CH1

1) What is Software Engineering

١) ما هي هندسة البرمجيات

Solving problems

حل المشاكل

Software products are large and complex

(عملية انتاج البرامج) كبيرة ومعقدة

Development requires analysis and synthesis

يتطلب التطوير (التحليل) و(التركيب)

Analysis : decompose a large problem into smaller, understandable pieces

التحليل : تحليل مشكلة كبيرة إلى أجزاء صغيرة ومفهومة

abstraction is the key

(التجريد)

Synthesis: build (compose) a software from smaller building blocks

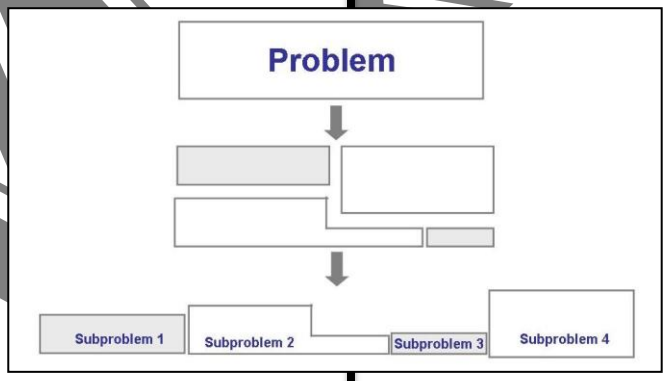
التركيب : (تجميع الأجزاء الصغيرة ليصبح برنامج)

composition is challenging

التجميع يشكل تحديا

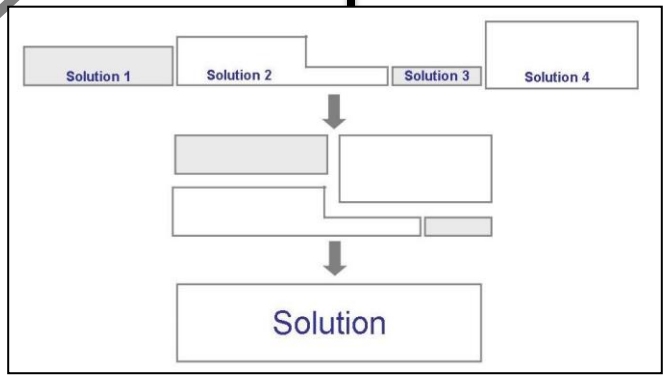
The analysis process

عملية التحليل



The synthesis process

عملية التركيب



Method: refers to a formal procedure; a formal “recipe” for accomplishing a goal that is typically independent of the tools used

أسلوب (**طريقة**) : (يصف كيفية تنفيذ شيء معين بطريقة رسمية يعدا عن الأدوات المستخدمة)

Tool: an instrument or automated system for accomplishing something in a better way

أداة : أداة أو نظام آلي لإنجاز شيء بطريقة أفضل

Procedure: a combination of tools and techniques to produce a product

الإجراء : مجموعة من الأدوات والتقنيات لإنتاج منتج

Paradigm: philosophy or approach for building a product (e.g., OO vs structured approaches)

النموذج : (النهج لبناء منتج)

Where does the software engineer fit in?

أين يركز مهندس البرمجيات ؟

Computer science: focusing on computer hardware, compilers, operating systems, and programming languages

