Chapter Four
Software design

4.1 Software design

The goal of design is to create a model a model of software that will implement all customer requirements correctly and bring delight to those who use it.

ان الهدف من مرحلة التصميم هو خلق model لل software الذي سوف يحقق كل متطلبات المستخدم وبشكل صحيح بدون اخطاء ويجلب البهجة لمستخدميه.

Software design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analysed and modelled, software design is the essential to start other phases (code generation and testing), that are required to build and verify the software.

يمثل ال Software design النواة التقنية لهندسة البرمجيات ويتم تطبيقه بغض النظر عن software process model المستخدمة. وتبدأ هذه المرحلة حالما يتم الانتهاء من تحليل المتطلبات وعمل النماذج model لل software ، وتعد مرحلة اساسية للبدأ ببقية المراحل من(كتابة ال code واختباره) ، اللازمة لإنشاء software والتحقق منه.

4.2 Software Design Stages

Design creates a representation or model of the software, but unlike the requirements model (that focuses on describing required data, function, and behaviour), the design model provides detail about software architecture, data structures, interfaces, and components that are necessary to implement the system. Figure (4-1) illustrate how to translate the analysis model to design model. These stages are:-

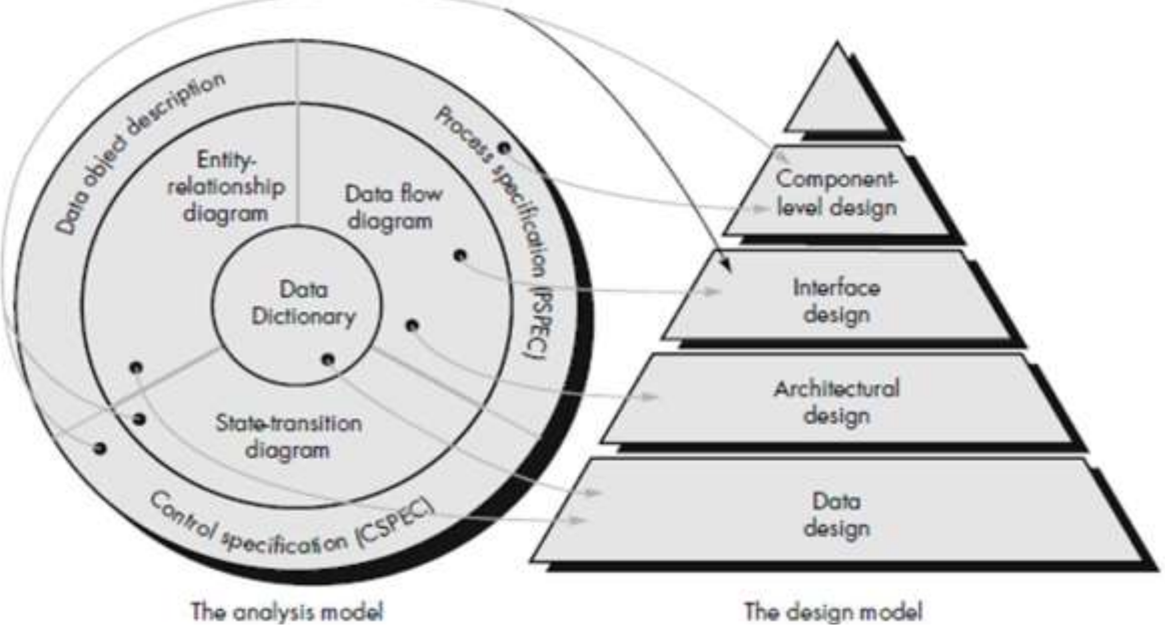


Figure (4-1) Translating the analysis model into a software design

1- Data design

Data design translates the **data objects** defined in the analysis model (using entity relationship diagrams (ERD) and the data dictionary (DD)) --------into-----------**data structures** that reside within the software.

The attributes that describe the object, the relationships between data objects and their use within the program all influence the choice of data structures.

The data design activity translates these elements of the requirements model into data structures at the software component level and, when necessary, a database DB architecture or a data warehouse DW at the application level.

