

Software development.

Practical case

At BK Programming everyone has already returned from their vacation.

A hectic September awaits them, as they have just received a request from a hotel chain to develop a software project.

Ada, the project supervisor at BK Programming, meets with Juan and María (company workers) to start planning the project.

Ana, whose specialty is graphic design of web pages, has just finished the Intermediate Degree Cycle in Microcomputer Systems and Networks and completed the FCT in BK Programming. She works in the company helping with the designs, and although she is happy with her work, she would like to actively participate in all phases of the project. The problem is that she lacks the necessary knowledge.

Antonio has learned about the possibility of studying the new Higher Degree Cycle in Multiplatform Application Design remotely, and is willing to do so. (She wouldn't have to leave work.) She tells her plans to her friend Antonio (who has basic computer skills), and he joins her.

After all... what do they have to lose?

1.- Software and program. Types of software.

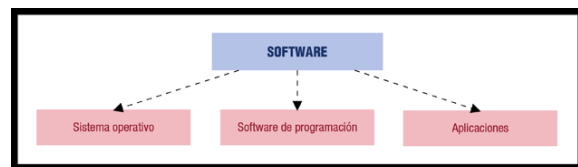
Practical case

Everyone in the company is excited about the project at hand. They know that the most important thing is to plan everything in advance and choose the most appropriate type of software. Ana listens to them talk and doesn't understand why they talk about "types of software." Wasn't the software just the logical part of the computer? What are the types of software?

It is well known that the computer is made up of two very different parts: [hardware](#) and [software](#).

Software is the set of computer programs that act on the hardware to execute what the user wants.

According to their function, three types of software are distinguished: operating system, programming software and applications.

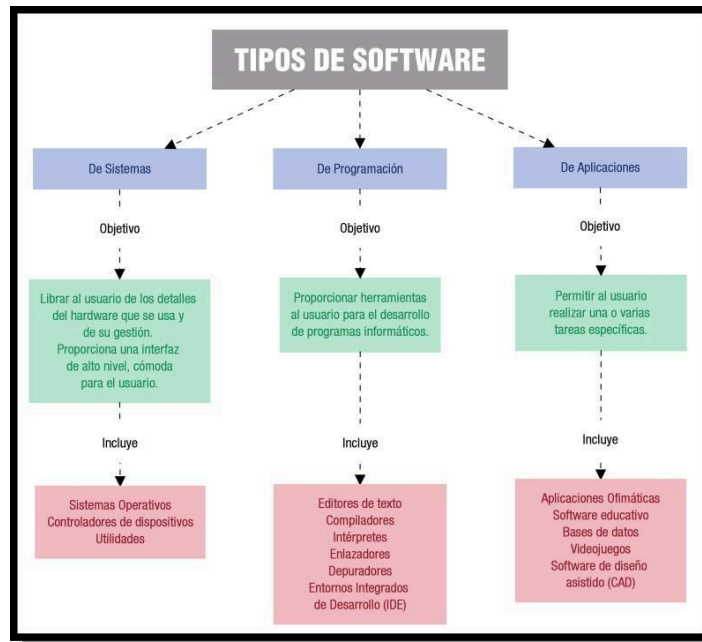


The operating system is the base software that must be installed and configured on our computer so that the applications can run and function. Examples of operating systems are: [Windows](#), [Linux](#), [macOS](#) ...

Programming software is the set of tools that allow us to develop computer programs, and computer applications are a set of programs that have a more or less specific purpose. Examples of applications are: a word processor, a spreadsheet, software for playing music, a video game, etc.

In turn, a program is a set of instructions written in a programming language.

In short, we distinguish the following types of software:



In this topic, our interest focuses on computer applications: how they are developed and what phases they must necessarily go through.

Throughout this first unit you will learn the fundamental software concepts and the phases of the so-called life cycle of a computer application.

You will also learn to distinguish the different programming languages and the processes that occur until the program works and performs the desired action.

To know more

In the following link you will find more information on the types of existing software, as well as examples of each one that will help you delve deeper into the topic.

<http://www.tiposdesoftware.com/>

Reflect

There are several operating systems on the market: Linux, Windows, Mac OS The best known is Windows. Despite that, why do we use Linux more and more?

2.- Hardware-software relationship.

Practical case

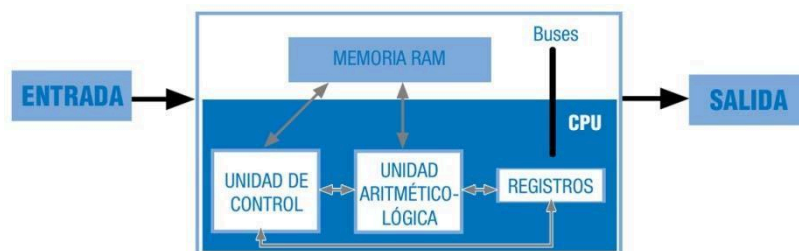
After already knowing how to differentiate the different types of software, Ana poses another question: The software, whatever its type, runs on the physical devices of the computer. What is the relationship between them?

As we know, the set of physical devices that make up a computer is called hardware.

There is an inextricable relationship between it and the software, since they need to be installed and configured correctly for the equipment to work.

The software will run on the physical devices.

The first stored program hardware architecture was established in 1946 by John Von Neumann:



We can highlight this software-hardware relationship from two points of view:

a. From the point of view of the operating system

The operating system is responsible for coordinating the hardware during the operation of the computer, acting as an intermediary between it and the applications that are running at any given time.

