Software Engineering
PART 2

System Design :
Definition of Software Design :

ان فكرة تصميم السوفت وير مبنيه بالكامل على انتاج SRS والذى شرحناه فى الجزء الاول لكن بتفاصيل اكتر عن النظام

خصائص التصميم او Properties :

- Correctness ان يكون صحيحا
- Verifiability ان يكون قابل للتحقق من صحته
- Completeness ان يكون كاملا
- Traceability الاحتفاظ بأصل كل اصدار من الاصدارات التى تنتج عن التصميم
- Efficiency الكفأه وذلك باستغلال اقل موارد ممكنه من هاردوير و سوفت وير
- Simplicity السهوله والبساطه وهى اهم معيار من معايير الكفأه

Design Principles:

- **Problem Partitioning**

تجزىء المشكله بهدف التغلب عليها

- **Abstraction**

اعطاء رؤيه مجرده عن النظام ليكون لديك فكره عامه من مساره والفكره المجرده لاغنى عنها فى عمل Design فهى تسمح لمن يقوم بالتصميم بالتركيز على جزء من مكونات النظام او Component ويكون لديه فكره على كل مكونات النظام والنوع العمليه التى تقوم بها

- **Architectural Design**

ويمثل التركيب او Structure للبيانات و المكونات ويوجد ايضا اطار العمل الذى يحيط بالنظام او Framework

والان بعد سرد مجموعه من مبادىء التصميم نستطيع ان نعطى مثالا مبسطا عنه

Data Centric Architecture:

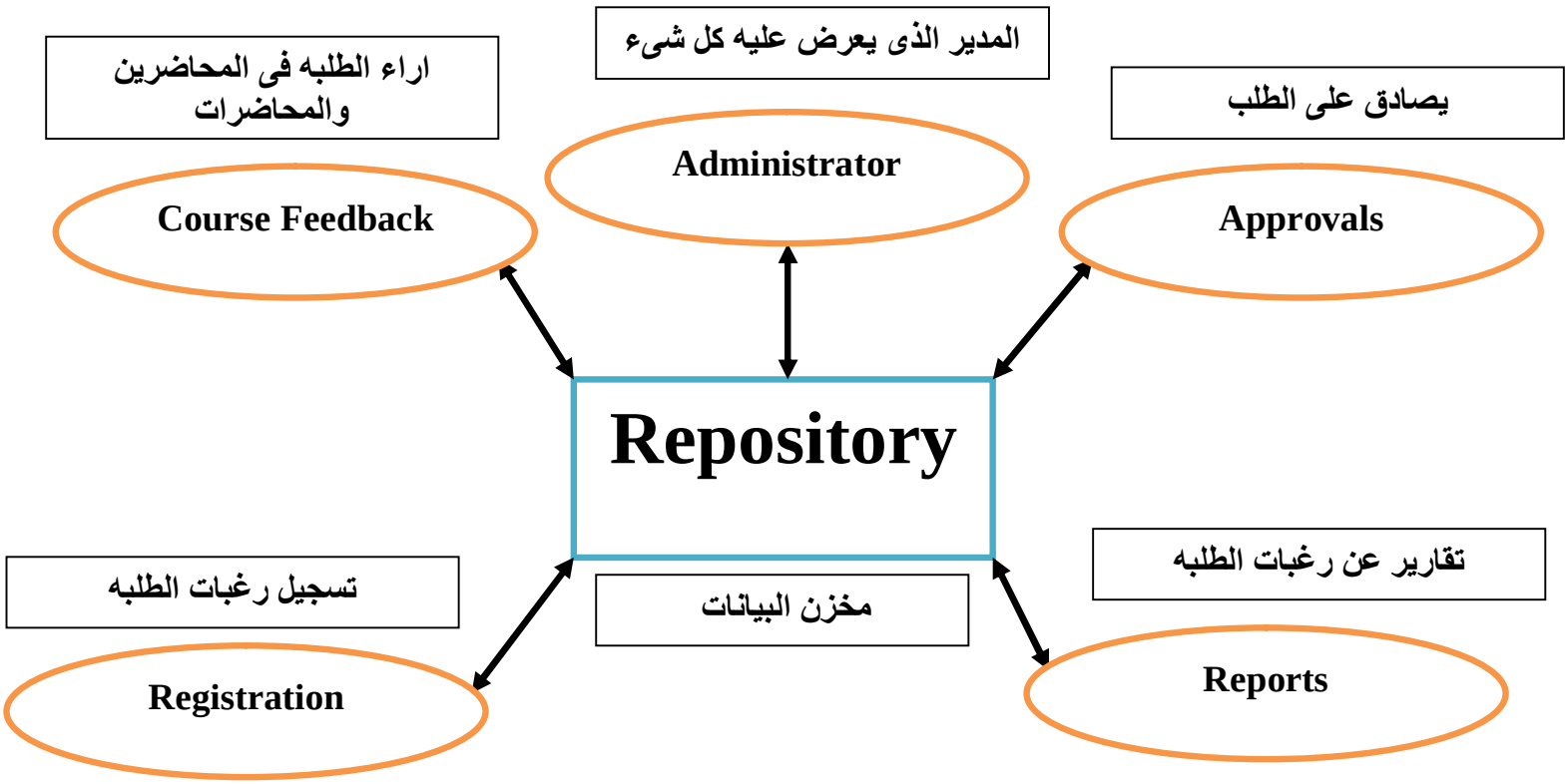
قاعدة البيانات المركزيه والتى يتعامل معها النظام او System

