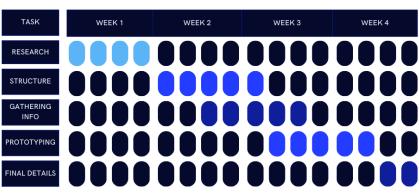


SOFTWARE DEVELOPMENT

ALL APPS ARE MADE THROUGH THE SAME STEP BY STEP PROCESS, WHICH STARTS WITH AN IDEA AND ENDS WITH THE SET OF FILES THAT MAKES IT DO WHAT IT HAS TO DO SUCCESSFULLY

GANTT DIAGRAM

JAVIER PEDRAGOSA LOZANO (DAW 1)



1.PROGRAM

TO ANALYZE HOW TO CREATE A COMPUTER PROGRAM, WE FIRST NEED TO UNDERSTAND WHAT A PROGRAM IS.

EVEN THOUGH «PROGRAM» ±S USED OUTSIDE OF THE IT CONTEXT, SUCH AS A MOVIE THEATER'S OR A TV CHANNEL'S, ±N COMPUTING, ±T REFERS TO A SET OF ORDERS THAT GET EXECUTED TO PROCESS DATA. OUR JOB AS PROGRAMMERS ±S TO SELECT AND ARRANGE THE APPROPIATE ENSTRUCTIONS FOR THE PROGRAM TO FULFILL A CERTAIN TASK. THE COMPUTER, UNABLE TO WORK BY ±TSELF, ALWAYS EXECUTES PROGRAMS, SUCH AS OPERATIVE SYSTEMS WHICH RUN ALWAYS ±N THE BACKGROUND.

U.LANGUAGES

THŁS PARADŁGM ŁS BASED ON REPRESENTŁNG REAL WORLD THROUGH OBJECTS, WHICH ARE COMBENATIONS OF DATA (ATTREBUTES) AND (FUNCTIONS) OOP CONSEST EN SETS OF THESE OBJECTS THAT ENTERACT WETH ONE ANOTHER THROUGH MESSAGES (METHOD CALL£NG). TH£S APPROACH **ENTRODUCES** FEVE KEY ACONCEPTS: ABSTRACTION, ENCAPSULATION, MODULARITY HIERARCHY AND POLYMORPHISM, MOREOVER, THIS PARADIGM FACILITATES REUSING CODE AND EMPROVES THE DESEGN OF COMPLEX APPL±CAT±ONS.

DEVELOPMENT PHASES

- 1. Vłability Study: TECHNICAL, ECONOMIC AND OPERATIVE Vłability OF THE SYSTEM ARE ANALYZED TO DECIDE IF THE PROJECT MUST CONTINUE.
- 2. SYSTEM ANALYSIS: THE SYSTEM'S FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS ARE DEFINED THROUGH NEED MODELS AND CATALOGUES.
- 8. SYSTEM DESIGN: THE SYSTEM'S ARCHITECTURE AND TECHNOLOGICAL ENVIRONMENT ARE SPECIFIED, JUST LIKE COMPONENTS, TESTS AND MIGRATION PROCEDURES.
- 4. SYSTEM BUŁLDŁNG: THE SYSTEM'S COMPONENTS ARE DEVELOPED AND TESTED, GENERATŁNG CODE AND DOCUMENTŁNG PROCEDURES.
- S. IMPLEMENTATION AND ACCEPTANCE: THE SYSTEM IS DELIVERED, TESTED AND ACCEPTED, ENSURING IT COMPLIES WITH THE REQUIREMENTS BEFORE STARTING PRODUCTION.

2.CODE

FOR CREATING A PROGRAM, A TEXT FILE IS WRITTEN WITH INSTRUCTIONS FOLLOWING THE RULES OF THE SELECTED PROGRAMMING LANGUAGE. THOSE INSTRUCTIONS MAKE UP THE SOURCE CODE, WHICH CAN BE WRITTEN IN A HIGH LEVEL LANGUAGE (CLOSER TO HUMAN LANGUAGE) OR LOW LEVEL (CLOSER TO MACHINE CODE).

DUE TO THE COMPUTER NOT UNDERSTANDING DIRECTLY THE SOURCE CODE, IT MUST BE TRANSLATED THROUGH A PROCESS CALLED COMPILATION, WHICH CONVERTS THE SOURCE CODE IN OBJECT CODE, WHICH CAN BE UNDERSTOOD BY THE PROCESSOR. THEN, THE LINKER INTEGRATES THE OBJECT CODE WITH THE FUNCTIONS OF THE NEEDED LIBRARIES, FINALLY GENERATING AN EXECUTABLE FILE THAT THE COMPUTER CAN INTERPRET AND EXECUTE.

8.PARADIGMS

- 1. IMPERATIVE/STRUCTURED PARADIGM: PROGRAMS ARE BUILT USING IMPERATIVE SENTENCES THAT EXECUTE OPERATIONS MODIFYING DATA. THE MOST COMMON LANGUAGES ARE C AND PASCAL. THIS APPROACH USES CONTROL STRUCTURES LIKE SEQUENCE, SELECTION, AND ITERATION, AND FOLLOWS A TOP-DOWN MODULAR DESIGN.
- 2. OBJECT-ORIENTED PARADIGM: FOCUSES ON OBJECTS, WHICH ARE ABSTRACTIONS OF REAL-WORLD ENTITIES THAT COMBINE DATA AND METHODS. THE MOST COMMON LANGUAGES ARE C++ AND JAVA, ALTHOUGH THERE ARE MANY MORE AS THIS IS THE MOST IMPLEMENTED PARADIGM. THERE IS GREAT EMPHASIS ON ENCAPSULATION, INHERITANCE, AND POLYMORPHISM.
- S. FUNCTIONAL PARADIGM: BASED ON MATHEMATICAL FUNCTIONS WHERE EACH RESULT BECOMES THE INPUT FOR THE NEXT, AIMING FOR DATA TRANSFORMATION. LANGUAGES LIKE LISP ARE TYPICAL IN THIS CATEGORY.
- 4. LOGECAL PARADEGM: USES LOGEC RULES TO ENFER CONCLUSIONS FROM GEVEN DATA, SEEN EN LANGUAGES LEKE PROLOG, OFTEN APPLIED EN AI AND EXPERT SYSTEMS.