All applications require hardware resources during their execution (runtime). [CPU](#), space in [RAM](#), treatment of [interruptions](#), management of Input/Output devices, etc.). The operating system will always be in charge of controlling all these aspects in a "hidden" way for the applications (and for the user).

b. From the application point of view

We have already said that an application is nothing more than a set of programs, and that these are written in some programming language that the computer hardware must interpret and execute.

There are many different programming languages (as we will see in due course). However, they all have something in common: being written with sentences from a language that humans can easily learn and use. On the other hand, computer hardware is only capable of interpreting electrical signals (absences or presences of voltage) which, in computing, are translated into sequences of 0 and 1 (binary code).

This makes us ask ourselves a question: How will the computer be able to "understand" something written in a language that is not its own?

As we will see throughout this unit, something will have to happen (a code translation process) for the computer to execute instructions written in a programming language.

Self appraisal

To make a computer program that runs on a computer:



Instructions must be written in binary code so that the hardware understands them.



You just need to write the program in some programming language and it runs directly.



You have to write the program in some Programming Language and have software tools that translate it into binary code.



Computer programs cannot be written: they are part of operating systems.

3.- Software development.

Practical case

In BK programming the work is already underway. Ada brings together her entire staff to develop the new project.

She knows better than anyone that it will not be easy and that we will have to go through a series of stages. Ana doesn't want to miss the meeting, she wants to discover why so many notes and hassles have to be taken before even starting.

By Software Development we understand the entire process that occurs from when an idea is conceived until a program is implemented on the computer and working.



The development process, which at first may seem like a simple task, consists of a series of mandatory steps, because only in this way can we guarantee that the programs created are insurance and they respond to the needs of the efficient, reliable,

end users (those who will use the program).

As we will see in more detail throughout the unit, software development is a process that involves a series of steps. Generally, these steps are as follows:

Stages in software development:

As we are going to see in the next point, depending on the order and way in which the stages are carried out, we will talk about different software life cycles.

Building software is a process that can become very complex and requires great coordination and discipline from the work group that develops it.

Reflection

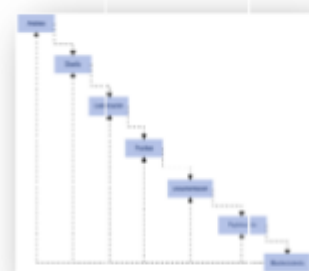
According to estimates, 26% of major software projects fail, 48% must be drastically modified, and only 26% are completely successful. The main cause of project failure is the lack of good planning of the stages and poor management of the steps to follow. Why is the failure rate so high? Why do you think these causes are so decisive?_

3.1.- Software life cycles.

We have already seen that the series of steps to follow to develop a program is what is known as the Software Life Cycle.

Each stage will be explained in more detail in the section of this unit dedicated to the phases of software development and execution.

Miscellaneous authors have raised different



life, but the most well-known and used are the ones below:

A life cycle model should always be applied to the development of any software project.

1. Waterfall Model

It is the classic software life model.

It is practically impossible to use it, since it requires knowing all the system requirements in advance. It is only applicable to small developments, since the stages pass from one to another with no possible return. (it is assumed that there will be no errors or variations in the software).

2. Feedback Cascade Model

It is one of the most used models. It comes from the previous model, but feedback is introduced between stages, so that we can go back at any time to correct, modify or debug some aspect. However, if many changes are expected during development, it is not the most suitable model.

It is the perfect model if the project is rigid (few changes, little evolutionary) and the requirements are clear.

3. Evolutionary Models

They are more modern than the previous ones. They take into account the changing and evolutionary nature of software.

We distinguish two variants:

1. Iterative Incremental Model

It is based on the waterfall model with feedback, where the phases are repeated and refined, and their improvement is propagated to the following phases.

2. Spiral Model

It is a combination of the previous model with the waterfall model. In it, the software is built repeatedly in the form of versions that are increasingly better, because they increase the functionality in each version. It is a quite complex model.

Self appraisal

If we want to build a small application, and it is expected that it will not undergo major changes during its life, would the spiral life cycle model be the most recommended?



Yeah.



No.

3.2.- Tools to support software development.

In practice, to carry out several of the stages seen in the previous point we have computer tools, whose main purpose is to automate tasks and gain reliability and time.

This will allow us to focus on system requirements and system analysis, which are the main causes of software failures.

CASE tools are a set of applications that are used in software development with the aim of reducing costs and process time, therefore improving the productivity of the process.

In what phases of the process can they help us?

In the design of the project, in the coding of our design based on its visual appearance, error detection...

Rapid application development or RAD is a software development process that involves iterative development, building prototypes, and using CASE utilities. Today it is often used to refer to the rapid development of graphical user interfaces or complete integrated development environments.

CASE technology tries to automate the phases of software development so that the quality of the process and the final result improves.

Specifically, these tools allow:

- Improve project planning.
- Give agility to the process.
- Being able to reuse parts of the software in future projects.
- Make applications respond to standards.
- Improve the task of program maintenance.
- Improve the development process, by allowing the phases to be visualized graphically.

CLASSIFICATION

Typically, CASE tools are classified based on the phases of the software life cycle in which they help:

- **UKASE**- Offers help in the planning and requirements analysis phases.
- **M-CASE**- Offers help in analysis and design.
- **L-CASE**- Helps in software programming, code error detection, program and testing debugging, and project documentation generation.

