

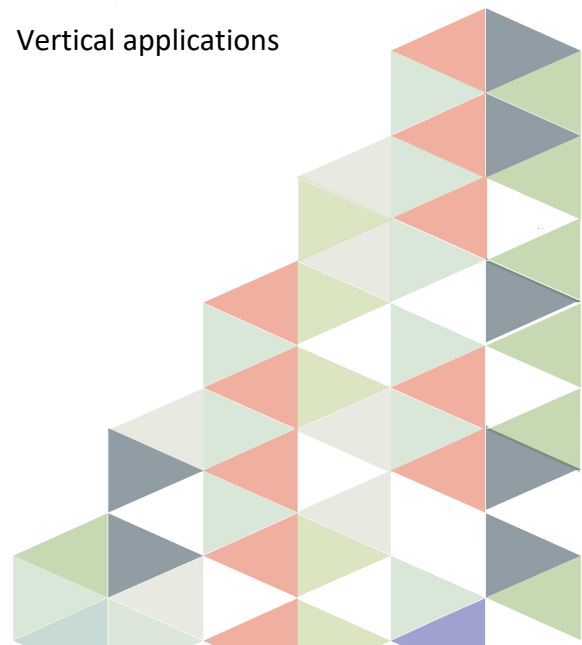
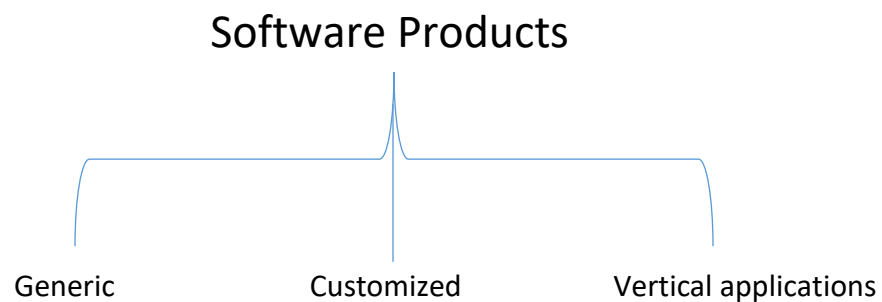
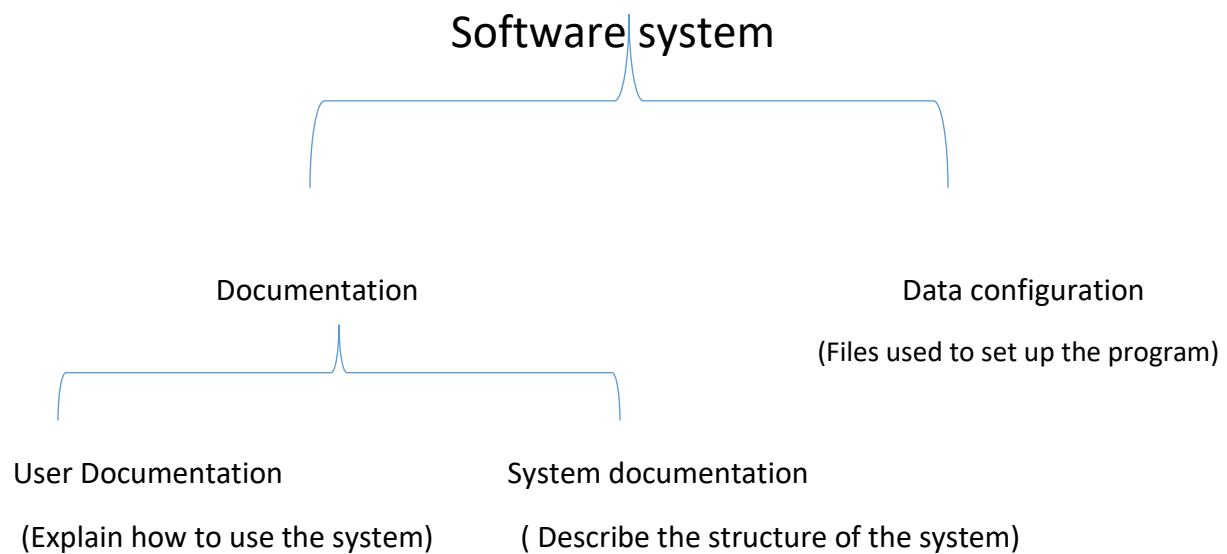


# Software Engineering

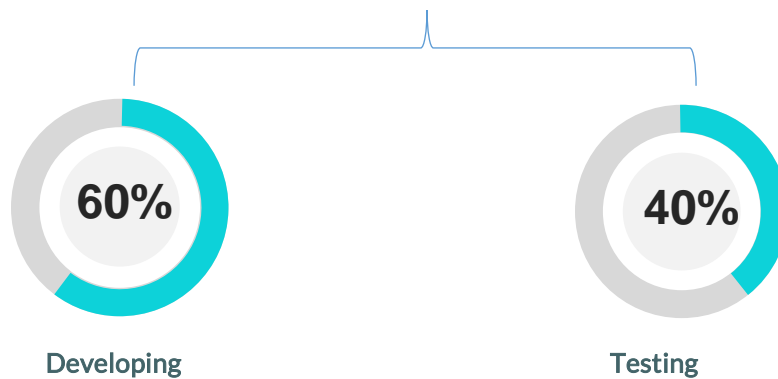
## Mind Map

- Software Engineering : Over all process to create SW product .

It's important because it produces trustworthy, quick and cheap systems



## Software cost



Testing of human-related software ( flight control, nuclear reactor monitoring) can cost 3 to 5 times

### Attributes of good software

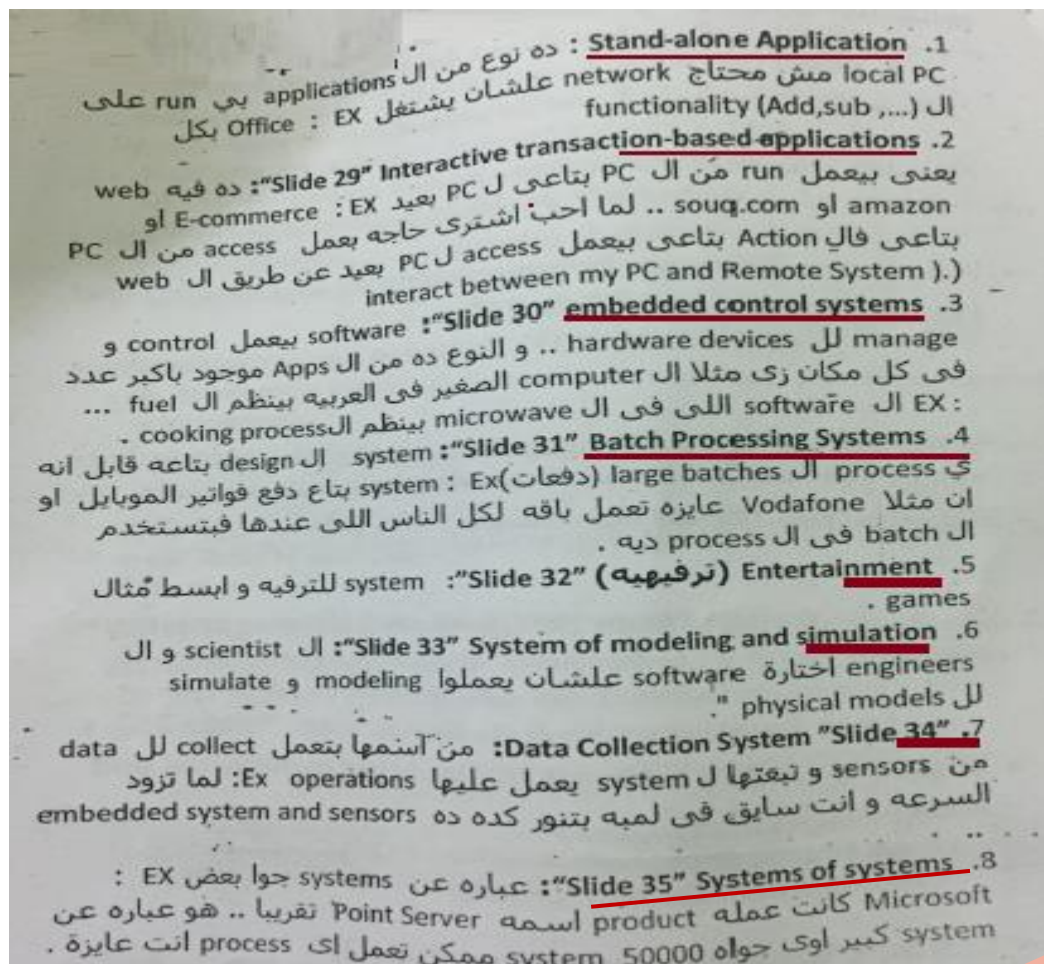
- **Maintainability**  
(meet customer's need)
- **Dependability and security**
- **Efficiency**
- **Acceptability**  
(acceptable to the users , understandable and compatible with other systems they use )

### Issues affect many types of software

- **Heterogeneity**  
So, it must cope with different platforms
- **Business and social change**  
So , we must lead to faster deliver of SW
- **Security and trust**  
**So, we must** make sure that malicious users can't attack our SW



