources related to it.
private	void	SetHeaderL()	Used to set header value to HTTP request. So, the IssueHTTPPostL calls it during its execution.
private	void	ReleaseData()	Inherited from MHTTPDataSupplier, it allows us to release resources needed for previous chunk. This function is called by MHFRunL in the

			case EGotResponseBodyData.
private	TBool	GetNextDataPart()	Inherited from MHTTPDataSupplier, this function is called when next data part is needed by MHFRunL in the case EGotResponseBodyData and it has to return ETrue if the received part is the last one.
private	TInt	Reset()	Inherited from MHTTPDataSupplier, it is called by framework to reset the data supplier that should return to the first part of the data. In the practise an error has occurred while sending data, and framework needs to resend data.
private	TInt	OverallDataSize()	Inherited from MHTTPDataSupplier, it is called by framework. We should return the expected size of data to be sent. If it is not know we can return KErrNotFound.
private	void	MHFRunL()	Inherited from MHTTPTransactionCallback and called by framework to pass transaction events.
private	TInt	MHFRunError()	Inherited from MHTTPTransactionCallback and called by framework when a leave occurs in handling a transaction event.
private	void	SetupConnectionL()	Call by IssueHTTPPostL to start a connection. The method set the internet access point and connection setups.
private	TPtrC	TimeEvent()	Return the date and time when it was called, it is used to get a timestamp for the events logged in the server in the POST transaction.

Table 6. Methods of the class CLibrary1

4. Global Variables of this class

As this class CLibrary1 do many tasks to log events and sent data to the server, the number of variables is high and some of them are used in several functions of the class, so

the following members are defined in the header file of the class for their use along the different methods.

Access	Type	Name	Description
public	TBool	iServer	Variable set true if user wants to log data in the server.
public	TBool	iLocally	Variable set true if user wants to log data locally.
public	RFs	fsSession	The file server connection opened to work with files when we log events locally.
public	RFile	file	Created as object of the class RFile to handle files when we log events locally.
public	TBuf<40>	fName	The name of the logged file to store data locally that will vary depending on the time when is created.
public	HBufC8*	iPostData	Contain the data for the HTTP POST transaction.
public	HBufC8*	iPostDataImage	Variable with the same size of the data to send, that is used to operate with it without writing on the original data to send and avoid problems of losing information.
public	TBool	iRunning	Variable set true, if the transaction is running.
public	RHTTPSession	iSession	Represents the session opened to send data to the server. It is an object of the class RHTTPSession explained previously.
public	RHTTPTransaction	iTransaction	Represents the transaction opened to send data to the server. It is an object of the class RHTTPTransaction explained previously.

public	RSocketServ	iSocketServ	Represents the socket used to open a connection.
public	RConnection	iConnection	Represents the connection opened in the socket to send data to the server.
public	TBool	iConnectionSetupDone	Variable set true to know that the connection is up.
public	HBufC*	iResponseBuffer	Variable to store the response body data.
public	TBool	iDataAvailable	Variable set true to know that there is data available to be sent.
public	TFileName	iFileName	Contain the name of the file to send to the server to include its data in the object iReqBodyFile.
public	RFs	iFileServ	The file server connection opened to work with files when we send data to the server.
public	RFile	iReqBodyFile	Object of the class RFile to handle files when data is sent to the server.

Table 7. Global variables in class CLibrary1

- Class Property -

This class is used to get and store the information from each property of the event that the user wants to log. It is composed by two fields, the name and the value of the property, and when these variables are loaded in an object of this class, the information is sent to an array that contains the list of all the properties of the event.

1. Syntax of the class

```
class Property
```

2. Methods

This class implements only one method to get the necessary information about the properties included in each event logged by the user of the application.

Type	Method	Description
Public	Property	This method receive name and value of each property in an event, and store this information to copy it in the array of properties PropertyArray.

Table 8. Methods of the class Property

3. Variables

This class uses these two variables to store inside them the name and the value of a property and them pass this information to the array of properties passed as parameter to the object Library1, and then this stored information is used in the method LogProperties to write it in the log file.

Access	Type	Name	Description
Public	TBuf<50>	iName	Store the name of the property to load it in the array PropertyArray.
Public	TBuf<50>	iValue	Store the value of the property to load it in the array PropertyArray.

Table 9. Global variables of the class Property

- **The array RArray<Property>** -

Even, not been a class, it is necessary to explain in a few lines what it is the task of this array in the tool. When an event is logged, it will have a group of properties with information about this event that the user wants to log, as important information related to the event.

This group of properties will be stored with its name and value in an element of the class Property, after that, we want to store this team of Property elements together to send it to the library and log it. That is because an array of elements of the class Property will be

created, and there, the list of properties will be stored and passes as parameter to the library class that will log the event with its properties.

In the library, this array, called `PropertyArray`, will be handle to extract each property and its value and log the series of properties correctly. For example, to log an event in the server this array has to contain at least the name of the device that is sending it and the time stamp when the event occurs.

- The Application -

Once the programmer has the DLL library explained previously, it is necessary to develop an application, which interact with the user and with the library, and this application is going to make use of the methods implemented in the library to log events.

In one side, it has to create an interface to interact with the user, as for example to show in the screen of the device a menu with the operations that can be done thanks to the methods developed and also to communicate to the user messages about the execution of the tool.

Also, this application has to create an object of the library to call the functions included there and set what he wants from the library, it means which methods from the library are needed to carry out the tasks that the user requires.

Moreover, this application has to include some properties that are necessary to the interaction with the library, as the methods to provide the parameters that the functions of the library need to be called. These basic methods are `SetConfiguration` and `LogEvent`; the first one set if the data has to be logged locally, it means in the own mobile device, or in a server, and in this case it has to create a connection with the appropriate server, it is also allow to log in both places at the same time. The second method, `LogEvent`, is used to call the needed functions of the library depending on the kind of configuration selected, and it has another uses as set the URL with the IP address of the server or set the unique identifier of the mobile device.

The application can also include another method to calculate the time necessary to log events, in the server or locally. This is a good way to test the tool and evaluate the time needed to log this data depending on the number of events logged and the number of properties of these events.

Next, there are explained the five classes contained in this application; one of them developed for our tool, and the other four classes are basics for every Symbian OS application like the one implemented.

Class	Description
CApplication1Engine	This class calls the functions includes in the library and set the configuration to use, besides it receives an object of the class CLibrary1 to use its methods. Here, it is implemented also some methods to calculate the time necessary to log events in a local file or in the server.
CApplication1App	It is the entry point for the operating system. It represents the properties that are the same for every instance of the application. This includes the information specified in the registration file and other information, like the UID. Besides it creates the class CApplication1Document.
CApplication1Document	It is constructed and returned by one of the functions of the class CApplication1App mentioned above. It takes care of the data that relates to the application and, it creates the interface between the user and the application.
CApplication1AppUi	It is the “controller” of the application. The main task of this class is to manage the views of the application, as to control the size of the screen and the option’s menu.
CApplication1Container	It is constructed by the class CApplication1AppUi and it handles the user-interface components.

Table 10. Classes in the application

For the development of this tool this application is called Application1 and this is the reason because the classes on it starts with the same name, to identify that it depends to that application, but if it would be necessary that names can be changed.

- CApplication1Engine Class –

This Symbian OS class was created as a part of the tool designed because it handles the interaction with the library, and from this class an object of the library is created and its methods are called.

There are developed three functions on this class, one of them is used to define is the user wants to log in the own device, in the server or in both places. Other of its methods is who calls the library, firstly creating an object of the library and then calling the necessary functions to log events using that object. And the last method is used as a evaluation tool, to calculate how much time is needed to log events, locally or in the server, and varying

variables like the number of events or the number of properties per event could be possible to do a test of the tool.

1. Syntax of the class

This is the syntax of this class `CApplication1Engine`, and it is shown that this class inherits from the class `CBase`, basic in Symbian OS for all classes instantiated on the heap.

```
class CApplication1Engine : public CBase
```

2. Constructors and Destructor

This class has a two-phase construction, so it implements the following methods to create an object of this class in several steps and one more to destroy the created object after its use.

To create an object of this class, firstly it is necessary to call the function `NewL()` that will call `NewLC()` to continue the construction, from there is called the constructor for performing the first stage construction `CApplication1Engine()` and the second stage `ConstructL()`.

Access	Type	Name	Description
public	<code>CApplication1Engine</code>	<code>NewL()</code>	Two-phased constructor.
public	<code>CApplication1Engine</code>	<code>NewLC()</code>	Two-phased constructor.
private	void	<code>ConstructL()</code>	EPOC default constructor for performing 2nd stage construction.
private		<code>CApplication1Engine()</code>	Constructor for performing 1st stage construction.
public		<code>~CApplication1Engine()</code>	Destructor.

Table 11. Constructors and Destructor of the class `CApplication1Engine`

3. Methods

As it was explained above, this class implements three classes developed by the user to give an application for the library designed.

The method `SetConfiguration()` set the configuration for the library, and for this task it is necessary to set a variable called `Configuration` that represents if the library has to log

events locally (when the value of Configuration is 1), to log in the server (when the value is 2) or in both places (when the value is 3).

The function LogEvent() create an object of the library and calls its methods, if the user wants to log locally (according to the configuration previously set) this method will call the library to create a new XML file on the device and log events there, also writing the properties of the event that this function will send in an array called PropertyArray to the library. If the user wanted to log in the server the connection will be created to the server, indicating in this function its url, and sending also to the library the array with the properties of an event.

Lastly, the method TimeStamp() takes from the system of the device the date and the time in that moment, and returns it to the function who called, then this time will be use to calculate the time wasted to log events calling it two times and displaying the difference in microsecond between them. This is implemented as a tool to measure the capabilities of the application designed.

Access	Type	Name	Description
public	void	SetConfiguration()	Set the kind of configuration that the user wants to use, logging in the device or/and in the server.
public	void	LogEvent()	Creates an object of the library and calls the functions on it, depending on the configuration set.
private	TTime	TimeStamp()	Returns the exact date and time from the device. It is called from LogEvent() to test how much time it is needed to log events and it could be a good tool to evaluate the application.

Table 12. Methods of the class CApplication1Engine

4. Global Variables of this class

As this class CLibrary1 do many tasks to log events and sent data to the server, the number of variables is high and some of them are used in several functions of the class, so the following members are defined in the header file of the class for their use along the different methods.

Access	Type	Name	Description
public	TInt	NEvents	The number of events that the user wants to log.

public	TInt	Configuration	The configuration that the user wants to use to log events.
public	TBool	Close	This variable is set false logging in the device to check if the file were the events must be log is already open. It is useful when the user wants to log several events at the same time in a XML file.
public	CLibrary1*	EventLogger	This object of the library will be referred with this name when it is created.

Table 13. Global variables of the class CApplication1Engine

- Application Class –

The application class is the point of entrance for the operating system, it is constructed and returned by the NewApplication() function described below that belongs to the Avkon class, that is part of the S60 UI framework.