Examples of free CASE tools are: ArgoUML, Use Case Maker, ObjectBuilder...

[To know more](#)

The following link presents an expansion of the types and specific aids of the CASE tools.

<http://temariotic.wikidot.com/tema-58-boe-13-02-1996>

4.- Programming languages.

Practical case

One of the aspects of the project that worries Ana the most is the choice of the programming language to use.

You need to be very clear about the client's requirements to correctly focus the choice, because depending on these, some languages will be more effective than others.

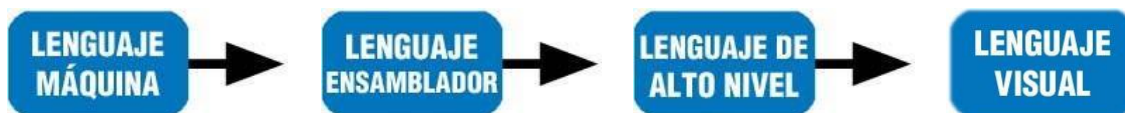
We already said earlier that computer programs are written using some programming language. Therefore, we can define a Programming Language as an artificially created language, formed by a set of symbols and rules that are applied to an alphabet to obtain a code that the computer hardware can understand and execute.

Programming languages are what allow us to communicate with computer hardware.

In other words, it is very important to be very clear about the function of programming languages. They are the instruments we have so that the computer performs the tasks we need.

There are many programming languages, each with different symbols and structures. In addition, each language is focused on programming specific tasks or areas. Therefore, the choice of the language to use in a project is a matter of extreme importance.

Programming languages have undergone their own evolution, as can be seen in the following figure:



Characteristics of Programming Languages

- **Machine language:**
 - Its instructions are combinations of ones and zeros.
 - It is the only language that the computer understands directly. (No translation needed).
 - It was the first language used.
 - It is unique for each processor (it is not portable from one computer to another).
 - Today no one programs in this language.
- **Assembly language:**
 - It replaced machine language to facilitate programming work.
 - Instead of ones and zeros, it is programmed using mnemonics (complex instructions).
 - It needs translation into machine language to be able to run.
 - Its instructions are statements that refer to the physical location of the files on the computer.
 - It is difficult to use.

- **High-level code-based language:**
 - o They replaced assembly language to make programming easier.
 - o Instead of mnemonics, sentences and commands derived from the English language are used. (Needs translation into machine language).
 - o They are closer to human reasoning.
 - o They are used today, although the trend is less and less.
- **Visual languages:**
 - o They are replacing high-level code-based languages.
 - o Instead of written statements, it is programmed graphically using the mouse and directly designing the appearance of the software.
 - o Its corresponding code is generated automatically.
 - o They need translation into machine language.
 - o They are completely portable from one computer to another.

To know more

In the following link, you will see the evolution between the different types of Programming Languages in history.

<http://www.monografias.com/trabajos38/tipos-programming-languages/types-languages-programming.shtml>

4.1.- Concept and characteristics.

We already know that programming languages have evolved, and continue to do so, always towards greater usability (that the greatest possible number of users use and exploit it).

The choice of programming language to code a program will depend on the characteristics of the problem to be solved.

CONCEPT

A programming language is the set of:

- **Alphabet:** set of allowed symbols.
- **Syntax:** permitted construction rules of language symbols.
- **Semantics:** meaning of constructions to make valid actions.

```
package calculadora;  
import junit.framework.TestCase;  
  
/**  
 * @author usuario  
 */  
public class CalculandoTest extends TestCase {  
  
    public CalculandoTest(String testName) {  
        super(testName);  
    }  
  
    @Override  
    protected void setUp() throws Exception {  
        super.setUp();  
    }  
}
```

CHARACTERISTICS

We can classify the different types of Programming Languages based on different characteristics:

- **Depending on how close it is to human language**

- o High-level Programming Languages: due to their essence, they are closer to human reasoning.
- o Low-level Programming Languages: they are closer to the internal functioning of the computer:
 - Assembly language.
 - Machine language.
- **Depending on the programming technique used:**
 - o Structured Programming Languages: They use the structured programming technique. Examples: Pascal, C, etc.
 - o Object-Oriented Programming Languages: They use the object-oriented programming technique. Examples: C++, Java, Ada, Delphi, etc.
 - o Visual Programming Languages: Based on the previous techniques, they allow graphical programming, with the corresponding code generated automatically. Examples: Visual Basic.Net, Borland Delphi, etc.

Despite the immense number of existing programming languages, Java, C, C++, PHP and Visual Basic concentrate around 60% of the interest of the global computing community.

To know more

On the following web page you will find a summary of the characteristics of the most used Programming Languages today.

<http://www.larevistainformatica.com/LENGUAJES-PROGRAMMING-list.html>

4.2.- Structured programming languages.

Although the current software requirements are much more complex than what the structured programming technique is capable of, it is necessary to at least know the bases of structured Programming Languages, since from them it evolved to other more languages and techniques. complete (event or object oriented) which are the ones currently used.

Structured programming is defined as a technique for writing programming languages that allows only the use of three types of statements or control structures:

- Sequential sentences.
- Selective (conditional) sentences.
- Repetitive sentences (iterations or loops).

Programming languages that are based on structured programming are called structured programming languages.