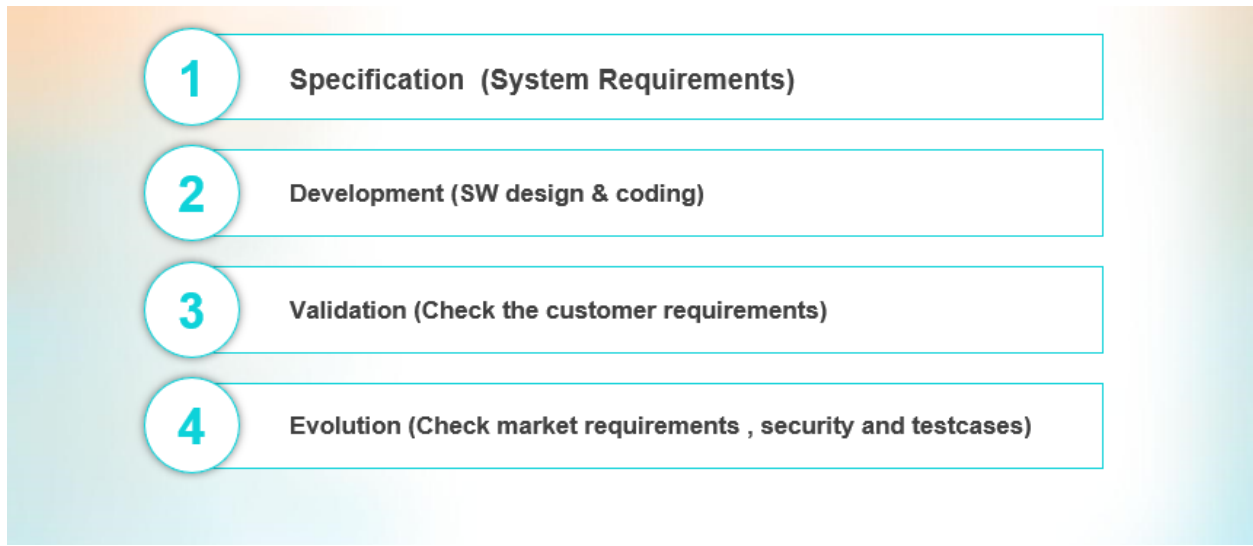
## Types of SW applications



كده خلصنا المحاضره الاولى

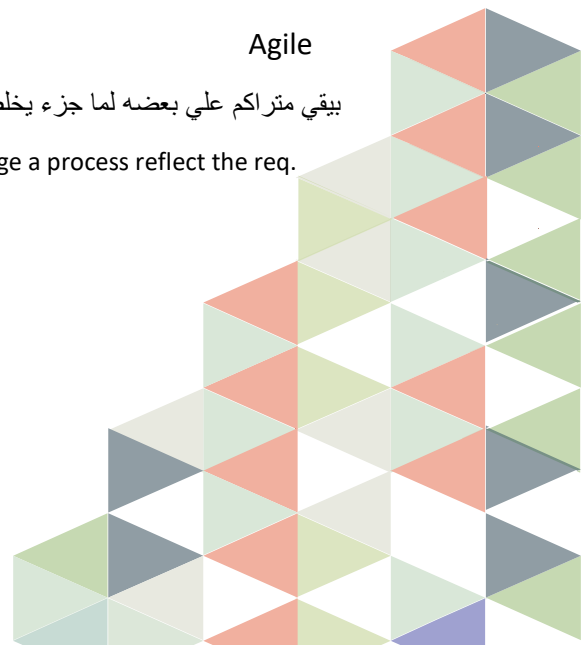
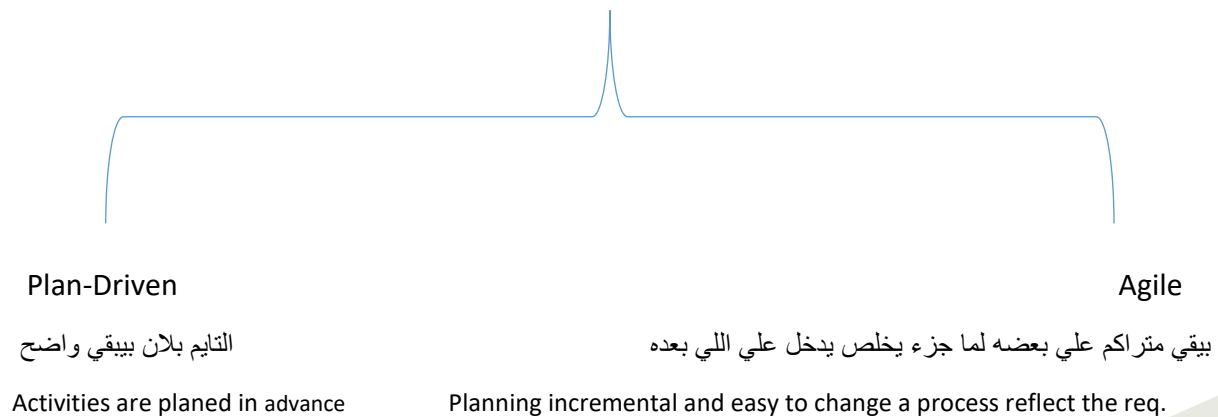


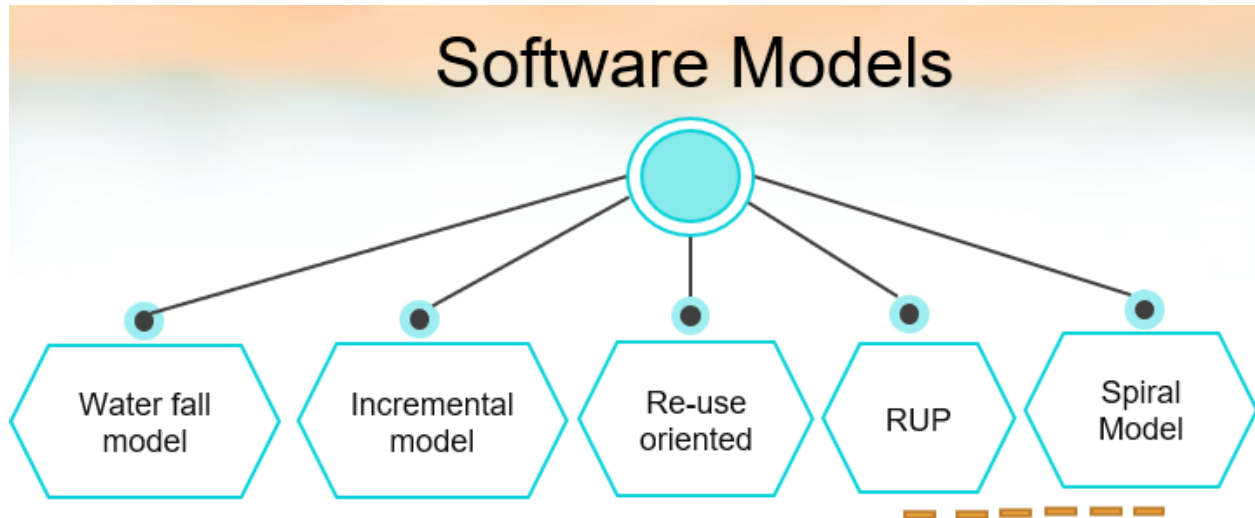
## Software **Activities**



- **Software process** : Are sequence of **activities** to produce needed software .

## Software process





- **Water fall model** : plan-driven .. يحدد وقت لكل تاسك و كل اكتيفتي يخلصها يدخل علي اللي بعدها
- Requirements are **distinct** from design and development.

### Advantage

- كويس للأنظمة الثابتة الكبيرة اللي لها فروع .
- كويس للأنظمة اللي فيها تغيير متطلبات .
- سهل عمله **Documentation** .
- الـ **Recruitments** بتبقى واضحة من البداية .
- وقت التوصيل مش بيبقي مشكله لانه بيبقي متحدد في الأول .
- **-Has staple frame work-**

### Disadvantage

- العمليات يتمشي بالتسلسل في لو عايز - غير حاجه في المتطلبات لازم امشي من الأول بنفس التسلسل .
- مينفعش اعمل تجزئه للعمليات -
- اليوزر مش بيستخدمه غير لما يخلص -



# Software Engineering

## Mind Map

- **Incremental Development** : Can be agile or plan-driven
- Requirements , development, and validation are **interleaved** متداخلة

### Advantage

- مناسب للمشاريع الصغيرة و المتوسطة
- ال. Requirements يتغير عادي
- مش لازم ال Customer يستني السيستم يخلص عشان يقدر يستخدمه
- ال. Customer بيشف السيستم بيتطور قدامه

### Disadvantage

- بياخد وقت طويل لو functions كثيره
- مش يقدر اشوف التقدم في ال process
- بالتالي مش يقدر عمله Document

- **Re-use oriented** : Can be agile or plan-driven
- the system is assembled from **existing** component .
- It focused can integrating the parts rather than developing them from scratch.

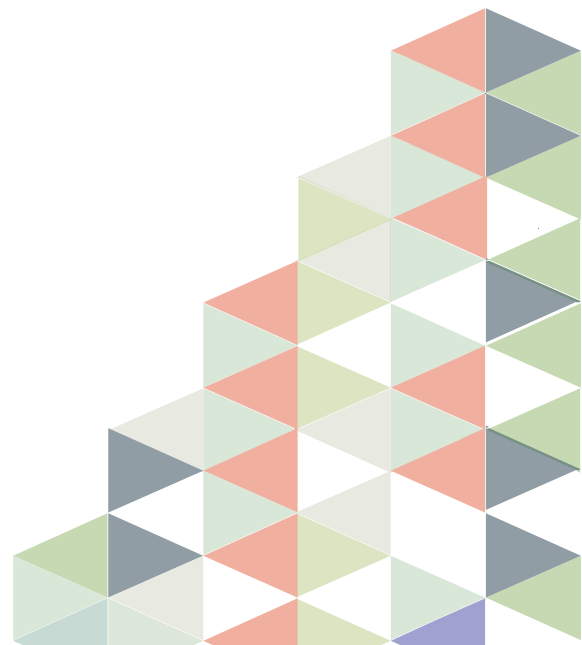
### Advantages

- Reduce risks.
- Faster delivery of SW .

### Disadvantage

- Can't be suitable for my system
- Copy rights

كده خلصنا المحاضرة الثانية





## Agile

ثالث و رابع محاضره معنا بيتكلموا عن الـ

طيب كنا اتكلمنا قبل كده عن الـ SoftWare process واللي كان في منها نوعين plan-driven و Agile  
بس قبل ما نتكلم عن الـ Agile عايز أوضح ان انا كـ developer ممكن اضحي ب requirements عشان الـ sw بتاعي يخلص  
اسرع و من هنا ظهر مصطلح **Rapid delivery of the Software** : هي ان انا ممكن اطلع السيستم يكون في وقت سريع و يكون  
فيه اهم النقط اللي انا محتاجها .