This application class, in our code called CApplication1App class, represents the properties of the application. This includes the information specified in the registration file, and other information, as for example the application UID. Besides, it creates an object of the Document class described below.

1. Syntax of the class

This is the syntax of this class CApplication1App, and it describes that this class is derived from the class CAknApplication, contained in Avkon library, basic to create a series S60 application.

```
class CApplication1App : public CAknApplication
```

2. Constructor

Access	Type	Name	Description
public	CApaApplication*	NewApplication()	Creates an application object.

Table 14. Constructor of the Application Class

3. Methods

This class implements only two basic functions for the creation of a Symbian OS application, `CreateDocumentL()`, which creates the document class explained below, and `AppDllUid()`, which returns the UID of the application.

Access	Type	Method	Description
private	CApaDocument*	CreateDocumentL()	Creates an object of the CApplication1Document class.
private	TUid	AppDllUid()	Returns the UID of the application.

Table 15. Methods of the Application class

- Document Class –

The document class is constructed and returned by the function of the application `CreateDocumentL()`, mentioned above. It represents the data that relates to a particular instance of the application, it means that this class takes care of the data model. Besides, it creates the application user interface, related to the next class explained, with the implementation of a function called `CreateAppUiL()`, that it is inherited from the document's base class `CEikDocument`.

1. Syntax of the class

In the following syntax of this class is shown that its name on the application designed is `CApplication1Document` and this class is inherited from `CAknDocument`, the base class for the S60 application documents.

```
class CApplication1Document : public CAknDocument
```

2. Constructors and Destructor

This class has two-phase construction also as the class `CApplication1Engine` explained before.

Access	Type	Name	Description
public	CApplication1Document*	NewL()	Two-phased constructor.

public		~CApplication1Document()	Destructor.
private		CApplication1Document()	EPOC default constructor
private	void	ConstructL()	Performs the second stage of the constructor.

Table 16. Constructors and Destructor of the Document class

3. Methods

This class implements only one method to, used to create an object of the class application user interface explained below.

Access	Type	Method	Description
private	CEikAppUi*	CreateAppUiL()	Creates an object of the application UI class.

Table 17. Methods in the Document class

- Application User Interface Class –

The application UI class is constructed and returned by the CreateAppUiL() function of the Document class, described above. Its main task is to create and initialize the views of the application and also it handles the commands sent from the application's graphical UI controls.

In the application designed, the user can do changes in the function HandleCommandL() of this class, and, in this way, modify at its pleasure the options menu created to, for example, integrate the configuration of the library explained above in this menu and then will not be necessary to implement in the CApplication1Engine class the function related to it.

1. Syntax of the class

The syntax of this class is the following, and its name on the application designed is CApplication1AppUi and this class is inherited from CAknAppUi, the base class for the S60 application User Interface.

```
class CApplication1AppUi : public CAknAppUi
```

2. Constructors and Destructor

This class has only one constructor and a destructor. Besides, this constructor creates an object of the Container class that follows.

Access	Type	Name	Description
private	void	ConstructL()	EPOC default constructor.
public		~CApplication1AppUi()	Destructor.

Table 18. Constructor and Destructor of the Application UI class

3. Methods

This class contains three methods, the most important for the development of our application is the HandleCommandL() function because there it can be change the options menu that appears when the application is open, and it can be modified by the programmer.

In the application developed, Application1, the menu has been change to set if the events can be logged on the device, on the server or in both places, as an alternative to modify the function Configuration of the class CApplication1Engine with the preferred configuration.

Access	Type	Method	Description
private	void	DynInitMenuPanelL()	The EIKON framework calls this function just before it displays a menu panel.
private	void	HandleCommandL()	It is inherited from CEikAppUi class and takes care of command handling.
private	TKeyResponse	HandleKeyEventL()	It is inherited from CEikAppUi class and handles key events.

Table 19. Methods of the Application UI class

4. Variables

In this class there is a global variable of the class used to refer to an object of the container class created by the Construct() function of the Application UI class.

Access	Type	Name	Description
private	CApplication1Container*	iAppContainer	This variable is an object of the Container class.

Table 20. Variables of the Application UI class

- Class CApplication1AppContainer -

The container class is created by the ConstructL() function of the application user interface class, and it is used for handling the UI components. It has the control of the graphical user interface (GUI). Besides, this class implements the main window and acts as a container for the other application controls.

1. Syntax of the class

The syntax of this class is the following, and its name on the application designed is CApplication1AppUi and this class is derived from CCoeControl, the basic control base class of the UI Control Framework.

```
class CApplication1Container : public CCoeControl,
MControlObserver
```

2. Constructors and Destructor

This container class has only one constructor and a destructor.

Access	Type	Name	Description
public	void	ConstructL()	EPOC default constructor.
public		~CApplication1Container()	Destructor.

Table 21. Constructor and Destructor of the Container class

3. Methods

In the application designed, this class implements five methods related to control the graphical view of the application for the user.

Access	Type	Method	Description
private	void	SizeChanged()	Called by framework when the view size and position are changed.
private	TInt	CountComponentControls()	It tells how many controls to draw.
private	CCoeControl*	ComponentControl()	This method controls the Draw() function.
private	void	Draw()	It is the actual draw operation.
private	void	HandleControlEventL()	It is inherited from MCoeControlObserver class and it acts upon the changes in the hosted control's state.

Table 22. Methods of the Container class

5 IMPLEMENTATION

In this chapter, it is presented the data collector tool since the point of view of the most important steps of the design explained above, so this time it is shown how exactly the tool prepare the data to be log, how it is log in the device and how it is log in the server, explaining also the connexion with it to send HTTP POST request.

5.1 DATA COLLECTOR TOOL FOR SYMBIAN PHONES

This data collector tool has been created for using it in Symbian phones, so as it was explained in the technology chapter, the language used for the implementation is Symbian C++, making use of the Carbide.c++ program to carry out this implementation. In the following pages, the main functions implemented are presented and in some cases also some pieces of the code are included so for that people do not familiarize with this language it could be difficult to understand but each sentence is explained trying to make easier the task of the reader.

5.1.1 Prepare the data to be log

One of the most important parts of this tool is to store the data to log correctly to send it to the logging methods of the library. If the implementation of this task is not good, the rest of the work would not be worthy, so it is necessary to pay attention on the development of this part of the logging process.

To store this data, that it is composed by a list of properties and the value that takes each one of them, it was defined a class called Property with two fields of 50 characters each one of them, one for the name of the property and the other one for its value. In the developing process it was decided to store the information in this way because it is not very complicated, and it allows keeping together the name and the value of the property in the same variable. Besides, the size of this fields were set to a string of 50 characters because it was considered big enough, and we avoid to keep so many resources for that, because probably it is unnecessary.

Then, in the code of each event to log, it is created an element of the Property class for each property that has to be logged, and there with are going to add the name and the value to store as follows:

```
Property i (_L("actionEvent"),_L("Edit File"));
```

These properties will be stored in a list of properties, inside an array of elements of the Property class declared as:

```
RArray<Property> PropertyArray;
```

The number of properties is not set, so the user can add as much as he wants to the array and send it to the logger method in the library that will extract the information. The elements are stored in the array as follows:

```
User::LeaveIfError(PropertyArray.Append((Property)i));
```

What it is said there, it is that the element named *i* from the class *Property*, will be append to the array called *PropertyArray* and if any problem happens during this operation, the function will leave.

5.1.2 Log events in the own device

When the user selects to log the data locally, in the own device that is using, firstly an object of the library will be created, and after that the code from the application will call the methods of library with that object created, as much times as it needs.

In the following pages, there are explained the most important steps to log events in the own device, including some lines of code that come from the methods that carry out these tasks.

- Create the XML file -

Then, the following step is to create that file where the events are going to be logged. It is just necessary to connect to the file server, and then to create a file in writing mode and, it is useful, to add at the beginning some data to describe the content of the document, like the encoding type, the application's ID or the device's ID.

Besides, the way to naming the file in our tool, it is calling a function that returns the exact date and time when the file was created, and in this way the user can order the files that he will create easily.

Below, it is shown some pieces of the CreateFile() method that includes the most relevant sentences to take a look about how the code carry out these tasks.

```
EXPORT_C void CLibrary1::CreateFile()
{
    // Connect to the file server
    fsSession.Connect();
    ...
    // Set the name of the file XML with the time when was created
    TBuf<40> fileName = GetDateTime();
    file.Create(fsSession, fileName, EFileWrite);
    ... // Write the first lines with useful information
}
```

- Write data on the file -

Once the file is created, the application calls the library to use the method `LogProperties()` that need as parameter the array with properties explained above.

As it can be logged several events in the same log file, it is necessary to identify each one of them, that is why before starting to write the properties, the time is written, and also, this is a good tool for the user to know which event occur in each moment.

After that the function will extract the property name and the property value from the array using a *for* sentence that will go through all of it, until the last element because it will count how many elements the array has thanks to the function `Count()` provided by the platform for the management of array information.

```
EXPORT_C void CLibrary1::LogProperties(RArray<Property>* PropertyArray)
{
    ... TimeEvent();
    ... //Open the event
    file.Write(_L8("\r\n\t<event time=\"));
    ...
    file.Write(_L8("\r\n\t\t<properties>"));
    for (TInt i=0;i<PropertyArray->Count();i++) // Browse the array
    {
        ... //Print property Name extrating the info from the array
        myName.Format(_L("%S"), &(*PropertyArray)[i].iName);
        TPtrC8 ptrC8Name((TUint8*)myName.Ptr(), myName.Size());
        file.Write(_L8("\r\n\t\t\t<prop name=\"));
        file.Write(ptrC8Name);
        file.Write(_L8("\t\t\t\t\t"));
        ... //In the same way print the value and close property
        file.Write(_L8("</prop>"));
    }
    ... // Close properties and close event
}
```

When all properties of the event are written, it will close it and if the user just select to log one event the file will be close as follows, but if he wants to log more events in the same file it could be done and he just need to call again to the `LogProperties()` function from the application sending a new array of properties.

- Close the file -

The user can log as much events as he wants in the same XML file, and after the last one, it is necessary to call the `FinishFile()` method to close correctly the document. If the application forgets to call this function, the file will be saved, but the log file will not be finished with the sentence `</logfile>` that indicate in XML that this logfile is finished, and the connection with the file server will not be closed neither.

