هندسة البرمجيات المتقدمة

د. م. علي ذياب

جدول المحتويات

- نظرة عامة
- فحص واختبار البرمجيات
- هندسة البرمجيات غرضية التوجه

نظرة عامة

محتويات المحاضرة

- ماهي البرمجية؟
- لماذا تُعتبر هندسة البرمجيات مهمة جداً؟
- أهداف المقرر

ماهي البرمجية؟

- البرمجية
  - المنتج الذي يقوم ببنائه مجموعة من المبرمجين المحترفين ويقومون بصيانته بشكل مستمر و لفترة زمنية طويلة

- ماذا تتضمن البرمجية
  - التعليمات البرمجية
  - بنية البيانات
  - التوثيق

لماذا تُعتبر هندسة البرمجيات مهمة جداً؟

- كل اقتصاديات الدول النامية تعتمد على البرمجيات بشكل أساسي

- دائماً تظهر أنظمة جديدة يتم التحكم بها عن طريق البرمجيات (أنظمة الإتصالات, الأنظمة العسكرية, الأجهزة المنزلية, ألعاب الأطفال, الخ)

- صيانة الأنظمة البرمجية عملية مكلفة للغاية
  - الحل: **هندسة البرمجيات**

  - **هندسة البرمجيات** هي العلم الذي يتعلق بالنظريات, الطرائق والأدوات المستخدمة لبناء احترافي للبرمجيات

أهداف المقرر

- تزويد الطلاب بمعلومات حديثة حول هندسة البرمجيات
- فهم متعمق لكيفية حل المشاكل المتعلقة بهذا المجال
- التركيز على
  - تعريف وخصائص البرمجية
  - تطوير البرمجيات

مقدمة

محتويات المحاضرة

- مدخل إلى المحاضرة
- طريقة هندسة البرمجيات (**Software Engineering Process**)
- تكلفة هندسة البرمجيات (**Software Engineering Costs**)
- موديل SDLC

مدخل إلى المحاضرة

كلفة البرمجيات (Software Costs)

- كلفة البرمجيات هي المسيطرة وذات النصيب الأكبر عند النظر إلى كلفة النظم المحوسبة
  - كلفة البرمجيات الموجودة في جهاز حاسوب أكبر عادةً من كلفة الـ hardware

- كلفة البرمجيات في قسمها الأكبر هي كلفة صيانة للبرمجيات وليس تطوير لها
  - الأنظمة مخططة لتعيش فترة زمنية طويلة, كلفة صيانة البرمجيات عدة أضعاف كلفة تطوير البرمجيات

- تتعلق هندسة البرمجيات ببناء برمجيات بأقل كلفة ممكنة مع الحفاظ على جودة عالية
  - لتخفيض الكلفة ينبغي
    - تخفيض كلفة تطوير المنتجات البرمجية
    - تخفيض كلفة صيانة المنتجات البرمجية
  - **مهم جداً**: أداء البرمجيات يتحسّن أيضاً عند تطبيق مبادئ هندسة البرمجيات

خصائص البرمجيات (Features of Software?)

- خصائص البرمجيات تجعلها مختلفة عن بقية الأشياء التي يصنعها الإنسان

- البرمجيات عبارة عن نظم منطقية لها الخصائص التالية
  - البرمجية يتم تطويرها أو هندستها (لايتم صناعتها كما هو مطبق بالمفهوم الكلاسيكي للمنتجات, الأمر الذي يترك تغييراً بمفهوم الجودة)
  - البرمجية لاتصاب بالعطب ولكن تتقادم

- بالرغم من أن الصناعة تتجه نحو صناعة مركبات وعناصر جاهزة يمكن استخدامها في بناء نظم متكاملة, تبقى البرمجيات تتجه بحو البناء بشكل مخصص باتجاه تطبيقات محددة

مصطلحات أساسية (Terminology)

السؤال	الجواب
ماهي البرمجية؟	عبارة عن برنامج حاسوبي, بنية بيانات و التوثيق المرتبط بهما.
ماهي مواصفات البرمجية الجيدة؟	البرمجية الجيدة يجب أت تُقدم الأداء المطلوب للمستخدم, ويجب أن تكون قابلة للصيانة, يمكن الإعتماد عليها و قابلة للإستخدام.
ماهي هندسة البرمجيات؟	هندسة البرمجيات عبارة عن الإختصاص العلمي المتعلق بمبادئ وأساسيات إنتاج البرمجيات.
مالفرق بين هندسة البرمجيات و المعلوماتية؟	تركز المعلوماتية على النظريات وأساسيات البرمجة, بينما تركز هندسة البرمجيات على التطوير الفعلي للبرمجيات القابلة للإستخدام.
مالفرق بين هندسة البرمجيات و هندسة النظم؟	تتعلق هندسة النظم على كل جوانب الأنظمة المعتمدة على الحاسوب, متضمنةً البرمجيات والعتاد الصلب. هندسة البرمجيات جزء من هندسة النظم و تركز على التطوير الفعلي للبرمجيات القابلة للإستخدام.

مصطلحات أساسية (Terminology)

الخاصة	الشرح
قابلية الصيانة (Maintainability)	يجب بناء البرمجية بطريقة تلبي حاجات الزبون وبنفس الوقت تكون قابلة للصيانة بسهولة في حال حدوث أخطاء.
الإعتمادية (Dependability) و الأمن (security)	تشمل الإعتمادية العديد من الخصائص الجزئية كالموثوقية (reliability), الأمن (security), والسلامة (safety). البرمجية المعتمد عليها يجب أن تؤدي عملاً موثوقاً, وألا تسمح إلا للمستخدمين المرخص لهم بالعمل عليها, وألا تؤدي لتلف في بيئة التشغيل البرمجية أو المادية عند حدوث خطأ ما.
الفعالية (Efficiency)	يجب أن تعمل البرمجية بفعالية عالية و ألا تضيع موارد النظام, كالذاكرة, نبضات ساعة المعالج, الخ
القبول (Acceptability)	يجب أن تكون البرمجية مقبولة من قبل المستخدمين الذين طورت البرمجية لأجلهم, حيث يجب أن تكون مفهومة وسهلة الإستخدام ومتوافقة مع الأنظمة التي يستخدمونها.

ماهو Computer-Aided Software Engineering؟

- عبارة عن نظام برمجي مخصص للمساعدة في أتمتة عملية بناء البرمجيات

- Upper-CASE
  - عبارة عن أدوات مخصصة لبناء البرمجيات في مراحل التصميم وجمع المتطلبات, كلغة التوصيف UML

- Lower-CASE
  - عبارة عن أدوات مخصصة لبناء البرمجيات في مراحل البناء الفعلي, كواجهات البرمجة, المنقحات, الخ

طريقة هندسة البرمجيات (Software Engineering Process)

ماهي طرائق هندسة البرمجيات (Software Engineering (Methods

- عبارة عن طرائق منهجية موجهة لبناء البرمجيات, تتضمن بناء موديلات للنظام, ملاحظات, قواعد و نصائح للتصميم بالإضافة لدليل لعملية التصميم
  - توصيف الموديل
    - يحتوي توصيف الموديلات الرسومية الواجب بناؤها
  - القواعد
    - القواعد المرتبطة بالموديلات الرسومية
  - التوصيات
    - لبناء عملي و جيد للبرمجية
  - دليل عملية التصميم
    - ماهي النشاطات والخطوات الواجب اتباعها

ماهو Computer-Aided Software Engineering؟

- عبارة عن نظام برمجي مخصص للمساعدة في أتمتة عملية بناء البرمجيات

- Upper-CASE
  - عبارة عن أدوات مخصصة لبناء البرمجيات في مراحل التصميم وجمع المتطلبات, كلغة التوصيف UML

- Lower-CASE
  - عبارة عن أدوات مخصصة لبناء البرمجيات في مراحل البناء الفعلي, كواجهات البرمجة, المنقحات, الخ

الطريقة البرمجية (Software Process)

- مجموعة من النشاطات هدفها تطوير البرمجيات

- الخطوات العامة للطريقة البرمجية هي
  - التوصيف
    - ماذا يجب أن يفعل النظام و ماهي التقييدات و الصعوبات التي يمكن أن يواجهها
  - البناء
    - بناء النظام البرمجي
  - التحقق و الإختبار
    - التحقق من أن البرمجية تؤدي مايريده المستخدم
  - تطوير البرمجية
    - تعيير البرمجية وتطويرها بسبب تغيير المتطلبات

ماهو موديل تطوير البرمجيات (Software Process Model)?

- عبارة عن تمثيل بسيط للطريقة البرمجية, يتم عرضه من عدة وجهات نظر

- أمثلة عن وجهات النظر
  - Workflow perspective
  - Data-flow perspective
  - Role/action perspective

- أمثلة عن موديلات تطوير البرمجيات
  - Waterfall
  - Iterative development
  - Component-based software engineering

تكلفة هندسة البرمجيات (Software Engineering Costs)

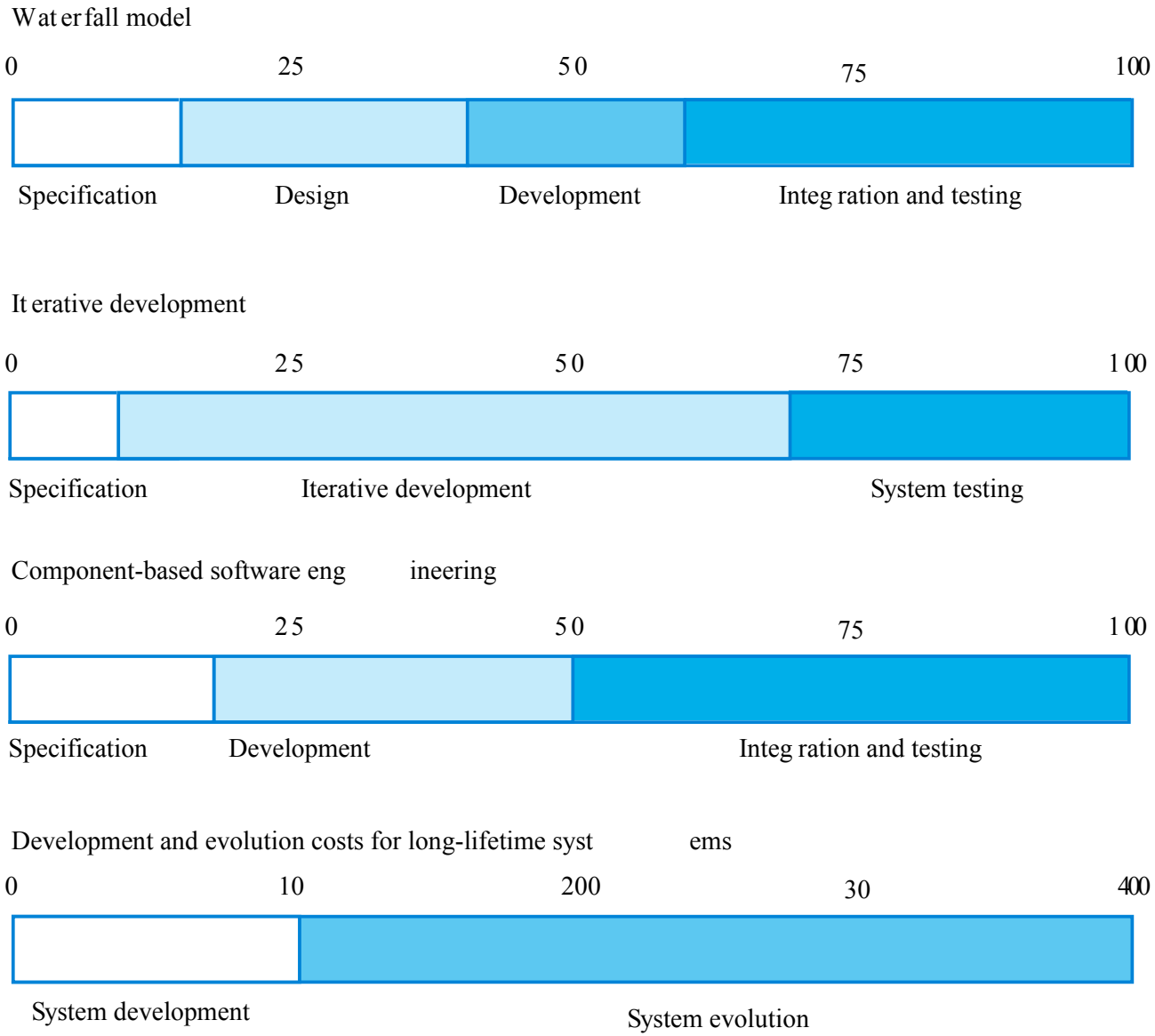
ماهي كلفة هندسة البرمجيات؟

- بشكل تقريبي
  - 60 % من الكلفة هي كلفة تطوير
  - 40 % من الكلفة هي كلفة فحص وصيانة
  - ملاحظة: من أجل البرمجيات المخصصة (custom software), تتجاوز كلفة الفحص كلفة التطوير

- تتغير الكلفة تبعاً لـ
  - نوع النظام الجاري تطويره
  - متطلبات النظام, كالأداء, الموثوقية, الخ.

- توزع الكلفة يتغير تبعاً لموديل (development model) هندسة البرمجيات المستخدم

توزّع كلفة هندسة البرمجيات



طرق تنفيذ دورة حياة المنتج البرمجي

- الطريقة المخطط لها (Plan-driven Process)
  - هي طريقة ذات خطوات منظمة
  - أمثلة عن الموديلات المستخدمة
    - Waterfall Model
    - Incremental Model
    - …
- الطريقة الرشيقة (Agile Process)
  - A way to manage a project by breaking it up into several phases
  - Once the work begins, teams cycle through a process of planning, executing, and evaluating
  - أمثلة عن الموديلات المستخدمة
    - Adaptive Model
    - Extreme Programing Model
    - …

الطريقة الرشيقة (Agile Process)

SDLC موديل

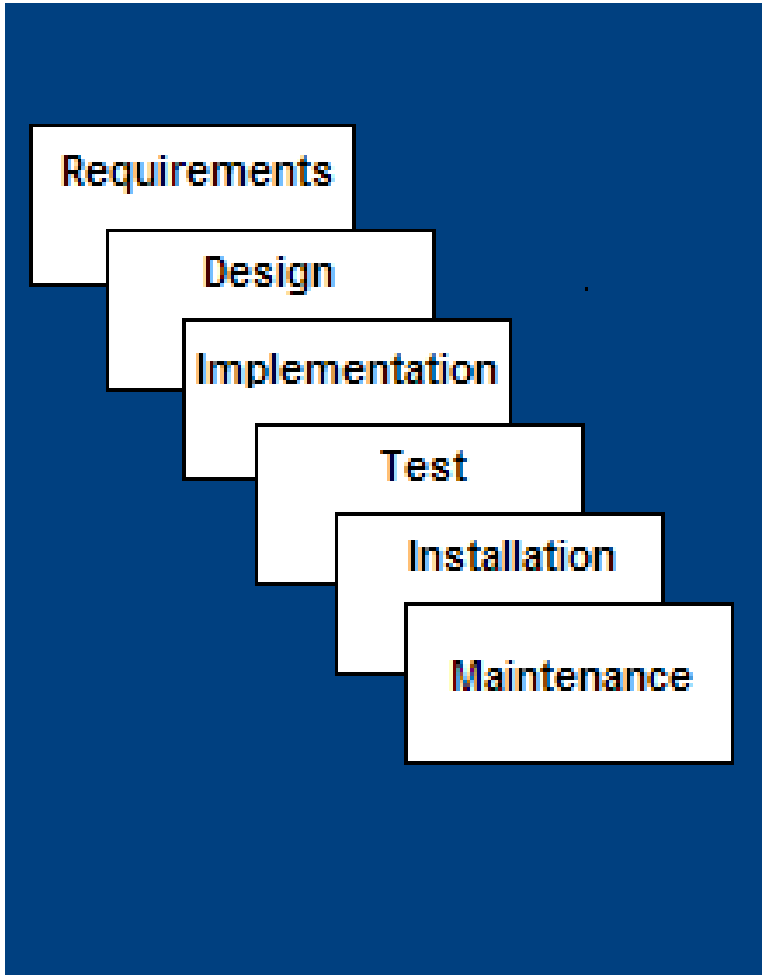
تعريف موديل SDLC

- عبارة عن إطار (framework) يصف النشاطات والخطوات المتبعة في كل مرحلة من مراحل تطوير البرمجية ضمن المشروع الجاري تنفيذه