Data Flow Architecture:

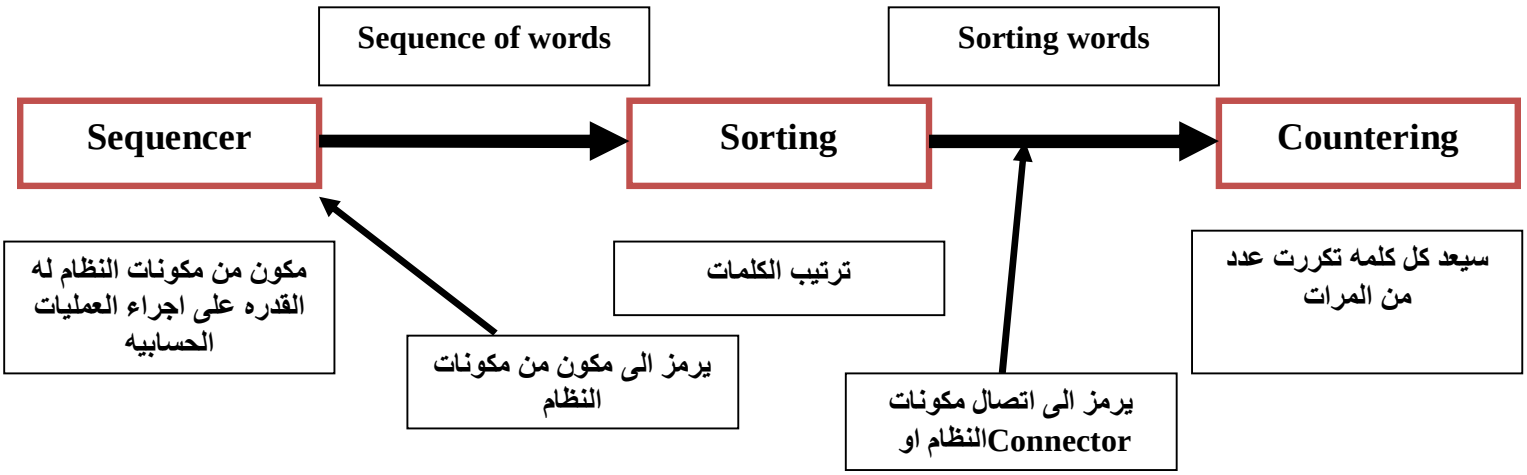
معلومات يتم العمل عليها عمليا ثم اعطاء خرج ثم اجراء عمليات على الخرج لاعطاء نتائج اخرى وهكذا

:Data Centric

:Student Registration System Example



:Pipe-and-Filter example



Low – Level Design:

Modularization:

عملية النمذجه بمعنى تقسيم النظام الى عناصر يمكن التعامل معها كل على حدى بحيث نفصل بينها فى عملية Implementation او كتابة الكود وتكون العلاقه بينها اقل ما يمكن وهذا يجعل الكود اسهل

Pseudo – Code:

الكود المزيف كما يطلقون عليه هو طريقه ليس لها شكل معين عباره عن وصف للتصميم باللغه الانجليزيه وموجود فيها بعض الكلمات المحجوزه داخل لغة البرمجه او Keywords

Layers:

السوفت وير عباره عن طبقات مختلفه تتعامل مع بعضها والنظام يطلق عليه مصطلح Modular اذا كان مقسم الى اقسام لا تعتمد على بعضها قدر الامكان

Flow Charts:

خرائط التدفق تغنيك عن كتابة الكود المزيف فهى تؤدى نفس الوظيفه لكن بكفاه اعلى وهى طريقه واسعة الاستخدام لانها تفيد كثيرا فى متابعة الحلقات التكراريه داخل البرمجه و الجمل الشرطيه Loops and Flow Control

جميع المصطلحات والادوات السابقه تندرج تحت مسمى المستوى البسيط من التصميم

قبل ان ندخل فى مرحلة التصميم ووصفه واجزائه كما وصفنا SRS يجب اولا ان نتعرف على بعض المفاهيم الهامه والتى على اساسها يبنى تصميمك لاى مشروع

Coupling & Cohesion:

الترابط والتماسك الداخلى وهما مفوهمان يتناسبان عكسيا مع بعضهما البعض ومن المعروف ان اثناء انشاء النظام ان تكون الاعتماديه بين اجزائه او Component اقل ما يمكن والان نأتى لوصف كلا منهما على حدى

Types of Coupling:

- **Data Coupling**

تكون هناك داله رئيسيه او Main داخل النظام وتقوم بأستدعاء داله فرعيه او Sub function وتعطى لها بيانات او Data اوليه بسيطه مثل نص – حرف – ارقام وهذا يعتبر اقل انواع الترابط وهو الافضل

- **Stamp**

هو نوع من اتصال الداله الرئيسيه بالدوال الفرعيه لتمرير Data لكن هذه المره نوع البيانات معقد مثل matrix – objects – struct

- **Control**

تمرر الداله الرئيسيه بعض الاوامر الى الداله الفرعيه لكى تتحكم الداله الفرعيه فى تنفيذها