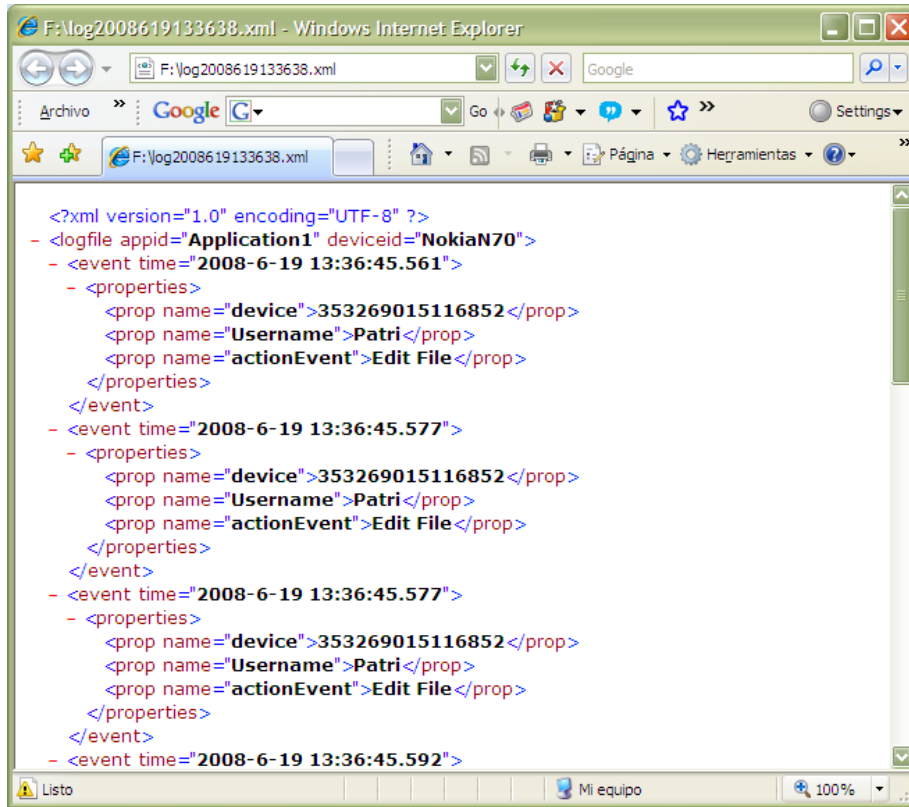


Figure 26. XML file created correctly

If the file was closed correctly, it can be opened with, for example, the Internet Explorer, to check that everything was written fine and all the labels were closed (Figure 26). If some label was not closed or some invalid character was written, when we will try to open it with the Internet Explorer it will appear a note to tell that something was wrong on the file (Figure 27). The file can be opened with other programs, like the Notepad but this is not going to tell us if there is any mistake on it, but the information can be read.

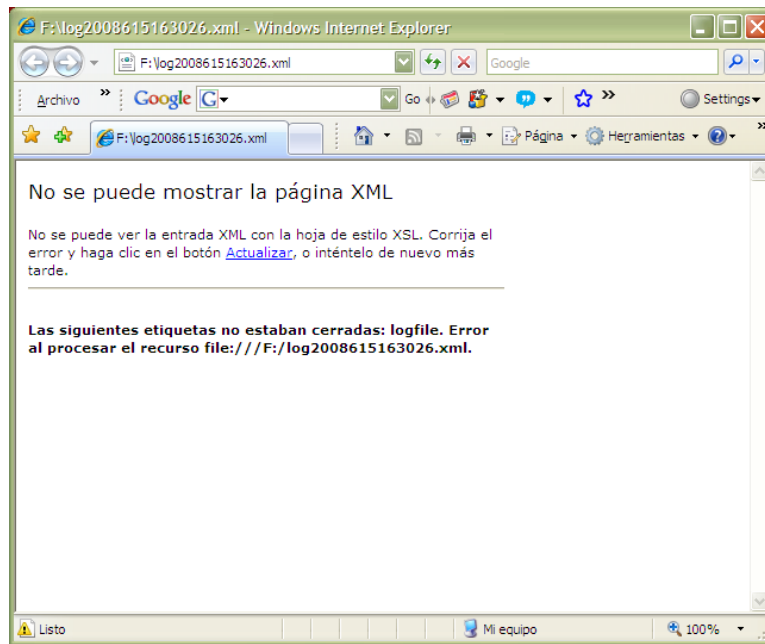


Figure 27. XML file wrong created

5.1.3 Log events in the server

When the user selects to log the data in the server, firstly an object of the library will be created, and after that the code from the application will call the methods of library with that created object.

Besides, from the application is necessary to send additional information, not only the array of properties. It is necessary to indicate the URL of the server where the data has to be logged; if the user wants to log a file, as for example, a jpeg image, it has to be indicated also, and for that it is needed the name of the file to send, including its path.

Then, it starts to work in the methods of the library to send the data to the server to be logged, and in the following pages, there are explained the most important parts of this process.

- Open a session and set up connection -

Firstly, it is needed to open a HTTP session where it is going to be open an Internet access point, and a socket server to start the connection to Internet through it. Then the preferences of this connection are defined, as the ID of the access point or the kind of dialog preferred. With this preferences set, the connection starts normally.

Once this connection is open, when the data will be ready is only necessary to send it to the preferred server through this connection opened.

```
void CLibrary1::SetupConnectionL()
{
    ... //Open socket server and start the connection
    User::LeaveIfError(iSocketServ.Connect());
    User::LeaveIfError(iConnection.Open(iSocketServ));
    ... // Define connection preferences
    connectPref.SetIapId(m_Iap);
    connectPref.SetDialogPreference(ECommDbDialogPrefDoNotPrompt);
    ... // Start a synchronous connection
    User::LeaveIfError(iConnection.Start(connectPref));
    //set the sessions connection info to use our socket server and
    connection
    ...
}
```

- Prepare data for the POST request -

The data cannot be sent to the server as series of parameters, one after the other, it needs to conform some specifications about the way to post this data to be understood and classified when it arrives to the server.

In the tool implemented, the request method used by the application is a POST request. In this kind of request, the each parameter of data to send has to be delimited by a boundary. For example, if the user wants to send three properties of an event to the server, first it is necessary to put the boundary, the content disposition that will be data, the name of the first property between inverted commas, two line breaks and then the value of this property; after that, to send the next property it has to follow the same schema, the boundary, the content disposition, the name, etc. At the end of all the data to be send it is necessary to put another boundary following by the characters "--" to indicate to the server that it is the last information.

An example of the upload format explained is the following:

```
EXPORT_C void CLibrary1::ConvertDatatoSend(...)
{
    ... ...

    ... //The variable iPostDataPtr is created to append there all the
    // data, because it is want we are going to send to the server
    iPostDataPtr.Append("\r\n");
    // The properties content in the array
    for (TInt i=0;i<PropertyArray->Count();i++)
    {
        iPostDataPtr.Append("--AaB03x"); //The boundary
        iPostDataPtr.Append("\r\n");
        iPostDataPtr.Append("Content-Disposition:form-data;name=\"");
        ... // Extract from the array the name
        iPostDataPtr.Append(myNameProp);
        iPostDataPtr.Append("\"");
        iPostDataPtr.Append("\r\n");
        iPostDataPtr.Append("\r\n");
        ... //Extract the value from the array
        iPostDataPtr.Append(myValueProp);
        iPostDataPtr.Append("\r\n");
    } ... ...
    ... ... //If a picture has to be send
```

```
if (File)
{
iPostDataPtr.Append("--AaB03x");
iPostDataPtr.Append("\r\n");
iPostDataPtr.Append("Content-Disposition:form-data;
name=\"file[]\"; filename=\"");
iPostDataPtr.Append(iFileName);
iPostDataPtr.Append("\"");
iPostDataPtr.Append("\r\n");
iPostDataPtr.Append("Content-Type:application/octet-stream");
iPostDataPtr.Append("\r\n");
iPostDataPtr.Append("Content-Transfer-Encoding: binary");
iPostDataPtr.Append("\r\n");
iPostDataPtr.Append("\r\n");
iPostDataPtr.Append(aPtr); //the file in binary
iPostDataPtr.Append("\r\n");
} ... //End of the format for the picture
... .. //Indicate that it is the last data
iPostDataPtr.Append("--AaB03x--");
iPostDataPtr.Append("\r\n");
... ..
}
```

It is important to take in consideration, that every event sent to the server has to include between its properties two important parameters, one of them is the unique ID of the device that is sending the information, identified in the server by the name “device” and the timestamp of the event, identified as “timeStamp”. In a transmission to the server, if these two properties do not appear, the event will not be logged. To avoid this situation, the library designed will send always these two parameters for every event to log in the library. The identifier of the device will be the IMEI code that every mobile phone has, and the time will be take from the system with the function TimeEvent() designed, including from year to millisecond to avoid the situation of several event with the same time, it is supposed thanks to the test done that a precision of millisecond is enough to differ between events.

- The connection to the server and sending the data by POST method -

When the data is ready to be sent, a transaction with the method HTTP POST is open, through the connection created previously, and this transaction is created in the URI indicated, that correspond to the server URL selected to log the data in.

Besides, it is needed to set the headers request for this transaction, with the accepted content type and other characteristics. After that the class will be set as a data supplier to indicate that the methods from this MHTTPDataSupplier class, as the OverallDazaSize(), GetNextPartData(), Reset() and ReleaseData() functions, must be called by the framework to send the body data already prepared to be sent.

This operations are related to the function IssueHTTPPostL() show below with its most important parts for this process:

```
EXPORT_C void CLibrary1::IssueHTTPPostL(... ..)
{
    ... ..
    // Get request method string for HTTP POST
    RStringF method = iSession.StringPool().StringF(HTTP::EPOST,
        RHTTPSession::GetTable());
    // Open transaction with previous method and parsed uri.
    iTransaction = iSession.OpenTransactionL(uri, *this, method);
    // Set headers for request with user agent, accepted content type
    // and body's content type.
    RHTTPHeaders hdr = iTransaction.Request().GetHeaderCollection();
    SetHeaderL(hdr, HTTP::EUserAgent, KUserAgent);
    SetHeaderL(hdr, HTTP::EAccept, KAccept);
    SetHeaderL(hdr, HTTP::EContentType, KPostContentType);
    // Set this class as a data supplier. Inherited MHTTPDataSupplier
    // methods are called when framework needs to send body data.
    MHTTPDataSupplier* dataSupplier = this;
    iTransaction.Request().SetBody(*dataSupplier);
    ... ..
}
```

Besides the POST method proposed, there is another possibility to implement this sending of data, the PUT method. The fundamental difference between POST and PUT requests is the meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request, the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource. If the server desires that the request be applied to a different URI [40].

In the following Figure 28, it is represented the information received from the server where the data has been sent from the mobile phone. In this example, the IP address 150.140.188.197 correspond to the device used, and the server has received several requests from this address using the method POST, and here it is also indicated the number of bytes received by the server, that vary depending on the number of data logged from an event. The code number 200 indicates that the information arrived to the server correctly.