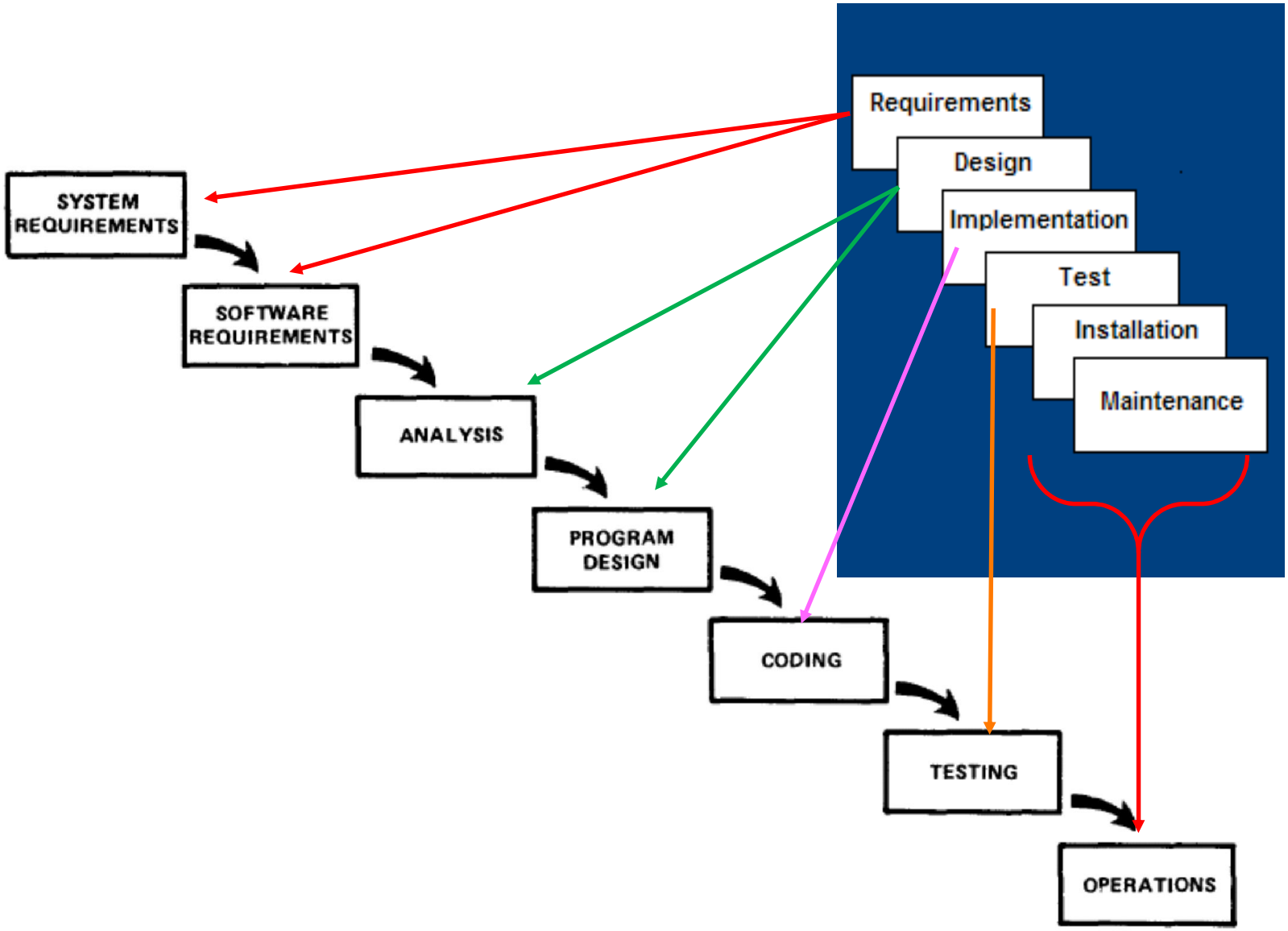
- أمثلة عن الموديلات المستخدمة
  - A Bug & Fix Model
  - Waterfall Model
  - Rapid Prototyping Model
  - Incremental Model
  - Extreme Programming Model
  - Synchronize-and Stabilize Model
  - Spiral Model
  - Object-Oriented Life-Cycle Models
  - Dynamic Systems Development Method (DSDM)
  - Adaptive Model
  - ....

موديل شلال الماء

- تحدد المتطلبات (Requirements)
  - المعلومات الضرورية
  - التوابع والسلوك
  - الأداء وواجهات الربط
- التصميم (Design)
  - بنية البيانات
  - بنية البرمجيات
  - تمثيل واجهات الربط
  - تفاصيل الخوارزميات
- التنفيذ الفعلي (Implementation)
  - كتابة الشيفرة البرمجية وبناء قاعدة المعطيات
  - توثيق البرمجية وفحصها



موديل شلال الماء



نقاط القوة

- سهل الفهم والإستخدام
- يزود فريق العمل من غير ذوي الخبرة ببنية مرتبة
- خطواته مفهومة بشكل جيد
- يقوم بوضع المتطلبات بطريقة منهجية
- سهل الإدارة
- يعمل بشكل جيد عندما تكون الجودة أكثر أهمية من التكلفة أو مدة الإنجاز

السلبيات

- يجب أن تكون المتطلبات كلها معروفة مسبقاً
- المستندات والعناصر المتولدة في كل مرحلة تعتبر ثابتة ولا يمكن تغييرها في الخطوات اللاحقة إلا بالعودة للوراء

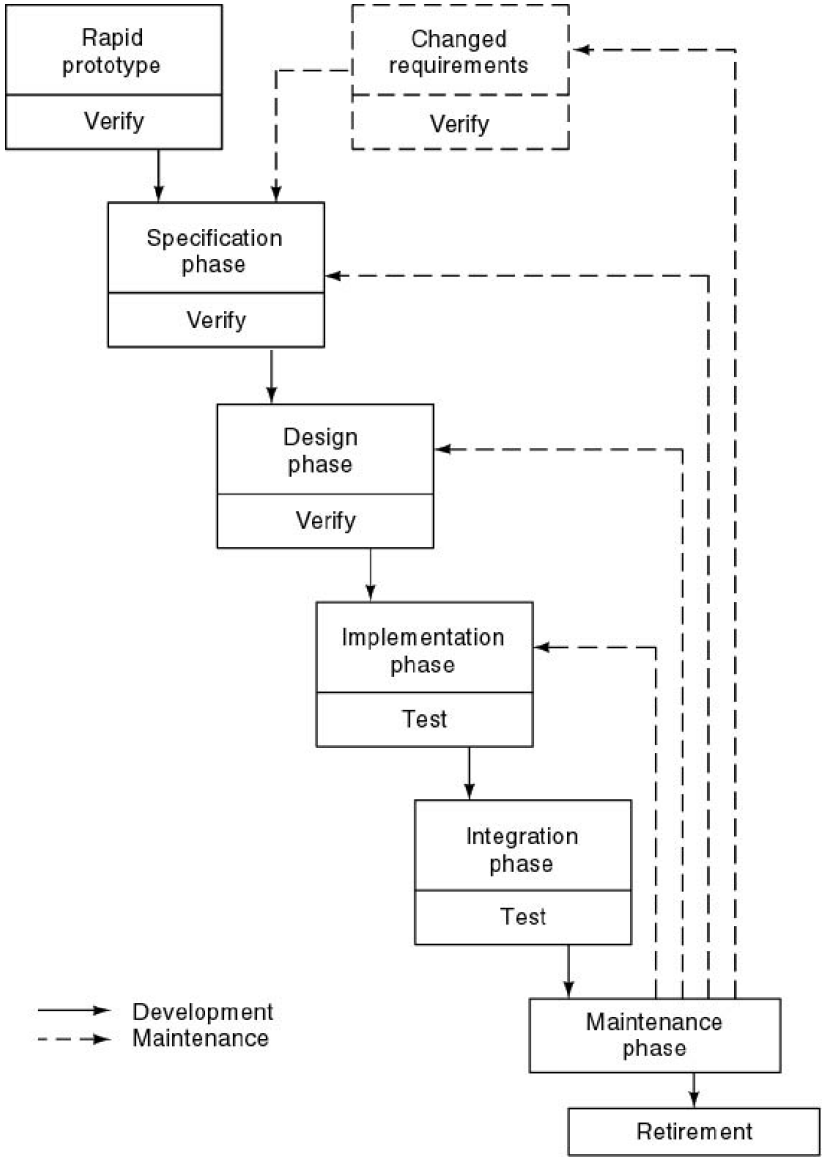
- يعطي انطباع سلبي عن التقدم
- لايعكس آلية العمل مشكلة–حل المستخدمة في تطوير البرمجيات بالطريقة التكرارية

- دمج البرمجية في البيئة أو مع برمجيات أخرى تعتبر مشكلة كبيرة في نهاية تطوير المنتج
- يتواجد فرص قليلة للمستخدم للتعرف عل المنتج البرمجي في مراحله الأولى (قبل وصول المنتج النهائي)

متى يجب استخدام موديل شلال الماء

- المتطلبات معروفة بشكل جيد
- البرنامج محدد بشكل جيد
- التكنلوجيا المستخدمة مفهومة بشكل جيد
- إصدار جديد من منتج موجود مسبقاً
- تصدير منتج برمجي للعمل ضمن بيئة جديدة (ضمن نظام تشغيل جديد مثلاً)

موديل النموذج السريع



موديل النموذج السريع

- يزود المستخدم بنموذج تجريبي عن المشروع البرمجي بأسرع فترة ممكنة
  - الهدف هو تسريع بناء البرمجية بأقصى مايمكن
- استخدام النموذج التجريبي يحدد بدقة ماهي احتياجات الزبون
- لايتم بناء وتنفيذ النموذج التجريبي بشكل فعلي
  - البنية الداخلية للنموذج التجريبي غير مهمة
  - النموذج التجريبي يمكن تعديله بسرعة لتلبية حاجات المستخدمين

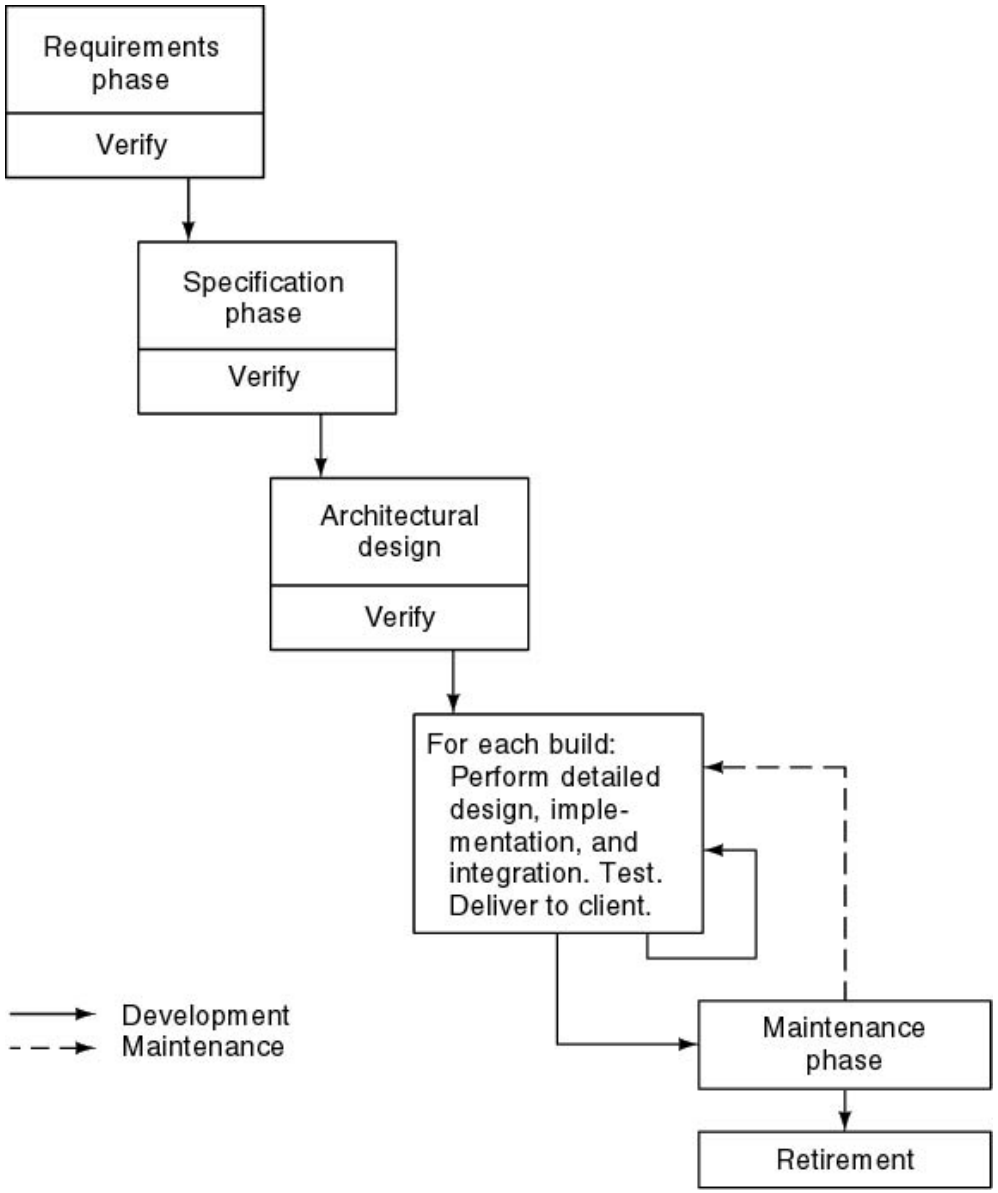
نقاط القوة ونقاط الضعف

- المزايا
  - مقارنةً مع موديل شلال الماء
    - عملية التطوير في خطية من مرحلة بناء نموذج تجريبي لمرحلة بناء منتج نهائي
    - عمليات العودة للوراء ضمن الموديل غير متوقعة
  - المساوئ
    - يصبح هذا الموديل غير نافع إذا كان من غير الممكن إدخال المستخدم في دورة حياة بناء المنتج البرمجي
    - إذا لم يكن من الممكن اإعادة استخدام مكونات وعناصر جاهزة, فزمن بناء المنتج البرمجي لا يمكن إنقاصه
    - يتطلب مبرمجين ذوي مهارة عالية
    - الزبائن يتوقعون أن يتم تلبية رغباتهم والتغييرات التي طلبوها بسرعة كبيرة كسرعة إنجاز الموديل التجريبي