### Rapid Software Development Fundamental Characteristics

- **The processes of specification, design, and implementation are interleaved.**

.. مش عايز details كتيره و الديزاين بيفي بسيط ممكن استعمل حاجات جاهزه و في الـ user requirements اهم الحاجات اللي انا محتاجها و بس .

- **The system is developed in a series of versions.**

السيستم بتاعي هينزل علي هيئه فيرزونز مين اللي يقد يشوفها و يخليها اعدل فيها 1 ( 2 Stack holders ) End user .

- **System user interfaces**

ان انا اقدر استخدم interactive development system يخليني اعمل interface (gui) بطريقه سريعه زي انه يخليني اعمل  
drag and drop للـ icons بتاعتي في الـ interface . ( c# مثلا افكرها كده )

## Principles of agile

1 اهم حاجه ارضي العميل انه يشوف السيستم شغال قدامه

2 كل أسبوعين ل ثلاثه بالكثير لازم اطلع فيرجن جديد و كل version جديد بتحقق Requirement جديده باخدها من اليوزر  
( **Incremental Delivery** ) .

3 الكاستمر يشتغل مع الديفيلوب ( **Customer Involvement** ) .

4 ببقا مؤمن مبهارات الـ Development team و اسبيهم بطوروا اساليهم الخاصه ( **people not process** )

5 اوفر للـ Developers اللي عندي كل الـ **support** اللي محتاجينه عشان يقدمولنا system كويس .

6 اقبال الـ Team بتاعي Face To Face .. لو هطلب Task اطلبها فيس تو فيس مش ابعثها Mail .

7 الـ Design بتبقي Simple و بحاول مضيعش وقت فيه .

8 بخلي الـ System بتاعي بيفي simple وانا شغال علي الـ Development ( **maintain simplicity** ) .

9 بتوقع دايمًا ان الـ Requirements بتاعتي هتتغير و ابتدي وانا شغال اخليه جاهز لكده ( **Embrace change** )

امتي الـ **Agile** بيبقي **successful** ؟؟

- (1) لو الـ system اللي شغال عليه Small او Mediem .
- (2) لو الـ customer مهم عندي .
- (1) لو مش بهتم بالـ Documentaion لأن الـ Agile شايف ان الـ DOC تضيق للوقت لانه كل شويه ممكن تتغير حاجات فبيعوضها بتبسيط الـ Code.

طيب امتي بيبقي **صعب** إني اشتغل Agile ؟؟

- (1) لو صعب الـ customer ينزل كل يوم و يتقابل لاني بحتاج ببقى فيه clear commitment من ناحيه الـ customer .
- (2) لو في members في الـ team بتاعي مش عايزه تشتغل في team و عايزه كل واحد ياخد Task لوحده يعملها و خلاص .
- (3) لو في عندي كذا stakeholder هيبقي الموضوع صعب لان ممكن يحصل خلاف بينهم وانا بشتغل ف الـ Agile بناءا علي طلباتهم .
- (4) الـ code لازم ببقى مفهوم وواضح ..ف ممكن الـ Developer ميلحش يعمل simplify للكوذ عشان أي حد بعديه يقدر يقرأه و يفهمه.
- (5) لو في customers عايزه تيجي تمضي عقد و تستلم الـ system في وقت معين .. لأن ده ضد **Principles of Agile** اللي اتكلمنا عنها فوق لأنني عايز الـ customer معايا علي طول .

امتي اشتغل Agile و امتي اشتغل Plan-driven ؟؟

plan - driven	agile
لو محتاج موافقه من جهات خارجيه	لو في incremental strategy و الـ customer شغال معايا.
لو الـ Design فيه تفاصيل كثيره	لو الـ members معايا ف نفس المكان مش بشتغل اون لاين.
لو كل واحد عايز يخلص التاسك بتاعته لوحده ويسلمه ف وقت معين	لو محتاج Analysis قبل الـ Implementaion .
لو شغال علي large project و كذا حد شغال من كذا دوله مثلا	لو في <b>support</b> متاح للـ Developers بتوعي .
لو المنظمه تقليديه و ماشيه بتخطيط و تايم لاين معين	لو في عمل جماعي و الـ members متفاهمين .
لو محتاج Long-life documentaion	الـ Agile محتاج ناس عندها قدرات و خبره عاليه .



هنتكلم دلوقتى عن اول approach للـ Agile و يعتبر هو الـ most widely used هو الـ

## Extreme programming (XP)

طيب ليه اسمه كده ؟؟

- pushing recognized good practice, such as iterative development, to 'extreme' levels

## Principles of Extreme programming

1ال xp بيركز علي الـ planning و بيهتم بالـ customer و بيعتبروه جزء من الـ team (On-site customer) وظيفته انه يجيب الـ Requirement بتاعته جدا لدرجه انه بيسميها user story و بيحطها في story card و بيتعامل معاها علي انها sequence of tasks و بيبتدي ينفذها (Incremental Planning)

2اول لما بخلص task بعملها integration مع باقي الـ system كله (continuous integration)

3 كل شويه بعمل release و اضيف functionality علي الـ release الأولاني (Small release)

4بخلي الـ Design يكون easy implemented عشان لو في future change هيحصل (simple design)

5زي ما قولنا ان الـ xp بيهتم بالـ planning ف هو ممكن يعمل test ل functionality معينه قبل ما يعمل development يعني بيشف سيناريو و بيتدي ي test السيستم عليه (Test-first development)

6 بحسن دايمنا في الكود بتاعي ولو في أجزاء متكرره في الكود او ملهش لازمه او مستخدمها بطريقه غلط زي الـ data structures و اسامي الـ attributer اللي في الكود (Refactoring)

7لو حاجه حصلت للـ project بتاعي كل الـ team بيثيل المسؤوليه ومحدث يرمي الحمل علي التاني (Collective ownership)

8ال xp مش بيقبل ان يبقى في large amounts of over time لانه شايفها بتقلل من الـ code qualite وال medium term productivity بتاعته (sustainable pace)

9 طبعا من اسمه ف هو extreme ف هو فعلا كده .. ف بيخلي كل اثنين من الـ team بتاعي يشتغل مع بعض كـ pair واحد و في الاخر بعد ما كله يشتغل بعمل integration للشغل كله مع بعض و اعلمهم test كانهم حاجه و احده (pair programming)

وده ادي الي : ان عدد الـ errors بقا يقل و الوقت اللي بقضيه كمان وانا بعمل testing .. مبقتش بعدل كثير ولا ببدا بدايات غلط .. ده طبعا غير الـ sharing of knowledge اللي بيحصل .. وبكده ابقا قللت المخاطر علي الـ project بتاعي لو كل الـ team مشي



كده خالصنا اول approach بس قبل ما ندخل في ال approach الثاني عايز أوضح حاجه بالنسبه للـ **Agile**

## Agile Project Management

لو انا عايز ا management أي project شغال Agile هيبقي **صعب** و ده علشان

\*ال Requirement معموله incrementa \*دائما بيبقي في Rapid increment \*في changing في ال Requirements

طب **اعدل** ايه علشان اقدر اعمل Management للـ Agile project

- اوفر Time و Resources للـ Team بتاعي
- اعمل adapt مع ال incremental و برکز علي النقط المهمه في ال Agile

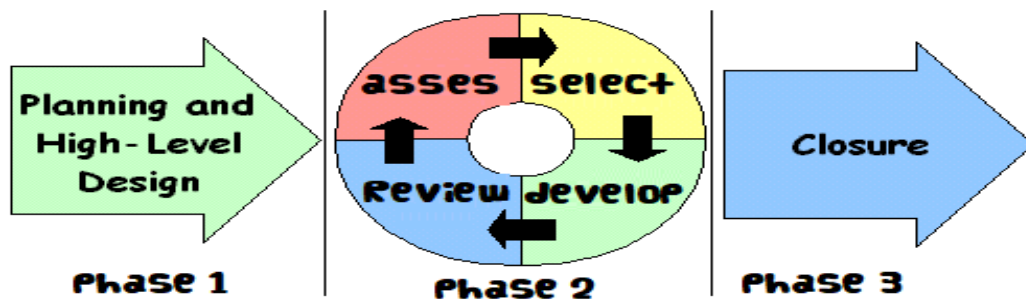
## Scrum

تاني approach معنا هو الـ

وده بيرکز علي ال Managing iterative development اكثر من تركيزه علي ال Agile و كمان بستخدمه في ال xp علشان بيقدملي Management frame work للـ project بتاعي .

الـ **Sprint cycle**

من اكثر اللي بيميز ال scrum هي



**FIXED LENTH DURATION 2-4 WEEKS**

**Phase 1 :** put outline planning and simple starting design .

**Phase 2 :**

Assess : Features are selected to development and the software is implemented.

Develop : most important stage produce new version at the end of every cycle.

Review : stack holders and end-user will see the new version.

**Phase 3 :** mistakes to avoid it.



كل الحاجات دي بتتخط في حاجه اسمها **backlog** بيبقا فيها ال project details و خلصنا قد ايه و Req



و بيشرف عليها و ال-Scrum Master و بيبقا هو الوحيد اللي بيقدر يتعامل direct مع ال customer ودي المهام بتاعته :

▣Arranges daily meetings

▣Tracks the backlog of work to be done

▣Measures progress against the backlog

▣Communicates with customers and management outside of the team

▣Records decisions( هو بيسجل القرارات اللي اتفقوا عليها لكن مش بياخد قرارات عشان كده اسمه ماستر مش مانجر )

## Principles of Scrum

daily ~15 minutes meetings, which are sometimes 'stand-up'

to Share information and know What did you do yesterday?

What obstacles are in your way?)

What will you do today?)