- **External**

يستخدم مع الداله الرئيسيه اكثر من الفرعيه وتعتمد على الارتباط بواجه خارجيه او Interface خارجى

- **Common**

يوجد اثنين من مكونات النظام او Tow Module مشتركين فى نفس البيانات وتسمى حالة البيانات Global Data بمعنى انهم يعملوا على نفس البيانات لكن فى هذه الحاله عمل مكون على نفس البيانات يمكن ان يلغى عمل الاخر او يعطله

- **Content**

اثنين من المكونات مختلفين ولا يشتركان فى نفس البيانات لكن واحد منهم يقوم بعمل Go to الى الاخر بمعنى انه يقوم باستدعاء المكون الثانى لاداء وظيفه معينه ثم يعود مسار التحكم اليه لاتمام باقى مهامه ويسمى هذا الامر Jump وتجده كثيرا فى لغة Assembly ويعتبر اخطر انواع الارتباط لانه صعب التتبع ويجب التقليل منه قدر الامكان

Types of Cohesion:

عباره عن التماسك بين الاجزاء نفسها بالنسبه لاعضاء الداله الواحده او Function

- Functional

جميع اعضاء الداله تتعامل مع بعضها لاداء وظيفه واحده

- Sequential

الدوال تسلم بعضها النواتج وهو اخطر من الاول لانه لو حدث خطاء فى ناتج الداله الاولى سوف تكون جميع نتائج الدوال التاليه لها خطاء

- Communicational

اعضاء الداله كلها تعمل على Data مركبه واحده لاداء وظيفه معينه كالتى تعمل على Matrix – structur

- Procedural

ومعناه الاجراء وهو عباره عن مجموعه من العمليات مرتبه بواسطة الكود ولكل داله فيه مهمه معينه تنفذها

- Temporal

كل عضو من Module له وقت معين فى تنفيذ مهامه

- Logical

تجد ان Component مرتبطه مع بعضها ارتباطا منطقيا ويجب الحذر عند استخدامه

- **Coincidental**

مجموعة الدوال ليس لهم علاقه ببعض وهذا اخطر انواع الارتباط

:Design Specifications

- **Scope**
- **Data Design**
- **Structure Charts**
- **Design Program Interface**

الاستفاده منه :

1 – يؤسس Requirements للمشروع

2 – قياس مدى حساسية Requirements مع كل جوانب المشروع

- **هناك قواعد محدده يسير عليها اى تصميم او Design**

1 – الطريقه العاديه وهى No executable ولا يوجد فيها ادوات او Tools

واشهرها الاجتماعات او Meeting ويشارك فيها عضو من Designers وعضو من Analysis Requirements وعضو من المبرمجين Programmers وعضو من مهندسين الكفأءه Quality وعضو من مهندسين الصيانه او Mentance

والهدف من هذه المراجعه هى اكتشاف الاخطاء

اشهر الاخطاء التى توجد فى التصميم انه لا يغطى جميع متطلبات النظام

Structure Charts and Basic Building Blocks:

التصميم يمكن ان يمثل على الورق بواسطة خرائط مركبه وهذا التوثيق يهدف الى

البناء الهرمى للنظام وكذلك كل اجزائه والتوصيلات بين اجزاء النظام

وهناك نوعان من تمثيلات النظام :