موديل النموذج السريع أو موديل شلال الماء

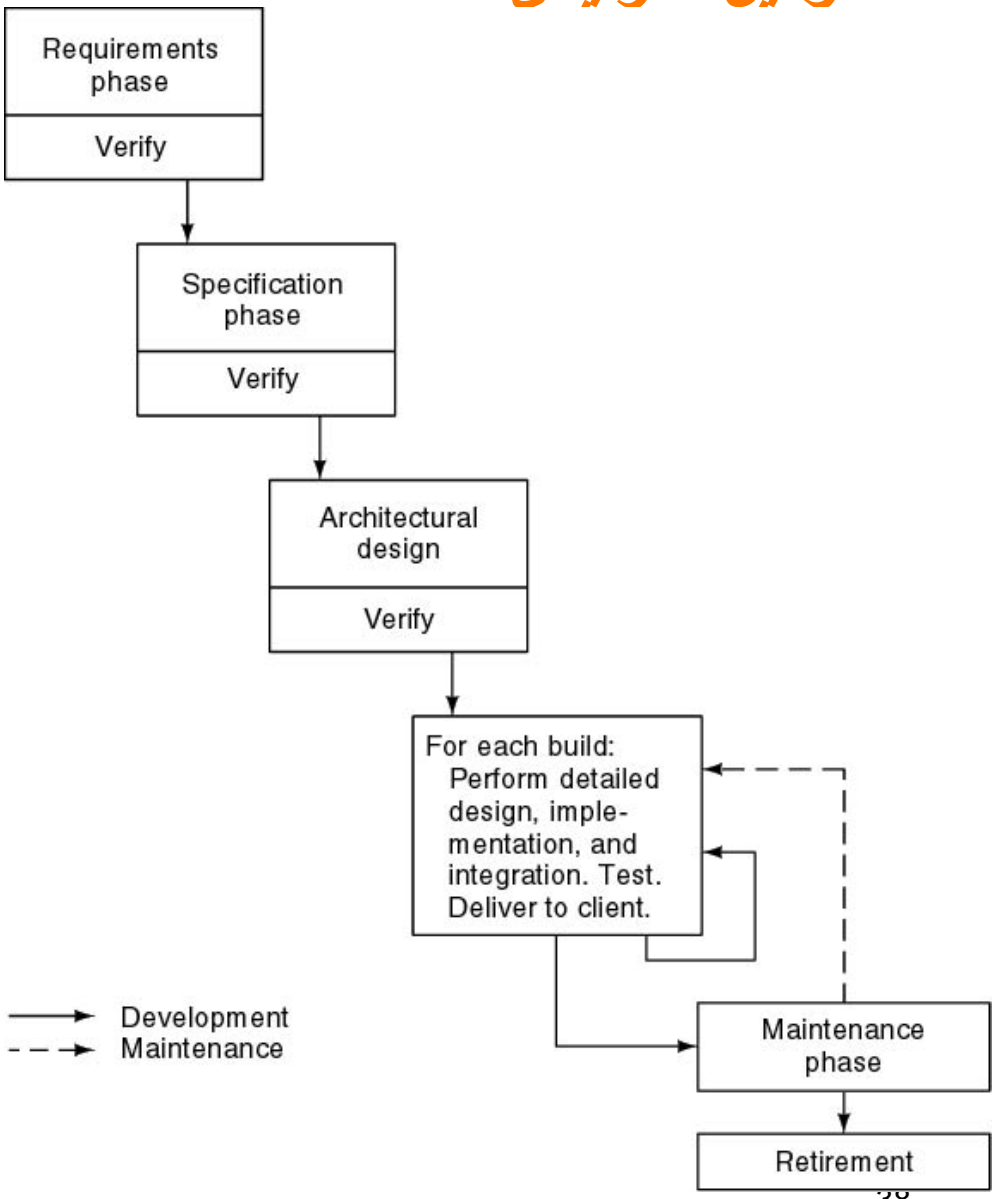
- موديل شلال الماء
  - نجاحات عديدة
  - يلبي حاجات المستخدم
- موديل النموذج السريع
  - غير مبرهن نجاحه
  - يمتلك مشاكل خاصة به متعلقة بالنموذج التجريبي
- الحل
  - موديل النموذج السريع لمرحلة المتطلبات
  - موديل شلال الماء لبقية المراحل

الموديل التزايدي



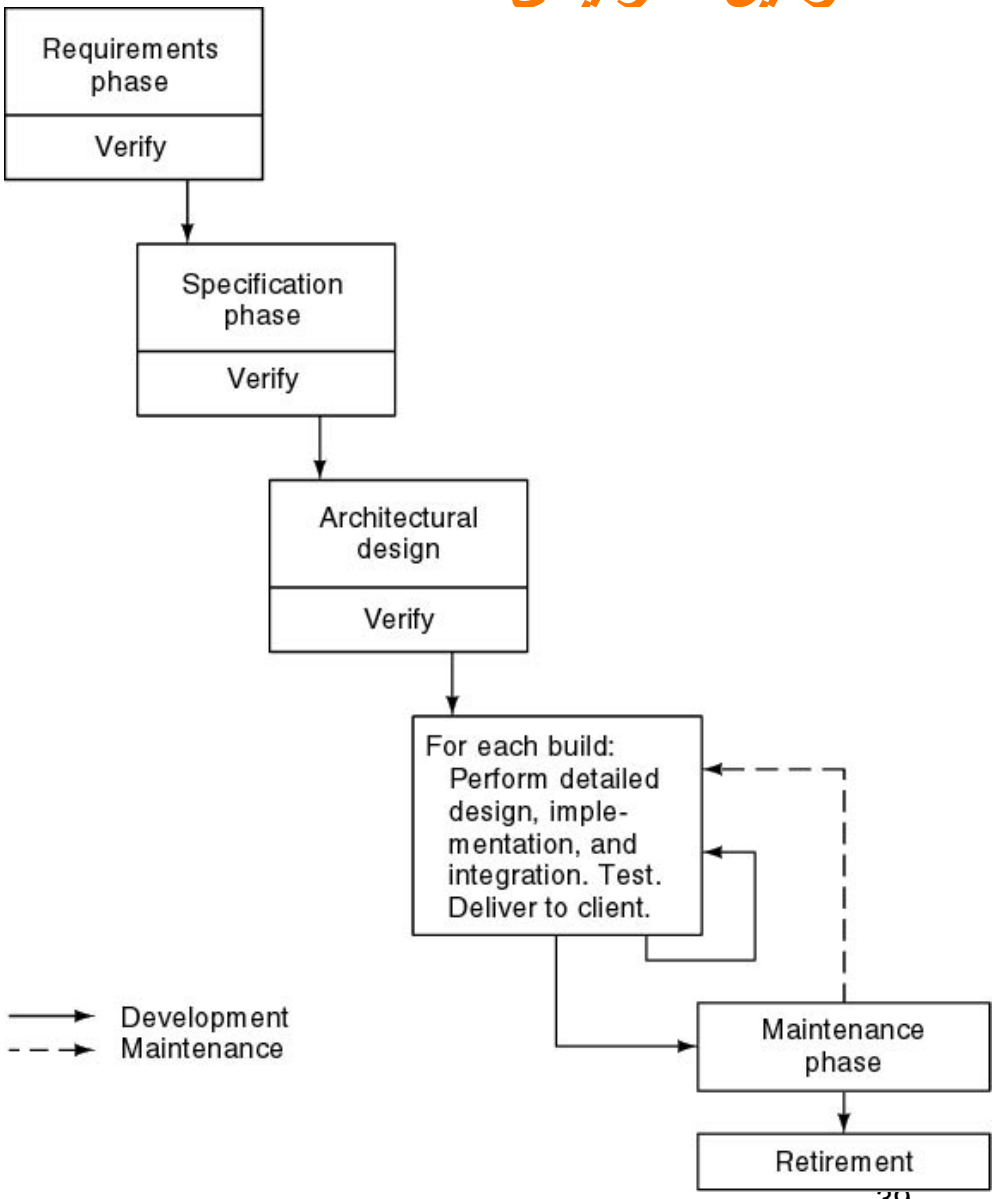
الموديل التزايدى

- يتم تصميم المنتج, تنفيذه, دمجه وفحصه كسلسلة من الأجزاء البرمجية (builds) المتتالية في البناء
  - الـ build عبارة عن مجموعة من التعليمات البرمجية من مجموعة من الموديلات المصممة لتنفيذ وظيفة معينة
- يتم في كل مرحلة بناء build جديد ومن ثم دمجه مع البرنامج الجاري تصميمه لتكوين برنامج كامل متكامل



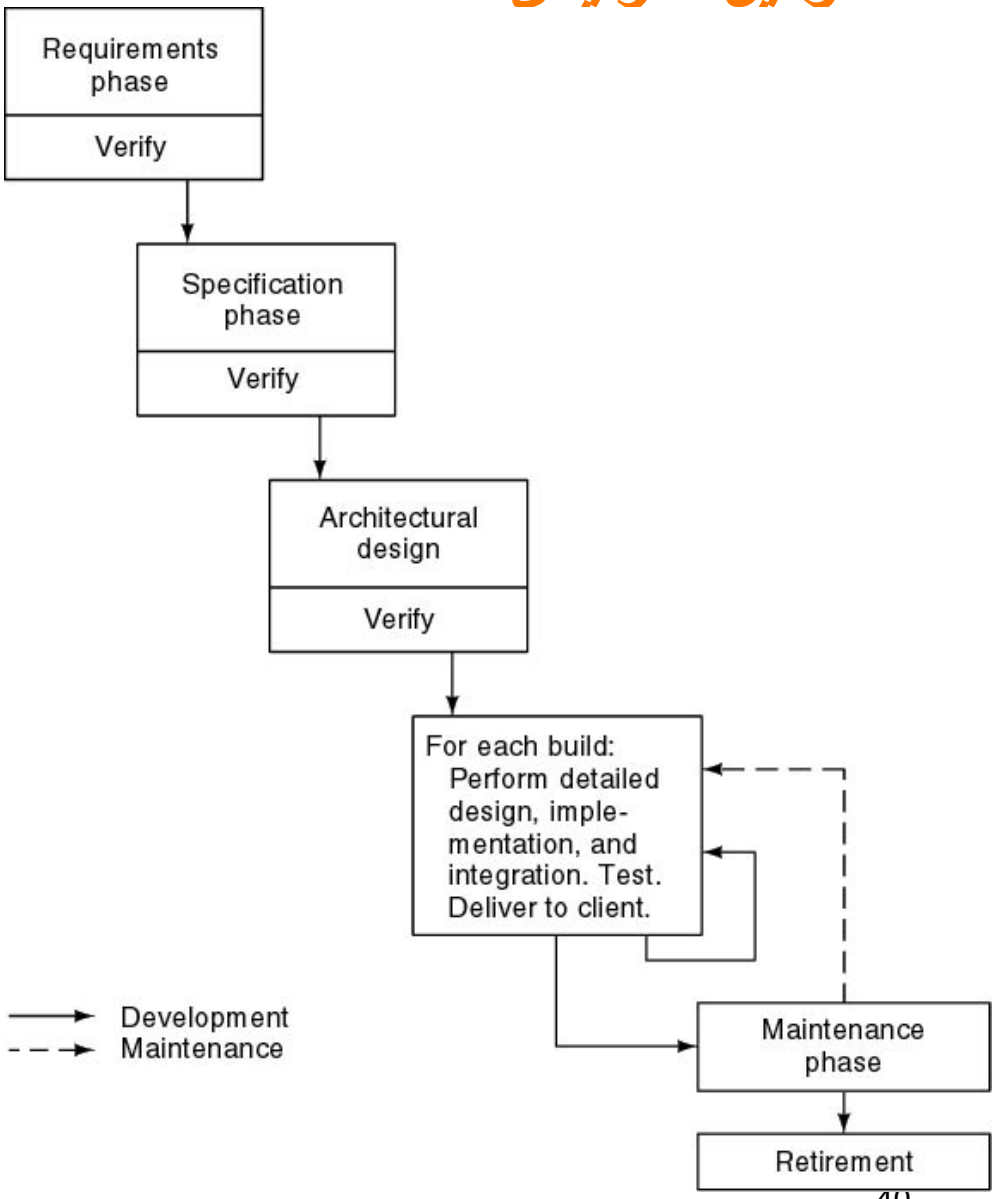
الموديل التزايدى

- يتم تقسيم المنتج البرمجي إلى مجموعة من الـ builds
  - كل build يتم دمجه بعد بنائه في المنتج البرمجي الموجود. المنتج الناتج يجب أن يكون قابلاً للفحص
- إذا كان المنتج البرمجي مؤلفاً من عدد كبير من الـ builds
  - زمن كبير لعملية الدمج والفحص, حيث بعد كل عملية دمج يتم فحص المنتج البرمجي بشكل كامل



الموديل التزايدى

- إذا كان المنتج البرمجي مؤلفاً من عدد قليل من الـ builds
  - يعود الموديل التزايدي إلى موديل build and fix



نقاط القوة ونقاط الضعف

- المزايا
  - التقديم التدريجي للمنتج البرمجي للزبون يعطي الزبون إمكانية الإطلاع على المنتج البرمجي و تعديله حسب رغبته
  - تغيير المنتج البرمجي و تأقلمه مع متطلبات المستخدمين أمر طبيعي

- المساوئ
  - كل build جديد يجب أن يتم دمجه مع المنتج البرمجي الموجود بدون إلحاق الضرر بالمنتج البرمجي ووظائفه الموجودة قبل الدمج
  - دمج الـ build الجديد يجب أن يكون سهل ومخطط له

  - لايميز الموديل التزايدي بين تطوير المنتج البرمجي و صيانته
  - يعبر عن المنتج البرمجي كسلسلة من الـ builds

فحص واختبار البرمجيات (Verification & Validation)

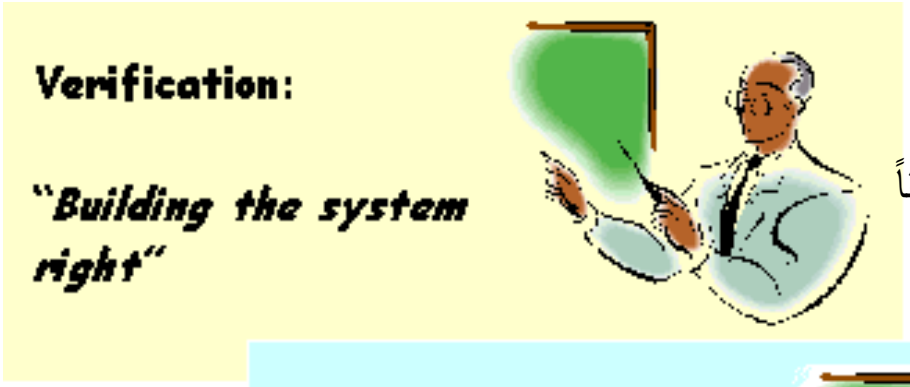
محتويات المحاضرة

- تعريفات
- التحقق و الإختبار الديناميكي (Dynamic & Static V&V)
- فحص البرنامج (Program Testing)
- التحري عن البرمجيات (Software Inspections)
- V-Model
- استراتيجيات الفحص (Testing Strategies)