وبكده بقا

everyone on the team knows what is going on

## Advantages of Scrum

1. The product is broken down into a set of **manageable and understandable chunks**.
2. **Unstable** requirements **do not hold up progress**.
3. The whole **team** has **visibility** of everything and consequently team **communication is improved**.
4. **Customers** see **on-time delivery** of increments and **gain feedback** on how the product works.
5. **Trust** between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.  
لأنه شاييف ال backlog وال project بتاعه بيتطور قدامه

# Scrum Framework

## □ Roles

- Product owner *desides what kind of software product it to be*
- Scrum Master
- Team

## □ Ceremonies

- Sprint planning
- Sprint review *What is accomplished during the spring*
- Sprint retrospective *What good .. what bad . .. what can be improved to improve the sprint which will be held in the future*
- Daily scrum meeting

## □ Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

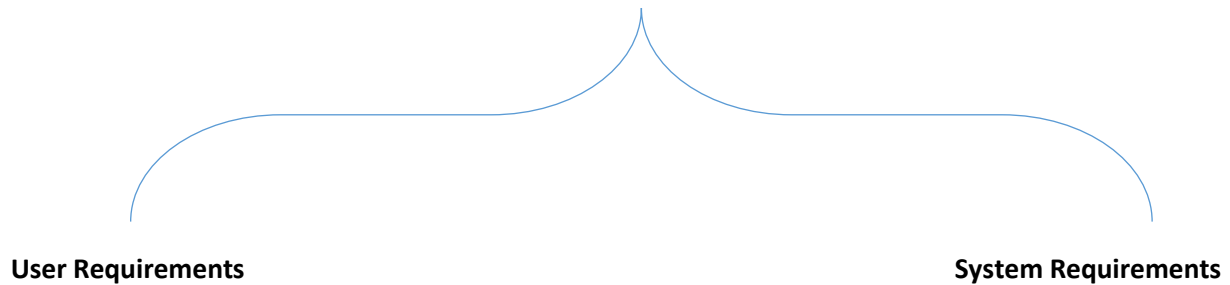




## Requirements Engineering

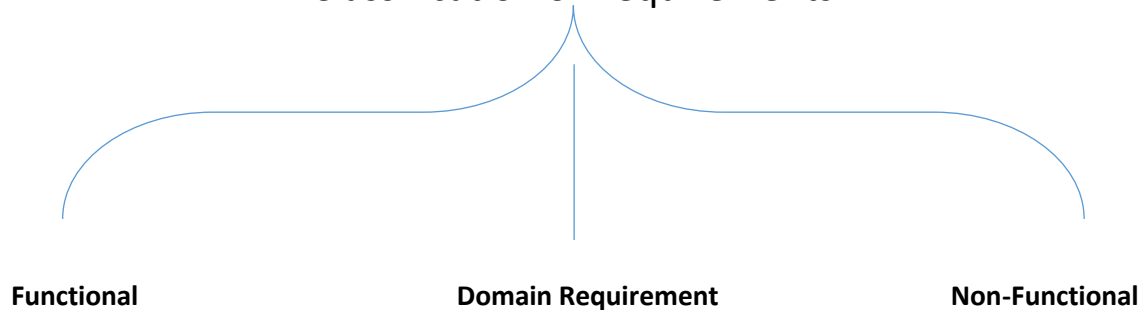
- Over all process to collect and analyze requirements.

### Types of Requirements



- **User Requirements** : Natural languages and diagrams of services the system provides .
- **System Requirements** describe system's functions, services and operation .
- **User Requirements + System Requirements** = MAIN REQUIREMENTS of THE SYSTEM .

### Classification of Requirements



- **Functional** : service provides by the program . what are its functions and operation? . . How does it behave in different situations?
- **Non-Functional** : describe constraints and functions like speed and security ...
- **Domain Requirement** : constraints of the system from the domain of operation

و ذكرنا مثال ان قولنا مثلا لو انا في السيستم المستشفى ف هنا في دومين ان المريض مينفعش ياخذ فوق 3 جرعات من دواء معين عشان غلط عليه ف هنا هي مش فانكشنال عشان اليوزر مقاليش عليها و مش نون فانكشنال زي السرعه و المساحه و كده



## \* Important \*

Goal	Non-Functional Req.
<ul style="list-style-type: none"><li>هدف معين عايز اوصله</li><li>مش يحدده بوقت</li></ul> <p>Ex: Ease of use سهوله استخدام</p>	<ul style="list-style-type: none"><li>الخطوات اللي يعملها عشان احقق الهدف ده</li><li>بيحدد بوقت معين</li></ul> <p>Ex: Stack holders can use the system after <u>4 hours</u> of training</p>

## Metrics for non-functional Req.

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

## non-functional Req. classifications





## Requirement Engineering processes

Feasibility study : بشوف هل هقدر اكفي احتياجات اليوزر بالتكنولوجيا الحاليه اللي معاها سواء هارد وير او سوفت وير ببصلها من منظور البيزنس

Requirements Elicitation استنباط and analysis : delivering the system requirements through *observation* of existing systems, *discussion* with users and *task analysis*.

Requirements specification: Requirements are documented.

Requirements validation: check for errors and conflicts in req.

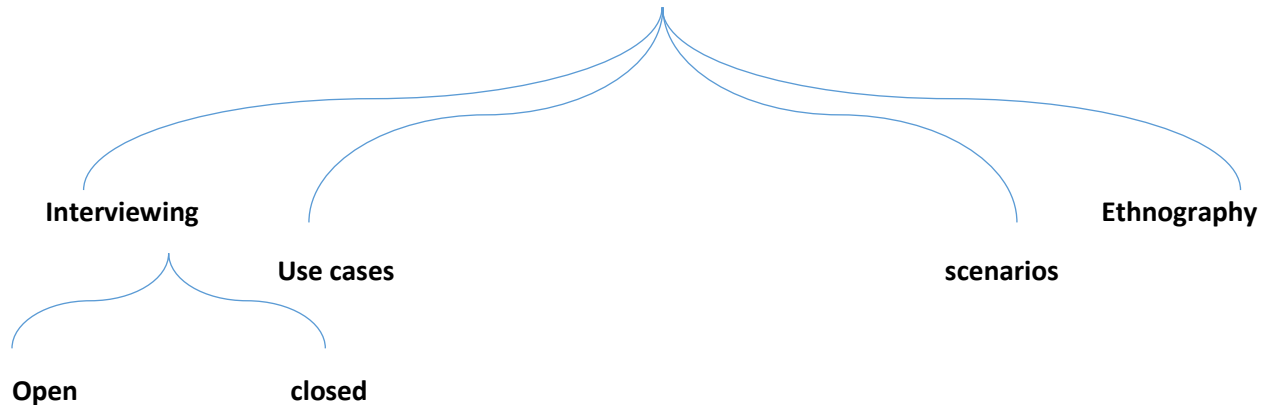
### Requirements Elicitation and analysis

- 
- ```
graph TD; A[Requirements Elicitation and analysis] --- B[1) Req. discovery]; A --- C[2)Req. Classification]; A --- D[3) negotiation]; B --- B1[By interactive with stack holders]; C --- C1[create relations among Req.]; D --- D1[negotiate about plan & cost];
```
- 1) Req. discovery  
By interactive with stack holders
  - 2)Req. Classification  
create relations among Req.
  - 3) negotiation التفاوض  
negotiate about plan & cost





## Methodology for Discover and Collect Requirements



### Interviewing

- Helps to get overall understanding of what stakeholders do and how they might interact with the system **NOTE : not effective to discover domain req.**

**Open :** ببقا موجود ف مكان الشغل و بكتشف المشاكل مع الستاك هولدر **Closed :** اسئلته محدده ببقا مجهزها

- في مشاكل بتقابلني وانا بجمع المتطلبات ان بيحصل تضاد في المتطلبات وان الستاك هولدرز مش بيوضحوا الدنيا اوي زي مثلا موظف شنون الطبله لو قولتله ع الاهتمام هيقولك دي ورقه و بتملاها و خلاص لكن مش هيجبك التفاصيل

### scenarios

- Scenarios are **real-life examples** of how a system can be used.
- They should include
  - A description of the **starting** situation;
  - A description of the **normal** flow of events;
  - A description of what can go **wrong**;
  - Information about other **concurrent activities**;
  - A description of the state when the scenario **finishes**.

### Ethnography

Analyzing how people actually work

- Effective in two types of req.** - لو عندني وعي عن نشاطات الناس - لو المتطلبات هتوصلني من اني اشوف الطريقه الفعلية اللي الناس بتادي بيها الانشطه احسن من انهم يحكولي ف ساعتها هيبقي مفيد

طيب انا دلوقتي جمعت كل المتطلبات بتاعتي ف هكتبها بقا ف دوكمنت بالترتيب و المحتوي ده

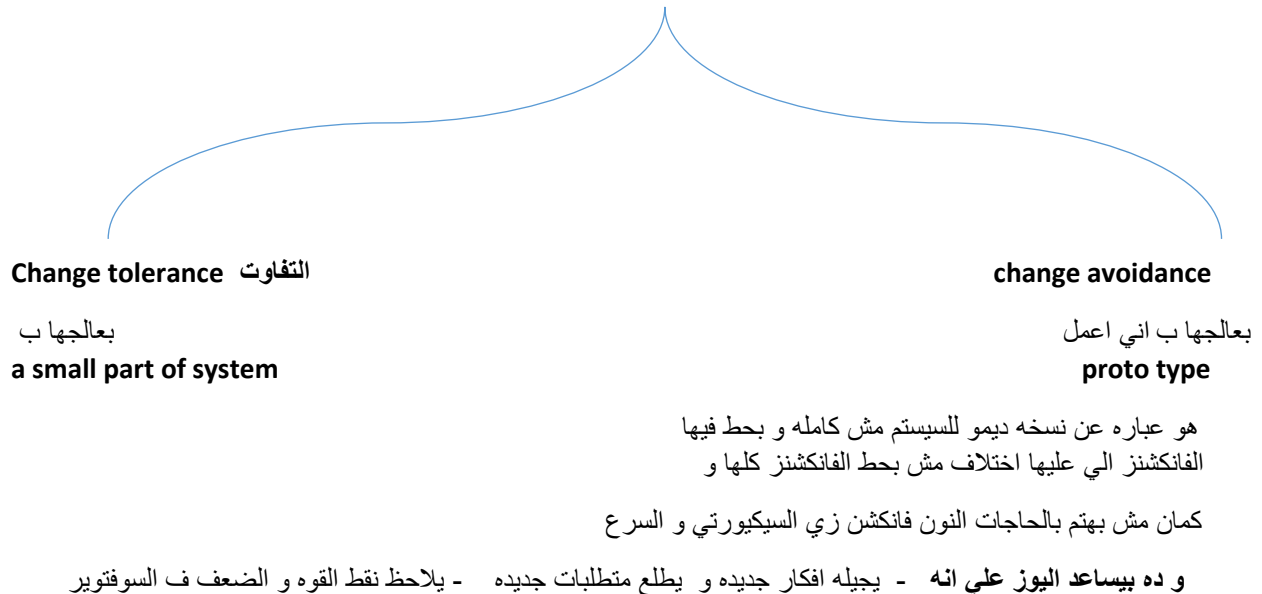
| Chapter                           | Description                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Preface                           | This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.                                                                                                                                                               |
| Introduction                      | This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.                                                                        |
| Glossary                          | This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.                                                                                                                                                                                                                         |
| User requirements definition      | Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.                                        |
| System architecture               | This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.                                                                                                                                                |
| System requirements specification | This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.                                                                                                                                                    |
| System models                     | This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.                                                                                                                                        |
| System evolution                  | This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.                                                |
| Appendices                        | These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data. |
| Index                             | Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.                                                                                                                                                                                                           |