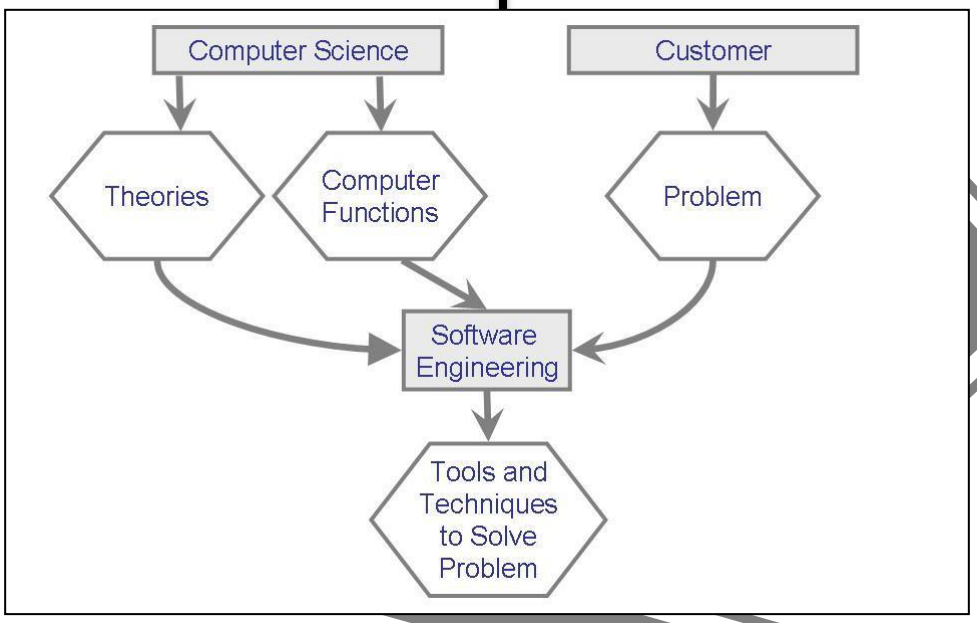
علم الحاسوب : التركيز على أجزاء الحاسوب ، وأنظمة التشغيل ، ولغات البرمجة

Software engineering: a discipline that uses computer and software technologies as a problem-solving tools

هندسة البرمجيات : نظام يستخدم تكنولوجيا الحاسوب والبرمجيات كأدوات لحل المشكلات

Relationship between computer science and software engineering

العلاقة بين علم الحاسوب وهندسة البرمجيات



2) How successful have we been?

٢) ما مدى نجاحنا؟

Perform tasks more quickly and effectively

أداء المهام بسرعة أكبر وفعالية

Word processing, spread sheets, e-mail

Support advances in medicine, agriculture, transportation, multimedia education, and most other industries

دعم التقدم في الطب والزراعة والنقل والتعليم متعدد الوسائط ومعظم الصناعات الأخرى

Many good stories

العديد من القصص الجيدة

However, software is not without problems

ومع ذلك ، فإن البرامج لا تخلو من المشاكل

Terminology for describing Bugs

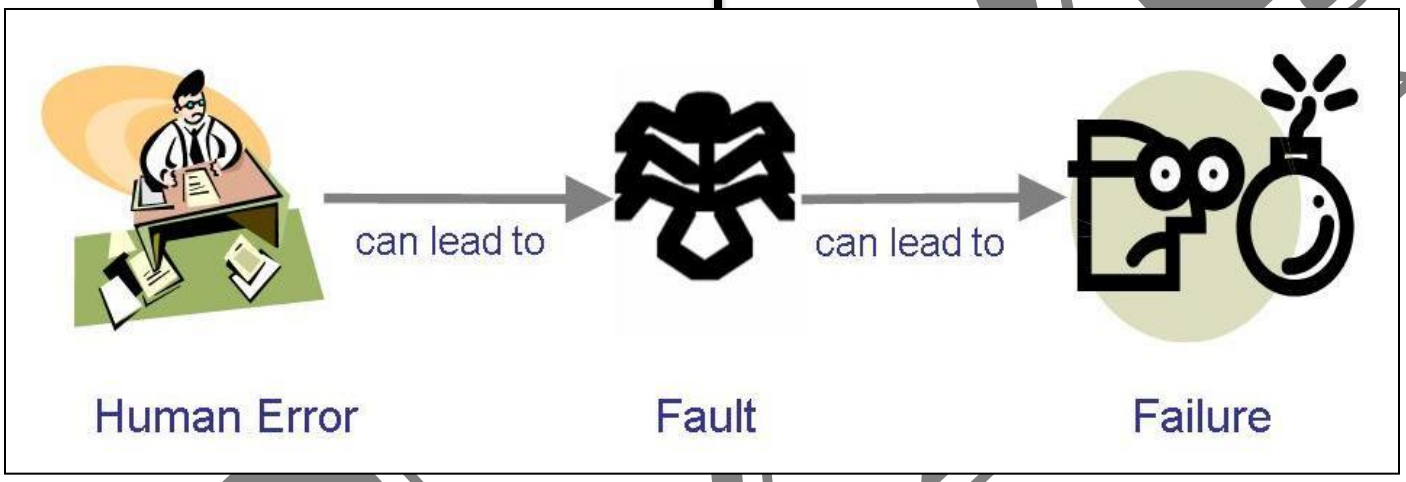
A fault: occurs when a human makes a mistake, called **an error**, in performing some software activities

A failure: is a departure from the system's required behaviour

مصطلحات لوصف الـ (Bugs)

(يحدث عندما يرتكب الإنسان خطأً)

(اختلاف عن السلوك المطلوب من النظام)



3) What is good software?