تعريفات

- التحقق (Verification)
  - السؤال: هل نقوم ببناء المنتج الصحيح؟
  - البرمجيات يجب أن تتوافق مع المواصفات الخاصة بها (specification)

- الإختبار (Validation)
  - السؤال: هل نقوم ببناء المنتج بشكل صحيح؟
  - البرمجية يجب أن تفعل مايريده المستخدم حقاً



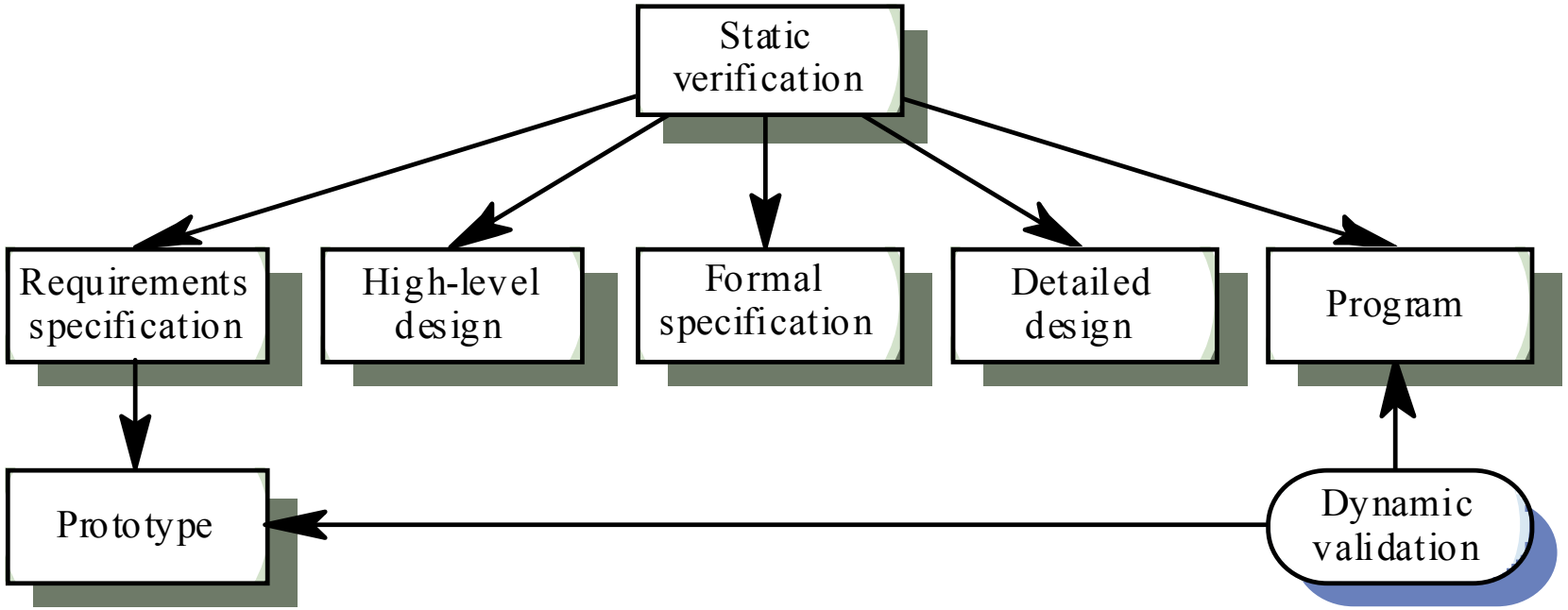
تعريفات

- التحقق و الإختبار (Verification & Validation (V&V))
  - تُستخدم للتحقق من أن البرمجية تحقق متطلبات المستخدم
  - هي عبارة عن طريقة بدورة حياة كاملة (life-cycle process), يجب أن يتم تطبيقها في كل مرحلة من مراحل تطوير المنتج البرمجي
  - له هدفين مبدئيين
    - اكتشاف الأخطاء الموجودة في النظام
    - تخمين قدرة النظام على العمل في ظروف العمل الفعلية (the system is usable in an operational situation)

التحقق و الإختبار الديناميكي (Dynamic & Static V&V)

- التحقق و الإختبار الديناميكي يهتم باختبار و مراقبة سلوك النظام **(testing)**

- التحقق و الإختبار الستاتيكي يهتم بتحليل بنية وتمثيل النظام لاكتشاف المشاكل



فحص البرنامج (Program Testing)

مدخل إلى فحص البرنامج

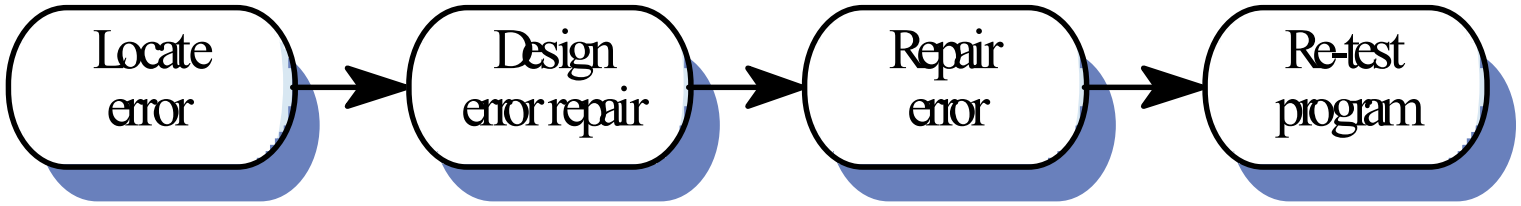
- يستطيع إثبات وجود أخطاء وليس غيابها
- فحص ناجح (successful test): عبارة عن فحص يكتشف خطأ أو أكثر
- المتطلبات الغير وظيفية (non-functional requirements) يجري فقط اختبارها (validation), ولايمكن التحقق منها (verification)
- يجب أن يُستخدم بالتناغم مع التحقق الستاتيكي (static verification) من البرنامج

أنماط الفحص

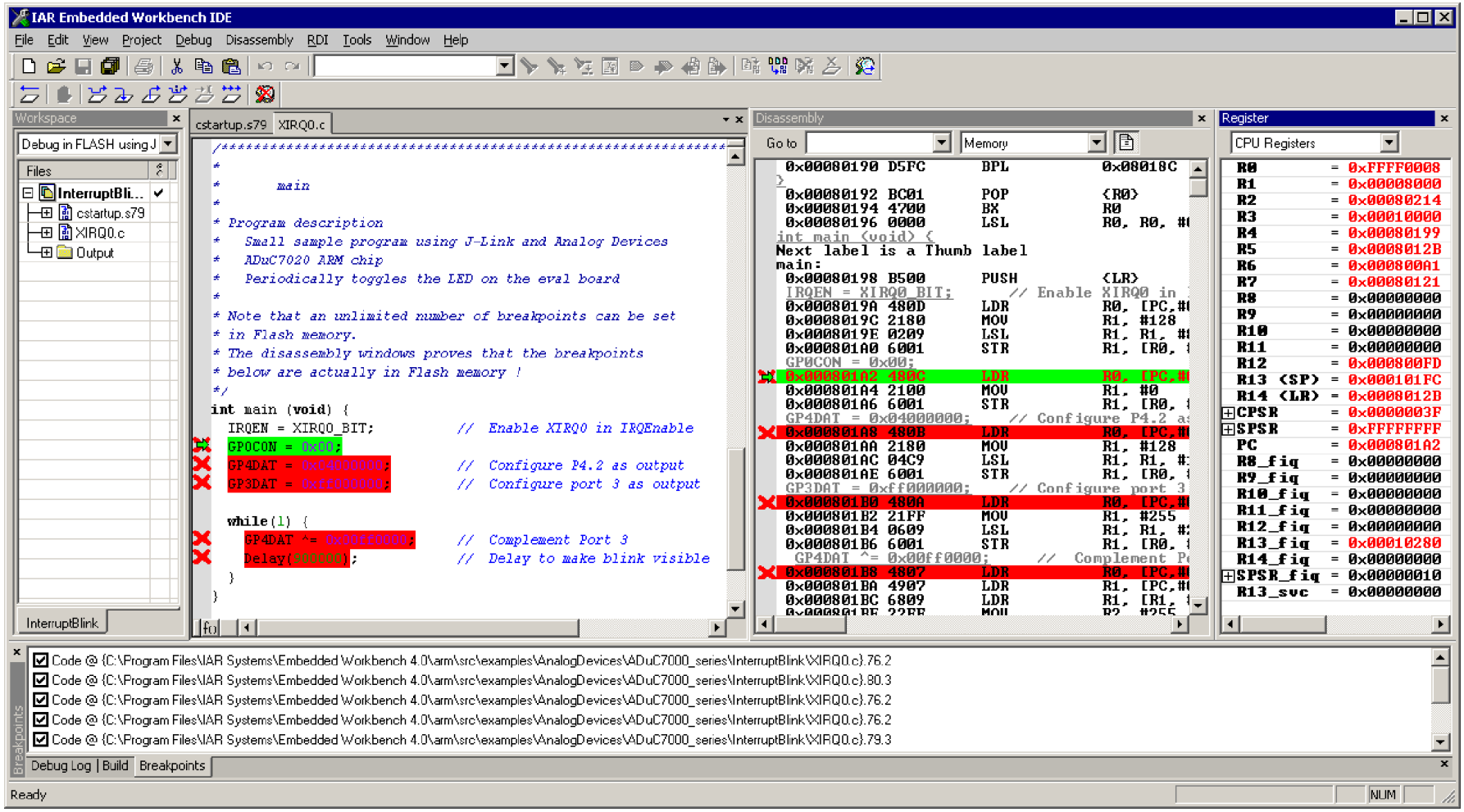
- فحص إحصائي (Statistical testing)
  - عبارة عن فحص موجه ليعكس تردد دخل المستخدم (the frequency of user inputs)
  - يستخدم لفحص الموثوقية (reliability estimation)
- فحص الأخطاء (Defect testing)
  - عبارة عن فحص موجه لاكتشاف الأخطاء الموجودة في نظام ما
  - فحص أخطاء ناجح (successful defect test): عبارة عن فحص أخطاء يكتشف وجود خطأ أو أكثر في نظام ما

الإختبار والتنقيح (Testing & Debugging)

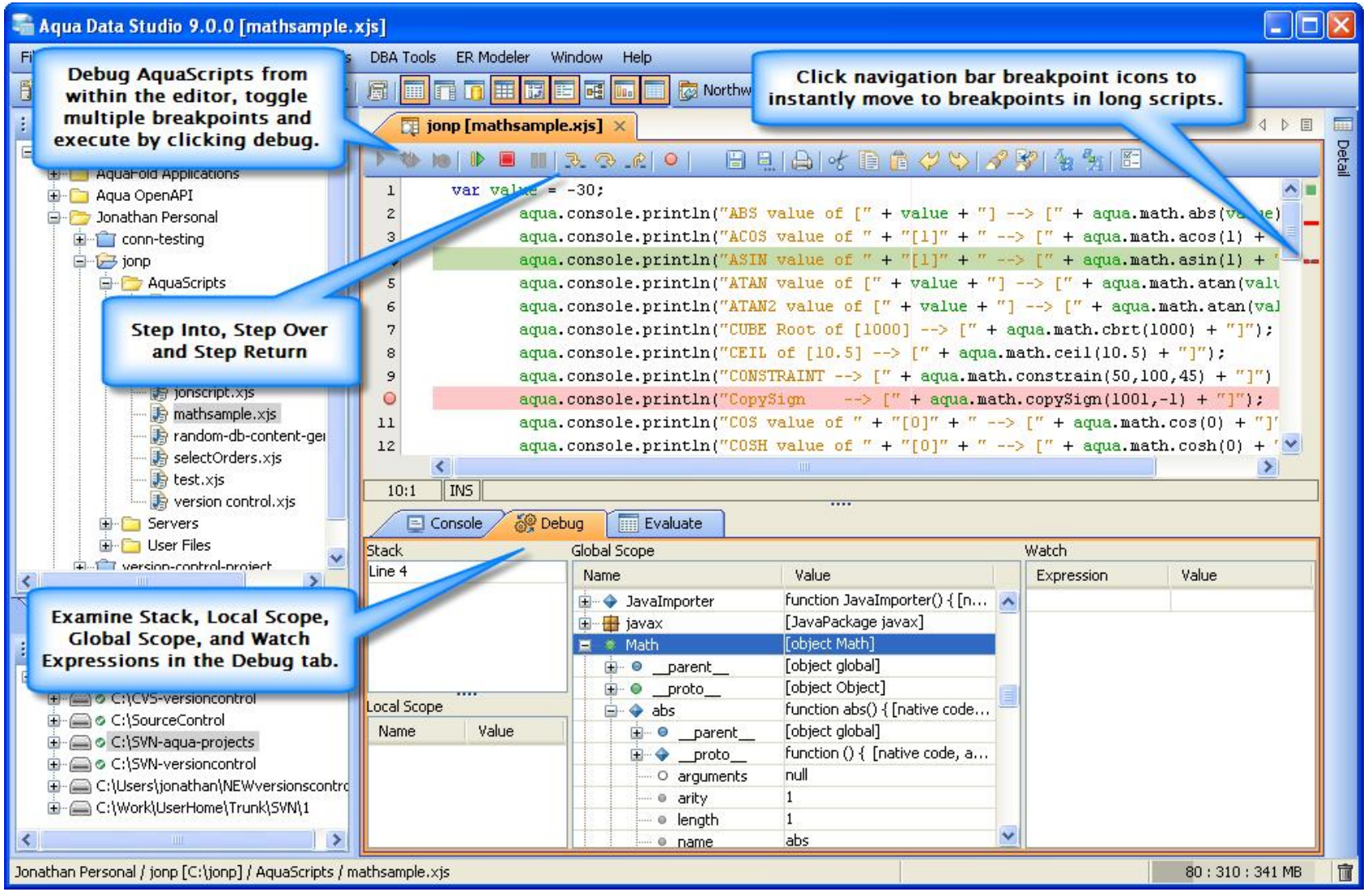
- فحص الأخطاء (Defect testing) و التنقيح (debugging) عمليتين مستقلتين (distinct processes)
- فحص الأخطاء (Defect testing) مرتبط باكتشاف الأخطاء الموجودة
- التنقيح (debugging) مرتبط بتحديد مكان الخطأ وإصلاحه
- عملية التنقيح (Debugging process)