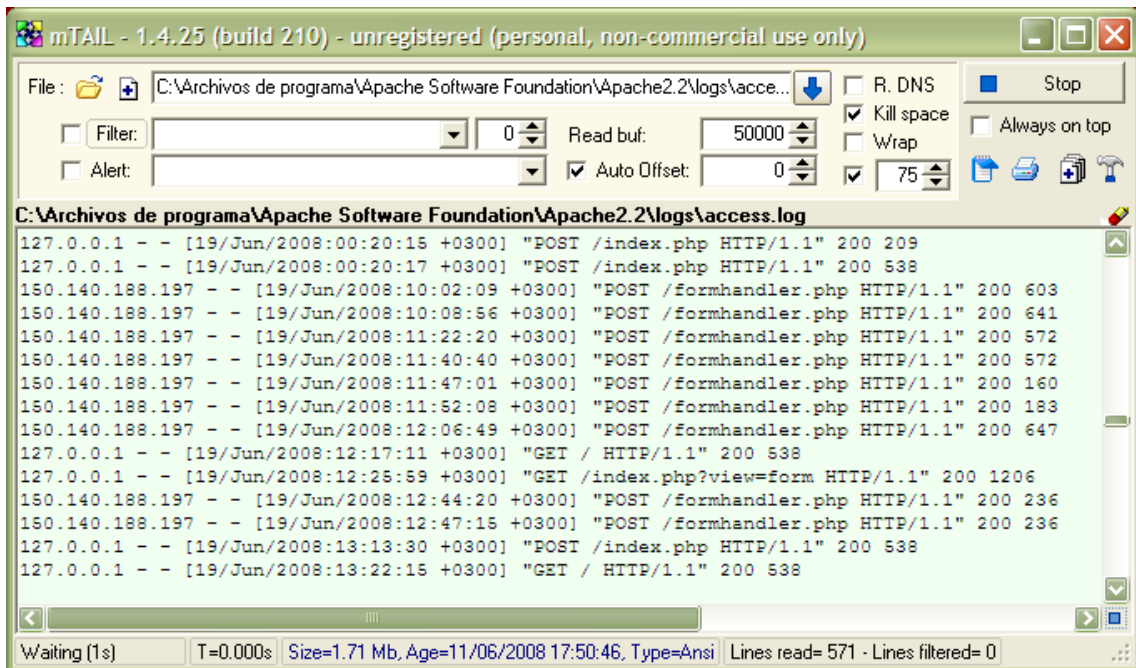


Figure 28. View of the requests received by the server

As it was explained before, the data sent to the server can vary in each event, and the kind of properties to log can be as varied as the user wants, it means that the name of the properties that the server can store is not set. Although, it will recognize the most common ones as device, time, actionEvent or file_name; these ones will be stored in its own database and for the rest of the properties there is a common database that is shown below, with some of the properties received, in the Figure 29 are represented the ones sent by the Symbian phone using the library developed.

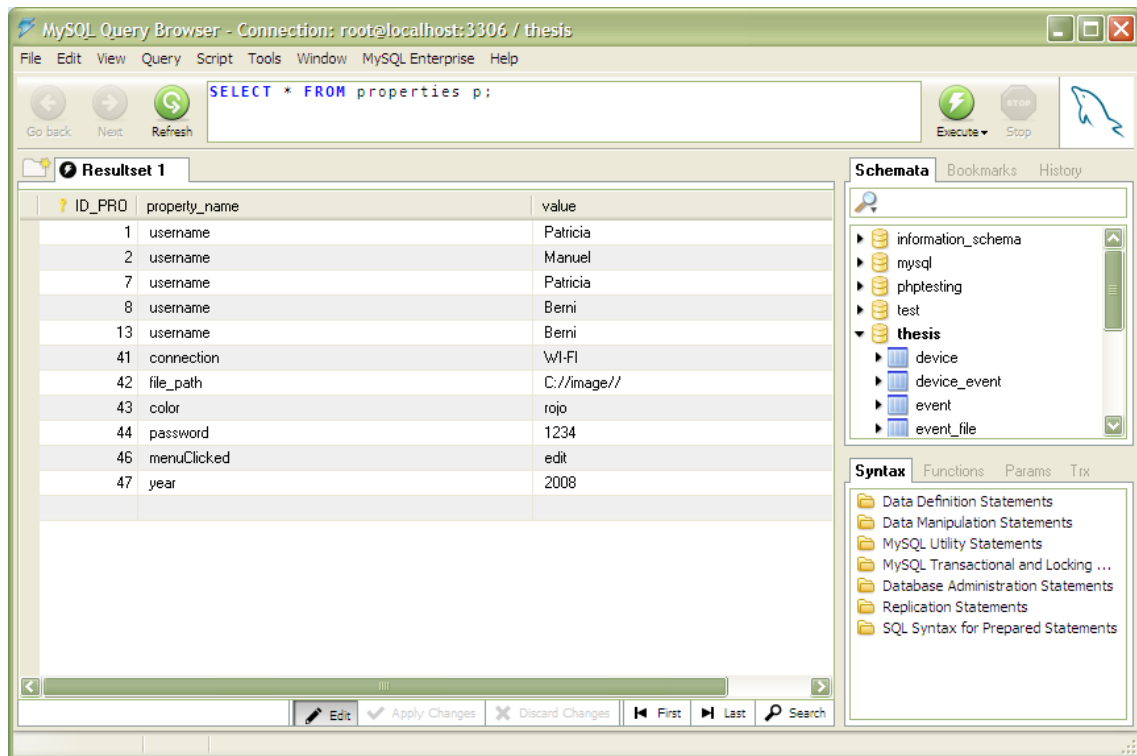


Figure 29. View of the database of properties in the server

5.1.4 Measure the time spent logging events

In the application created, it has been developed also a method to measure the time spent logging data, in the server side or locally, and with the combination of number of events, properties, etc preferred by the user.

This function has been really useful to test the tool as it will be explained in the evaluation section, because to know exactly, with a precision of milliseconds, how much time needs an event to be log is very important to analyse if the tool is enough fast for the work that has to realise.

To implement this TimeStamp() method, it has been used one of the function of the RTimer class provided by the platform, called HomeTime() that return the time of the system in that moment and then, the result will be stored in a TTime variable that will be return to the code who called. Therefore, this method will be called two times, one before starting to log events and another one just after the library finished to log the events; and then, the difference between these two variables will be calculated and shown in the screen. Also, if there are several events, the medium time per event will be shown.

It was decided to use this function from the RTimer class and not another way to measure the time, because it is very easy to use and understand the result, providing also a function MicroSecondsFrom() to calculate the difference between the values of time received.

6 EVALUATION

6.1 INTRODUCTION

For the evaluation of this tool, there were used two different methods; one of them was to provide the library designed to a developer with a document explaining the instructions step by step to add the data collector tool to an application already working, and test it, recording some videos of the process and giving us feedback about it.

The other way to evaluate the data collector tool was testing the number of events and properties that it can process, and analysing the results obtained, especially about the time necessary to carry out these tasks, getting an idea of the speed of the tool.

6.2 EVALUATION MADE BY A DEVELOPER

6.2.1 Method

In this project, the method used to evaluate the data collector tool designed for Symbian phones, is the Developer Feedback, that is one of the methods of peer review.

Peer review methods include inspections, walkthroughs, peer deskchecks, and other similar activities [36].

This method is supported for Karl Wiegers, who says; “Peer review -- an activity in which people other than the author of a software deliverable examine it for defects and improvement opportunities -- is one of the most powerful software quality tools available”. He said too; “After experiencing the benefits of peer reviews for nearly fifteen years, I would never work in a team that did not perform them.”

More authors like, for example, McConnell, give many evidences for the efficiency of code reviews in Code Complete. Saying: “software testing alone has limited effectiveness -- the average defect detection rate is only 25 percent for unit testing, 35 percent for function testing, and 45 percent for integration testing. In contrast, the average effectiveness of design and code inspections is 55 and 60 percent” [37].

Some cases have been studied of peer review technique, and the results have been [37]:

- ⇒ In a software-maintenance organization, 55 percent of one-line maintenance changes were in error before code reviews were introduced. After reviews were introduced, only 2 percent of the changes were in error. When all changes were considered, 95 percent were correct the first time after reviews were introduced.

Before reviews were introduced, fewer than 20 percent were correct the first time.

- ⇒ In a group of 11 programs developed by the same group of people, the first five were developed without reviews and the other six were developed with reviews. After all, the programs were released to production; the first five ones had an average of 4.5 errors per 100 lines of code. The other six had an average of only 0.82 errors per 100. Reviews cut the errors by over 80 percent.
- ⇒ The Aetna Insurance Company found 82 percent of the errors in a program by using inspections and was able to decrease its development resources by 20 percent.
- ⇒ A study of an organization at AT&T with more than 200 people reported a 14 percent increase in productivity and a 90 percent decrease in defects after the organization introduced reviews.
- ⇒ Jet Propulsion Laboratories estimates that it saves about \$25,000 per inspection by finding and fixing defects at an early stage.

With all this positive data about this method of evaluation, it was decided that this technique is really useful for the situation of a new application tool ready to start working; so, it was used with a developer that agreed to study the data collector tool and give some feedback about it, besides showing us the process followed thanks to a recording of the steps followed by him.

The videos recorded by the developer were used lately in the explanation of the steps followed by the developer to add the library to an application and start using it. The final time to do this task, following the instructions provided to him last around 15 minutes. And then, another 5-10 minutes of tests with different capabilities of the tool.

To make easier the understanding of the steps followed, there is shown in these pages the procedure that they are doing each time, and thanks to this method the tool can be evaluated. The videos are attached to this report in a special DVD with all the information necessary to understand it, and here there are presented only some screenshots with the steps to add the tool designed in another application, which needs to be log.

6.2.2 Data Analysis

Firstly, a document was prepared to provide to the developers as an explanation of the instruction they have to follow to add the library designed to a application already working, it represents all the information about the procedure that they received some days before to read and understand it with calm.

These instructions are shown in the following pages:

INSTRUCTIONS OF USE

- Firstly, the project with the library has to be added to the same workspace were the application that the user wants to log is, and import it to the `carbide.c++` program.

- Add to the header file where the application's functions are included the reference to the library `#include "Library1.h"`. Also, link against this library editing the `mmp` file of the project that can be found in the folder *Group*, looking for the library `LIBRARY1.LIB` and save with the new changes.

- Be sure that this header file includes also the following libraries provided by the platform:

```
#include <e32std.h> // Default included when a project is create
#include <e32base.h> // Default included when a project is create
#include <eikenv.h> //Necessary to receive information in the screen from the program.
If is not included, link against its library also in the previous mmp. file, the name of its library is avkon.lib.
```

- Create in the header file of the application the following variables:

- A public variable type *TInt* called *Configuration*, to send the type of configuration that the user wants to the library.
- A public variable type *TBool* called *Close*, to be used to check if an object of the library has been already created.
- A public variable type *TInt* called *NEvents*, to take the account of the number of events to log.
- A public variable that will store an object of the library. The syntax is: `CLibrary1* EventLogger;` Where *EventLogger* is the name that will be used in the application to refer the object of the library.