## وفي كمان 5 طرق تانيه نعمل بيهم دوكمنتيشن للمتطلبات بتاعتي

| Notation                     | Description                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Natural language sentences   | The <u>requirements are written using numbered sentences</u> in natural language. Each sentence should express one requirement.                                                                                                                                                                                                                                   |
| Structured natural language  | The requirements are written in natural language on a <u>standard form or template</u> . Each field provides information about an aspect of the requirement.                                                                                                                                                                                                      |
| Design description languages | This approach uses a language like a <u>programming language</u> , but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.                                                                                           |
| Graphical notations          | Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; <u>UML use case and sequence diagrams</u> are commonly used.                                                                                                                                                                                   |
| Mathematical specifications  | These notations are based on mathematical concepts such as <u>finite-state machines or sets</u> . Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract. |

طيب كده المحاضره الرابعه خلصت بس كان فيه جزء من المحاضره التانيه حبيت أقوله هنا عشان ليه علاقه بالموضوع عشان يبقي التسلسل مفهوم ..

## Problem of changing system requirements



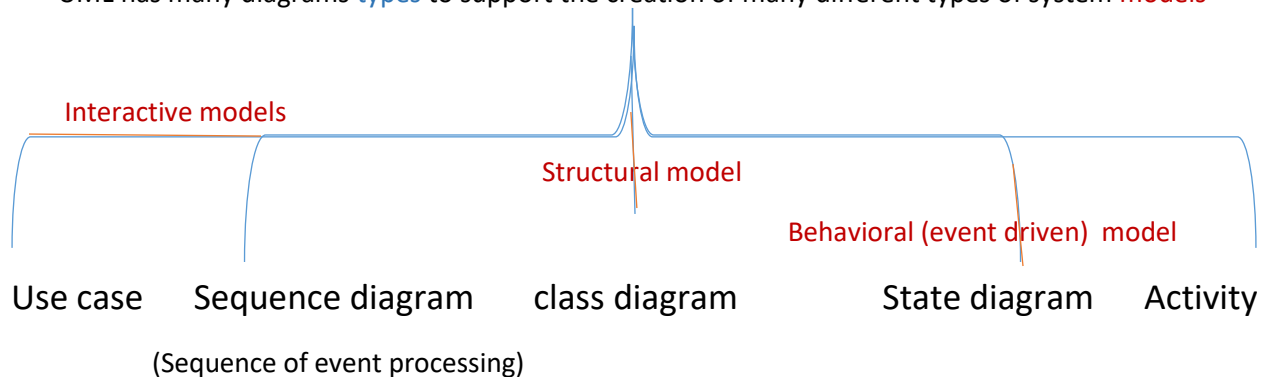


# Software Engineering

M i n d   M a p

## UML

UML has many diagrams **types** to support the creation of many different types of system **models**



**Interactive models :**

- ▣ User interaction, which involves user inputs and outputs (helps to identify **user requirements**)
- ▣ Interaction between the system being developed and other systems (highlights the communication problems that may arise)
- ▣ Interaction between the components of the system (helps us understand if a proposed system structure is likely to deliver the required system performance and dependability) (Sequence diagrams). but.

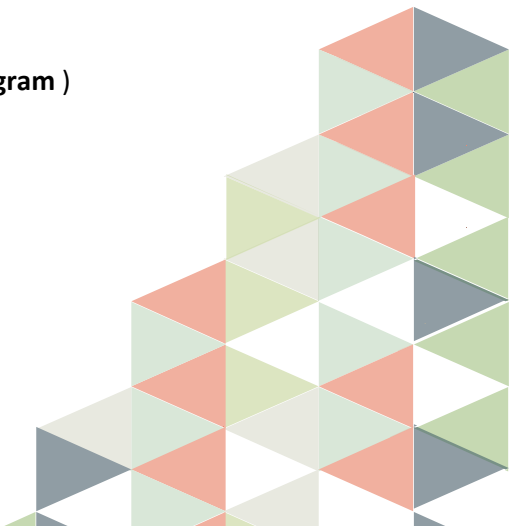
Use case modeling, which is mostly used to model interactions **between a system and external actors** **users or other systems**

**Behavioral model :**

- Data flow معتمد علي الداتا اللي جياي ( **Data-Flow model** )
- Event- driven معتمد علي الايفينت اللي جياي ( **State diagram** )

**Structural model :** Define structure and relationships between classes .

**Semantic model :** Define associate relation only and has no operation .



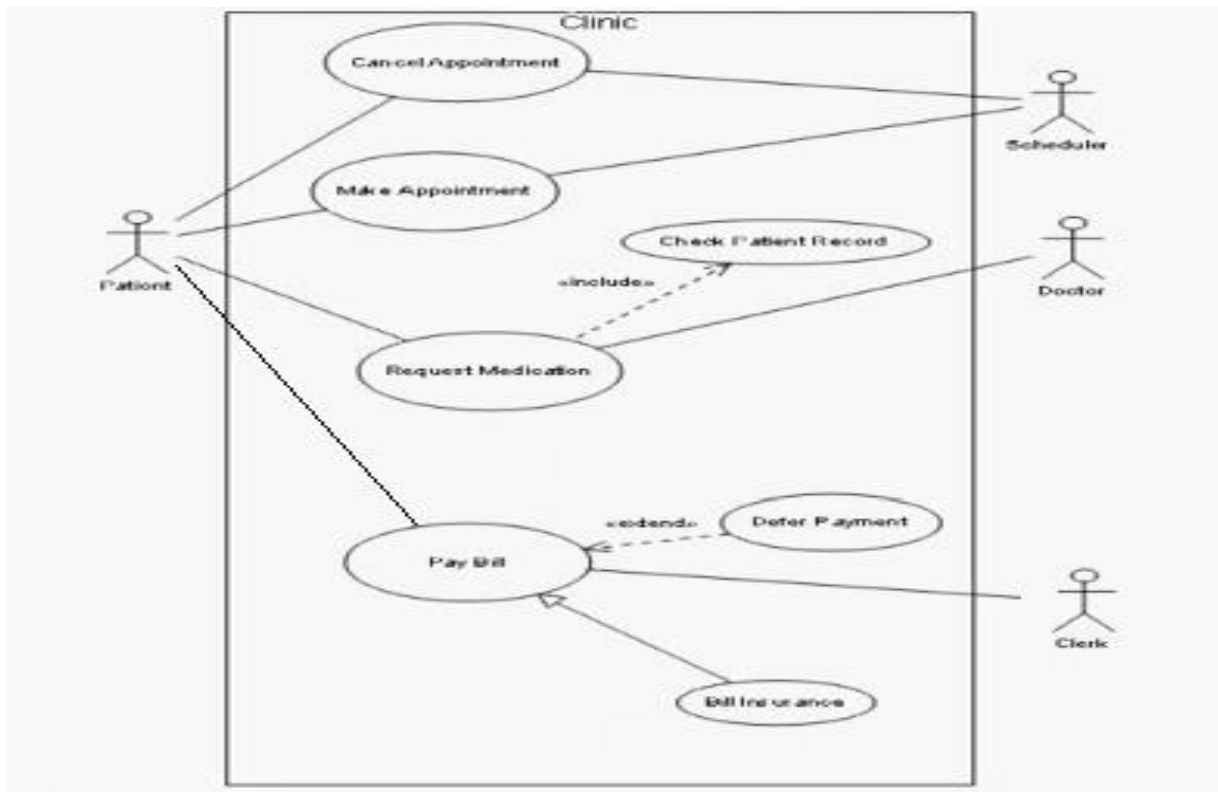
# Use Case

## Notations



Association      X Generalization Y      X Include Y      X Extend Y  
ارتباط      يعني اكس حالة خاصه من واي      يعني المكان اكس بيحصل جواه الاوبيريشن واي      يعني اكس امتداد ل واي