2- Architectural design

The architectural design defines the relationship between major structural elements of the software. It depicts the structural elements, their properties, and the connections between them.

يعرف العلاقة بين structural elements المكونة للsoftware . وذالك بتوضيحها وتوضيح خصائصها وطريقة التوصل بينها.

Software components include program modules and the various data representations that are manipulated by the program. Therefore, data design is an integral part of the software architecture. The primary objective of architectural design is to develop a modular program structure and represent the control relationship between modules.

مكونات ال Software هي program modules وحدات برمجية التي تحتوي على بيانات ممثلة بشكل مختلف ويتم معالجتها بواسطة البرنامج. لذالك data design هي جزء من software architecture . الهدف الاساسي لهذه المرحلة هو تطوير هيكلية لل program modules وتمثيل العلاقة بينها

3- Interface design

User interface design begins with the identification of user, task, and environmental requirements.

لتصميم الواجهات يجب التعرف على المستخدم user والمهمة task والمتطلبات requirement

The interface design involves:-

- Design the interfaces between human (user) and computer
- Design the interfaces between the software and other nonhuman producers and consumers of information (other external entities)

Three important principles for effective user interfaces design: لتصميم الواجهات يجب مراعاة عدة شروط

- Place the user in control
- Make the interface consistent
- Reduce the users' memory load

4- component-level design

- Also called procedural design

It transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC, and STD serve as the basis for component design.

Component-level design depicts the software at a level of abstraction that is very close to code.

At the component level, the software engineer must represent data structures, interfaces and algorithms in sufficient detail to guide in the generation of programming language source code. To accomplish this, the designer uses one of a number of design notationsالترميزات that represent component level design detail in either graphical, tabular, or text based formats, as follows:

Component-level design techniques:

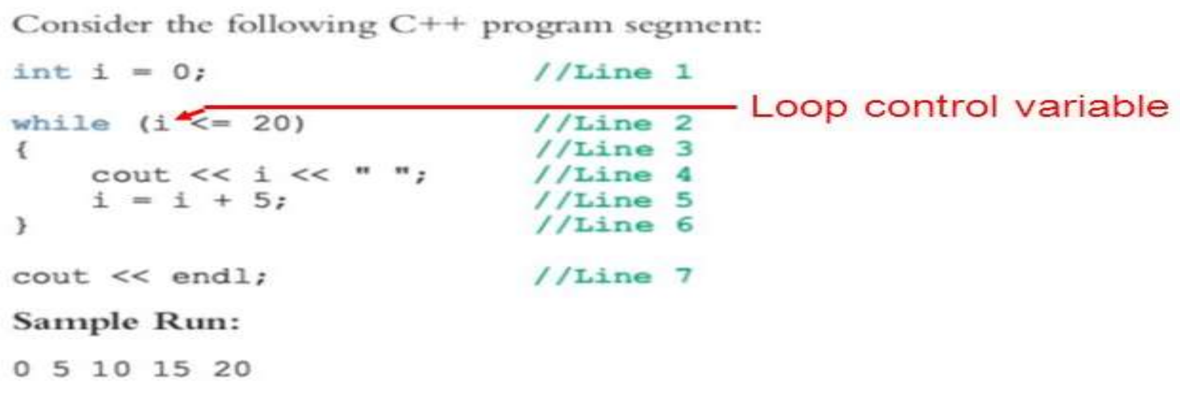
A. Structured programming

هي تقنية تصميم فلسفية التي تحدد كيفية التدفق المنطقي logic flow للنظام او ال software الى ثلاثة تراكيب constructs وهي sequence, condition, and repetition, used to represent algorithmic detail التي تستخدم لتمثيل تفاصيل الخوارزمية.

the intent of structured programming is to assist the designer الهدف من هذة التقنية من اجل دعم المصمم لتحديد وتحويل

To limit **the procedural design of software** -----to---------- **a small number of predictable operations,** defining algorithms that are less complex and therefore easier to read, test

لتعريف خوارزمية اقل تعقيد وسهلة القرءاة والاختبار


Consider the following C++ program segment:

```
int i = 0;                    //Line 1

while (i <= 20)               //Line 2
{                             //Line 3
    cout << i << " ";         //Line 4
    i = i + 5;                //Line 5
}                             //Line 6

cout << endl;                 //Line 7
```

Sample Run:

```
0 5 10 15 20
```

B. Graphical design

The activity diagram allows a designer to represent sequence, condition, andrepetition-all elements of structured programming-by using a flowchart.