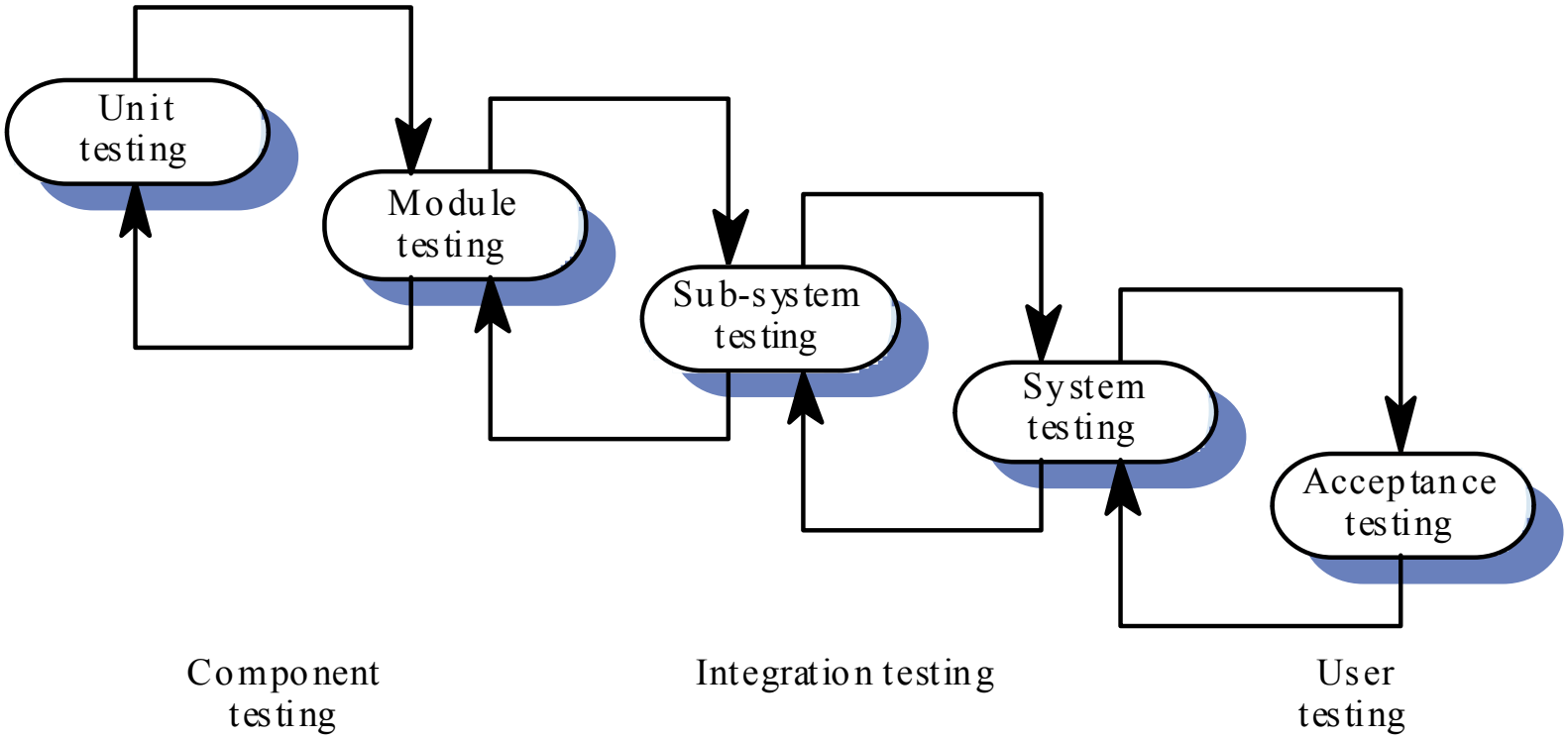
الإختبار والتنقيح (Testing & Debugging)



الإختبار والتنقيح (Testing & Debugging)



مراحل الإختبار (Testing Stages)



مراحل الإختبار (Testing Stages)

- Unit testing
  - فحص المكونات المفردة (Testing of individual components)
- Module testing
  - فحص مجموعة من المكونات المرتبطة ببعضها (Testing of collections of dependent components)
- Sub-system testing
  - فحص مجموعة من الموديلات المرتبطة ببعضها والتي تكون نظام فرعي (Testing collections of modules integrated into sub-systems)

مراحل الإختبار (Testing Stages)

- System testing
  - فحص النظام بأكمله قبل تسليمه للزبون
- Acceptance testing
  - يتم إجراؤه من قبل المستخدمين لفحص فيما كان البرنامج يلبي المتطلبات
  - يُدعى أحياناً alpha testing

تخطيط و جدولة عملية الفحص (Test Planning & Scheduling)

- تصف المراحل الأساسية لعملية الفحص
- تصف إمكانية تتبع فحص المتطلبات (trace-ability of tests)
- تقوم بتخمين الفترة الزمنية اللازمة (overall schedule) و المصادر المطلوبة (resource allocation)
- تصف العلاقة مع مراحل المشروع الأخرى (relationship with other project plans)
- تصف إجرائيات تسجيل نتائج الفحص (recording method for test results)

تخطيط و جدولة عملية الفحص (Traceability Scenario)

BRD- Section	FSD- Section	Test scenario ID	Test case ID	Status	Defects
1- Loan Process	1.1- New users	TS_Loan_001- Validate the "Apply Loan" feature as a new user	TC_newuser_01	Passed	
			TC_newuser_02	Passed	
			TC_newuser_03	Failed	Defect_01, Defect_02
		TS_Loan_002- validate the "Apply Loan" feature as a already existing user	TC_newuser_04	Passed	
			TC_newuser_05	Blocked	Defect_01
			TC_newuser_06	Failed	Defect_03
			TC_newuser_07	Passed	
		TS_Loan_003- For a new user in the "Apply loan", check the guest customer option and apply loan	TC_newuser_08	Passed	
		TS_Loan_004 - For a new user in the "Apply loan", check the Register option and apply loan	TC_newuser_09	Passed	
	1.2- Existing users	TS_Loan_005- Login to the loan portal as an already a customer with a loan and check the information displayed	TC_Exist_User_01	Passed	
		TS_Loan_006-Check the Loan whose status is "Sent for review"	TC_Exist_User_02	Passed	
		TS_Loan_007-Check the Loan whose status is "Reviewed and Accepted"	TC_Exist_User_03	Passed	
		TS_Loan_008-Check the Loan whose status is "Reviewed and deleted"	TC_Exist_User_04	Passed	
2- Ease of use	2.1- Ease os use	TS_Loan_009-Check for a visitor if the information on the site is accessible in less than 3 clicks or not	TC_EasyUse_01	Passed	
		TS_Loan_010-Check for a registered user if the information on the site is accessible in less than 3 clicks or not	TC_EasyUse_02	Passed	
		TS_Loan_011-Check for a banker if the information on the site is accessible in less than 3 clicks or not	TC_EasyUse_03	Passed	

خطة الفحص (Test Plan)

- عملية الفحص (testing process)
  - تصف المراحل الاساسية لعملية الفحص (major phases of the testing process)

- إجرائيات لتتبع تحقيق المتطلبات (Requirements traceability)
  - تسمح هذه الإجرائيات بمتابعة تحقيق المتطلبات أو عدم تحقيقه. لذلك يجب أن يهتم المبرمجين بجعل كل متطلب ممكن التحقق منه بشكل منفرد.

- النقاط التي يجب فحصها (Tested items)
  - يجب في هذا البند أن يتم تحديد المنتجات والعمليات الجزئية التي يجب أن يتم فحصها (الوظائف الاساسية, العناصر المرئية, الخ)

خطة الفحص (Test Plan)

- جدول الفحص (Testing schedule)
  - يجب أن يتم في هذا البند تحديد الفترة الزمنية اللازمة (overall schedule) و المصادر المطلوبة (resource allocation)
  - بالبطبع يجب ربطه مع خطوات بناء النظام البرمجي ككل (project development schedule)
- المتطلبات البرمجية والمادية (Hardware and software requirements)
  - يجب أن يتم في هذا البند تحديد الأدوات البرمجية اللازمةلملية الفحص و الاستهلاك ال ستستهلكه هذه العملية من موارد النظام (hardware utilisation)

خطة الفحص (Test Plan)

- إجرائيات تسجيل نتائج الفحص (Test recording procedures)
  - تهتم هذه الإجرائيات بتسجيل نتائج الفحص بشكل منظّم, بحيث يمكن معاودة فحص هذه النتائج لاحقاً, وذلك للتحقق من العمل الصحيح للنظام.
  - يمكن عرض النتائج بطرق مختلفة, الأمر الذي يُسهل دراستها.
- التقييدات (Constraints)
  - يجب أن يتم في هذا البند تحديد ماهي الحدود التي يمكن أن تصل إليها عملية الفحص وماهي التقييدات التي تحد من هذه العملية

التحري عن البرمجيات (Software Inspections)

مدخل إلى التحري عن البرمجيات

- يُعتبر التحرّي عن البرمجيات (Software Inspection) من المواضيع الأساسية التي يقوم بها مجموعة من الأشخاص لفحص بنية البرمجيات وآليات تمثيلها (Source representation)
  - طريقة منظمة ذات خطوات منظمة
  - تهدف إلى اكتشاف وجود أخطاء برمجية وغير برمجية (Anomalies and defects)
- التحرّي عن البرمجيات (Software Inspection) لايُقدم حلولاً للأخطاء المكتشفة
- ليس هناك داعي لأن يكون البرنامج منتهياً وقابلاً للعمل (Runnable) حتى يُمكن استخدام تقنيات التحرّي عن البرمجيات

مدخل إلى التحري عن البرمجيات

- يمكن أن يتم تطبيقه على أي تمثيل للنظام (requirements, design, (configuration data, test data, etc.

- أثبتت أنها تقنية جيدة للكشف عن الأخطاء

نجاح التحري عن البرمجيات

- يمكن أن يتم اكتشاف العديد من الأخطاء في عملية تحري واحدة

- في عملية الفحص, خطأ ما يُمكن أن يحجب عدة أخطاء (one defect may mask another)
  - لذلك يجب إجراء عملية التحري ثانيةً بعد كل مرة يتم فيها اكتشاف وجود أخطاء

- يتم الإستفادة هنا من الخبرة بشكل كبير
  - الفاحصون يكتشفون بسرعة الأخطاء التي تتواجد عادةً

التحري والفحص (Inspections & Testing)

- التحري والفحص عمليتان مكلمتان لبعضهم وليسوا عمليات تحقق متعارضة
- التحري والفحص يستخدم أثناء الـ V & V process
- تستطيع عملية التحري أن تكتشف التوافق مع المواصفات (specification) وليس مع المتطلبات الحقيقة (the customer's real requirements)
- لاتستطيع فحص الخصائص غير الوظيفية (non-functional characteristics), كالأداء, سهولة الإستخد

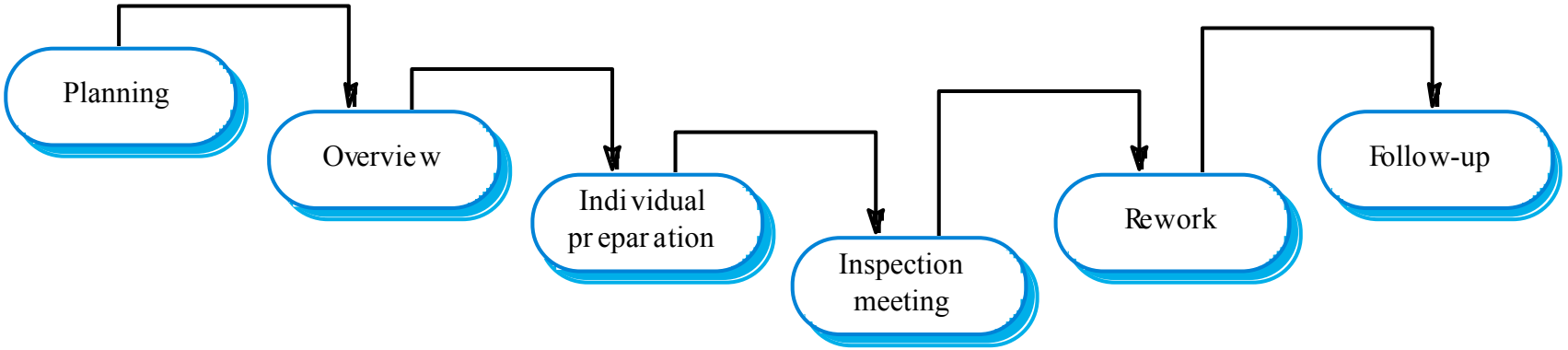
الشروط المسبقة اللازمة لنجاح عملية التحري عن البرنامج

- يجب أن يتواجد توصيف دقيق (Precise specification) للنظام البرمجي.
- يجب أن تتواجد لدى فريق العمل خبرة ودراية بالمعايير الدولية (Organization standards).

- يجب أن يتواجد شيفرة برمجية قابلة للتنفيذ (Syntactically correct code), أو أي تمثيل آخر للنظام.
- يجب أن يتواجد قائمة للفحص.

- يجب الأخذ بعين الإعتبار أن عملية التحري عن البرنامج سوف تُؤدي لزيادة الكلفة.
- يجب ألا تُؤخذ عملية التحري عن البرنامج وسيلة للإساءة لأعضاء الفريق.

عملية التحري (Inspection Process)



عملية التحري (Inspection Process)

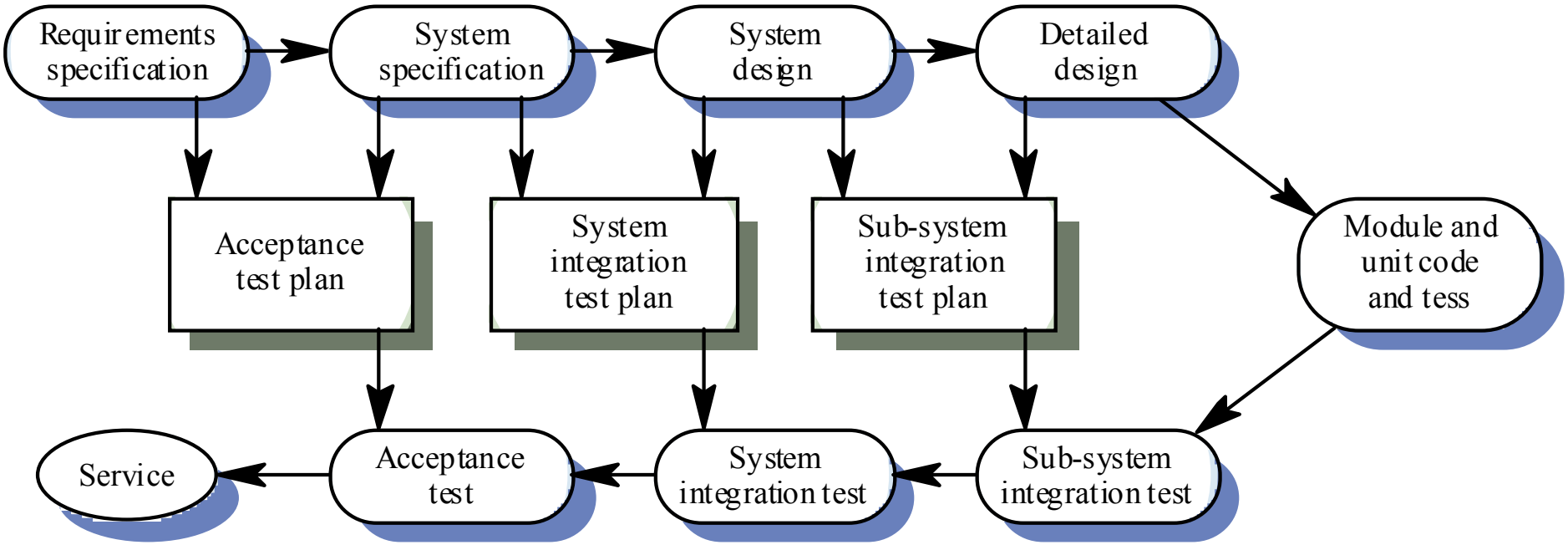
- يتم في المرحلة الأولى (Planning) التخطيط لعملية التحرّي عن البرمجيات (Software Inspection)
- في المرحلة الثانية (Overview) يتم استعراض الخطة وتوضيح بنودها,

- يتم في مرحلة (Individual preparation) إجراء عمليات التحري من قبل الأشخاص المشتركين بهذه العملية
- يتم عقد اجتماع (Inspection meeting) لتدارس النتائج وتقويم العمل