To know more

In the following link you will find a brief document that explains what each control statement is for with some simple examples written using the C language.

http://www.juntadeandalucia.es/educacion/adistancia/cursos/file.php/420/ED01/ED01_Web/index.html#annex_i_control_sentences_of_structured_programming.html

Structured programming was very successful because of its simplicity in building and reading programs. It was replaced by modular programming, which allowed large programs to be divided into smaller pieces (following the well-known "divide and conquer" technique). In turn, then object-oriented languages triumphed and from there to visual programming (it's always easier to program graphically than in code, don't you think?).

ADVANTAGES OF STRUCTURED PROGRAMMING

- The programs are easy to read, simple and fast.
- Program maintenance is simple.
- The structure of the program is simple and clear.

DISADVANTAGES

- The entire program is concentrated in a single block (if it becomes too large it is difficult to manage).
- It does not allow efficient reuse of code, since everything goes "in one". This is why structured programming was replaced by modular programming, where programs are coded by modules and blocks, allowing greater functionality.

Examples of structured languages: Pascal, C, Fortran.

Structured Programming evolved towards Modular Programming, which divides the program into pieces of code called modules with a specific functionality, which can be reusable.

4.3.- Object-oriented programming languages.

After understanding that structured programming is not useful when programs become very long, another programming technique is necessary to solve this problem. Thus, Object Oriented Programming (hereinafter, OOP) was born.

Object-oriented programming languages treat programs not as an ordered set of instructions (as was the case in structured programming) but as a set of objects that collaborate with each other to perform actions.

In OOP, programs are made up of mutually independent objects that collaborate to perform actions.

The objects are reusable for future projects.

Its first disadvantage is clear: it is not as intuitive as structured programming.

Despite that, around 55% of the software that companies produce is made using this technique.

Reasons:

- The code is reusable.
- If there are any errors, they are easier to locate and debug in an object than in an entire program.

Characteristics:

- The program objects will have a series of attributes that differentiate them from each other.
- A class is defined as a collection of objects with similar characteristics.
- Through so-called methods, objects communicate with others, causing a change in their state.
- Objects are, therefore, like individual and indivisible units that form the basis of this type of programming.

Main object-oriented languages: Ada, C++, VB.NET, Delphi, Java, PowerBuilder, etc.

To know more

In the following link there is a very interesting document introducing object-oriented programming, specifically, the C++ language.

<http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf>

5.- Phases in the development and execution of the software.

Practical case

In the BK meeting about the new project Ada, the supervisor, made it clear that the first and most important thing is to be clear about what we want the software to do and what tools we have: the rest would come later, because if this is not right raised, that error will propagate to all phases of the project.

-Where do we start? —asks Juan.

—REQUIREMENTS ANALYSIS—answers Ada.

We have already seen in previous points that we must choose a life cycle model for the development of our software.

Regardless of the model chosen, there are always a series of stages that we must follow to build reliable and quality software.

These stages are:

1. ANALISYS OF REQUIREMENTS.

The functional and non-functional requirements of the system are specified.

2. DESIGN.

The system is divided into parts and the function of each one is determined.

3. CODING.

A Programming Language is chosen and the programs are coded.

4. EVIDENCE.

Programs are tested for errors and debugged.

5. DOCUMENTATION.

From all stages, all information is documented and saved.

6. EXPLOITATION.

We install, configure and test the application on the client's computers.

7. MAINTENANCE.

Contact is maintained with the client to update and modify the application in the future.

Self appraisal

Do you think we should wait until a stage is completely closed to move on to the next?



Yeah.



No.

5.1.- Analysis.

This is the first phase of the project. Once finished, we move on to the next one (design).

It is the most important phase in the development of the project and everything else will depend on how well detailed it is. It is also the most complicated, since it is not automated and depends largely on the analyst who performs it.



It is the first stage of the project, the most complicated and the one that most depends on the capacity of the [analyst](#).

What is done in this phase?

The functional and non-functional requirements of the system are specified and analyzed.

Requirements:

- **Functional:** What functions the application will have to perform. What response will the application give to all inputs. How the application will behave in unexpected situations.
- **Non-functional:** Program response times, applicable legislation, treatment of simultaneous requests, etc.

The fundamental thing is good communication between the analyst and the client so that the application to be developed meets their expectations.

The culmination of this phase is the ERS (Software Requirements Specification) document.

This document specifies:

- Planning the meetings that are going to take place.
- Relationship between the objectives of the client user and the system.
- List of the functional and non-functional requirements of the system.
- List of priority objectives and timing.
- Recognition of requirements that are poorly stated or that entail contradictions, etc.

Quotes to think about

Anything that is not detected, or is misunderstood in the initial stage, will cause a strong negative impact on the requirements, propagating this degrading current throughout the entire development process and increasing its damage the later it is detected (Bell and Thayer 1976 (Davis 1993).

As an example of functional requirements, in the application for our cosmetic store clients, we would have to consider:

- If you want the products to be read using barcodes.
- If they are going to detail the purchase invoices and how they want it.
- If store workers work on commission, have information on each person's sales.
- If you are going to operate with credit cards.
- If you want control of the stock in the warehouse.
- Etc.

5.2.- Design.

Practical case

Juan is overwhelmed by the project. They have already maintained communications with the client and know exactly what the application should do. He also has a list of the hardware features of his client's equipment and all the requirements. He has so much information that he doesn't know where to start.