تمكن المصمم من تمثيل الشروط والتكرارات والتسلسل لكل عناصر البرنامج باستخدام diagram (**flowcharts**)

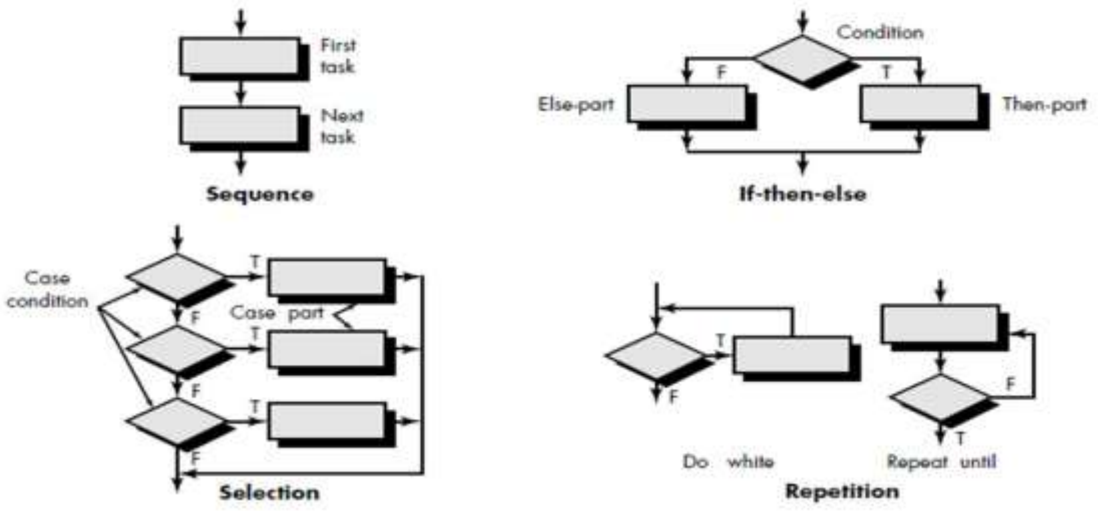


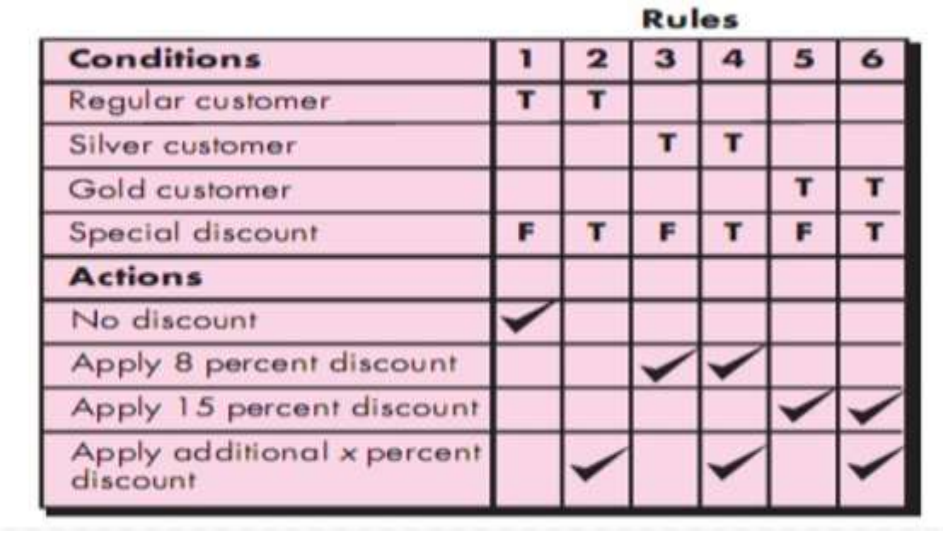
Figure (4-2): Flowchart constructs

C. Tabular design notation

In many software applications, a module may be required to evaluate a complex combination of conditions and select appropriate actions based on these conditions. <u>Decision tables provide a notation that translates actions and conditions.</u>