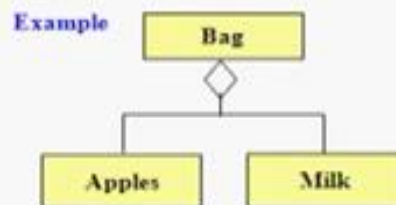
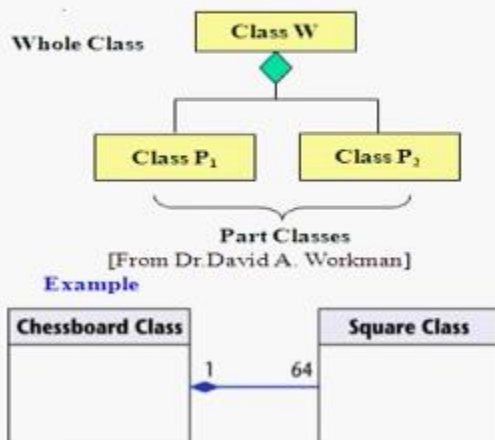
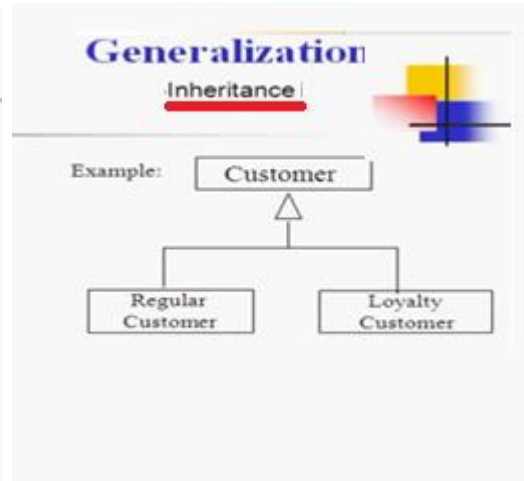
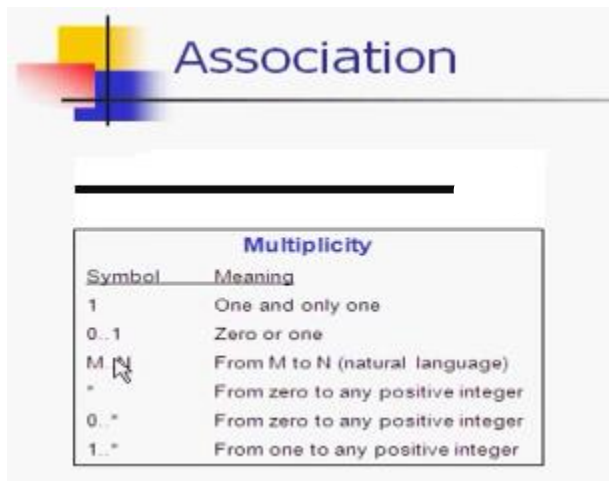
## EXAMPLE



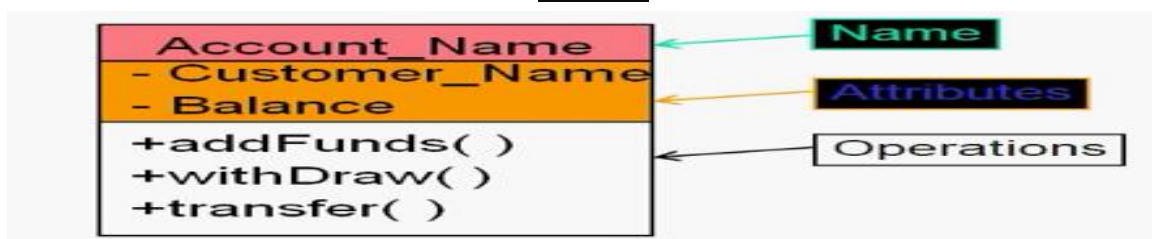


# Class Diagram

## OO Relations



## Class



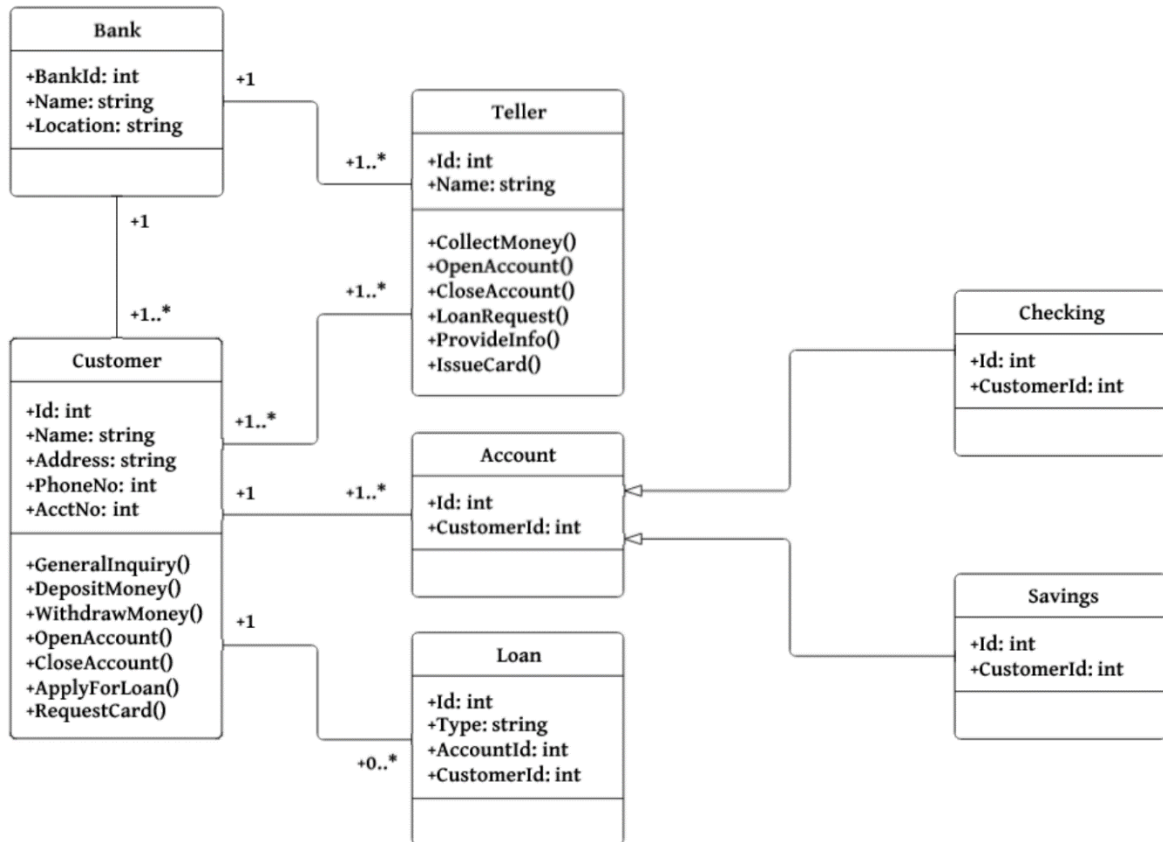
# Attributes

+ public (every one) # protected( can be called by other classes) - private ( no one can call it if itself not ) / derived

**NOTE** by default attributes Are hidden and operations are visible

## Example

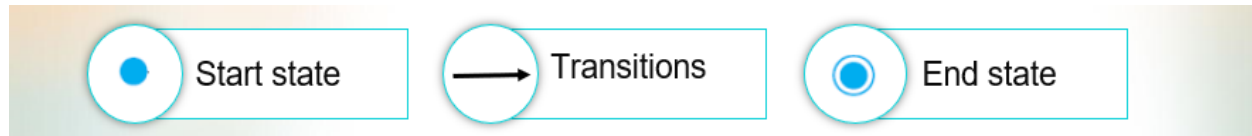
المثال ده للتوضيح بس



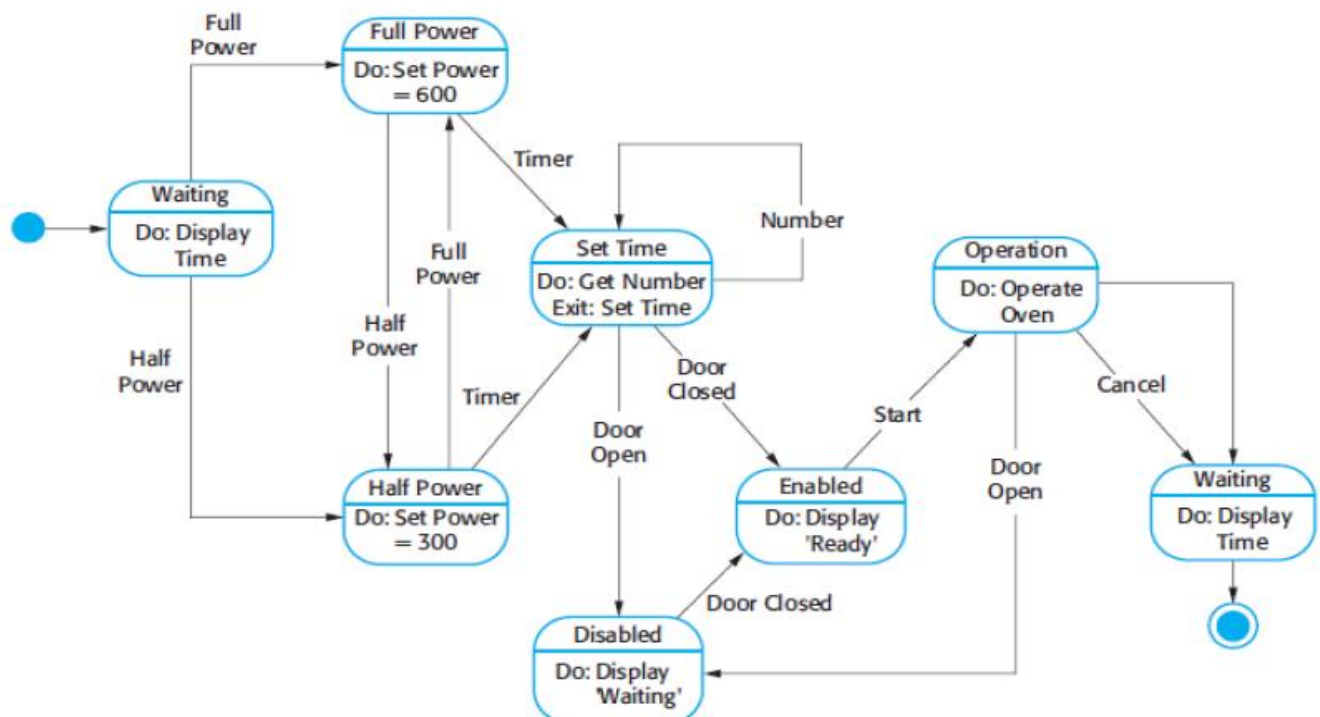
# State Diagram

Example for Event- driven

## Notations



## Example

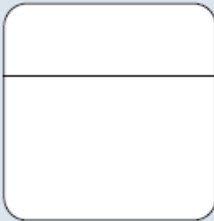







# Data Flow Diagram

- Is used to describe the data exchange between a system and other systems in its environments.
- May be used to model the system's data processing From a functional representative.