Perspective on Quality

The **transcendental** view: quality is something we can recognize but not define

The **user** view: quality is fitness for purpose

The **manufacturing** view: quality is conformance to specification

The **product** view: quality tied to inherent product characteristics

The **value-based** view: depends on the amount the customers is willing to pay for it

٣) ما هو البرنامج الجيد ؟

وجهة نظر على الجودة

وجهة نظرا كـ (transcendental) : شيء يمكننا التعرف عليه ولكن لا يمكن تعريفه

وجهة نظرا كـ مستخدم : (قوة الغرض الملموسة)

وجهة نظرا كـ (manufacturing) : مطابقة للمواصفات

وجهة نظرا كـ منتج : جودة مرتبطة بخصائص المنتج الكامنة

وجهة نظرا كـ قيمة المنتج : يعتمد على المبلغ الذي يرغب العميل في دفعه مقابله

Good software engineering must always include a strategy for producing quality software

يجب أن تشتمل هندسة البرمجيات الجيدة دائمًا على إستراتيجية لإنتاج برامج ذات جودة عالية

Three ways of considering quality

ثلاث طرق للنظر في الجودة

- The quality of the product
- The quality of the process
- The quality of the product in the context of the business environment

- جودة المنتج
- جودة العملية
- جودة المنتج في سياق بيئة الأعمال

The Quality of the Product

جودة المنتج

Users judge external characteristics (e.g., correct functionality, number of failures, type of failures)

يحكم المستخدمون على الخصائص الخارجية (على سبيل المثال الوظيفة الصحيحة ، عدد مرات الفشل ، نوع الفشل)

Designers and maintainers judge internal characteristics (e.g., types of faults)

المصممين والمشرفين يحكمون على الخصائص الداخلية (على سبيل المثال أنواع الأعطال)

Thus different stakeholders may have different criteria

وبالتالي قد يكون لأصحاب العلاقة معايير مختلفة

Need quality models to relate the user's external view to developer's internal view

يحتاج إلى نماذج ذات جودة لربط العرض الخارجي للمستخدم بالمشاهدة الداخلية لمطوّر البرامج

The Quality of the Process

جودة العملية

Quality of the development and maintenance process is as important as the product quality

تعتبر جودة عملية التطوير والصيانة بنفس أهمية جودة المنتج

The development process needs to be modeled

تحتاج عملية التطوير إلى نمذجة

Modeling will address questions like :

النمذجة ستعالج أسئلة مثل :

Where to find a particular kind of fault
How to find faults early
How to build in fault tolerance
What are alternative activities

أين تجد نوع معين من الخطأ
كيفية العثور على أخطاء في وقت مبكر
نسبة الخطأ المسموح بها
ما هي الأنشطة البديلة

4) Who Does Software Engineering?

Customer: the company, organization, or person who pays for the software system

Developer: the company, organization, or person who is building the software system