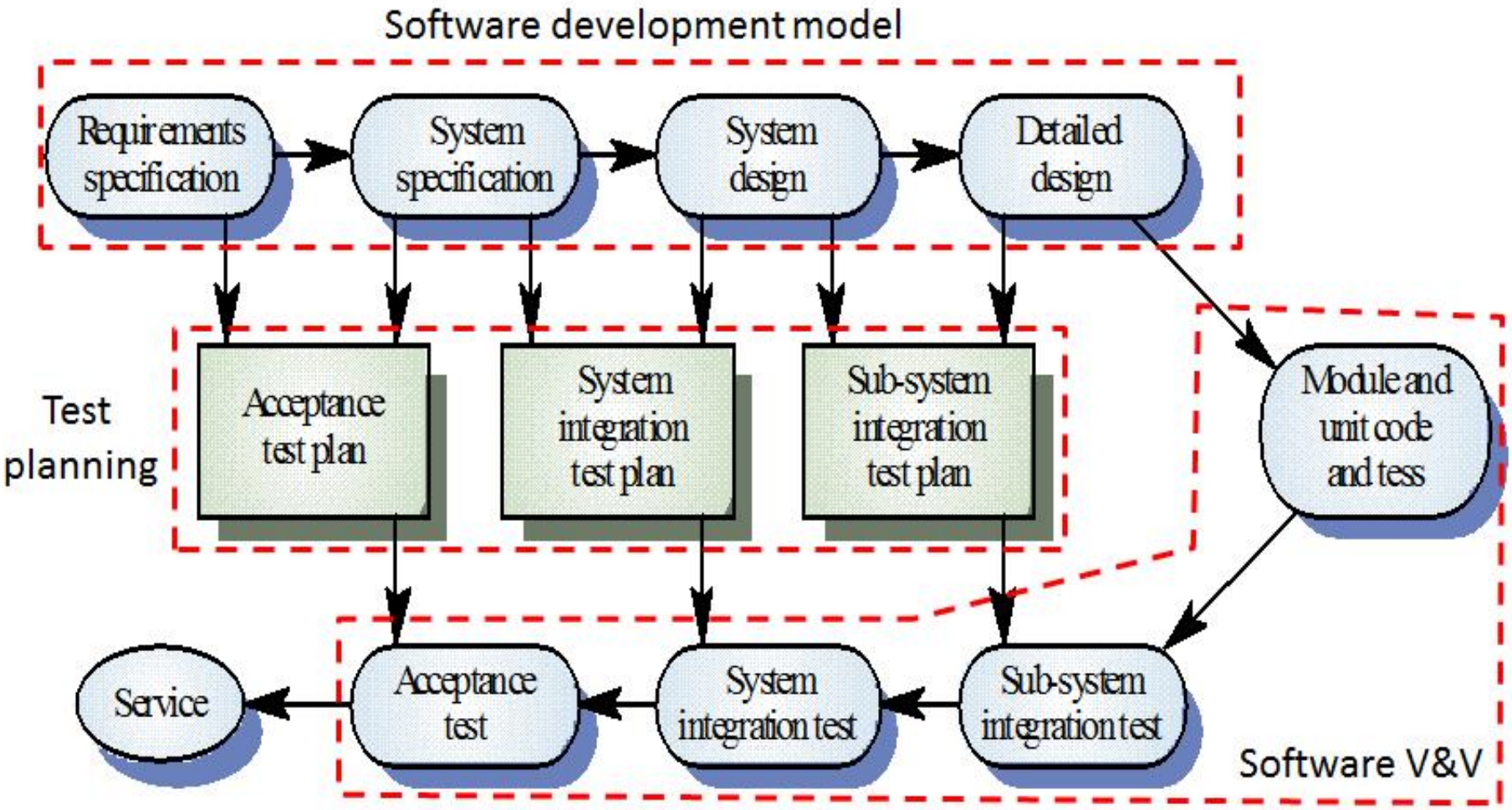
- في المرحلة التالية (Rework) يتم إعادة العمل لتصحيح الأخطاء في خطة العمل,
- يتم تكرار تنفيذ الخطة (Follow-Up)

V-Model

تطوير الـ V-Model



تطوير الـ V-Model



استراتيجيات الفحص (Testing Strategies)

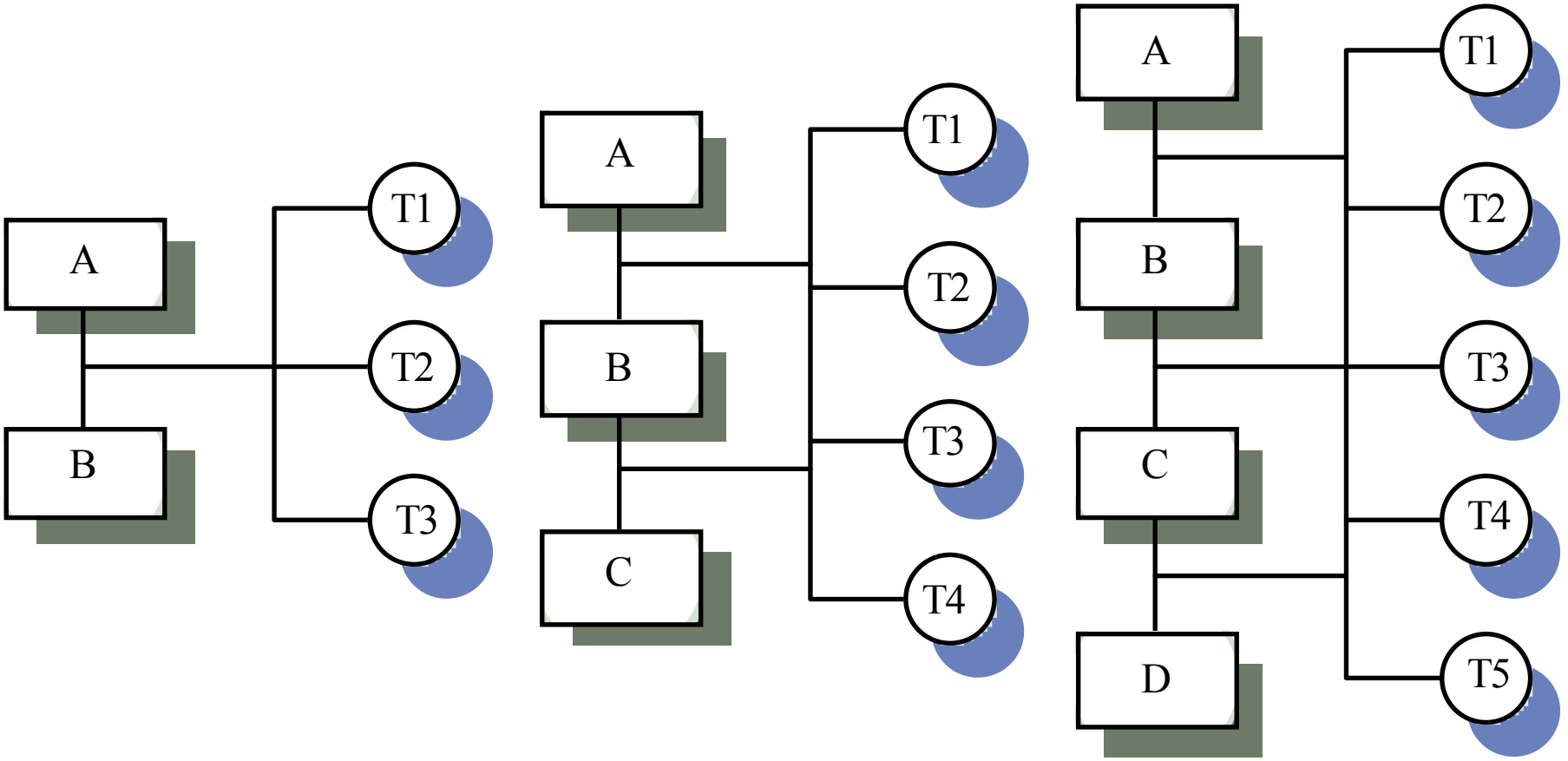
مدخل إلى استراتيجيات الفحص (Testing Strategies)

- استراتيجيات الفحص هي طرق منهجية لتنفيذ عملية الفحص (ways of approaching the testing process)

- يمكن أن يتم تطبيق استراتيجيات مختلفة في مراحل مختلفة من مراحل عملية الفحص

- أمثلة عن الاستراتيجيات
  - Top-down testing
  - Bottom-up testing
  - Object-oriented system testing
  - Thread testing
  - Stress testing
  - Back-to-back testing

Incremental Testing



Test sequence 1

Test sequence 2

Test sequence 3

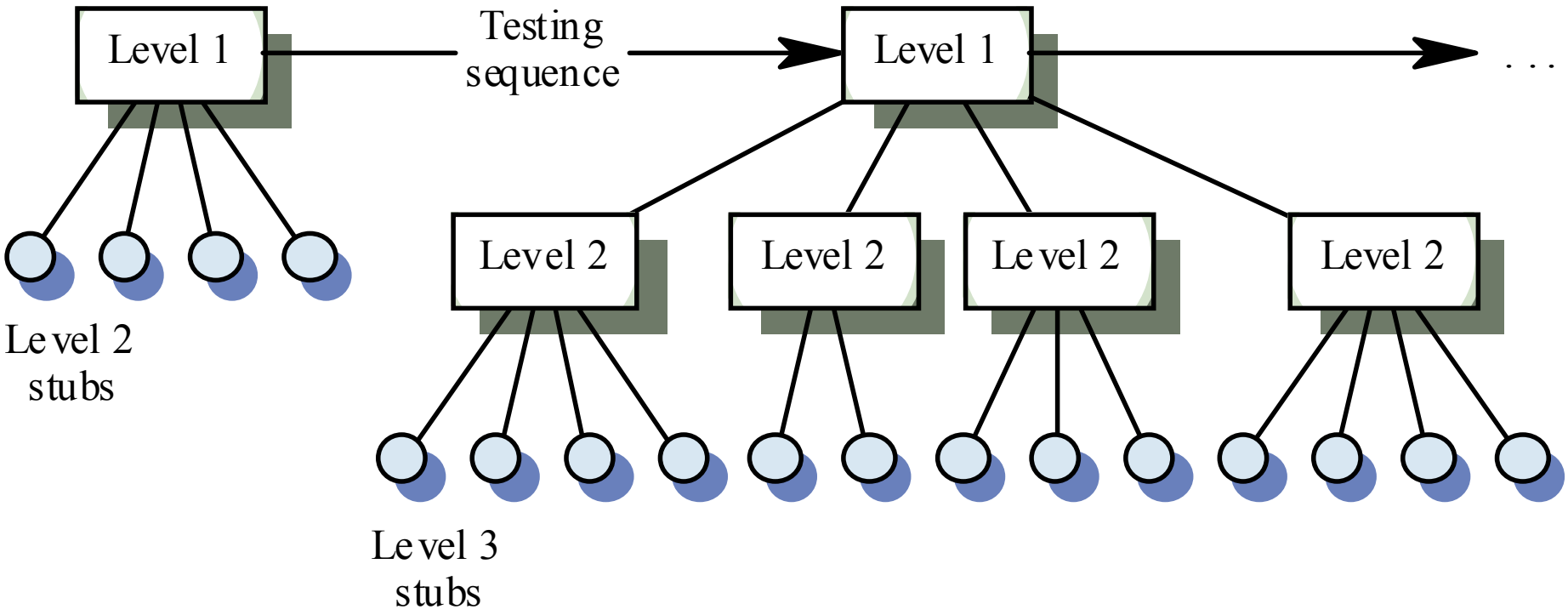
Incremental Testing

- تُعتبر طريقة Incremental Testing أحد الاستراتيجيات المستخدمة لفحص البرمجيات (Testing Strategies)

- تقوم هذه الطريقة بفحص المنتج البرمجي على شكل أجزاء, حيث يتم فحص الجزء الأول, ثم يتم إضافة جزء آخر من البرمجية و من ثمّ يتم فحص البرمجية المكونة من الجزأين كاملةً.

- هذه الطريقة مناسبة لفحص البرمجيات المبنية باستخدام الموديل التزايدي (Incremental model).

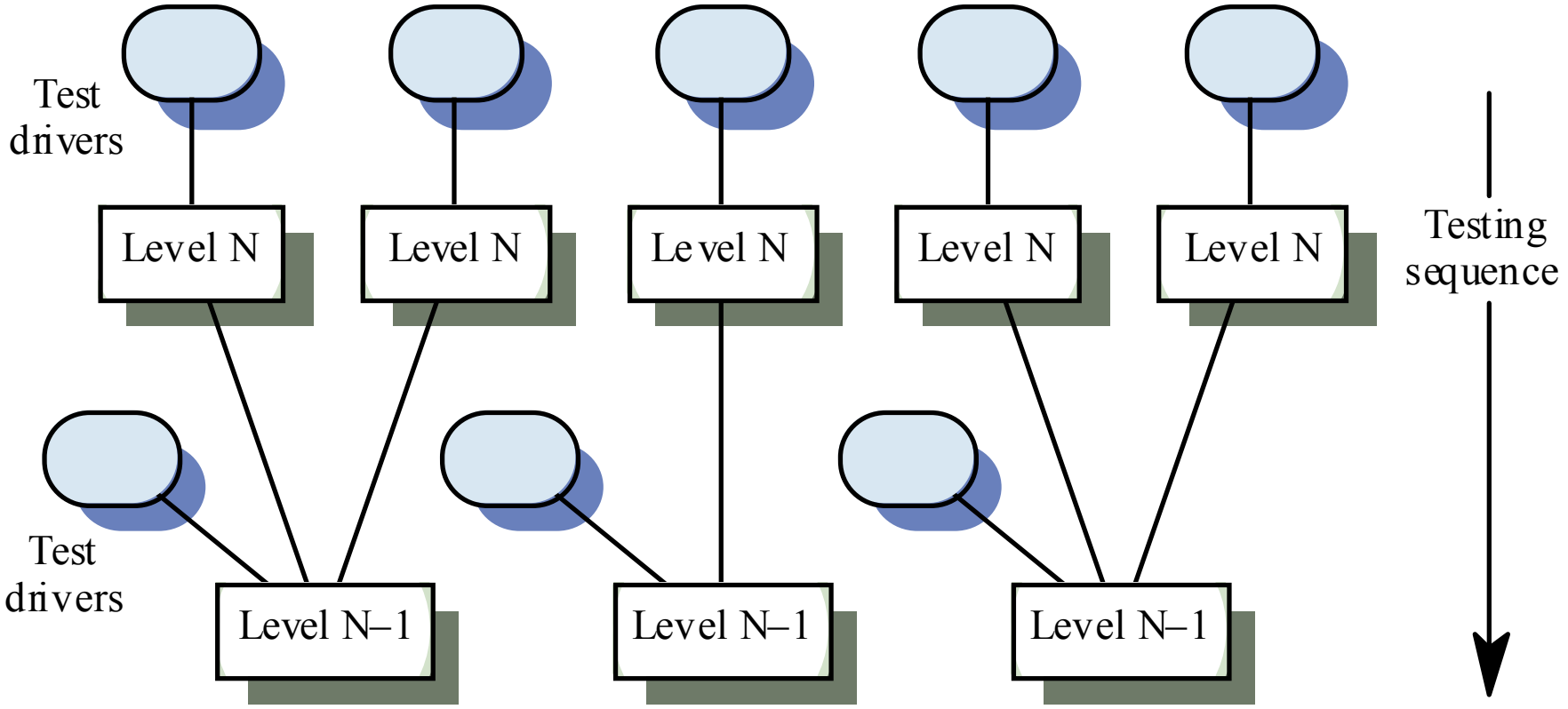
Top-Down Testing



Top-Down Testing

- تُعتبر طريقة Top-Down testing أحد الاستراتيجيات المستخدمة لفحص البرمجيات (Testing Strategies)
- تبدأ هذه الطريقة بفحص المنتج البرمجي من المستويات العليا (high levels) إلى المستويات الدنيا (low levels)
- تُستخدم مع موديلات التطوير التي تعمل وفق مبدأ Top-Down, موديل شلال الماء على سبيل المثال
- تستطيع اكتشاف وجود الأخطاء الأساسية في تصميم النظام (architectural errors)
- من المحتمل أن يكون هناك حاجة لبنية تحتية قبل عملية الفحص