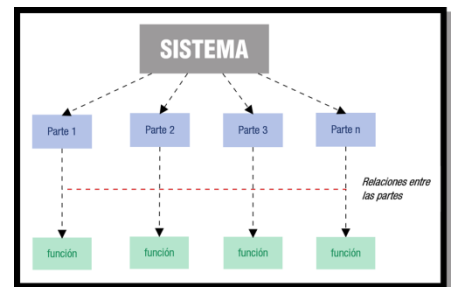
He decides to talk to Ada. Her supervisor, kind as always, suggests that she start dividing the problem into the parts involved.

—Okay, Ada, but how do I divide it?

During this phase, where we already know what to do, the next step is How to do it?

The system must be divided into parts and establish what relationships there will be between them.

Decide what exactly each part will do.



In short, we must create a functional-structural model of the requirements of the global system, to be able to divide it and address the parts separately.

At this point, important decisions must be made, such as:

- Database entities and relationships.
- Selection of the programming language to be used.
- Selection of the Database Management System.
- Etc.

Quotes to think about

Design is not just what it looks like and feels like. Design is how it works. Steve Jobs ("Design is not just what it looks like and how it looks. Design is how you work.")

Reflect

According to estimates, the organizations and companies that grow the most are the ones that invest the most money in their designs.

5.3.- Coding. Types of code.

Practical case

At BK, they already have the project divided into parts.

Now comes a key part: codifying the steps and actions to follow so that the computer can execute them. In other words, program the application. They know it won't be easy, but fortunately they have CASE tools that will be of great help. Ana would like to participate, but when talking about "source code", "executable", etc. She knows that she has no idea and that she will have no choice but to study it if she wants to collaborate on this phase of the project.

During the coding phase the programming process is carried out.

It consists of choosing a certain programming language, encoding all the above information and converting it to source code.

This task is carried out by the programmer and must exhaustively comply with all the data imposed in the analysis and design of the application.

The desirable characteristics of all code are:

1. Modularity: that it is divided into smaller pieces.
2. Correction: do what is really asked of you.
3. Easy to read: to facilitate future development and maintenance.
4. Efficiency: making good use of resources.
5. Portability: it can be implemented on any computer.

During this phase, the code goes through different states:

- **Source code:** is the one written by programmers in a text editor. It is written using some high-level programming language and contains the necessary set of instructions.
- **Object Code:** is the binary code resulting from compiling the source code.

Compilation is the one-time translation of the program, and is done using a compiler. Interpretation is the simultaneous translation and execution of the program line by line.

The object code is not directly intelligible by humans, nor by computers. It is intermediate code between the source code and the executable and only exists if the program is compiled, since if it is interpreted (line-by-line translation of the code) it is translated and executed in a single step.

- **Executable Code:** It is the binary code resulting from linking the object code files with certain [routines](#) and [libraries](#) necessary. The operating system will be in charge of loading the executable code into RAM and proceeding to execute it. It is also known as machine code and is directly intelligible by the computer.

Interpreted programs do not produce object code. The transition from source to executable is direct.

5.4.- Phases in obtaining code.

Practical case

Juan and Maria have already decided on the Programming Languages they are going to use. They know that the program they create will go through several phases before being implemented on the client's computers. All these phases will produce transformations in the code. What characteristics will the code take on as it progresses through the coding process?

5.4.1.- Source.

The source code is the set of instructions that the computer must carry out, written by programmers in some high-level language.

This set of instructions is not directly executable by the machine, but must be translated into machine language, which the computer will be able to understand and execute.

A very important aspect in this phase is the prior development of an algorithm, which we define as a set of steps to follow to obtain the solution to the problem. We designed the algorithm in pseudocode and with it, the subsequent coding to a given Programming Language will be faster and more direct.

To obtain the source code of a computer application:

1. It must be based on the previous stages of analysis and design.
2. An algorithm will be designed that symbolizes the steps to follow to solve the problem.
3. An appropriate high-level Programming Language will be chosen for the characteristics of the software to be coded.
4. The previously designed algorithm will be coded.

The culmination of obtaining source code is a document with the coding of all the modules (Each part, with a specific functionality, into which an application is divided), functions (Very small part of code with a very specific purpose.), libraries and procedures (Same as functions, but when executed they do not return any value.) necessary to code the application.

Since, as we have said before, this code is not machine intelligible, it will have to be TRANSLATED, thus obtaining an equivalent code but already translated into binary code, which is called object code. It will not be directly executable by the computer if it has been compiled.

An important aspect to consider is your license. Thus, based on it, we can distinguish two types of source code:

- Open source code. It is one that is available so that any user can study, modify or reuse it.
- Closed source code. It is the one that we do not have permission to edit.

Self appraisal

To obtain source code from all the necessary problem information:

- ☐ The most appropriate Programming Language is chosen and coded directly. It is
- ☐ coded and then the most appropriate Programming Language is chosen.



The most appropriate Programming Language is chosen, an algorithm is designed and coded.

Very good. The design of the algorithm (the steps to follow) will help us ensure that subsequent coding is done faster and has fewer errors.

5.4.2.- Object.

The object code is an intermediate code.

It is the result of translating source code into an equivalent code made up of ones and zeros that cannot yet be executed directly by the computer.

That is, it is the code resulting from the compilation of the source code.

It consists of a bytecode (Binary code resulting from the translation of high-level code that cannot yet be executed.) that is distributed in several files, each of which corresponds to each compiled source program.

Object code is only generated once the source code is free of syntactic and semantic errors.

