6. Coding/Implementation

Road map

(1) What is implementation ?

(2) Choosing the correct language

(3) Evolution of programming languages

(4) Good coding practices

(1) What is Implementation ?

1. Transforms the design specification to source code that can be executed on a computer.
2. This is the final stage of the series of front end activities we have been dealing with.
3. Coding is relatively straight forward given a design specification.
4. Coding is a minor activity compared to the other phases of development.

5. A good design may be **spoiled by the bad choice of a language.**
6. However a bad design **cannot be corrected** through coding.

5. Choice of the language and the coding style are important issues to consider.
6. The programmer translates the design into source code of the chosen programming language.
7. The language translator converts the source code in to executable code in several steps.

10. Certain design issues may not be supported by the language in which case the coder may choose to violate design
11. Although design quality should not be compromised because of a language issue, design approach may depend on the language choice.

(2) Choosing the correct language

Language characteristics

1. Technical characteristics (Supports the design to be translated to an executable version)
2. Psychological characteristics (Supports the coding process)
3. Engineering characteristics (Supports the software development process)

(1) Technical characteristics

1. Type of the system
2. Modularity support (The extent to which the language supports information hiding and modularity)
3. Control structures (How the language allows control flow to be defined)

(2) Psychological characteristics

1. Consistency (The language should use consistent notation)
2. Readability (The language should help to produce unambiguous and clear source code)
3. Expressiveness (How powerful the language constructs are)
4. Separation of concerns (This is promoted by structured programming)

(3) Engineering characteristics

1. Correspondence with the design
2. Availability and efficiency of compilers
3. Portability
4. Development support
5. Maintainability support
6. Application area
7. Expertise required
8. The adequacy of the computational model

(3) Evolution of programming languages

1. Machine Language.
2. Assemblers
3. Fortran, Cobol, Algol, Basic, and Pascal
4. Fourth generation languages (logic languages like Prolog, object oriented languages like C++, Java, domain specific languages like MATLAB, Query languages such as SQL)

Language Features

Language	Features	Strengths	Weaknesses
C	Procedural Weak by type checking Very low level Pointers Flexible	Close to hardware/OS Fast and efficient applications can be built Widely used	Poor exception handling support Memory handling leads to unreliable code
C++	OO extension to C Weak type checking Flexible Pointers	All those of C and OO concepts of Polymorphism Inheritance (single and multiple) Encapsulation	As for C
COBOL	Procedural Strong I/O handling Defined standard	Suited for batch transaction processing	Language run time system required Old –many features added later

Language Features

Language	Features	Strengths	Weaknesses
FORTRAN	Procedural Strong arithmetic support through libraries	Suited for scientific computing where significant numerical processing required	Old- many modern languages provide most of the features
Java	OO Better type checking than C but still reasonably weak Standard defined by the Sun Microsystems	Platform independent Dynamic downloading of classes Good user interface and network support through libraries Ideal for network applications	Requires own runtime environment Controlled by a commercial organization
Pascal	Procedural Strong type checking Well structured programming	Good teaching language	Not widely used in industry

Language Features

Language	Features	Strengths	Weaknesses
Visual Basic	Simple procedural language Interpreted Extensive Windows programming support	Suited for small applications and prototyping Some OO concepts	Performance Complex data structures cannot be modelled
Prolog	Logical language	Suitable for Intelligent systems	Numerical operations
Visual C++	C++ programming environment for MS Windows	Support for Windows programming User inteface design Some code generation	Portability of code

Language Features

Language	Features	Strengths	Weaknesses
.NET	NET is the Microsoft Web services strategy to connect information, people, systems, and devices through software	Good for developing integrated business solutions (distributed) Agile development, quickly build and security enhanced solutions.	
C#	The first component oriented language in C/C++ family Integrated documentation using XML No header files Can be embedded in web pages	New primitive types: Decimal, SQL, .. No memory leaks and stray pointers Error handling Interoperability: MS C# implementation talks to XML, SOAP,COM and any .NET language	

(4) Good coding practices

1. Coding should be done with an aim toward maintenance.
2. This could be done by adhering to coding standards (standards allow the design rationale to be reflected in the code, and consistent code to be produced)
3. Documentation (comments, pre & post conditions, and references)
4. Understandable identifiers (here there is a trade off between speed and readability)

1. Indentation and organization (this should be done to reflect the logical structure of the program)
2. Explicit declarations
3. Use of characters, spaces and colors effectively so as to promote readability
4. Breaking complicated expressions to simple ones (This not only enhances readability but also increases the chances of efficient compilation)

Question : Identify what languages are the most appropriate in the following situations.

(1) Program to implement a pocket calculator on a PC.

(2) Develop a web application for buying and selling vehicles over the internet

(3) Develop a knowledge based system for skin disease diagnosis

(4) Develop a prototype interface

(5) Develop the back end of an inventory system

(6) Writing an operating system utility