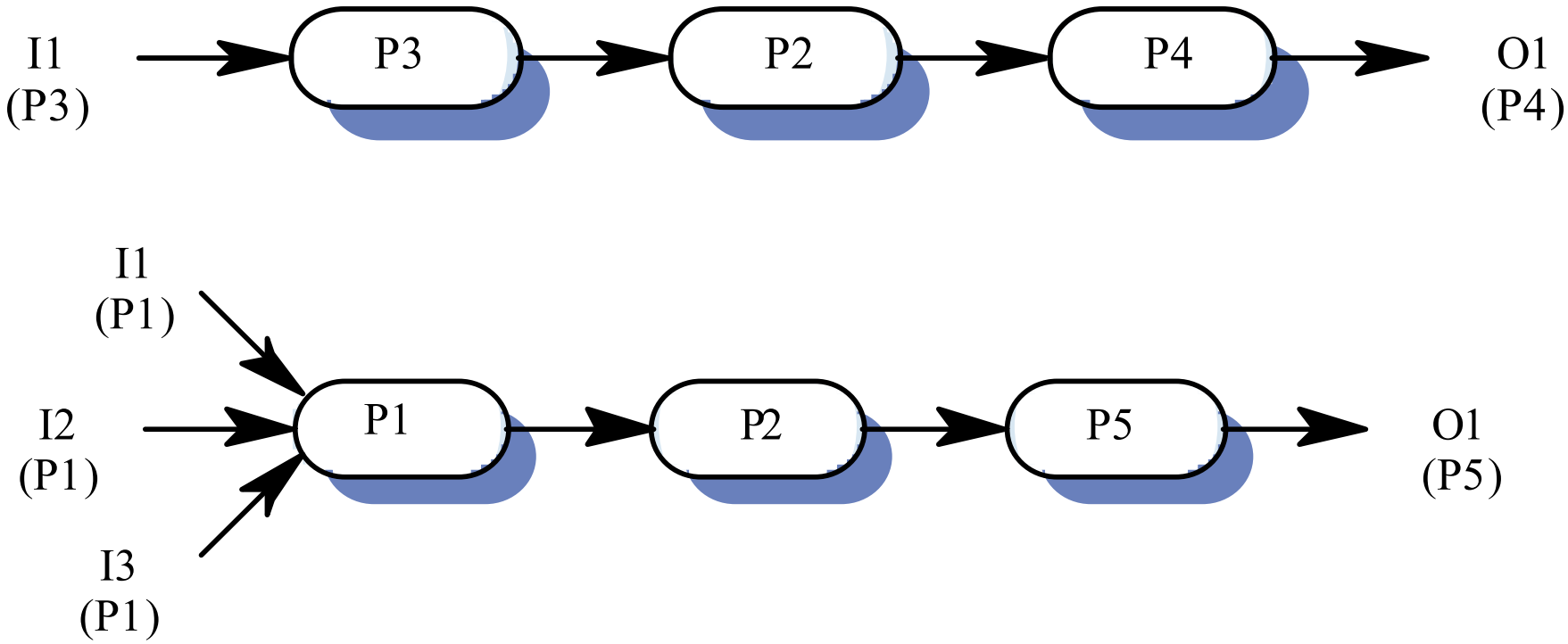
Bottom-Up Testing



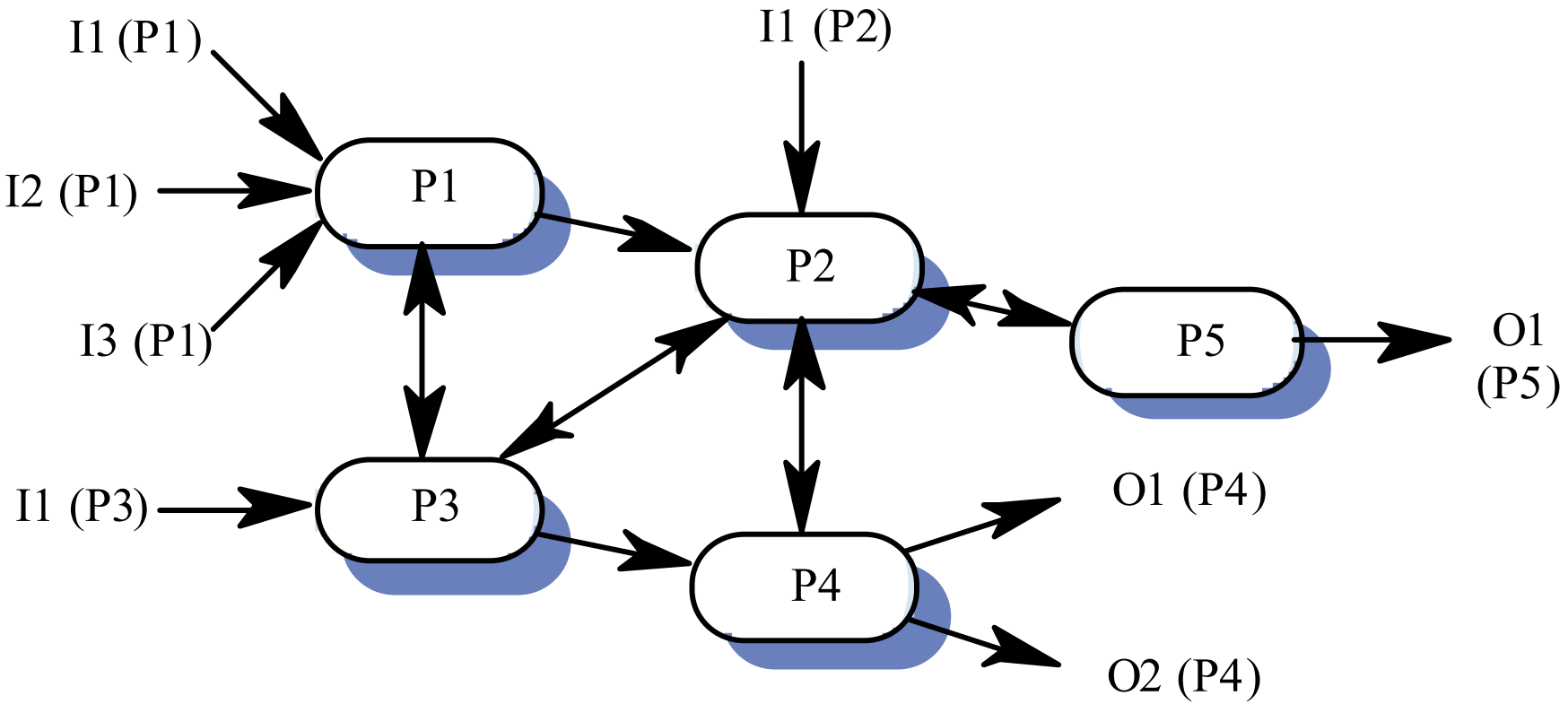
Bottom-Up Testing

- تبدأ هذه الطريقة بفحص المنتج البرمجي من المستويات الدنيا (low levels) إلى المستويات العليا (high levels)
- ضروري لفحص مكونات البنية التحتية الحساسة (critical infrastructure components)
- لاتستطيع اكتشاف وجود الأخطاء الأساسية في تصميم النظام حتى مراحل متأخرة من عملية الفحص.
- مناسبة للـ object-oriented systems

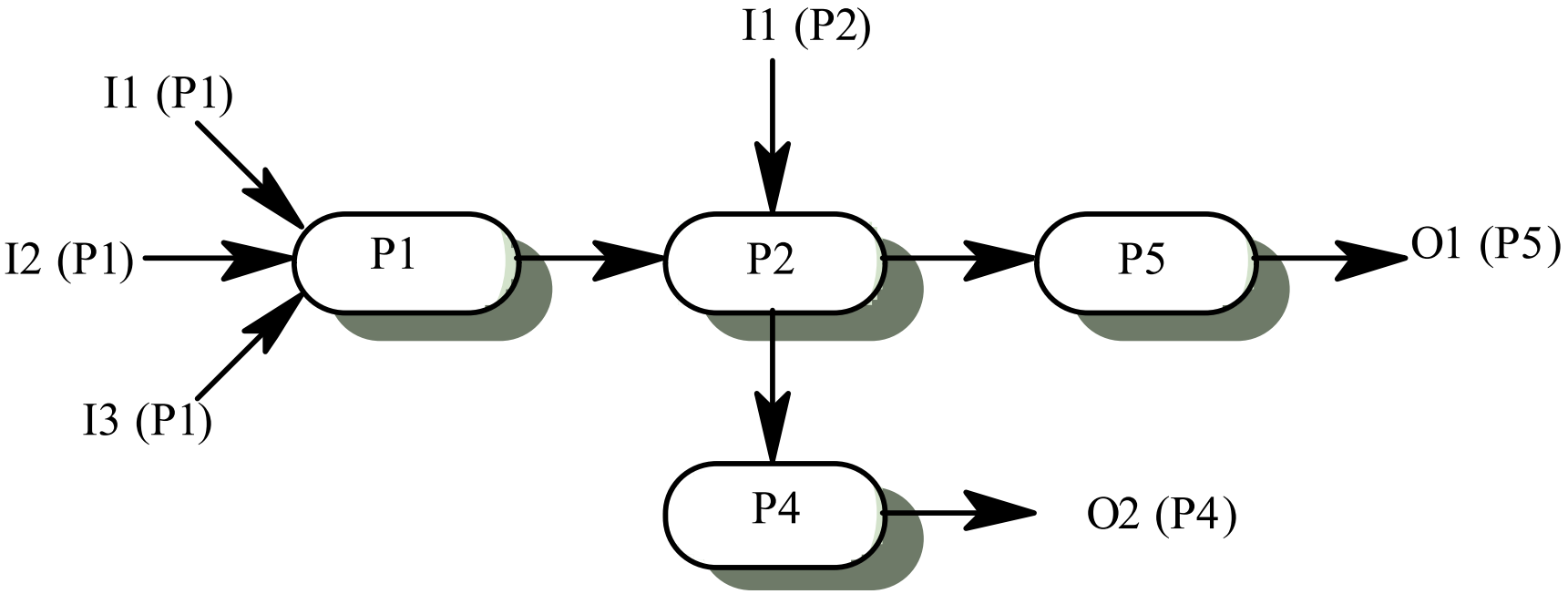
Thread Testing



Process Interactions



Multiple-Thread Testing



Stress Testing

- يتم تشغيل النظام البرمجي بشكل يتجاوز الحد الأعظمي للحمل
  - تحميل البرنامج بشكل زائد تُسبب إظهار الأخطاء التي لاتظهر عادةً

- تحميل البرنامج بشكل زائد تفحص إمكانية فشل النظام ضمن ظروف عمل غير اعتيادية
  - يجب ألا يتم تدمير عمل البرنامج وضياع البيانات, حتى في ظروف عمل غير اعتيادية

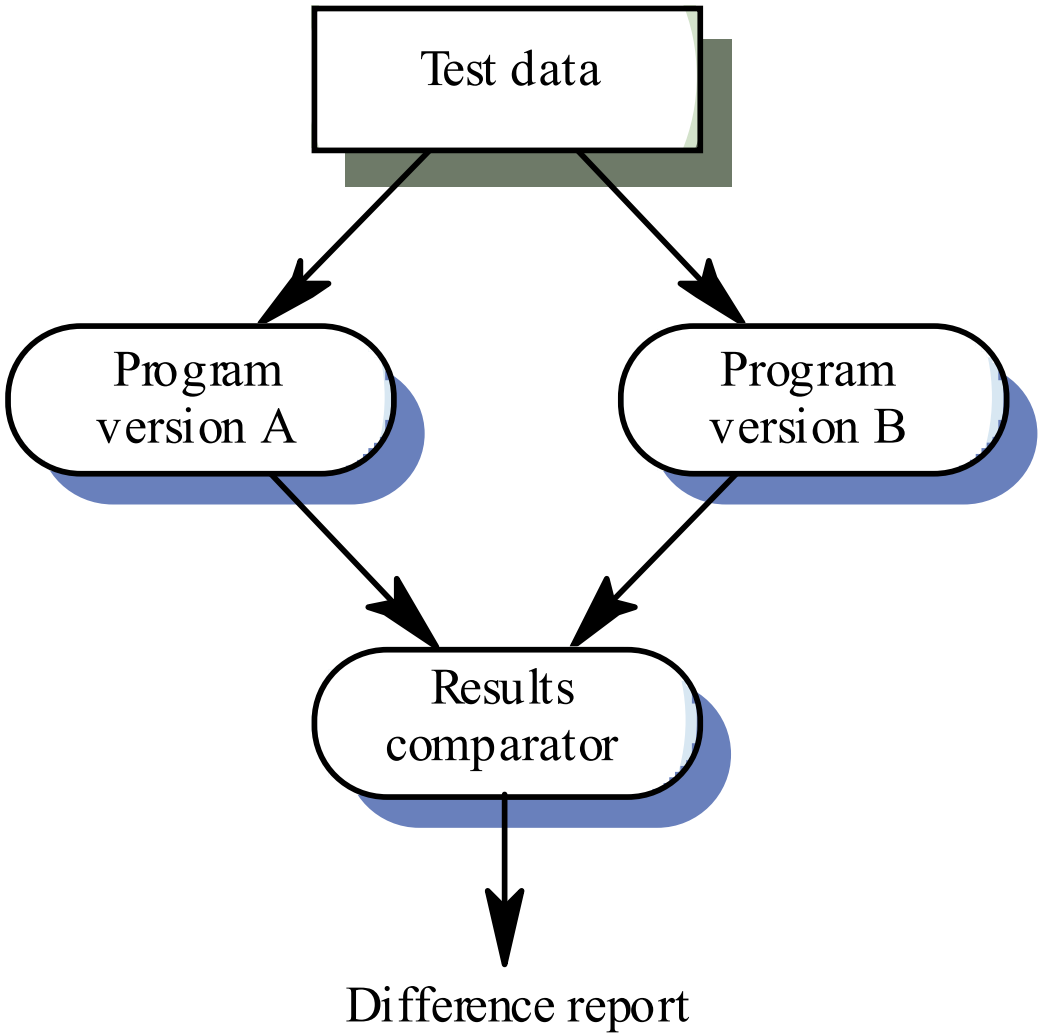
Back-to-Back testing

- يتم استخدام استراتيجية Back-to-Back testing لفحص فيما إذا كان المنتج البرمجي يعمل مع إصدارات قديمة,
  - بمعنى أن النتائج من إصدار معين للبرمجية يُمكن قراءتها ومعالجتها من نفس الإصدار للبرمجية أو من إصدارات أحدث.