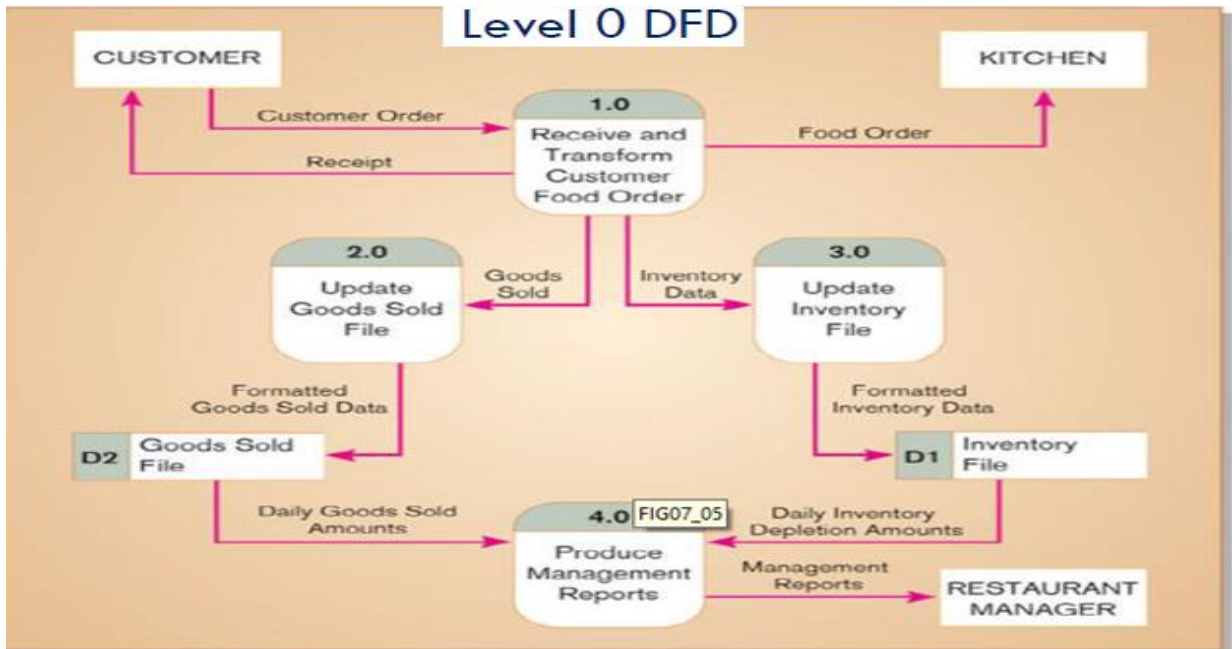
## Notations

| Element                       | Notation                                                                            | Representation                                                                                                                                                                  |
|-------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Process                       |    | Work or actions performed on data (inside the system)<br>Labels should be verb phrases<br>Receives input data and produces output                                               |
| Data Flow                     |  | Arrows depicting movement of Data<br>Labels should be noun phrases                                                                                                              |
| Data Store                    |  | It is used in a DFD to represent data that the system stores (inside the system)<br>Labels should be noun phrases<br>Must have at least one incoming and one outgoing data flow |
| Source/Sink (External Entity) |  | External entity that is origin or destination of data (outside the system)<br>Labels should be noun phrases                                                                     |

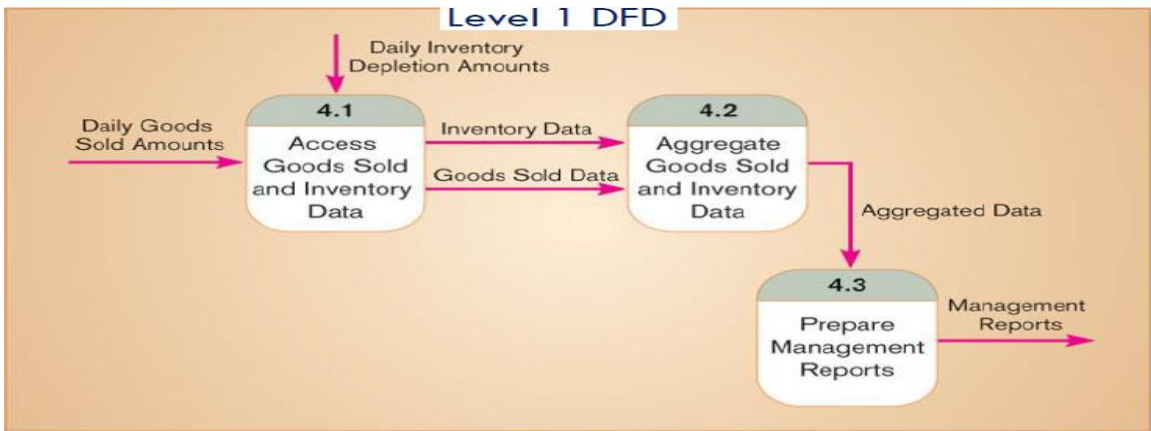
## Example



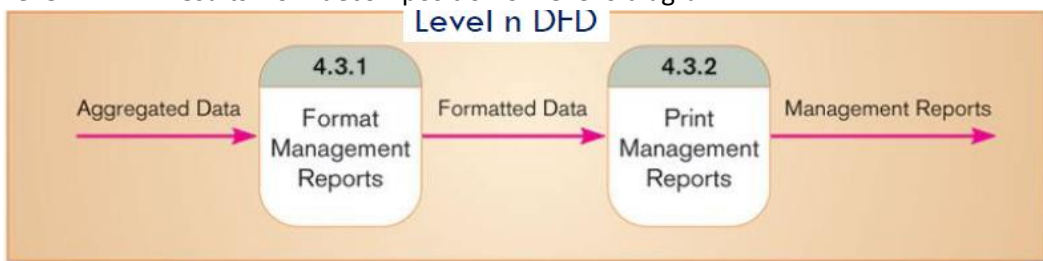
Context diagram : Overview of the system and show boundaries , external entites that interact with the system and mojour information flows between entities and the system



- Level 0 DFD : Representation of system's major processes at high level of abstraction .



- Level 1 DFD : Results from decomposition of Level 0 diagram .



- Level n DFD : Results from decomposition of level n-1 diagram .

## Important RULES

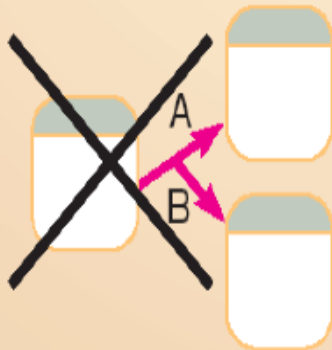
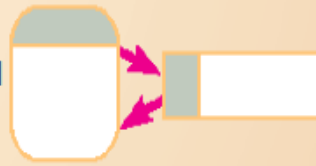
\*



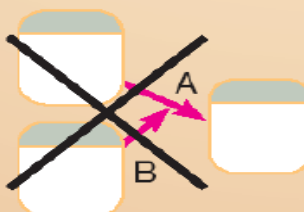
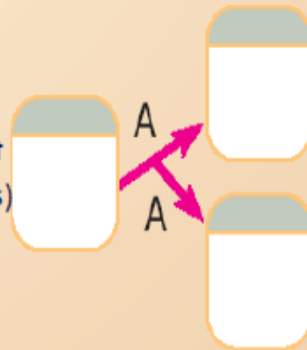
No data moves directly between external entities without going through a process.



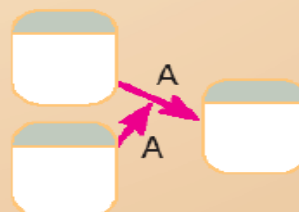
Bidirectional flow between process and data store is represented by two separate arrows.



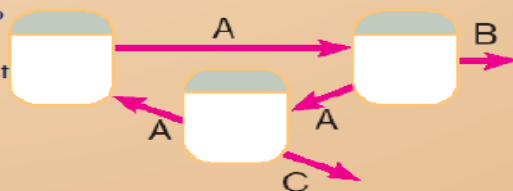
Forked data flow must refer to exact same data item (not different data items) from a common location to multiple destinations.



Joined data flow must refer to exact same data item (not different data items) from multiple sources to a common location.

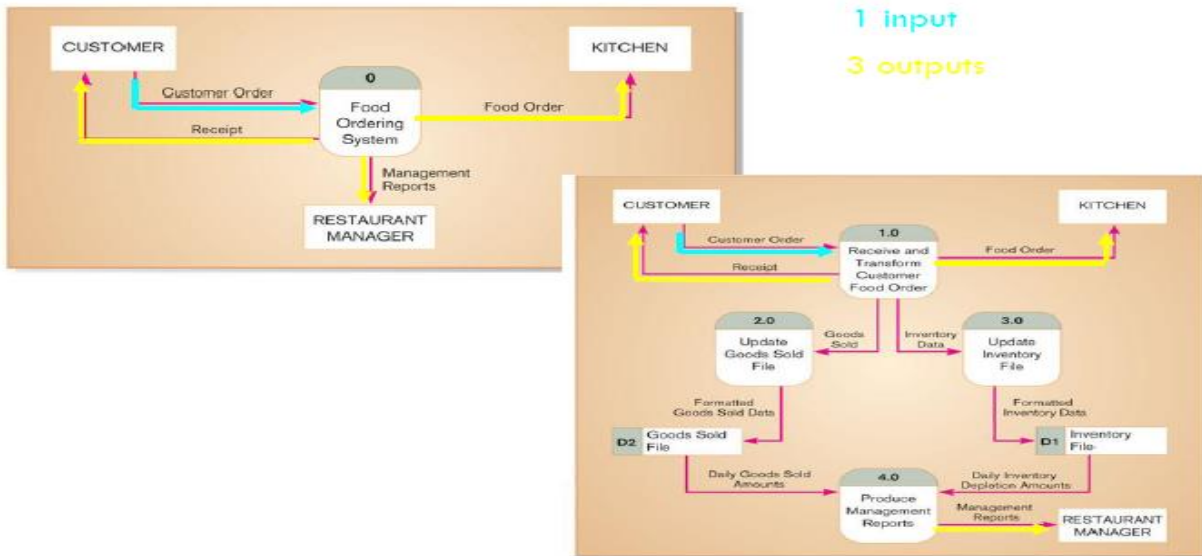


Data flow cannot go directly from a process to itself, must go through intervening processes.