The translation process from source code to object code can be done in two ways:

- a. **Compilation:** The translation process is carried out on the entire source code, in a single step. Object code is created that will have to be linked. The software responsible is called a compiler (Software that translates, in one go, a program written in a high-level programming language into its machine language equivalent.).
- b. **Interpretation:** The source code translation process is done line by line and executed simultaneously. There is no intermediate object code. The software responsible is called an interpreter (Software that translates, instruction by instruction, a program written in a high-level language into its machine language equivalent). The translation process is slower than in the case of compilation, but it is recommended when the programmer is inexperienced, since it gives more detailed error detection.

The object code is binary code, but it cannot be executed by the computer



To know more

In the following link you can visit a web page, which will allow you to learn more about object code generation:

<http://www.monografias.com/trabajos11/compil/compil2.shtml#co>

5.4.3.- Executable.

The executable code, the result of linking the object code files, consists of a single file that can be directly executed by the computer. You don't need any external application. This file is executed and controlled by the operating system.

To obtain a single executable file, you will have to link all the object code files, through a software called linker (Linker. Small software in charge of joining files to generate an executable program.) and thus obtain a single file that is already executable directly by the computer.

To know more

In the following link you can visit a web page, which will allow you to learn more about the generation of executables:

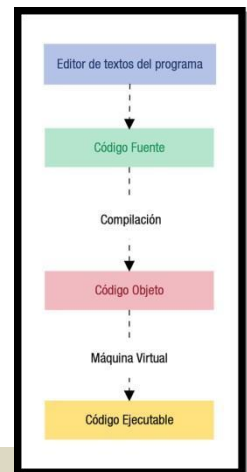
http://www.palomatica.info/juckar/sistemas/software/generacion_de_ejecutable.html

In the executable code generation scheme, we see the complete process for generating executables.

Starting with an editor, we write the source language with some programming language. (Java is used in the example.)

The source code is then compiled to obtain object code or bytecode.

That bytecode, through the virtual machine (will be seen in the next point), becomes machine code, already directly executable by the computer.



Self appraisal

Relate the code types to their most relevant characteristic, writing the number associated with the characteristic in the corresponding space.

Relationship exercise

Code type. Relationship Characteristics.

Source code	<input type="text" value="2"/>	1. Written in Machine Language but not executable.
Object Code	<input type="text" value="1"/>	2. Written in some high-level Programming Language, but not executable.
Executable Code	<input type="text" value="3"/>	3. Written in Machine Language and directly executable.

The source code written in some high-level programming language, the object written in machine language without being executable and the executable code, also written in machine language and already executable by the computer, are the different phases through which our programs go.

5.5.- Virtual machines.

A virtual machine is a special type of software whose mission is to separate the operation of the computer from the installed hardware components.

This software layer plays a very important role in the operation of programming languages, both compiled and interpreted.

With the use of virtual machines we can develop and run an application on any computer, regardless of the specific characteristics of the installed physical components. This ensures portability (Ability of a program to be executed on any physical architecture of a computer.) of the applications.

The main functions of a virtual machine are the following:

- Make applications portable.
- Reserve memory for objects that are created and free unused memory.
- Communicate with the system where the application is installed (guest), to control the hardware devices involved in the processes.
- Compliance with application security standards.

CHARACTERISTICS OF THE VIRTUAL MACHINE

When the source code is compiled, object code (bytecode, intermediate code) is obtained.

To run it on any machine, it is necessary to be independent of the specific hardware that is going to be used.

To do this, the virtual machine isolates the application from the physical details of the computer in question.

It works as a low-level software layer and acts as a bridge between the application bytecode and the physical devices in the system.

The Virtual Machine verifies all bytecode before executing it.

The Virtual Machine protects memory addresses.

The virtual machine acts as a bridge between the application and the specific hardware of the computer where it is installed.

To know more

In the following link we present the installation process of the JVM (Java Virtual Machine) and its appearance.

http://www.unabvirtual.edu.co/ayuda/manuales_pdf/maquinavirtualjava.pdf

5.5.1.- Frameworks.

A framework (Platform, environment, framework for rapid application development) is a structure to help the programmer, based on which we can develop projects without starting from scratch.

It is a software platform where support programs, libraries, interpreted language, etc. are defined, which helps to develop and unite the different modules or parts of a project.

With the use of framework we can spend more time analyzing the system requirements and technical specifications of our application, since the laborious task of programming details is solved.

- Advantages of using a framework:
 - o **Rapid development** of software.
 - o **Reuse** of code parts for other applications.
 - o **Design** software uniform.
 - o **Portability** of applications from one computer to another, since the bytecodes generated from the source language can be executed on any virtual machine.
- Disadvantages:
 - o Great dependence of the code on the framework used (without changing the framework, a large part of the application will have to be rewritten).
 - o The installation and implementation of the framework on our computer consumes a lot of system resources.

To know more

The increasing use of frameworks means that we have to constantly recycle ourselves. In the following link, there is a very interesting document of its main characteristics, advantages and ways of use:

<http://www.maestrosdelweb.com/editorial/los-php-frameworks-speed-up-your-work/>

Examples of Frameworks:

- **.NET** It is a framework for developing applications on Windows. It offers the "Visual Studio .net" that gives us facilities to build applications and its engine is the ".Net framework" that allows us to run these applications. It is a component that is installed on the operating system.
- **Java Spring**. They are sets of libraries (API's) for the development and execution of applications.

you must know

The installation and configuration process of the Spring Java framework, as well as several usage examples. In the following link you will find a very useful guide detailing the steps to follow:

[http://pabloig.wikispaces.com/file/view/spring_tutorial_v0.271.p
df](http://pabloig.wikispaces.com/file/view/spring_tutorial_v0.271.pdf)

5.5.2.- Execution environments.