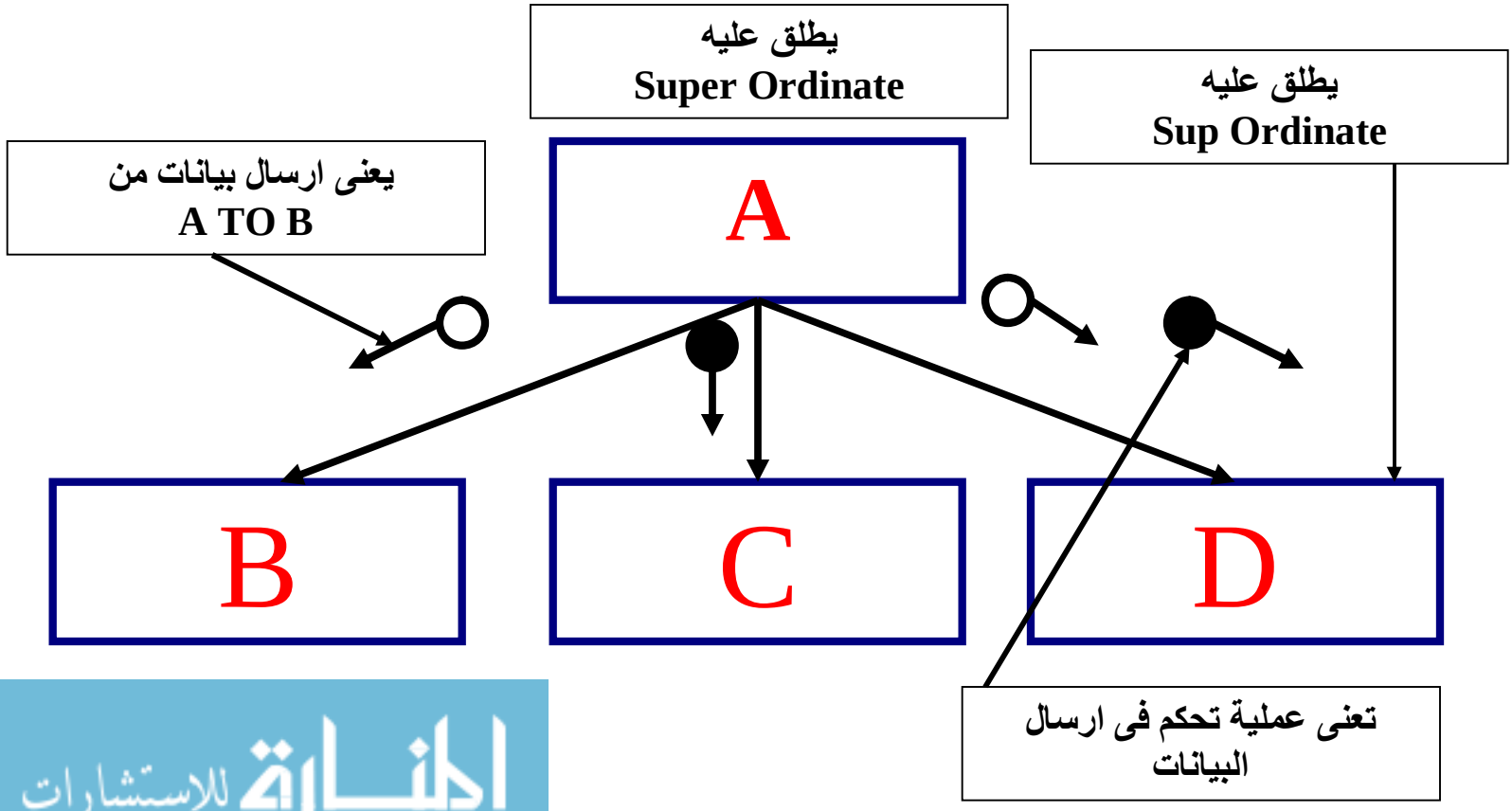
Function Oriented – Object Oriented

وكلاهما واسع الاستخدام داخل تصميمات الانظمه وناتى للنوع الاول وهو

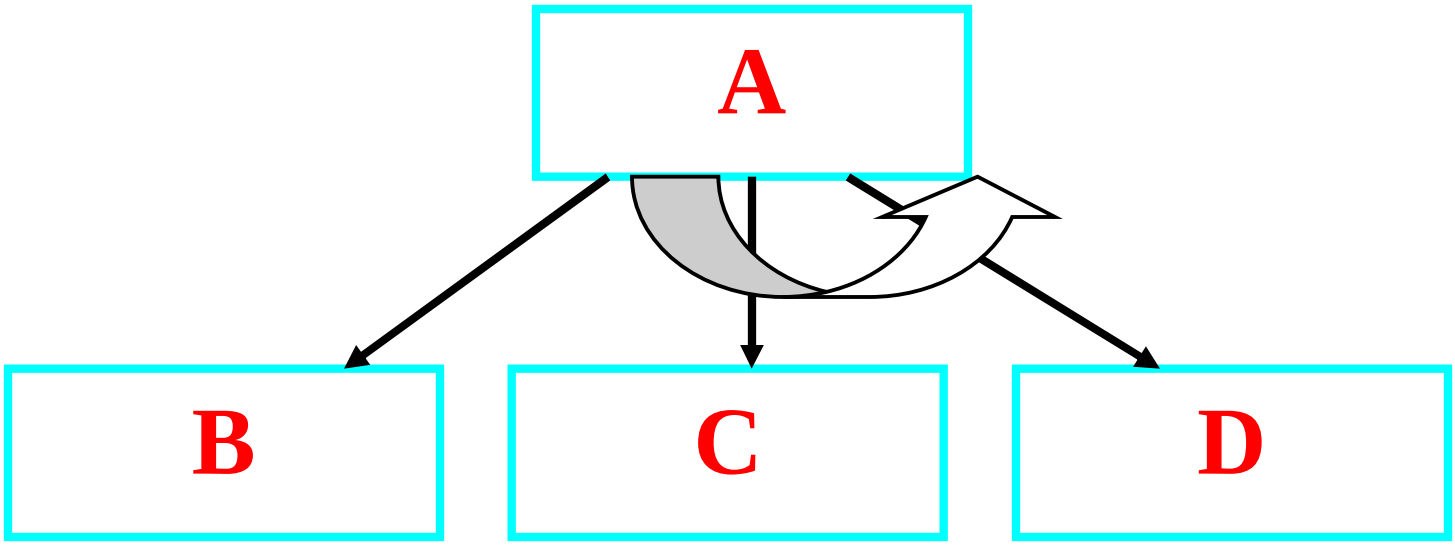
Function oriented التصميم يعتمد على مجموعة دوال والتى تمثل اجزاء النظام

ةالتى تمثل العمليات الرئيسيه التى تحدث ويتم الربط بينهما وتمثل الدوال او المكونات