User: the person or people who will actually use the system

Participants (stakeholders) in a software development project

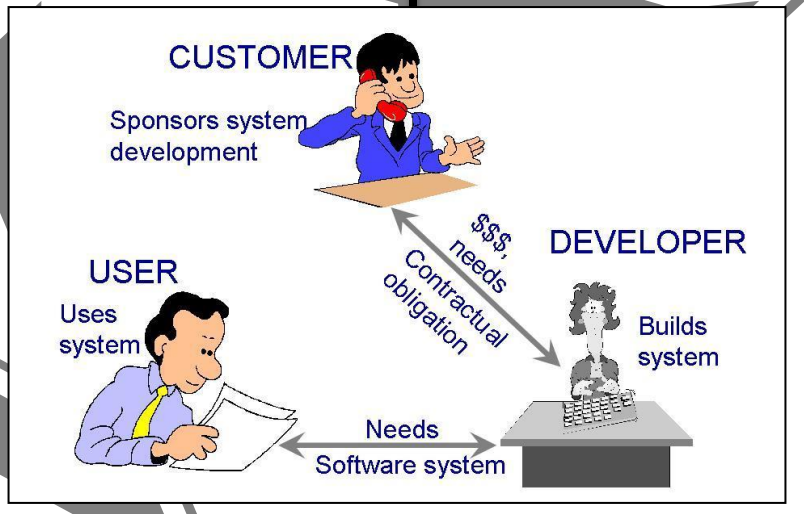
٤) من يقوم بـ هندسة البرمجيات؟

العميل : الشركة أو المنظمة أو الشخص الذي يدفع مقابل نظام البرامج

المطور : الشركة أو المنظمة أو الشخص الذي يقوم ببناء نظام البرمجيات

المستخدم : الشخص أو الأشخاص الذين سيستخدمون النظام بالفعل

المشاركون (أصحاب العلاقة) في مشروع لتطوير البرنامج



5) System Approach

Hardware, software, interaction with people

Identify activities and objects

Define the system boundary

Consider nested systems, systems interrelationship

٥) نهج النظام

الأجهزة ، البرامج ، التفاعل مع الناس

تحديد (شيء ، سلوك)

تحديد حدود النظام

الأخذ بالاعتبار النظم المتداخلة و نظم العلاقات المتبادلة

The Elements of a System

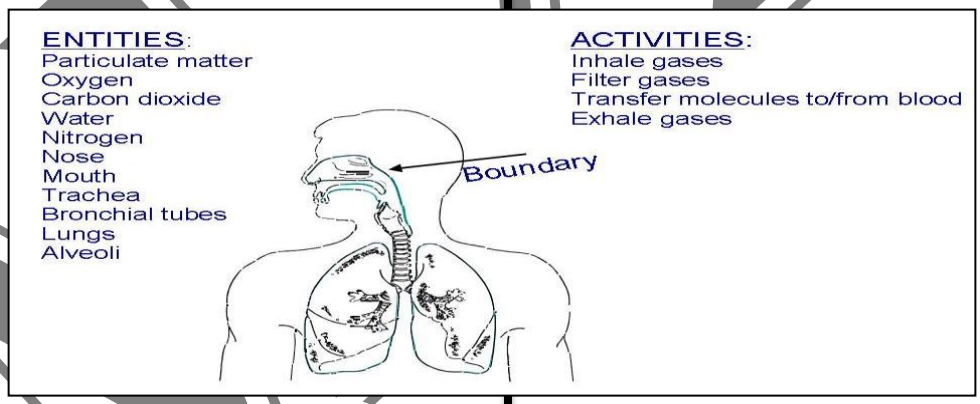
Activities and objects

- An activity is an event initiated by a trigger
- Objects or entities are the elements involved in the activities

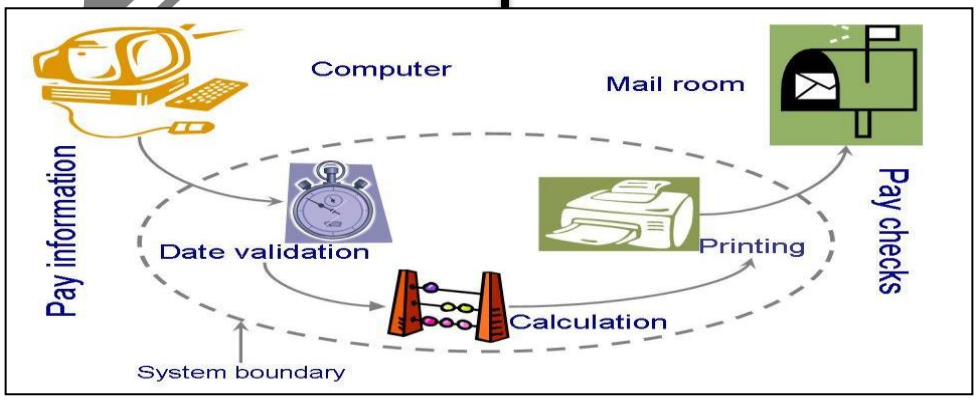
Relationships and the system boundaries

- A relationship defines the interaction among entities and activities
- System boundaries determine the origin of input and destinations of the output