يستخدم لتقييم مجموعة شروط معقدة واختيار الاجابة المناسبة لها اعتمادا على هذه الشروط. لذالك فأن Decision tables يوفر ترميز يترجم الشروط والاجابات.

the table is divided into four sections. The upper left-hand quadrant contains a list of all conditions. The lower left-hand quadrant contains a list of all actions that are possible based on combinations of conditions. The right-hand quadrants form a matrix that indicates condition combinations and the corresponding actions that will occur for a specific combination.



	Rules					
Conditions	1	2	3	4	5	6
Regular customer	T	T				
Silver customer			T	T		
Gold customer					T	T
Special discount	F	T	F	T	F	T
Actions						
No discount	✓					
Apply 8 percent discount			✓	✓		
Apply 15 percent discount					✓	✓
Apply additional x percent discount		✓		✓		✓

Figure (4-3): Decision tables

D. Program Design language(PDL)

Is also called pesudocode or structured English, it uses the vocabulary of one language (i.e., English) and the overall syntax of another (i.e., a structured programming language)".

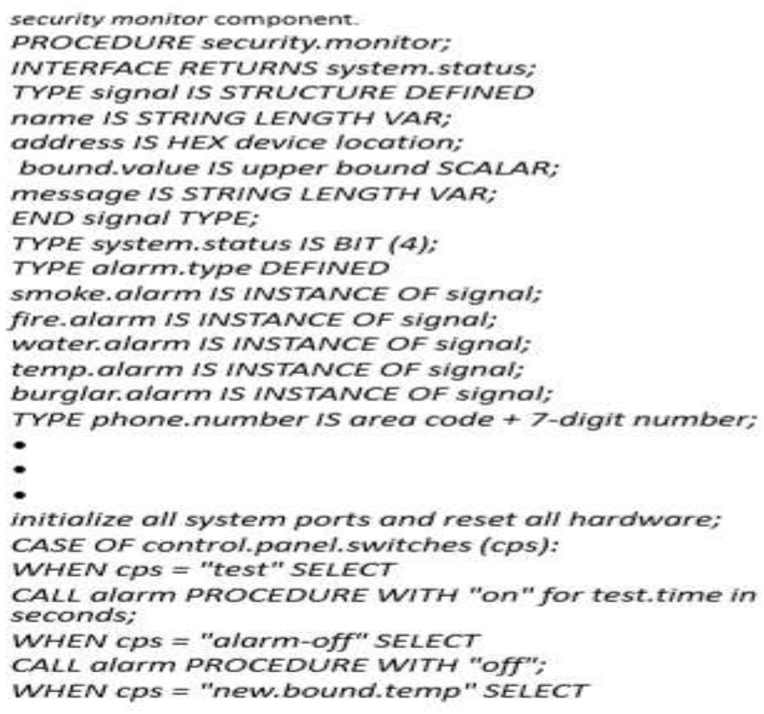
عبارة عن لغة مبسطة تستخدم المصطلحات لاحد اللغات مثل (الانكليزي) وتتبع قواعد لغة اخرى

The differences between PDL and real programming language is the use of narrative text (e.g., English) embedded directly into a syntactical structure

هو استخدام اسلوب السرد النصي كمثال استخدام اللغة الانكليزية مضمنة في هيكلية لها قواعدها بينمافي لغات البرمجة كل لغة لها هيكليتها

PDL tools currently exist to translate PDL into programming language is "graphical representation (flowchart),