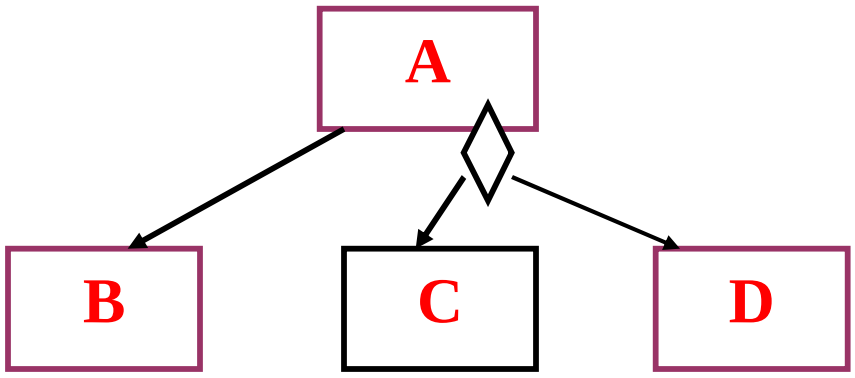
بواسطة مربع او صندوق ويكتب بداخله اسم المكون او الداله



ان طريقة تمثيل الحلقات داخل التصميم او ما يعرف داخل البرمجه Loops يكون بوتسطة سهم دائرى يمر على اسهم الاتصال المشتركه فى نفس الحلقه ثم يعود مره اخرى الى الداله الرئيسيه



اذا كان الاتصال بين الدوال يعتمد على قرارت خارجيه Outcome Decision تمثل بواسطة شكل ماسى وتخرج منه الاسهم المؤديه الى الدوال المتصله



:Four Major Steps in the Structured Design Methodology

هناك اربع طرق رئيسيه متبعه فى بناء تركيب التصميم وهى :

1 – اعادة وضع المشكله فى صورة مخطط بيانات او DFD

2 – تعريف بالظبط ما هى عناصر الدخل والخرج

3 – وضع المستوى الاول فى التصميم First Level Factory

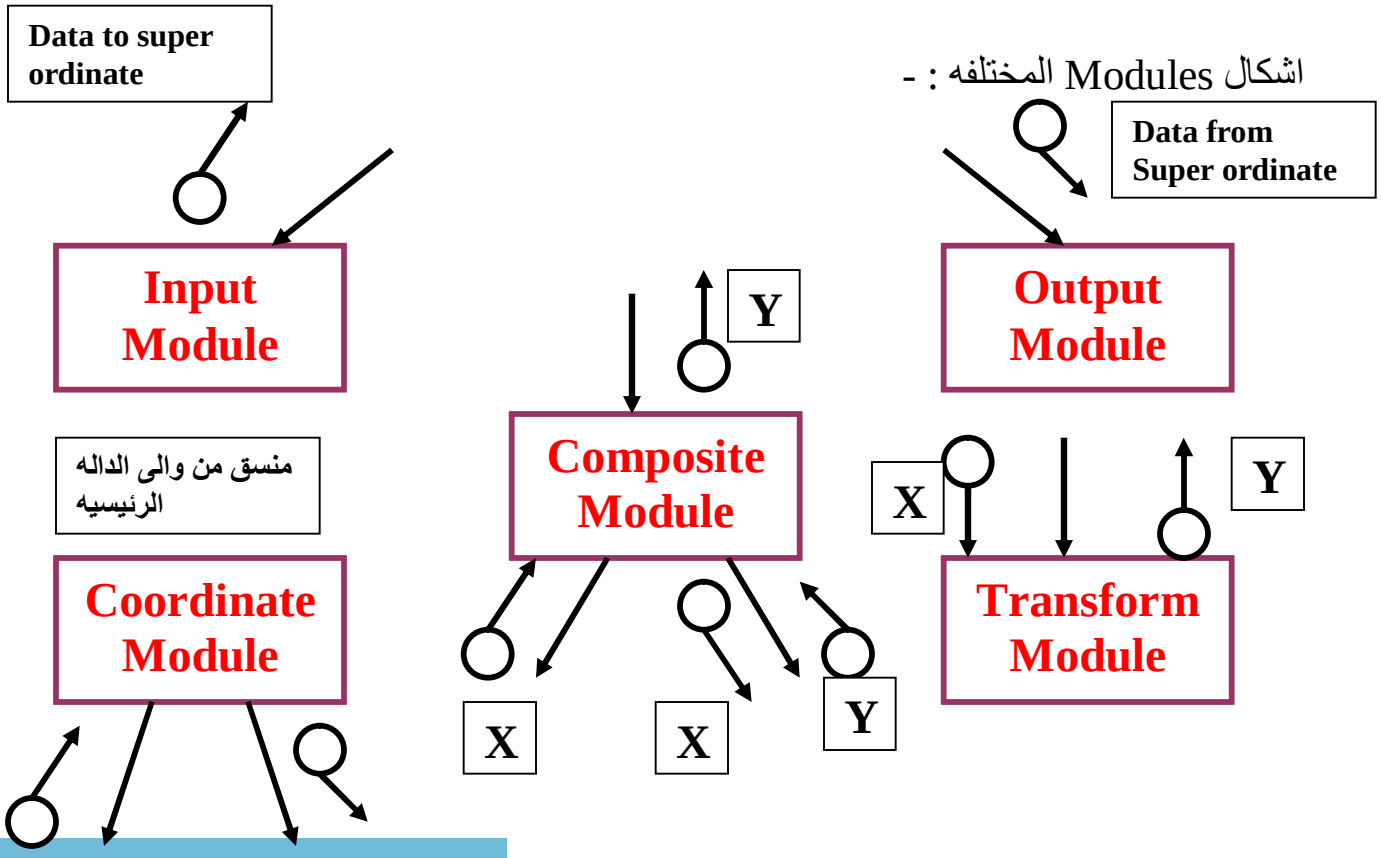
4 – وضع التصميم الشامل للنظام والذى يشمل داخله جميع عناصر الدخل والخرج