Example of systems: a human respiratory system



A computer system must also be clearly described : System definition of a paycheck production



عناصر النظام

(شيء ، سلوك)

(طريقة فعل شيء ما)

(العنصر الأساسي)

العلاقات وحدود النظام

علاقة تحدد التفاعل بين (الشيء ، السلوك)

حدود النظام يبين أصل المدخلات ووجهات الإخراج

مثال على الأنظمة: نظام تنفس بشري

يجب أيضًا وصف نظام الكمبيوتر بشكل واضح : تعريف نظام لإخراج الراتب

Interrelated Systems

Some systems are dependent to other systems

- The interdependencies may be complex

It is possible for one system to exist inside another system

If the boundary definitions are detailed, building a larger system from the smaller ones is relatively easy

نظم مترابطة

تعتمد بعض الأنظمة على أنظمة أخرى

قد يكون الارتباط معقد

من الممكن وجود نظام داخل نظام آخر

إذا تم معرفة الحدود بين الأنظمة ، فإن إنشاء نظام أكبر من الأنظمة الأصغر يكون سهلاً نسبيًا

6) Engineering Approach

Building a System

Requirement analysis and definition

System design

Program design

Writing the programs

Unit testing

Integration testing

System testing

System delivery

Maintenance

٦) النهج الهندسي

بناء النظام

تحليل المتطلبات وتعريفها

تصميم النظام

تصميم البرنامج

كتابة البرامج

تجربة الأجزاء

اختبار التكامل

تجربة النظام

تسليم النظام

الصيانة

7) Members of the Development Team

Requirement analysts: work with the customers to identify and document the requirements

Designers: generate a system-level description of what the system is supposed to do

Programmers: write lines of code to implement the design

Testers: catch faults

Trainers: show users how to use the system

Maintenance team: fix faults that show up later

Librarians: prepare and store documents such as software requirements

Configuration management team: maintain correspondence among various artifacts

Typical roles played by the members of a development team

٧) أعضاء فريق التطوير

محللو المتطلبات : العمل مع العملاء لتحديد وتوثيق المتطلبات

المصممون : إنشاء وصف على مستوى النظام لما يفترض أن يقوم به النظام

المبرمجين : كتابة أسطر من الكود لتطبيق التصميم

مختبرين : البحث عن أخطاء

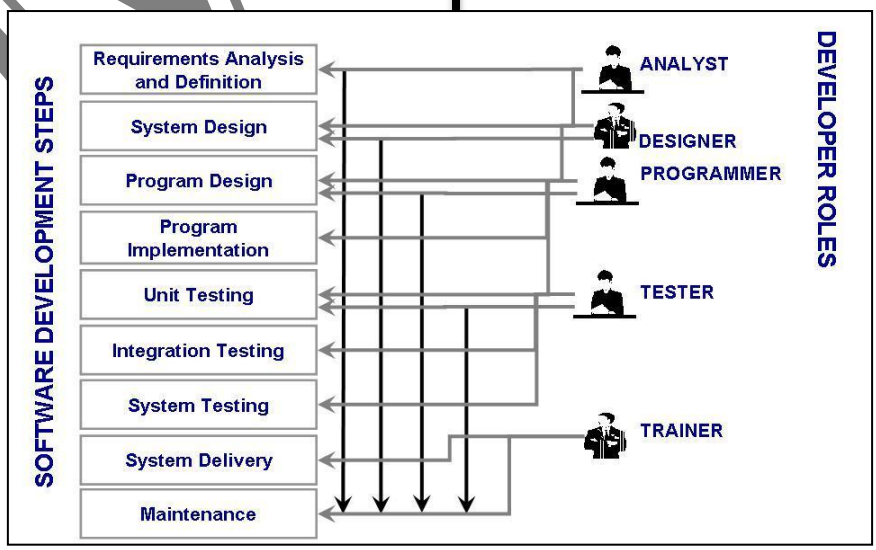
المدربون : تعليم المستخدمين كيفية استخدام النظام

فريق الصيانة : إصلاح الأخطاء التي تظهر في وقت لاحق

(معد الوصف): إعداد وتخزين المستندات مثل متطلبات البرامج

فريق إدارة التكوين : الحفاظ على التواصل بين مختلف المجالات (الأفرقة)

أدوار نموذجية يلعبها أعضاء فريق التطوير



8) How Has Software Engineering Changed?