PDL example (part of it): The safe Home security system software



```
security monitor component.
PROCEDURE security.monitor;
INTERFACE RETURNS system.status;
TYPE signal IS STRUCTURE DEFINED
name IS STRING LENGTH VAR;
address IS HEX device location;
 bound.value IS upper bound SCALAR;
message IS STRING LENGTH VAR;
END signal TYPE;
TYPE system.status IS BIT (4);
TYPE alarm.type DEFINED
smoke.alarm IS INSTANCE OF signal;
fire.alarm IS INSTANCE OF signal;
water.alarm IS INSTANCE OF signal;
temp.alarm IS INSTANCE OF signal;
burglar.alarm IS INSTANCE OF signal;
TYPE phone.number IS area code + 7-digit number;
•
•
•
initialize all system ports and reset all hardware;
CASE OF control.panel.switches (cps):
WHEN cps = "test" SELECT
CALL alarm PROCEDURE WITH "on" for test.time in
seconds;
WHEN cps = "alarm-off" SELECT
CALL alarm PROCEDURE WITH "off";
WHEN cps = "new.bound.temp" SELECT
```

4.3 Architectual Design

أي ما هي الطريقة التي سيتم بناء النظام وكيف يتم التحكم بالانظمة الفرعية وماهي طريقة التواصل بين الانظمة وطريقة تمرير ومعالجة البيانات وتتكون من الاتي:

A. Structural System Models طرق هيكلة الانظمة (بناء الانظمة)

B. Control Models .طرق التحكم بالانظمة

C. Modular Decomposition طرق الاتصال

A– **Structural System Models :** ما هي الطريقة التي سيتم بناء النظام ومعالجة الاخطاء

ولدينا ثلاث طرق لبناء الانظمة هي كالتالي:

١- **Repository model:** في هذة الطريقة يكون شيء مشترك بين كل الدوال او اجراءات النظام كان تكون قاعدة بيانية مجموعة جداول او دالة معينة مثل بناء compilers (توجد فيها بيانات مشتركة)

٢- **client / server model** : في هذة الطريقة نقوم بانشاء برنامج client ومرتبط ببرنامج server وفق شبكة وال server خادم لتلك الاجهزة المستنفيدة وكذلك يقوم بعملية التحكم والمراقبة مثل تقنية الانترنت

٣- **stract Machine Model** : تعتمد هذة الطريقة على شكل طبقات وكل طبقة من الطبقات تعتمد على التي تحتها فعلى سبيل المثال شاشة MDI ومثال عليها انظمة التشغيل تعتمد اساسا على هذة الطرقة فغلق اي واحد من هذذة البرامج لا يؤثر على نظام التشغيل والعكس غير صحيح

B. Control Models طرق التحكم بالانظمة

استراتيجة السيطرة على الانظمة الفرعية او الاجراءات ولدينا نوعان

١. centralized يتم التحكم بالنظام ككل من خلال برنامج رئيسي او دالة رئيسة

٢. Events Based في هذة الطريقة فان التحكم على اساس الاحداث فعند انطلاق الحدث يتم تنفيذ عمل معين

Modular Decomposition : استراتيجية وكيفية الاتصال بين مكونات النظام

A- object oriented programming

B- data flow model

4.4 Software Design Approaches

1- Top - Down Design

takes the whole software system and decomposes it to achieve more than one sub-system or component based on some characteristics. Each sub-system or component is then treated as a system and decomposed further. This process keeps on running until the lowest level of system in the top-down hierarchy is achieved. Top-down design is more suitable **when the software solution needs to be designed from scratch and specific details are unknown**.

High level of abstraction ------------------ low level of abstraction

2- Bottom-up Design

starts with most specific and basic components. It proceeds with composing higher level of components by using basic or lower level components. It keeps creating higher level components until a complete system is arrived. With each higher level, the amount of abstraction is increased. Bottom-up strategy is more suitable **when a system needs to be created from some existing system, where the basic primitives can be used in the newer system.**

Low level of abstraction ------------------ high level of abstraction

3- Hybrid design

Mix the benefit of top down and bottom up approaches