واخرى انتقال البيانات بين اجزاء النظام وبعضها البعض

بعد الانتهاء من جميع تلك العمليات تاتى مرحلة المصادقه على التصميم او

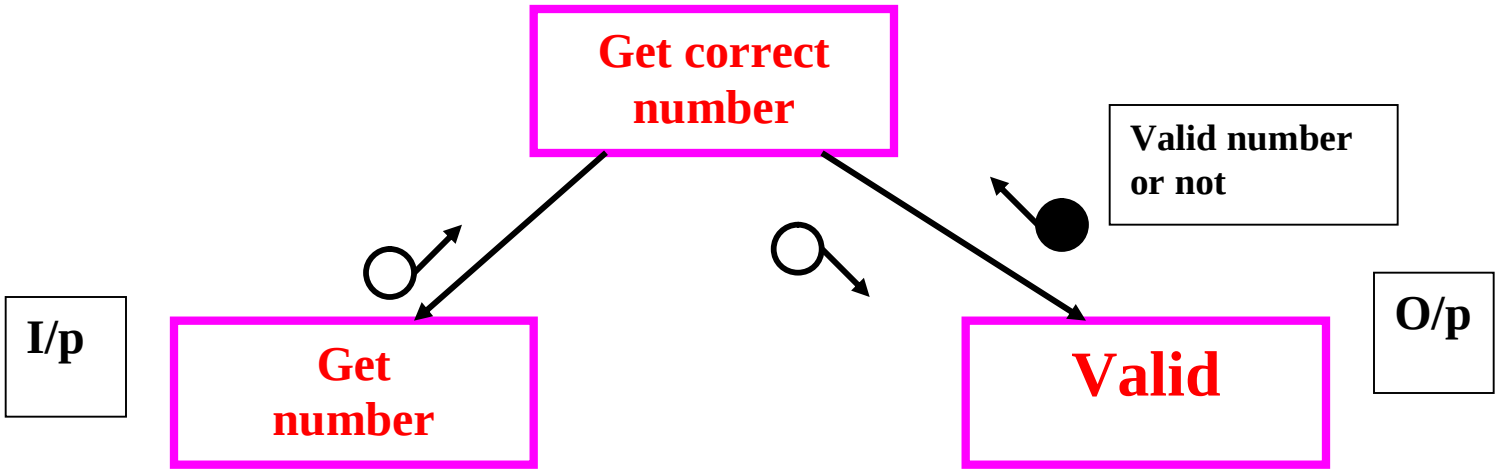
Improving and Verification Design

اشكال Modules المختلفه : -



المثال الاول لدينا عمل تصميم لنظام يختبر صحة الارقام المدخله اليه حسب ما يرى

مستخدم النظام



لدينا مثال اخر :