The Nature of the Change

Before 1970s

- Single processors: mainframes
- Designed in one of two ways
  - as a **transformation**: input was converted to output
  - as a **transaction**: input determined which function should be performed

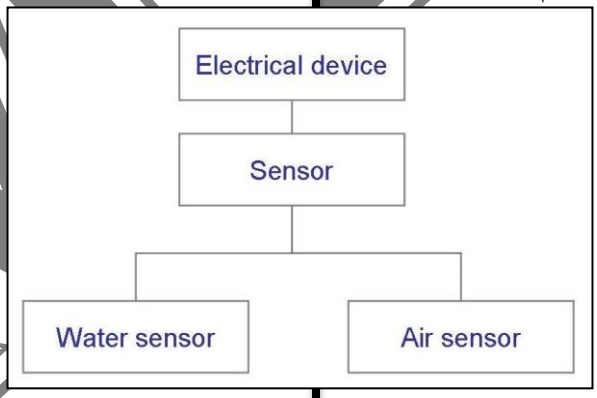
After 1970s

- Run on multiple systems
- Perform multi-functions

Abstraction

A description of the problem at some level of generalization

- Hide details



Analysis and Design Methods and Notations

Provide documentation
Facilitate communication
Offer multiple views
Unify different views

٨) كيف تغيرت هندسة البرمجيات ؟

طبيعة التغيير

قبل عام ١٩٧٠

- معالجات فردية: حواسيب كبيرة صممت بإحدى طريقتين
  - **Transformation :** تم تحويل المدخلات إلى مخرجات
  - **Transaction :** تحديد المدخلات على أي الاقترانات يجب أن تقوم بها

بعد عام ١٩٧٠

- تعمل على أنظمة متعددة
- أداء وظائف متعددة

التجريد

وصف للمشكلة على مستوى معين من التعميم

- أخف التفاصيل الداخلية (لا يهم العميل ما برمج بالداخل)

طرق تحليل وتصميم والملاحظات و آليات العمل

تقديم الوثائق
تسهيل التواصل
تقديم وجهات نظر متعددة
توحيد وجهات النظر المختلفة

User Interface Prototyping

Prototyping: building a small version of a system

- Help users identify key requirements of a system
- Demonstrate feasibility

Develop good user interface

نموذج واجهة المستخدم الأولية

النمذجة: بناء نسخة صغيرة من النظام

مساعدة المستخدمين على تحديد المتطلبات الأساسية للنظام
إثبات الجدوى
تطوير واجهة مستخدم جيدة

Software Architecture

A system's architecture describes the system in terms of a set of architectural units and relationships between these units

Architectural decomposition techniques

- Modular decomposition
- Data-oriented decomposition
- Event-driven decomposition
- Outside-in-design decomposition
- Object-oriented decomposition

معمارية البرمجيات

معمارية النظام تصف النظام من حيث مجموعة من الوحدات المعمارية والعلاقات بين هذه الوحدات

تقنيات إعادة التقسيم المعماري

إعادة تقسيم الوحدات
إعادة تقسيم البيانات الموجهة
إعادة تقسيم القائم على الحدث
إعادة تقسيم خارج التصميم
إعادة تقسيم البرمجة الموجهة

Software Process

Many variations

Different types of software need different processes

- Enterprise-wide applications need a great deal of control
- Departmental applications can take advantage of rapid development