An execution environment is a virtual machine service that serves as a software base for executing programs. Sometimes it belongs to the operating system itself, but it can also be installed as independent software that will work underneath the application.

That is, it is a set of utilities that allow the execution of programs.

Runtime is the time it takes for a program to run on the computer.

During execution, the environments will be responsible for

- Configure the main memory available in the system.
- Link program files with existing libraries and with created applets. Considering that libraries are the set of subprograms that are used to develop or communicate software components but that already exist previously and the subprograms will be those that we have created on purpose for the program.
- Debug programs: check the existence (or non-existence) of semantic errors in the language (the syntactic ones were already detected in the compilation).



Runtime environment operation:

The Execution Environment is made up of the virtual machine and the API's (standard class libraries, necessary so that the application, written in a Programming Language, can be executed). These two components are usually distributed together, because they need to be compatible with each other.

The environment works as an intermediary between the source language and the operating system, and manages to execute applications.

However, if what we want is to develop new applications, the execution environment is not enough.

Going forward to what we will see in the next unit, to develop applications we need something else. That "something else" is called the development environment.

Self appraisal

Point out the false statement regarding runtime environments:



Its main use is to allow rapid development of applications.



It acts as a mediator between the operating system and the source code.



It is the set of the virtual machine and libraries necessary for execution.

5.5.3.- Java runtimeenvironment.

This section will explain the operation, installation, configuration and first steps of the RuntimeEnvironment of the Java language (it is extensible to other programming languages).

Concept.

The Java RuntimeEnvironment is called JRE.

The JRE is made up of a set of utilities that will allow the execution of Java programs on any type of platform.

Components.

JRE is made up of:

- A Java Virtual Machine (JMV or JVM if we consider the acronym in English), which is the program that interprets the application code written in Java.
- Standard class libraries that implement the Java API.
- Both: MVP and jointly. Java APIs are consistent with each other, which is why they are distributed

The first thing is to download the JRE program. (Java2 RuntimeEnvironment JRE 1.6.0.21). Java is free software, so we can download the application freely.

Once downloaded, the installation process begins, following the steps of the wizard.

you must know

The process of downloading, installing and configuring the program execution environment. The following link explains the steps to do it under the Linux operating system.

http://www.java.com/es/download/help/linux_install.xml

To know more

In the following link you will find a tutorial of the Java language, with its main characteristics and main orders and commands.

<http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf>

5.6.- Tests.

Practical case

Maria gathers all the designed codes and prepares them to be implemented on the client's computer. Juan realizes this, and reminds his friend that they have not yet been tested. Juan remembers well the time that happened to him: two years ago, when he went to present an application to his clients, he kept making all kinds of errors... the clients, of course, did not accept it and Juan lost a month of hard work and was about to lose his job... —Not so quickly Maria, we have to TRY the application.

Once the software is obtained, the next phase of the life cycle is testing.

Typically, these are performed on a test data set, which consists of a selected and predefined set of boundary data to which the application is subjected.

Testing is essential to ensure the validation and verification of the built software.

Among all the tests carried out on the software we can basically distinguish:

UNIT TESTS

They consist of testing, one by one, the different parts of the software and checking their operation (separately, independently). JUnit is the testing environment for Java.

INTEGRATION TESTING

They are carried out once the unit tests have been successfully carried out and will consist of checking the operation of the entire system: with all its interrelated parts.

The final test is commonly called Beta Test, this is carried out in the production environment where the software will be used by the client (if possible, on the client's computers and under normal operation of their company).

The trial period will normally be the one agreed with the client.

Self appraisal

If the unit tests are carried out successfully, is it mandatory to carry out the integration tests?



Yes, if the application is made up of more than five different modules.



Yes, in any case.

To know more

You can visit the following website, which details the types of tests that are usually performed on the software and the function of each one.

<http://www.sistedes.es/TJISBD/Vol-1/No-4/articles/pris-07-raja-ctps.pdf>

5.7.- Documentation.

Practical case

Ada has arranged to meet her client in two days. She asks Maria about all the documentation dossiers. The young woman's pale expression makes Ada burn in despair: "—Haven't you documented the stages? How am I going to explain to the client and his employees how the software works? How are we going to maintain it?"

All the stages in the development of software they must meet perfectly documented.

Why do we have to document all phases of the project?

To give all the information to the users of our software and to be able to undertake future reviews of the project.

We have to document the project in all its phases, to move from one to another in a clear and defined way. Correct documentation will allow the reuse of part of the programs in other applications, as long as they are developed with a modular design.

We distinguish three major documents in software development:

<i>Documents to elaborate in the software development process</i>			
	GUIDE TECHNIQUE	USE GUIDE	GUIDE OF FACILITY

They are reflected:	<ul style="list-style-type: none"> • The design of the application. • The coding of the programs. • The tests carried out. 	<ul style="list-style-type: none"> • Description of the functionality of the application. • Way to start running the application. • Examples of use of the program. • Application software requirements. • Solution of possible problems that may arise. <p>To the users who are going to use the application (customers).</p>	<p>All the information Necessary for:</p> <ul style="list-style-type: none"> • Start up. • Exploitation. • System security.
Who is it addressed to?		Give end users all the information necessary to use the application.	To the IT staff responsible for the installation, in collaboration with the users who are going to use the application (clients).
What's your objective ?	To the technical staff in computer science (analysts and programmers).		Provide all the necessary information to guarantee that the implementation of the application is carried out safely, reliably and accurately.