- Add the collection of properties to each event that you want to log, or we can suppose that each function already has it, and these properties have to be stored with the name of the property and its value in a class *Property* already defined in the library, so it is just necessary to write the name and value as the following example:

```
Property a (_L("property_name"),_L("property_value"));
```

And only change the bold variables, the name of the property (for example, instead of *a*, continue the alphabet *b c d...*), and the name of the property and its value, for example *property_name* could be *Username* and *property_value* could be *Patricia*. Note: The name of the property and its value are limited to a string of 50 characters.

- Now, store this information in an array to send it to the library. It will be declared in each function to be logged, and the type of the array is *RArray*, defined as follows:

```
RArray<Property> PropertyArray; //PropertyArray is the name gave to the array
```

To include the properties in the array, just after adding the name and the value of the property in a class *Property*, as it was explained above, use the following expression:

```
User::LeaveIfError(PropertyArray.Append((Property)a));
```

- Then add the following two functions to the class that contains the application, because these functions implement the creation of an object of the library and call the methods from the library.

```
void CApplication_TestEngine::SetConfiguration(TInt NEvent)
{
    NEvents=NEvent;
    //Set the appropriate configuration to 1 if the user wants to log the events locally, to 2 if he
    //wants to log it in the server and 3 if he wants to log it in both places.
    Configuration=3;
    Close=ETrue;
}

void CApplication_TestEngine::LogEvents(RArray<Property> PropertyArray)
{
    if(Close) //If the object is not created yet
    {
        //Create an object of the library with the variable EventLogger defined in the library
        EventLogger = CLibrary1::NewL(Configuration);
        Close=EFalse; //To indicate that the object is created
    }
    if (Configuration==2 || Configuration==3) // Log in the server
    {
        if (NEvents==1) //Only log one event on the library, if there are several, only the last one.
        {
            //Local variables
            HBufC8 *postbuf = HBufC8::NewLC(512);
            HBufC8 *postbuf2 = HBufC8::NewLC(512);
            TBuf<256> iUri;
            TBool File; //Use to set if the user wants to send a file to server or not

            iUri.Zero(); //Initialise iUri to zero.
            //Add the url of the server where you have to send the data
            iUri.Copy( _L("http://150.140.188.170/formhandler.php") );
            postbuf->Des().Zero();
            postbuf->Des().Append(iUri); //Prepare the url in this format to send it to the library
            File=EFalse; //No file

            //This part must be added to the code only if the user wants to send a file to the server if not,
            //comment it
            File=ETrue; //File
            //Add the name of the file to the array of properties to log the file in the server as file_name.
            // In this example the file to send is Audrey.jpeg
            Property y( _L("file_name"),_L("audrey.jpeg") );
            User::LeaveIfError(PropertyArray.Append((Property)y));
            TBuf<256> iFileNameToSend;
            iFileNameToSend.Zero(); //Initialise iFileName to zero.
            //Add here the name of the file with the path were it is stored.
            iFileNameToSend.Copy( _L("C:\\image\\audrey.jpeg") ); //The file to
send
```

```
postbuf2->Des().Zero();
postbuf2->Des().Append(iFileNameToSend); //Format to send it to the library
//End of the necessary code to send a file

//It is necessary to send the device ID to the server to log the data, add it to the array
//The IMEI is the unique identifier of a device, so use it as ID here.
//Note: The IMEI can be find easily in the documentation of your device.

Property x (_L("device"),_L("353269015116852"));
    User::LeaveIfError(PropertyArray.Append((Property)x));
//Call the function ConvertDatatoSend from the library to log the event
EventLogger->ConvertDatatoSend(&PropertyArray, postbuf->Des(),
postbuf2->Des(), File);
    _LIT(KTextPrueba, "The event has been logged ");
    _LIT(KTextPrueba2, "in the server ");
    CEikonEnv::InfoWinL(KTextPrueba,KTextPrueba2);
}
}
if (Configuration==1 || Configuration==3) // Log in the device
{
//Send to the function in the library that Edit the XML file
EventLogger->LogProperties(&PropertyArray);

    if (NEvents==1) //If it is the last event
    {
//Finish the file that we are editing to log events when the last one is logged.
EventLogger->FinishFile();
    _LIT(KTextPrueba3, "The event has been logged ");
    _LIT(KTextPrueba4, "locally");
    CEikonEnv::InfoWinL(KTextPrueba3,KTextPrueba4);
    }
}
NEvents=NEvents-1;
//Delete the array
PropertyArray.Close();
}
```

- Add also the declaration of these functions in the header file of this class, and declare it as public. The functions must be declared as follows:

```
void SetConfiguration(TInt NEvent);
void LogEvents(RArray<Property> PropertyArray);
```

- Finally, call the SetConfiguration() function and then, LogEvents() function from each event that wants to be log after storing the properties of the event in the array, and pass this array as parameter to the function. The sentences to call these functions are:

```
SetConfiguration(NEvent);

LogEvents(PropertyArray);
```

- Now, the user can start logging events thanks to the library.

After following these instructions to add the tool to an application used to create, edit and delete files, while recording all the steps; the developer had to give some feedback about this process, focusing in the list of quality dimensions show below, that was received at the beginning of the creation process to focus on in for the development of the tool. This list of dimensions is:

- Speed of execution
- Flexibility
- Brevity of code
- Fullness of features
- Time spent coding
- Robustness

The main goals during the development have been the flexibility, the quantity of features and the speed of execution, followed by the rest of parameters shown. It can be remarked that some of these parameters are kind of contradictory as, for example, robustness and flexibility or the fullness of features and the time spent coding.

In the following sections there are explained this steps of the evaluation done by the developer to test the data collector tool designed.

6.2.3 Video recorded by the developer

In this section there are attached some screenshots taken from the video recorder by the developer, that will help to the reader to understand the process followed to add the data collector tool to an application already existing.

In this first Figure 30, it is shown the application that has been used for the test, imported to the workspace of application where the developer is working with.

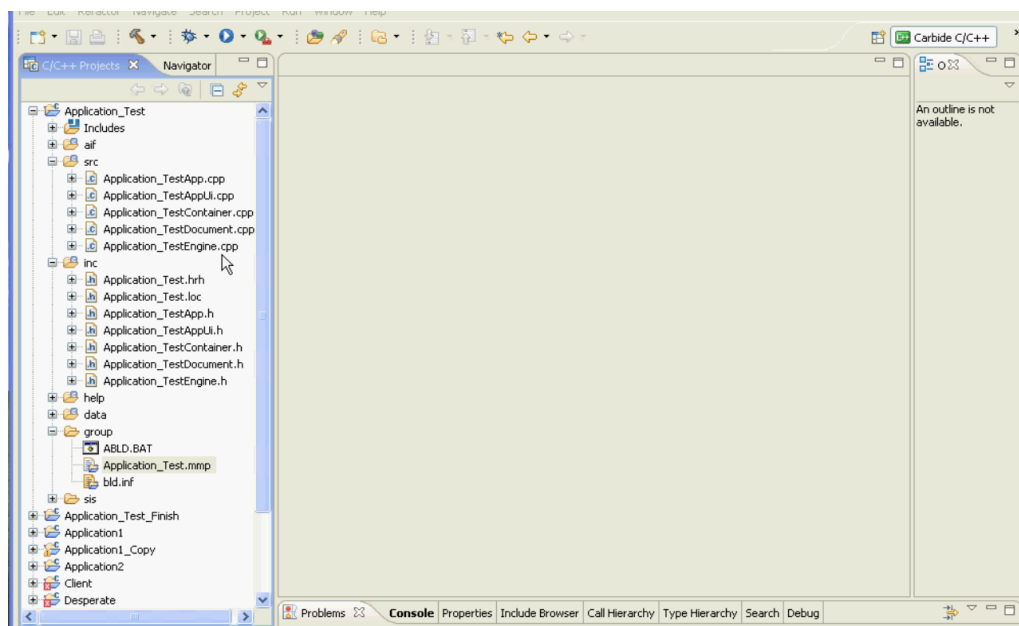


Figure 30. Screenshot with the application imported to the program

Then, the library is referred in the application to allow its use on it, and call its functions. This step is shown in the

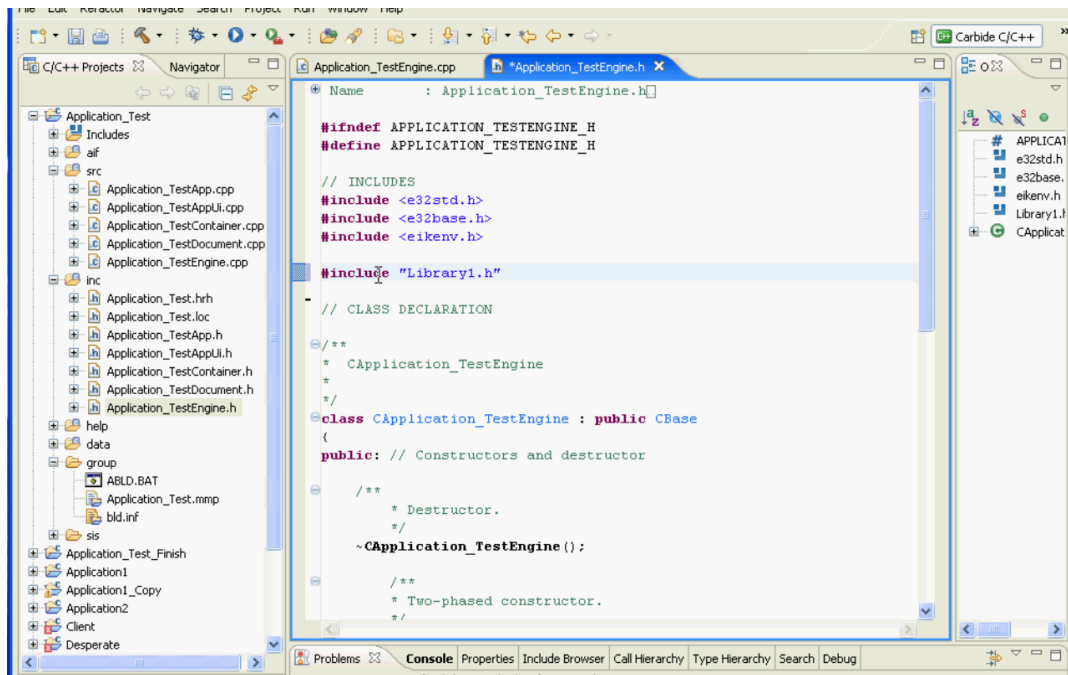


Figure 31. Screenshot of the Library referred in the header file of the application

After that, it is necessary to link against this library also, from the mmp file of the application, where it has to be added the Library.lib file:

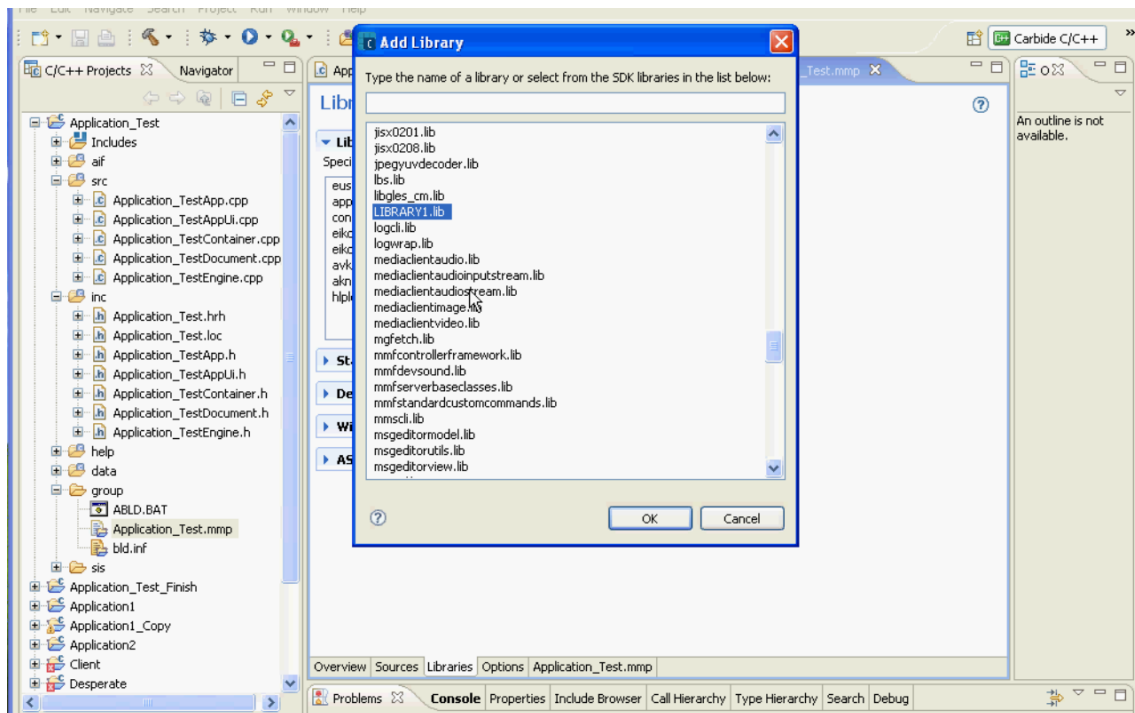


Figure 32. Screenshot of adding the library to the mmp file

After that, the code correspondent to the functions `SetConfiguration()` and `LogProperties()`, necessities to call the method of the library are included in the application, below the functions that are going to be log.

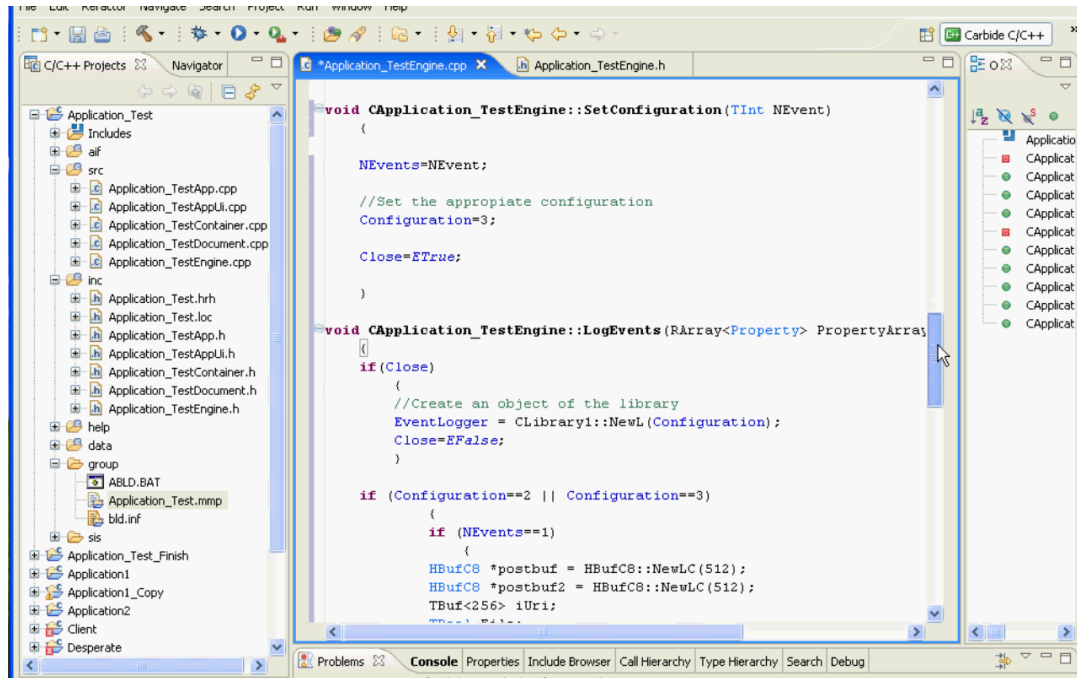


Figure 33. Screenshot of the application once the functions are included

Also, there is important for the operation that these functions are included its definition in the header file of this application:

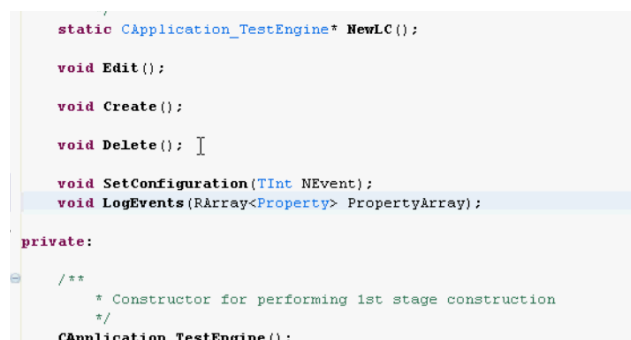


Figure 34. Screenshot of defining functions in header file

To continue the process, the developer has to add the name and value of all properties of the event that has to be logged in the array of properties `PropertyArray` to send it to the library, as it is shown in the Figure 35. Also, it is representing the sentence to call the functions `SetConfiguration()` and `LogEvents()` that interact with the library.

```
void Application_TestEngine::Edit ()  
{  
    //The function will do its tasks to for example open a file and E  
    _LIT(KTextPrueba5, "The method Edit ");  
    _LIT(KTextPrueba6, "has been selected");  
    CEikonEnv::InfoWinL (KTextPrueba5, KTextPrueba6);  
  
    Property a (_L("Username"), _L("Tester"));  
    Property b (_L("MenuClicked"), _L("Edit"));  
  
    RArray<Property> PropertyArray;  
  
    User::LeaveIfError (PropertyArray.Append((Property) a));  
    User::LeaveIfError (PropertyArray.Append((Property) b));  
  
    SetConfiguration(1);  
    LogEvents (PropertyArray);  
}
```

Figure 35. Screenshot of adding properties of the event to the array

Until now, all the elements has been added to the application, so in this moment the project has to be built, and if there is not any problem, the emulator can be used to run the application.

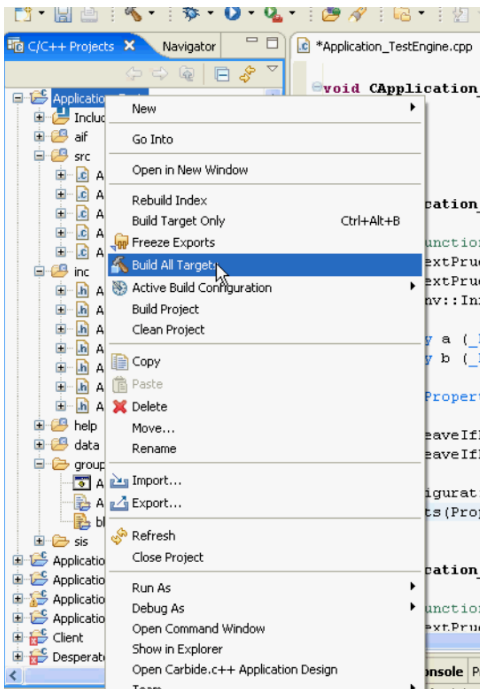


Figure 36. Building the application

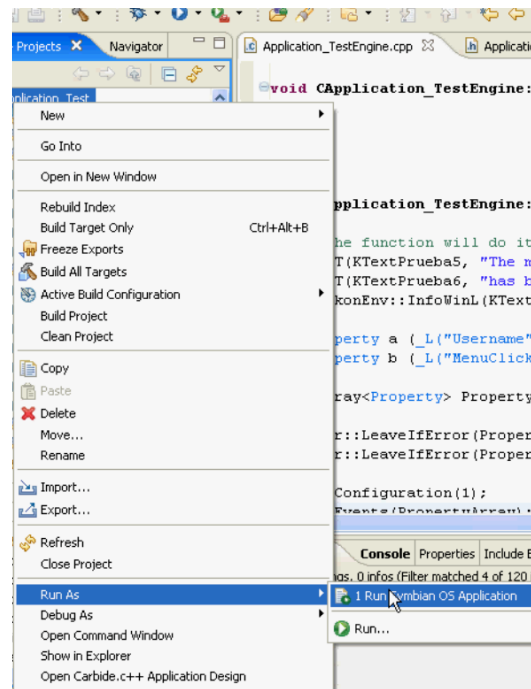


Figure 37. Running the application

Then, from the menu of the program, the event to do and log it is selected and when it is logged a message appear in the screen to inform the user (Figure 38).

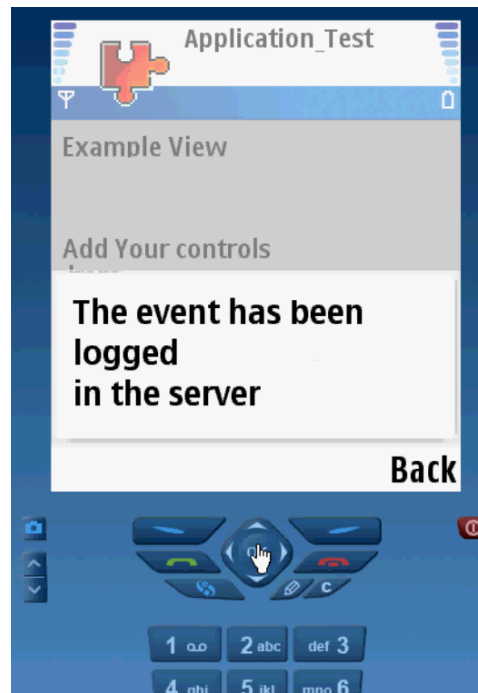


Figure 38. Message of event logged succesfully

6.2.4 Feedback from the developer

The main point for this feedback from the developer is to fulfill the list of quality dimensions presented next to the instructions with his objective mark due to the test realized with the tool. The evaluation should be with a scale from 1 to 5, been the 5 mark the best one.