عملية البرمجيات

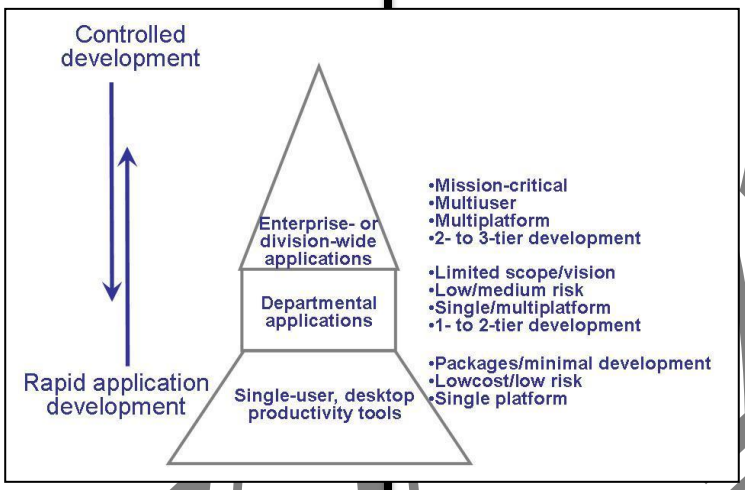
الكثير من الاختلافات

الأنواع المختلفة من البرامج تحتاج إلى عمليات مختلفة

التطبيقات على مستوى المؤسسة تحتاج إلى قدر كبير من التحكم
تطبيقات الإدارات يمكنها الاستفادة من التطور السريع

Pictorial representation of differences in development processes

التمثيل التصويري للاختلافات في عمليات التنمية



Software Reuse

Commonalities between applications may allow reusing artifacts from previous developments

- Improve productivity
- Reduce costs

Potential concerns

- It may be faster to build a smaller application than searching for reusable components
- Generalized components take more time to build
- Must clarify who will be responsible for maintaining reusable components
- Generality vs specificity: always a conflict

إعادة استخدام البرامج

قد تسمح الخصائص المشتركة بين التطبيقات بإعادة استخدام الأدوات من تطويرات سابقة

- تحسين الإنتاجية
- خفض التكاليف

مخاوف محتملة

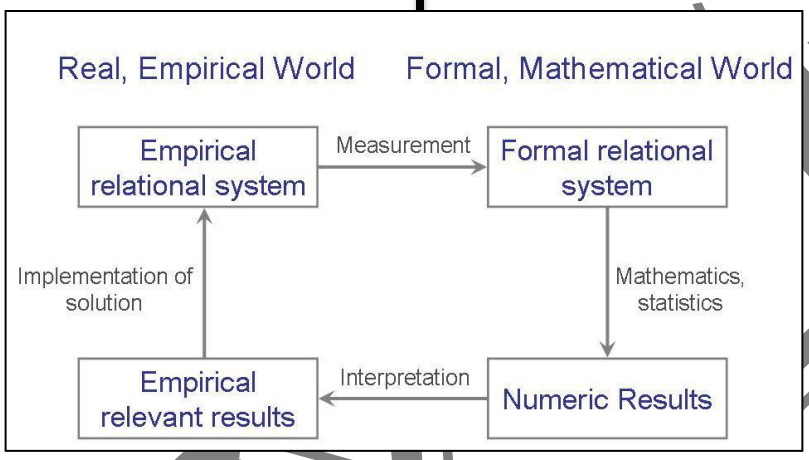
- قد يكون إنشاء تطبيق صغير أسرع من البحث عن مكونات قابلة لإعادة الاستخدام
- تستغرق المكونات العامة وقتًا أطول للبناء
- يجب توضيح من المسؤول عن الحفاظ على المكونات القابلة لإعادة الاستخدام
- العمومية مقابل الخصوصية : دائما في تعارض

Measurement

Objective: describe quality goals in a quantitative way

القياس

الهدف : وصف جودة الأهداف بطريقة كمية (رقم)



11) What this Chapter Means for You

١١) ماذا يعني هذا الفصل لك

Given a problem to solve

طريقة حل المشكلة

- Analyze it
- Synthesize a solution

تحليل المشكلة إلى مشاكل أصغر
تركيب حل المشاكل الأصغر لحل المشكلة الكبيرة

Understand that requirements may change

الأخذ بالاعتبار أن المتطلبات قد تتغير

Must view quality from several different perspectives

لكل جانب تقييم الكفآئة الخاص به

Use fundamental software engineering concepts (e.g., abstractions and measurements)

استخدام مفاهيم هندسة البرمجيات الأساسية (على سبيل المثال ، التجريد والقياس)

Keep system boundary in mind

أخذ حدود النظام في عين الاعتبار

Chapter 1 END

انتهى الفصل الأول