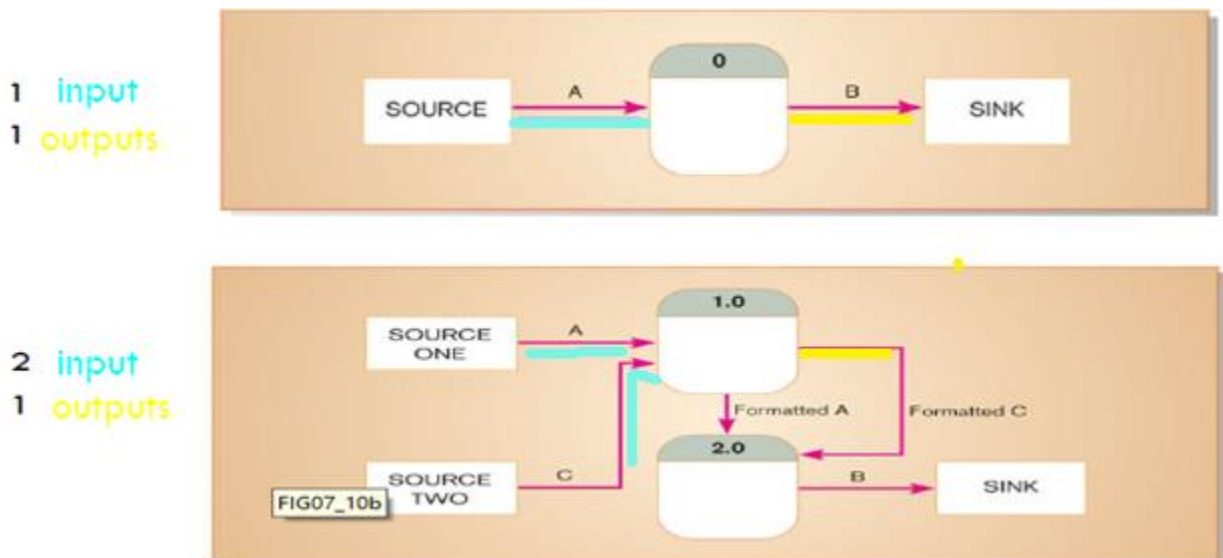


## Balanced or NOT ?

- **Balanced** : if #inputs & outputs to lower level DFD equals # inputs & outputs to High level DFD



- **Not Balanced**



D: كده الحمد لله خلصنا كل اللي قبل الميد ما عدا المحاضره الثالثه ^^ دعواتكم



## Software Testing

is the process to help identify the **correctness**, **completeness** and **quality** of the developed software.

\*في مصطلحين لازم اخذ بالي منهم\*

□A **good test case** is one that has a high probability of finding an as-yet undiscovered error. يعني بيبقي احتمال كبير انها تكتشف ايور

□A **successful test** is one that uncovers an as-yet-undiscovered error عدت عادي من غير ما تكتشف ايور

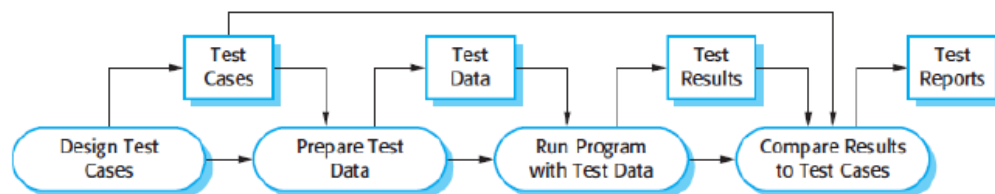
\*في مصطلحين كمان لازم اخذ بالي منهم\*

**Verification:** "Are we building the product right". اني شغال بالطريقه الصح

**Validation:** "Are we building the right product". اني شغال علي البروجيكت الصح

## A Model of the Software Testing Process

اني بعمل بلان قبل التيست ف ب ديزاين التيست كيس و اجهز الداتا اللي هتستها و ارن البروجرام بالداتا دي و اقارن اللي طلعتي باللي المفروض يطلعتي و بعدين اقدم ده في تيست ريبورت





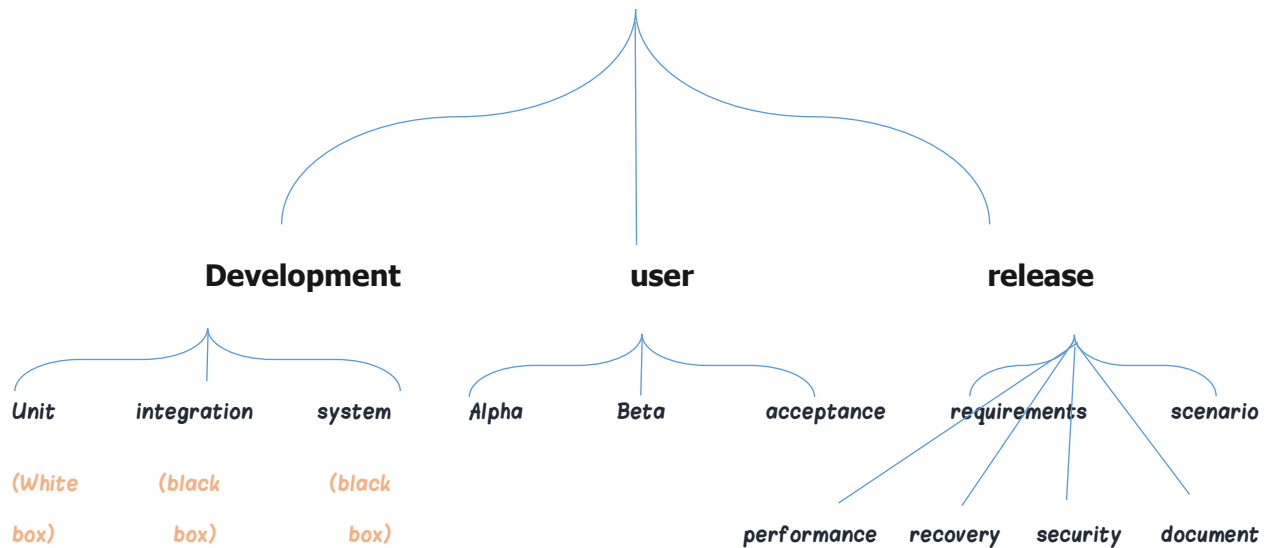
## Testing Principles

- ☐ All tests should be **traceable to customer requirements**. (Why?)
- ☐ Tests should be **planned** long before testing begins. (When?)
- ☐ Testing should begin “**in the small**” and progress toward testing “**in the large.**” (How?)
- ☐ **Exhaustive testing** is not possible. (Why?) يعني صعب اني اعمل تيست كامل 100%



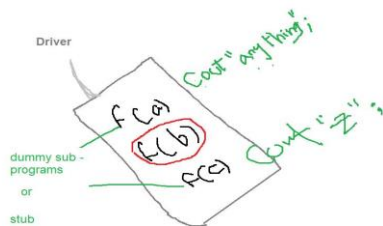


## Testing Stages



### Development

ال unit testing : بتشيك علي الكود unit ب unit يعني سطر سطر ف و انا بعمل unit test فانا عايز مثلا اعمل تيست لفانكشن b بس هي مش لوحدها هي معاها انتين فانكشن كمان مثلا اللي هما a و c فعمل ايه ؟؟؟!!  
ف انا بمنتهي البساطه هخلي الفانكشنز a و c بيطلعولي اي قيم ملهاش تاثير عليا او اعلمهم كومنت مثلا و احطهم في الفيچوال و اعلمهم رن .....طيب اللي عملته ده بلغه السوفت وير اسمه ايه ؟؟  
الفانكشنز اللي ملهاش لازمه عندي اللي هي في حالتنا هنا هي a و c اسمها Dummy sub-programs و الفيچوال او اي حاجه بعمل عليها رن او main يعني بيبقي اسمها في السوفت وير driver بس كده



مليش دعوه باي حاجه غير ال انبوت اللي داخل و الاوبوت black box

بعضي سطر سطر علي الكود white box

الـ integration testing : يشوف ال interaction بين ال functions و بعضها يعني بين ال individual units و بعضها  
الـ system testing : يشوف ال integration ما بين whole system components

### user

الـ alpha testing : بخلص البروجيكت و ادبه لليوز يعمله test و ببقا واقف جنبه و يشوف ال errors و ال problems  
الـ Beta testing: بديله نسخه من البروجيكت و مش ببقى معاه و هو بيشتغل عليها و بيديني report كل فتره  
الـ acceptance : الـ user بيعمل الـ test علي البروجيكت في الـ environmen اللي بيشتغل فيها و يشوف هيبقي مقبول ولا لا

### Release

يعني بجيب independent third party يعني testing team من برا ال team بتاعي و مكانش شغال علي البروجيكت  
الـ requirements : بيـ check ال req. بتاعه الـ system  
الـ scenario : بيـشوف story معينه فيها استخدام الـ system وبتقي typical scenario من الـ real life وبيـ test بيها  
الـ performance: بحط الـ system تحت stress و اشوف ايه اللي هيخليه يـfail  
الـ recovery : بجبر وقتها الـ system انه يقع عشان اشوف هيقوم في قد ايه لانه هيسبب كده damage كبير و يشوف دول  
**reinitialization, checkpointing mechanisms, data recovery, and restart are evaluated for correctness**