After that, as a good feedback, it includes a list to positive and negative characteristic about the tool that should be taken in consideration. And it was already discussed between the developer of the tool and the developer doing the evaluation, to get a list of face-to-face impressions about the point of view of both of them.

- Evaluation of the list of quality dimensions provided -

After the testing, the evaluator fulfill this list of dimensions (in a scale 1 to 5) as follows:

Speed of execution: 4 (the information obtained about the time spent logging events, in the device as in the server, is satisfactory but it cannot be compared with other tools carrying out the same tasks)

Flexibility: 4 (the number of properties to log is big, and include also the time when an event is logged, and it can include the file logging in server, but it do not have the option to take a screenshot)

Brevity of code: **2** (the code to add it is more than the expected, but the modifications on the code are easy to do)

Fullness of features: **3** (the library has a good variety of features, and a versatile configuration but it is necessary to reload the application each time the configuration is changed)

Robustness: **3** (problems of network bottleneck)

Time spent coding: **1** (the time coding was more than expected but primarily due to unfamiliarity with Carbide.c++)

- Positive characteristics -

- Support of local logging of the data, not only in the server side.
- Good points of flexibility that allows logging a single event with variable number of properties and optionally a file. Also could be logged several events at the same time in the same log file on the device.
- The events logged include a time stamp of the moment when were logged, with milliseconds accuracy, that make easier the search of an event by time, and also could be useful for another applications.

- Negative characteristics -

- It can be useful the implementation of the singleton pattern of design instead of creating a global variable to control if an object of the library has been already created.
- It will be easier to add the library to an existing application to do more calls to methods in the library, instead of including functions in the application. It would seem tidier.
- Network failures will probably finish with data loss or wasted of time in the execution.
- Changes in configuration of use of the library or server used to log requires rebuilt of the application using the library. An external configuration file would be more acceptable.

6.3 PERFORMANCE EVALUATION

6.3.1 Method

The method used in this case consists on carry out a group of tests with the tool designed, implemented in an application, and in this different tests a group of parameters were changed, as logging in the own device or in the server, different number of events logged, different number of properties per event, and including or not a file between this data to be logged.

In these different tests, it was taken in consideration the time necessary to log the data, and with all the time obtained, there was prepared some tables and graphs, that were analysed to create an idea to the reader of the speed and capability of the application.

6.3.2 Data Analysis and presentation of results

The first experiments done logging events locally, in the own device, and we were changing the number of events to log and the number of properties per event. Starting with a small number of properties and increasing the number of events, because we were interested in testing the capability of events to log in a file, after that the number of properties was incremented also, but especially tests were focus in number of events to log.

As it is shown in the Figure 39, the tool was tested for a number of events of 100, and the tool needed approximately 280 millisecond to log all of them; if the number of properties is double, 6 properties per event, this time increased to around 485 milliseconds, and for a number of 12 properties this time is increased until 812 milliseconds.

If the number of events to log is bigger, these numbers are bigger also. Logging 500 events, the tool need around 1 second to achieve this task with the smaller number of properties, if the properties increase until for example 12 properties per event, the tool needs around 3 second to log all the data. Increasing the number of events to 1000, for a small number of properties will last 2 seconds to log it, and with a bigger number of properties this number will be increased proportionately as in the other experiments.

This results let us interpret that the time needed for the tool to create a new file were the data is logged is reduce, compared with the time necessary to write on it the information. Because if we take a look at the graphics, each time the number of properties is doubled, the time necessary to complete the operation is almost doubles too. So we can deduced that most of the time for this data logging is spent to write on the file, not to create it and finish it.

Besides, it was considered not necessary to show more results for this operation because it was checked that these numbers continuing increasing in the same way if the

number of events is increased and the number of properties is bigger too. Moreover, the system does not crash if the number of events is too big, the only problem is that, for example, with a number of 100000 events to log it will need around 2 minutes to complete the operation and the size of the file created will be around 20MB, and it is considered too heavy for a mobile phone memory.

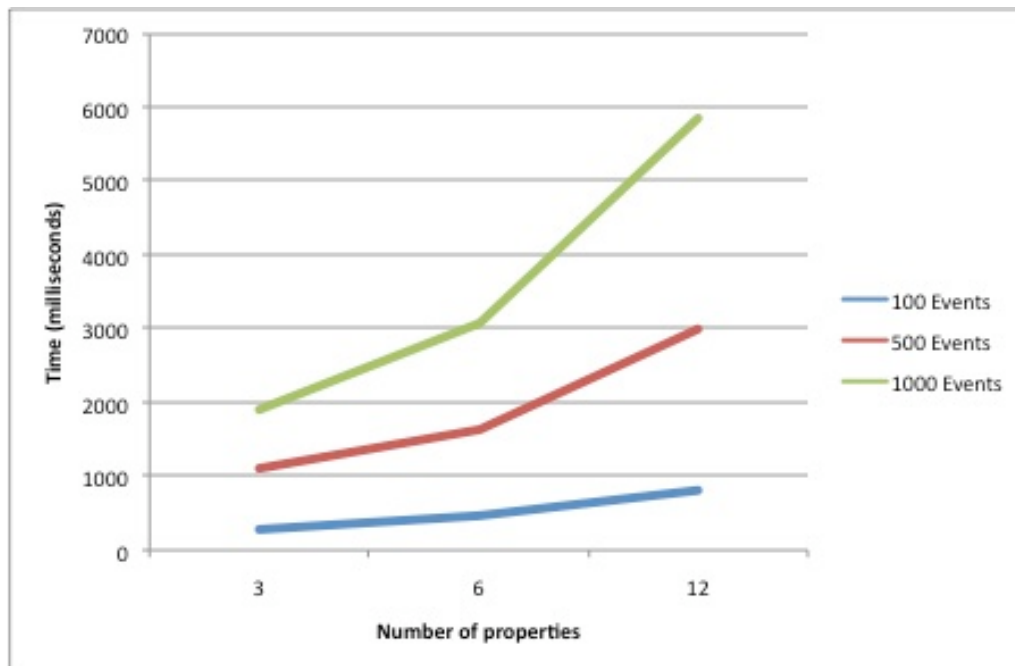


Figure 39. Time spent logging events locally

Next, the experiments were done logging in the server side. In this case, the parameters changed were the number of properties to log in an event and the number of files logged. The number of events is fixed to one event because per time it is only possible to log one event in the server as the library is designed; in contrast, the number of properties can be bigger so it was decided to do some test increasing the number of properties to check what happens, and then include also pictures in the events to log.

In the first tests, it was send to the server only properties, starting from a small number and increasing it until one hundred properties, for higher values the system crashes because the tool was implemented to send to the server a limited size content of properties, set in one hundred properties with 50 characters for the name and 50 characters for its value.

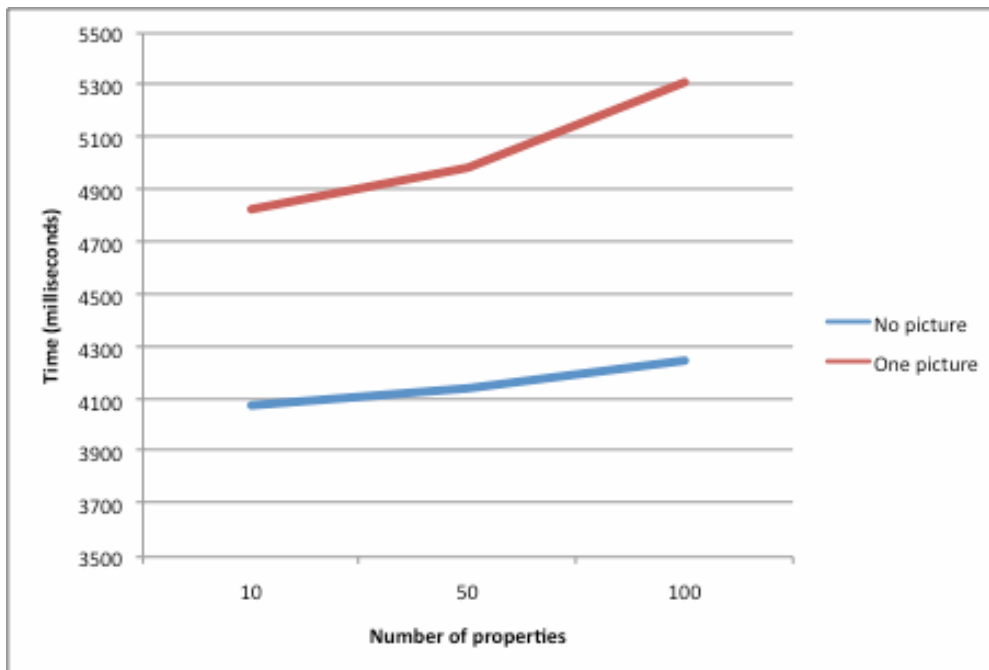


Figure 40. Time spent logging events in the server

If we analyze this results, for example, in the first situation, were the data to log contains only properties, it can be seen in the Figure 40, that the difference between logging 10 or 100 properties is not very high, in the order of 200 milliseconds, and the complete operation lasts about 4 seconds; therefore, it can be deduced that the most of the time is spent in the other operations done to log in the server, most probably this time is spent creating a connection and sending this data to the server through Internet, because in this situation the time wasted does not depend only in the system designed, it also depends in the traffic and the capabilities of the network.

Including also files in the transaction, the time spent to log is incremented but proportionately, so the conclusion explained in the previous paragraph makes sense, still needing to send also pictures that will increase the traffic on the network and increase the time of the logging because of the size of this files.

7 CONCLUSION

In the present moment, the spreading of the new technologies around the world is a really important source of business, and each day more and more programs, devices and applications are launch to the market. In this context, the tools related to test new programs gathering data are been developed quickly as a good opportunity to discover fails and negative points of new applications created, and in this way, fix this problems on the first stages of the creating process, in order to get good programs working correctly in less time and getting a better result between costumers.

Besides, it is necessary to take in consideration, that this big spreading of technologies causes a lot of new users, usually not accustomed to treat with this devices, so it is needed to make the new application easy to use, in order to facilitate the understanding of the operation mode of them. In this situation, it is when appear the Human-Computer Interaction field, trying analyse the behaviour of different groups of people, and in this way, get applications well accepted between common people, not only between specialized person on the subject.

Analyzing the techniques to improve applications designed and starting to be developed, the method of Usability Evaluation of the Data Collected is one of the most popular between analyzers, due to its number of possibilities and its big amount of advantages of use. For this reason, this is a very active field of work nowadays, and several researches and studies about it are in the market.