Facilitate correct development, carry out corrections in the programs and allow for future maintenance.

5.8.- Exploitation.

Practical case

The day of the appointment with the hotel chain arrives. Ada and Juan head to the hotel where the application is going to be installed and configured. If all goes well, it will be implemented in the other hotels in the chain.

Ada doesn't want to miss a single detail: she carries the user guide and the installation guide.

After all the previous phases, once the tests show us that the software is reliable, free of errors and we have documented all the phases, the next step is exploitation.

Although various authors consider exploitation and maintenance as the same stage, we are

going to differentiate them based on the moment in which they are carried out.

Exploitation is the phase in which end users learn about the application and start using it.

Exploitation is the installation, development and operation of the application on the client's final computer.

In the installation process, the programs are transferred to the client user's computer and subsequently configured and verified.

It is recommended that future clients be present at this time and tell them how the installation is being planned.

At this time, Beta Tests are usually carried out, which are the last tests that are carried out on the client's own computers and under normal workloads.

Once installed, we move on to the configuration phase.

In it, we assign the normal operating parameters of the company and test that the application is operational. It may also happen that the configuration is carried out by the end users themselves, as long as we have previously given them the installation guide. And also, if the application is simpler, we can program the configuration so that it is done automatically after installing it. (If the software is "custom", it is most advisable that it be made by those who have manufactured it).

Once it has been configured, the next and final step is the normal production phase. The application passes into the hands of the end users and the exploitation of the software begins.

It is very important to have everything prepared before presenting the product to the client: it will be the critical moment of the project.

Reflect

You carry out a software project for the first time and you don't realize how to document it. You manage to sell it at a good price to a company. After a couple of months they ask you to update some of the functions, to have greater functionality. You are happy because that means extra income. You stop for a moment...Where are the codes? What exactly did the app do? How was it designed? You don't remember... You've probably lost extra income and some good clients.

5.9.- Maintenance.

Practical case

Ada gathers her team for the last time during these weeks. Everyone celebrates that the project has been successfully implemented and that their clients have been satisfied.

"This is not over yet," Ada comments, "we have many things left to do." This afternoon I meet with clients. How are we going to manage the maintenance of the application?

It would be logical to think that with the delivery of our application (the installation and configuration of our project on the client's computers) we have finished our work.

In any other work sector this is true, but the case of software construction is very different.

The maintenance stage is the longest in the entire software life cycle.

By its nature, software is changeable and will need to be updated and evolved over time. It must adapt in parallel to hardware improvements on the market and face new situations that did not exist when the software was built.

In addition, errors always arise that will have to be corrected and new versions of the product better than the previous ones.

For all these reasons, a maintenance service for the application is agreed upon with the client (which will also have a temporary and economic cost).

Maintenance is defined as the process of controlling, improving and optimizing software.

Its duration is the longest in the entire software life cycle, since it also includes future updates and evolutions of the software.

The types of changes that require software maintenance are the following:

- **Perfectives:** To improve the functionality of the software.
- **Evolutionary:** The client will have new needs in the future. Therefore, code modifications, expansions or deletions will be necessary.
- **Adaptive:** Modifications, updates... for adapt to the newmarket trends, new hardware components, etc.
- **Correctives:** The application will have errors in the future (it would be utopian to think otherwise).

Self appraisal

What is, in your opinion, the most important stage of software development?



The requirements analysis.



The coding.



Testing and documentation.



Operation and maintenance.

Annex I.- Structured programming control sentences.

SEQUENTIAL SENTENCES

Sequential statements are those that are executed one after the other, according to the order in which they were written.

Example in C language:

```
printf ("variable declaration");  
integer_number; space=start_space +  
speed*time;
```

SELECTIVE (CONDITIONAL) SENTENCES

They are those in which a condition is evaluated. If the result of the condition is true, a series of action or actions are executed and if false, others are executed.

if → indicates the condition to be evaluated

then → All actions after this reserved word will be executed if the if condition is true (in C, this word is omitted).

else → All actions after this other reserved word will be executed if the if condition is false.

Example in C language:

```
if (a >= b)  
    c=ab;  
else  
    c=a+b;
```

REPETITIVE STATEMENTS (ITERATIONS OR LOOPS)

An iterative loop of a series of actions will cause them to repeat as long as or until a given condition is false (or true).

while → marks the beginning of the loop and is followed by the loop stop condition.

do → From this reserved word, all the actions to be executed while the loop is executed will be found (in C, this word is omitted).

done → marks the end of the actions that are going to be repeated while we are inside the loop (in C, this word is omitted).

Example in C language:

```
int num; num = 0;
while (num <= 10) {
    printf("Repetition number %d\n", num);
    num = num + 1;
};
```

Annex.- Resource licenses.

Resource data (1)
Author: Scott Schram. License: CC by 2.0. Origin: http://www.flickr.com/photos/schram/21742249/
Author: Francisco Palacios. License: CC by -NC-ND 2.0. Origin: http://www.flickr.com/photos/wizard_/3303810302/
Author: fsse8info. License: CC by -SA 2.0. Origin: http://www.flickr.com/photos/fsse-info/3276664015/