لدينا نظام يقراء 3 ارقام ويعمل Read لهم ثم يحسب RMS لكل منهم ويطبعها على الشاشه

Input → يقراء → 3numbers → اختبار صحة الارقام → If it is right

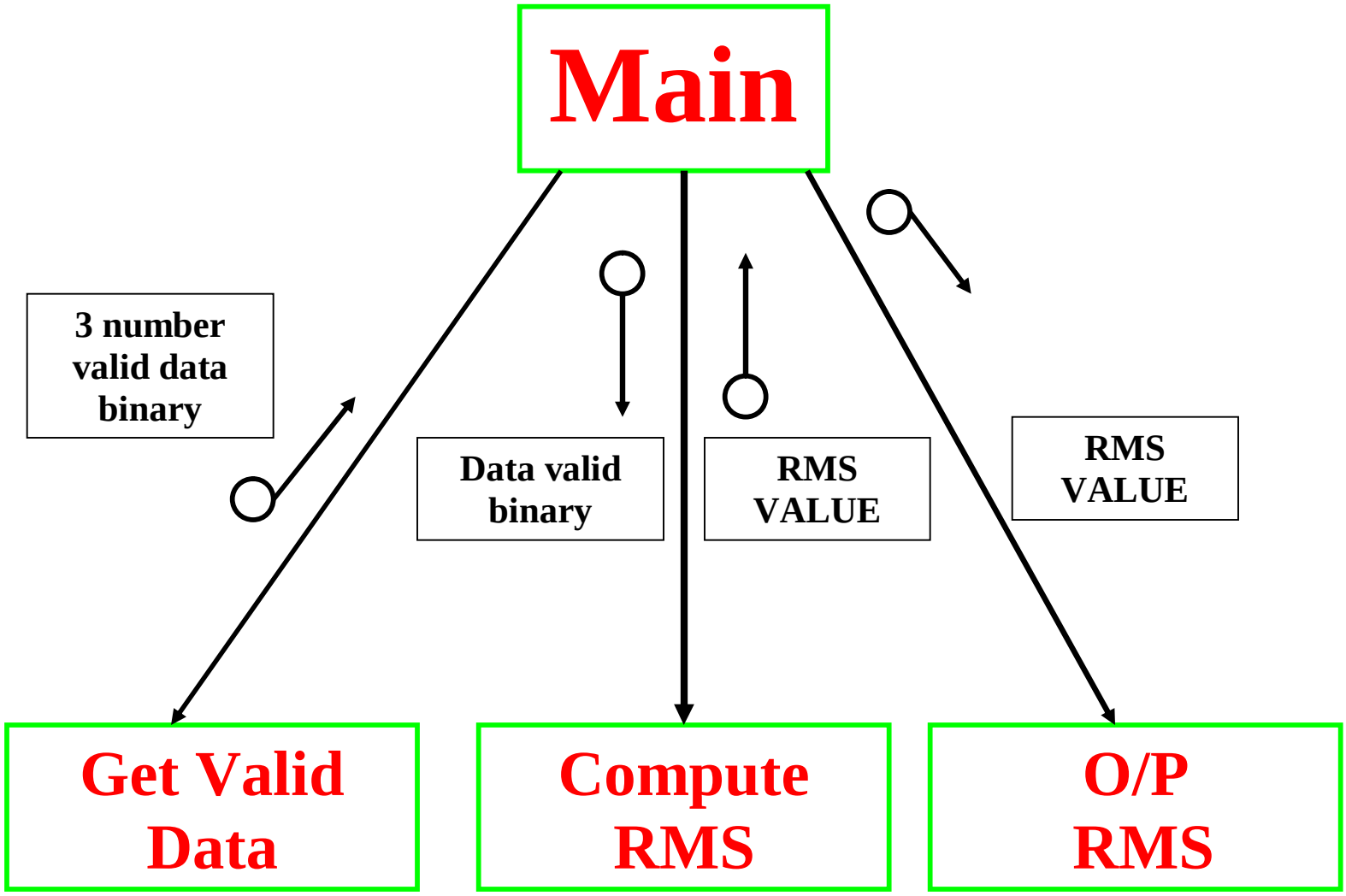
Send to → main function

Send to → Module of RMS

Send to → Module o/p

DFD Diagram

First Level Factory



ثم يتم تقسيم الدخل الى ثلاثة اقسام

1 – قراءة الارقام Read

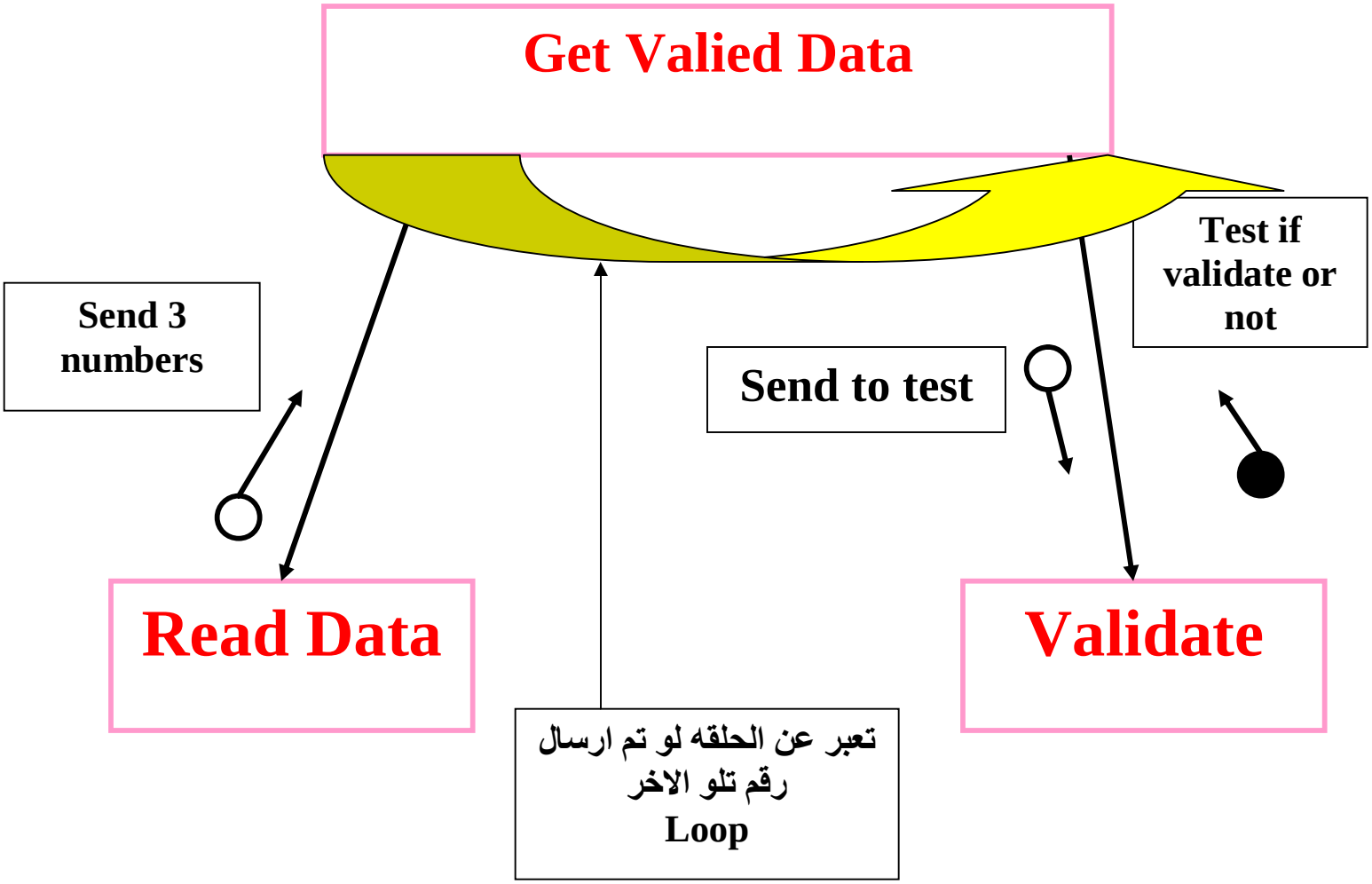
2 – اختبار صحة الارقام المدخله حسب ما يرى مصمم النظام Validate