Between all the technologies existing these days in the market to gather data and send it to another entity, as a server, to be stored in; we realised that there is not an application valid for every system, because it would have to include a big amount of features and requirements for all the programming languages and platform existing. So, for the development of this project it was selected the Symbian Operating System for mobile devices, due to its huge expansion in the world nowadays.

The application of a data collector tool for Symbian phones proposed, it is designed and implemented, and it accomplished the specifications to collect data needed for this project that the Human-Computer Interaction group is developing. But, as it has been remarked on the evaluation done to this tool, it can be improved, fixing the fails in the system that the evaluator has shown and trying to solve the negative points found. A good idea would also to test it between a bigger number of people, to even improve more the Data Collector tool.

Besides, the use of this tool for the system that proposed the initial requirements, once has been implemented and it is working correctly, it could be useful to open it for more applications that wants to log data from Symbian phones, and thereby the market for the tool would be extended for more applications. Even, as future work, it will be great the possibility to implement this tool to other devices, that use Symbian OS as operating system.

As a future work in this field, it would be a good idea to solve the negative points of the tool marked by the developer during its evaluation, as making easier the work to add the library to an application already working in order to facilitate it for the future users of the library, or the modification of the code to add the singleton pattern of design to control the creation of objects of the library created by the application instead of control it by a variable.

On the other hand, the Data Collector tool explained in this report could be modified, as also the Data Collector for Pocket PCs created by Raúl Esgueva, in order to standardize its interfaces to communicate with the programmer, because the characteristics of both of them at the moment are not exactly similar, due to the difference on time when these tools were developed and the difference between the programming languages used. But as a currently good point, both tools use the same interface with the server, so the same server can receive logging data from these devices.

8 APPENDIXES

8.1 APPENDIX A: EVALUATION WITH SLOCCOUNT BASED IN COCOMO MODEL

SLOCCount is a free tool created with the objective to analyze software projects, that it is based in the COCOMO model, an algorithmic of Software Cost Estimation Model developed by Barry Boehm. This tool counts the number of lines of code of the application that needs to be evaluated, even if different languages compose it, doing an estimation of cost, length and number of developers.

For the Data Collector Tool created, the results obtained from this method of evaluation show us that for the development of the tool would be necessary almost 3 months and a half of work from a developer, spending the 65 percent of the working hours on this task, including the coding time and some additional time for researching additional information.

Next, there are represented the complete results obtained using this method of evaluation:

Computing results:

SLOC	Directory	SLOC-by-Language (Sorted)
526	Library1	cpp=438,ansic=88
397	Application1	cpp=397
0	top_dir	(none)

Totals grouped by language (dominant language first):

cpp:	835 (90.47%)
ansic:	88 (9.53%)

Total Physical Source Lines of Code (SLOC)	= 923
Development Effort Estimate, Person-Years (Person-Months)	= 0.18 (2.21)
(Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{1.05})$)	
Schedule Estimate, Years (Months)	= 0.28 (3.38)
(Basic COCOMO model, Months = $2.5 * (person-months^{0.38})$)	
Estimated Average Number of Developers (Effort/Schedule)	= 0.65
Total Estimated Cost to Develop	= \$ 24,837

(Average salary = \$56,286/year, overhead = 2.40).

SLOCCount, Copyright (C) 2001-2004 David A. Wheeler

SLOCCount is Open Source Software/Free Software, licensed under the GNU GPL.

SLOCCount comes with ABSOLUTELY NO WARRANTY, and you are welcome to redistribute it under certain conditions as specified by the GNU GPL license; see the documentation for details.

Please credit this data as "generated using David A. Wheeler's 'SLOCCount'."

- What is COCOMO? -

COCOMO was first published in 1981, in the book *Software engineering economics*, from Barry J. Boehm's, as a model for estimating effort, cost, and schedule for software projects. For its development, it drew on a study of 63 projects at TRW Aerospace where Barry Boehm was Director of Software Research and Technology that year. The study examined projects ranging in size from 2000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I, a huge variety of them; after the study of all this project, it carried out a statistic to estimate the variables that he was interested in: effort, cost and schedule.

REFERENCES

- [1] N. Eagle, “Machine Perception and Learning of Complex Social Systems”, Ph.D. Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, June 2005.
- [2] ACM SIGCHI (Association for Computing Machinery Special Interest Group on Computer-Human Interaction) - Curricula for Human-Computer Interaction. Available online at http://sigchi.org/cdg/cdg2.html#2_1 . Site last visited: April 2008.
- [3] Kirakowski, J. and Corbett, M. SUMI, “The software usability measurements inventory.” *British Journal of Educational Technology* 24, 3 (1993), 210-212.
- [4] HCI: Human-Computer Interaction. Available online at <http://recuperacioninformacion-interfazhci.iespana.es/>. Site last visited: April 2008.
- [5] Kjeldskov, J. and Paay, J., “Indexical Interaction Design for Context-Aware Mobile Computer Systems.” *Proceedings of Workshop on Context in Mobile HCI, Mobile HCI 2005, Salzburg, Austria, ACM, pp. 23-30.*
- [6] Naur, P. Intuition in software development, in H. Ehring et al. “Formal Methods and Software Development,” Vol. 2, Berlin, 1985. Also in Naur, P. “Computing: A Human Activity.” ACM Press/Addison-Wesley, New York, 1992, 449-466.
- [7] Rasmussen, J. “Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models.” *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*, 3 (1983), 257-266.
- [8] PDA Data Collection: A Quick Overview. Available online at http://www.lokilogic.com/pda_data_collection.htm. Site last visited: May 2008.
- [9] Molich, R., Bevan, N., Butler, S., Curson, I., Kindlund, E., Kirakowski, J. and Miller, D., 1998, “Comparative evaluation of usability tests.” Usability Professionals Association 1998 Conference, 22 – 26 June 1998.
- [10] Molich, Rolf and Dumas, Joseph “Comparative usability evaluation (CUE-4)”, *Behaviour & Information Technology*, 1 – 19. 2006.
- [11] Johnson, P., “Usability and Mobility; Interactions on the move”. In *Proceedings of the First Workshop on Human-Computer Interaction with Mobile Devices, Glasgow, Scotland, GIST Technical Report G98-1.*
- [12] Tamminen, S., Oulasvirta, A., Toiskallio, K., & Kankainen, A. “Understanding mobile contexts.” *Special Issue of Journal of Personal and Ubiquitous Computing*. 8, 135-143. 2004.

- [13] Kjeldskov, J., and Graham, C. “A Review of Mobile HCI Research methods.” In Proceedings of the 5th International Mobile HCI 2003 Conference, Udine, Italy, Springer-Verlag. 2003.
- [14] Indexical Interaction Design (IID). Available online at <http://www.cs.aau.dk/~jesper/iid/index.html>. Site last visited: May 2008.
- [15] Jesper Kjeldskov, Mikael B. Skov and Jan Stage, “Instant Data Analysis: Conducting Usability Evaluations in a Day”, Department of Computer Science, Aalborg University.
- [16] Affinity diagramming. Available online at <http://www.usabilitynet.org/tools/affinity.htm>. Site last visited: March 2008.
- [17] Think aloud protocol. Available online at http://en.wikipedia.org/wiki/Think_aloud_protocol. Site last visited: February 2008.
- [18] How to Conduct a Heuristic Evaluation. Available online at http://www.useit.com/papers/heuristic/heuristic_evaluation.html. Site last visited: March 2008.
- [19] Kjeldskov J., Skov M. B., Als B. S. and Høegh R. T. “Is it Worth the Hassle?” Exploring the Added Value of Evaluating the Usability of Context-Aware Mobile Systems in the Field . In Proceedings MobileHCI conference, Glasgow, UK. Springer- Verlag, 2004. 61-73.
- [20] Roto, V. Oulasvirta, A. Haikarainen, T. Lehmuskallio, H. & Nyysönen, T. “Examining mobile phone use in the wild with quasi-experimentation.” HIIT Technical Report 2004-1, August 13.
- [21] Nielsen J. and Landauer T.K. “A Mathematical Model of Finding Usability Problems.” In proceedings of INTERCHI '93. 206-213. 1993.
- [22] Device details – Nokia N70. Available online at <http://www.forum.nokia.com/devices/N70>. Site last visited: March 2008.
- [23] S. Babin, “Developing Software for Symbian OS: An Introduction to Creating Smartphone Applications in C++”, John Wiley and Sons, Ltd. 2006.
- [24] J. Stichbury, “Symbian OS Explained: Effective C++ Programming for Smartphones”, John Wiley and Sons, Ltd. 2006.
- [25] LiMo Foundation. Available online at <http://www.limofoundation.org>. Site last visited: April 2008.
- [26] Wikipedia – The free encyclopedia. Available online at <http://en.wikipedia.org>. Site last visited: June 2008.

- [27] The Symbian Investor – Symbian History. Available online at <http://www.metalgrass.com/symbianinvestor/SymbHist.html>. Site last visited: April 2008.
- [28] All Nokia tools and Nokia SDKs for developers. Available online at <http://www.forum.nokia.com/>. Site last visited: June 2008.
- [29] Technology news – CNET News.com. Available online at <http://news.cnet.com/>. Site last visited: June 2008.
- [30] Android. Available online at <http://code.google.com/android/>. Site last visited: June 2008.
- [31] Apple. Available online at <http://www.apple.com/>. Site last visited: May 2008.
- [32] InformationWeek – Analysis: How Smartphone platforms compare. Available online at http://www.informationweek.com/1122/ID_chart.jhtml. Site last visited: April 2008.
- [33] Access Company. Available online at <http://www.access-company.com/products/garnet/>. Site last visited: April 2008.
- [34] HowStuffWorks “Smartphones operating systems”. Available online at <http://communication.howstuffworks.com/smartphone2.htm>. Site last visited: April 2008.
- [35] LiMo (Linux for Mobiles) is Ready to Go Prime Time - GigaOM. Available online at <http://gigaom.com/2007/11/05/limo-2/>. Site last visited: May 2008.
- [36] What’s S60. Available online at <http://www.s60.com/business/whatss60/softwareversions>. Site last visited: June 2008.
- [37] R. Esgueva, “A networked usage data logger for Pocket PC”, Final Thesis, Human-Computer Interaction Department, Panepistimio Patron, May 2008.
- [38] S. Hernandez, “Design and implementation of a matrix-code reader application suitable for mobile devices”, Final Thesis, Human-Computer Interaction Department, Panepistimio Patron, March 2007.
- [39] A. Jakl, “Symbian OS: Quickstart and Carbide.c++ UI-Desing”. University of Applied Sciences in Hagenberg, Austria, 2007.
- [40] Hypertext Transfer Protocol - HTTP/1.1. Available online at: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Site last visited June 2008.
- [40] K. Wiegens, Peer Reviews, “Software: A Practical Guide”. Addison-Wesley Information Technology Series.
- [41] S. McConnell, “Code Complete”. Microsoft Press.