3 – ارسال الارقام الى الداله الرئيسيه Send to Main Function

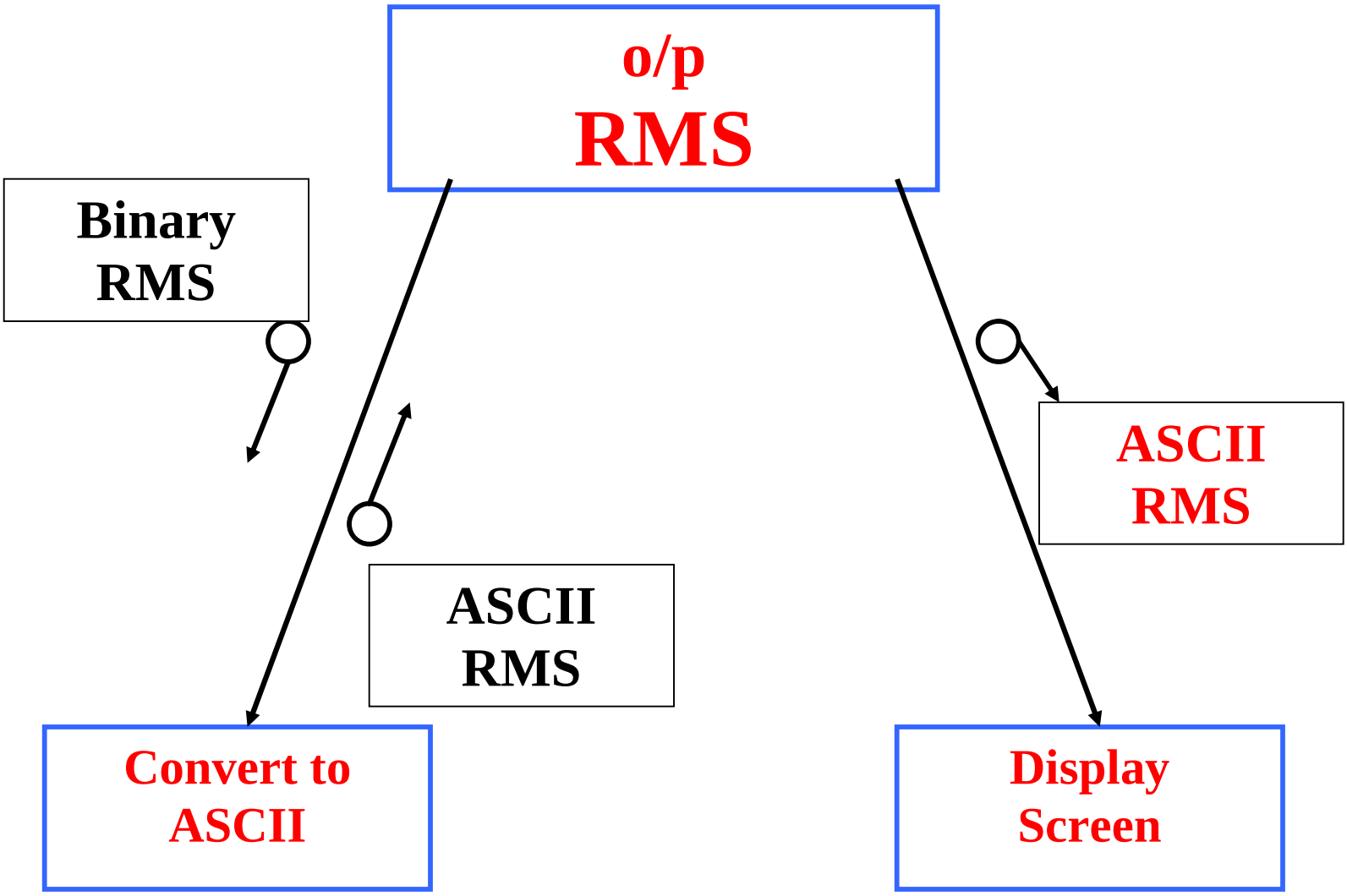
وبعدها نحصل على تصميم الخرج والدخل

Factory of i/p and o/p transform

Factory i/p



Factory o/p



انتهى الجزء الثانى من كتاب هندسة البرمجيات

وسيتم شرح واحده من اهم الطرق لتصميم والتى تكاد تكون هى السائده اليوم فى عالم البرمجيات فى الجزء الثالث من هذا الكتاب

Memorycode_84@yahoo.com