- يمكن أتمتة هذه العملية بشكل شبه كامل

- تختاج تواجد منتج جاهز أو prototype على الأقل

Back-to-Back Testing



هندسة البرمجيات غرضية التوجه

Object-Oriented Software Engineering (OOSE)

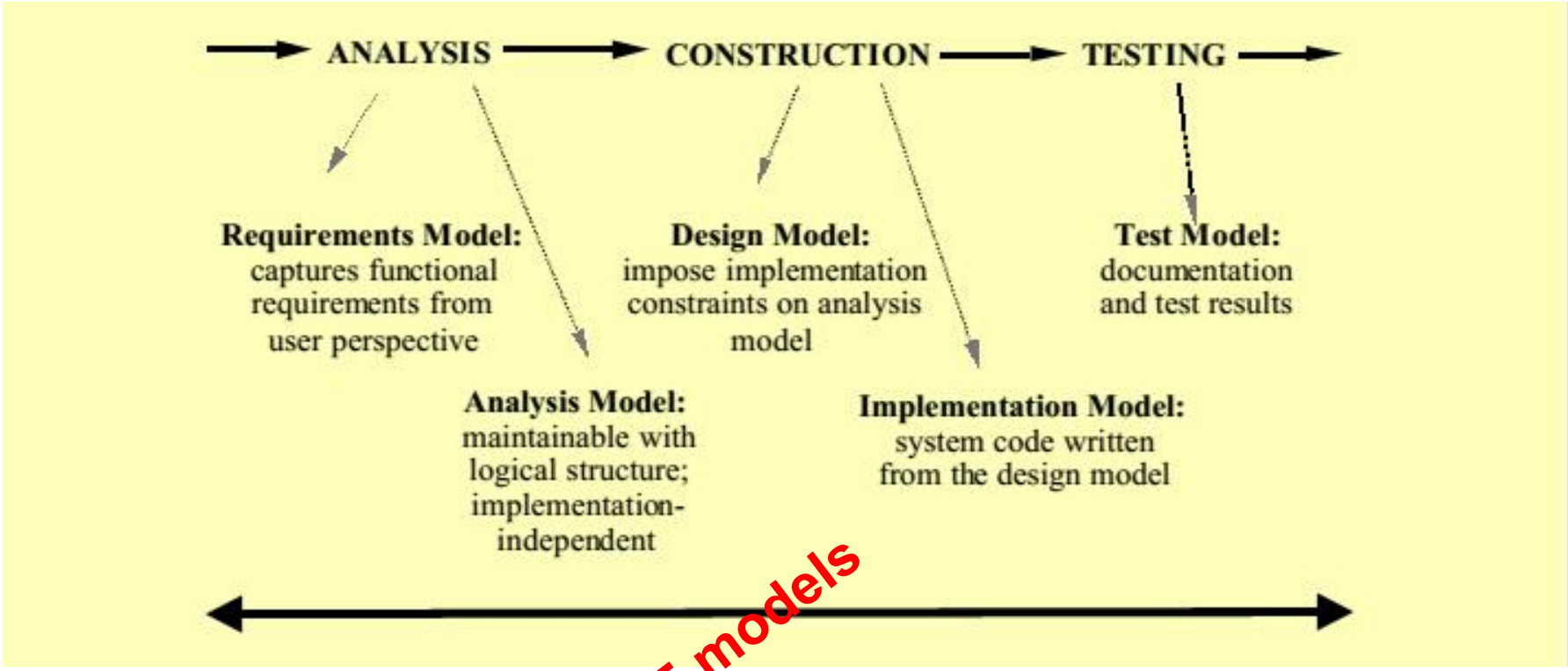
مقدمة عامة

- عبارة عن طريقة تعتمد على حالات الاستخدام (Use Case Driven Approach)
- طرريقة براغماتية (Pragmatic method) تعتمد على الخبرة
- شائعة الاستخدام و ناجحة
- عبارة عن طريقة متكاملة

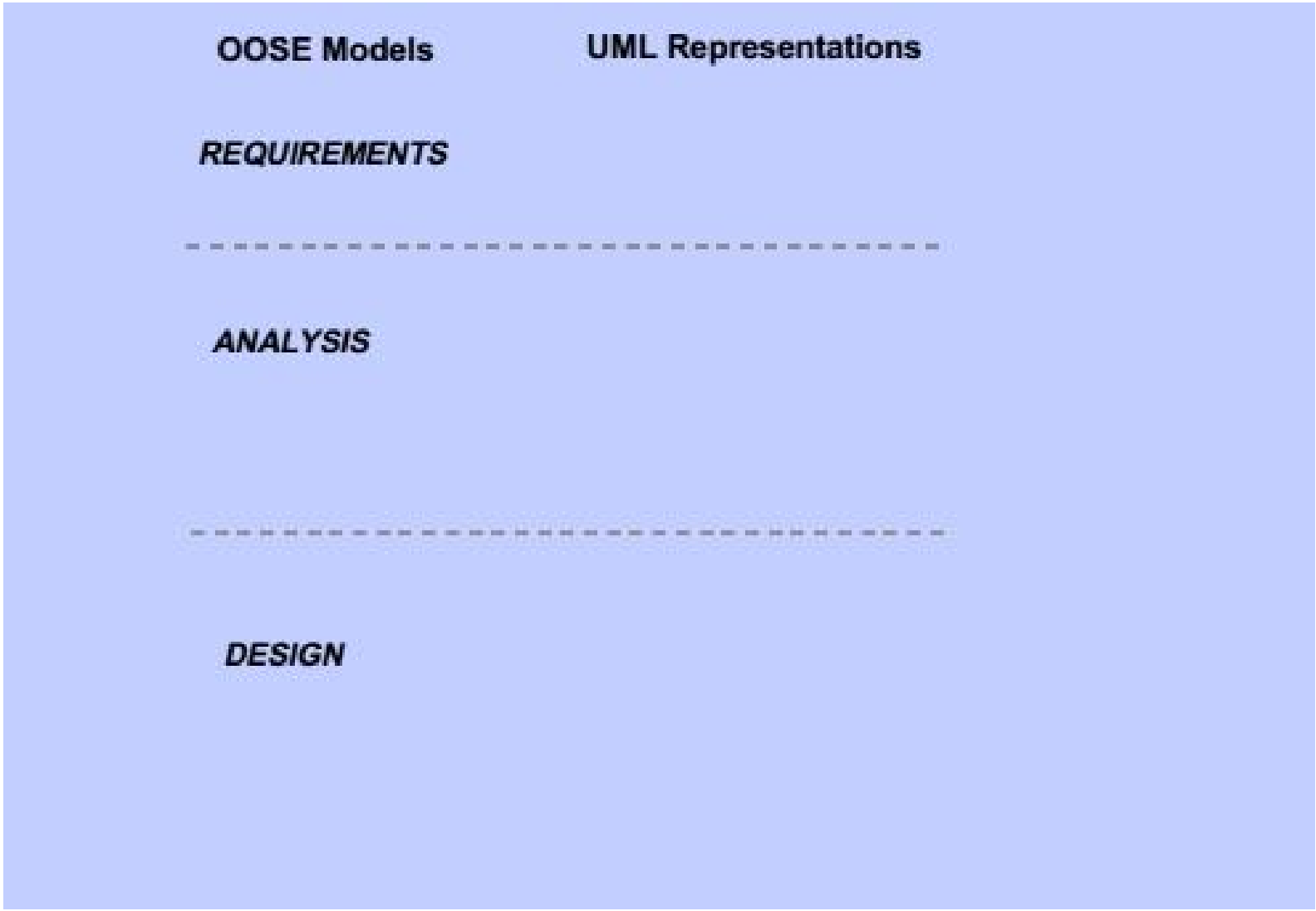
ماذا تتضمن الطريقة

- Syntax (how it looks)
- Semantics (what it means)
- Pragmatics (heuristics, rules of thumb for use)

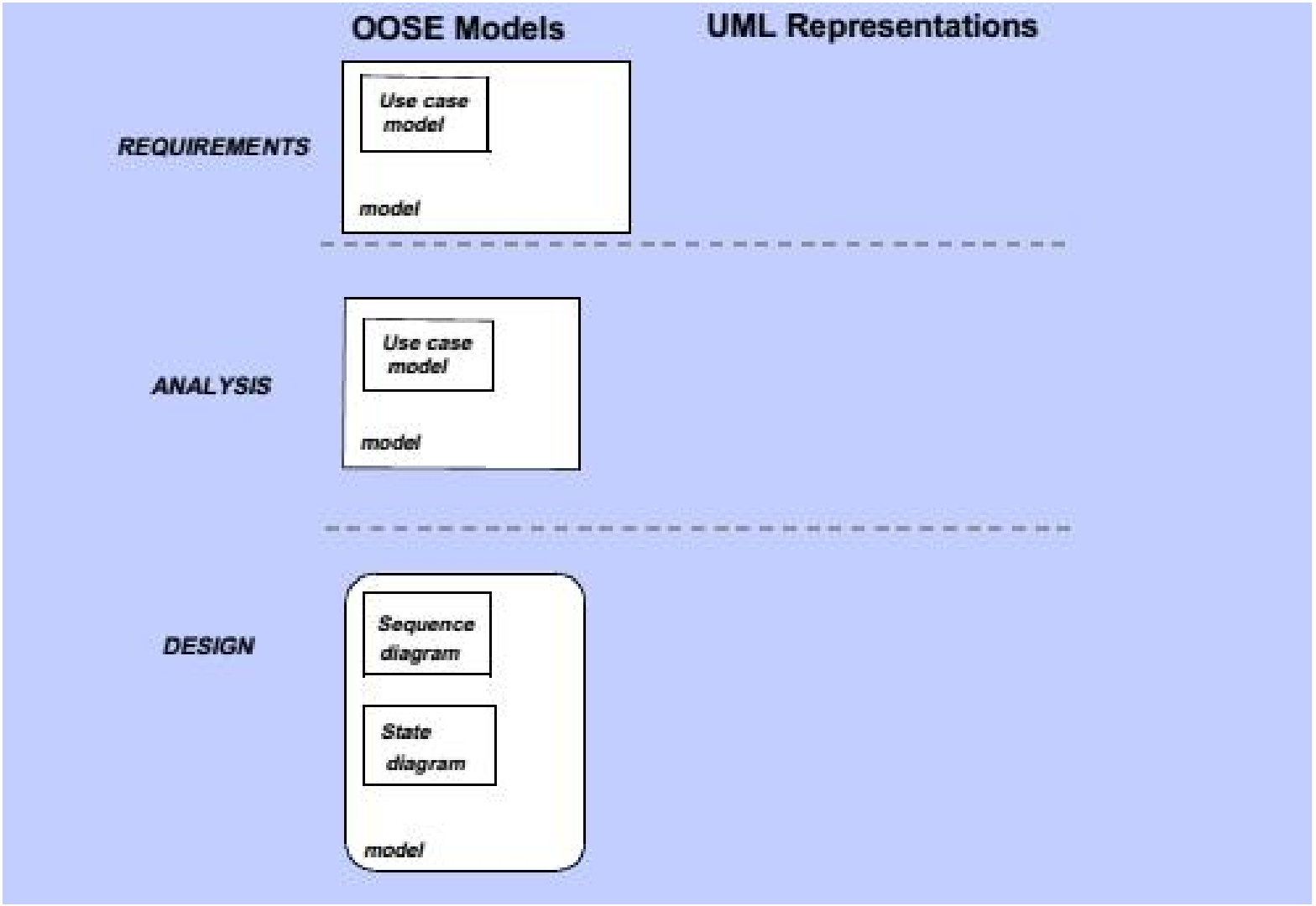
Building Models



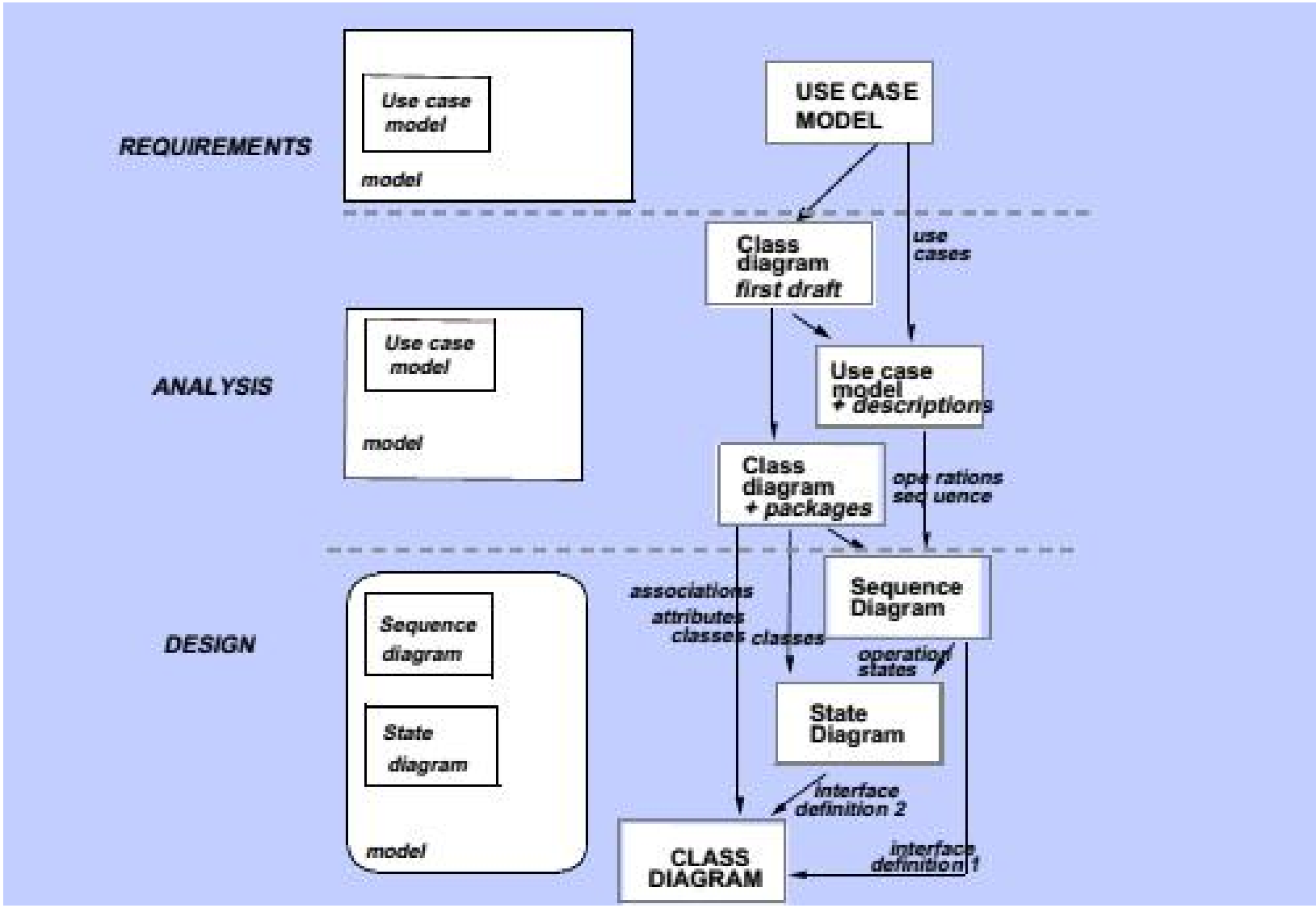
Building Models



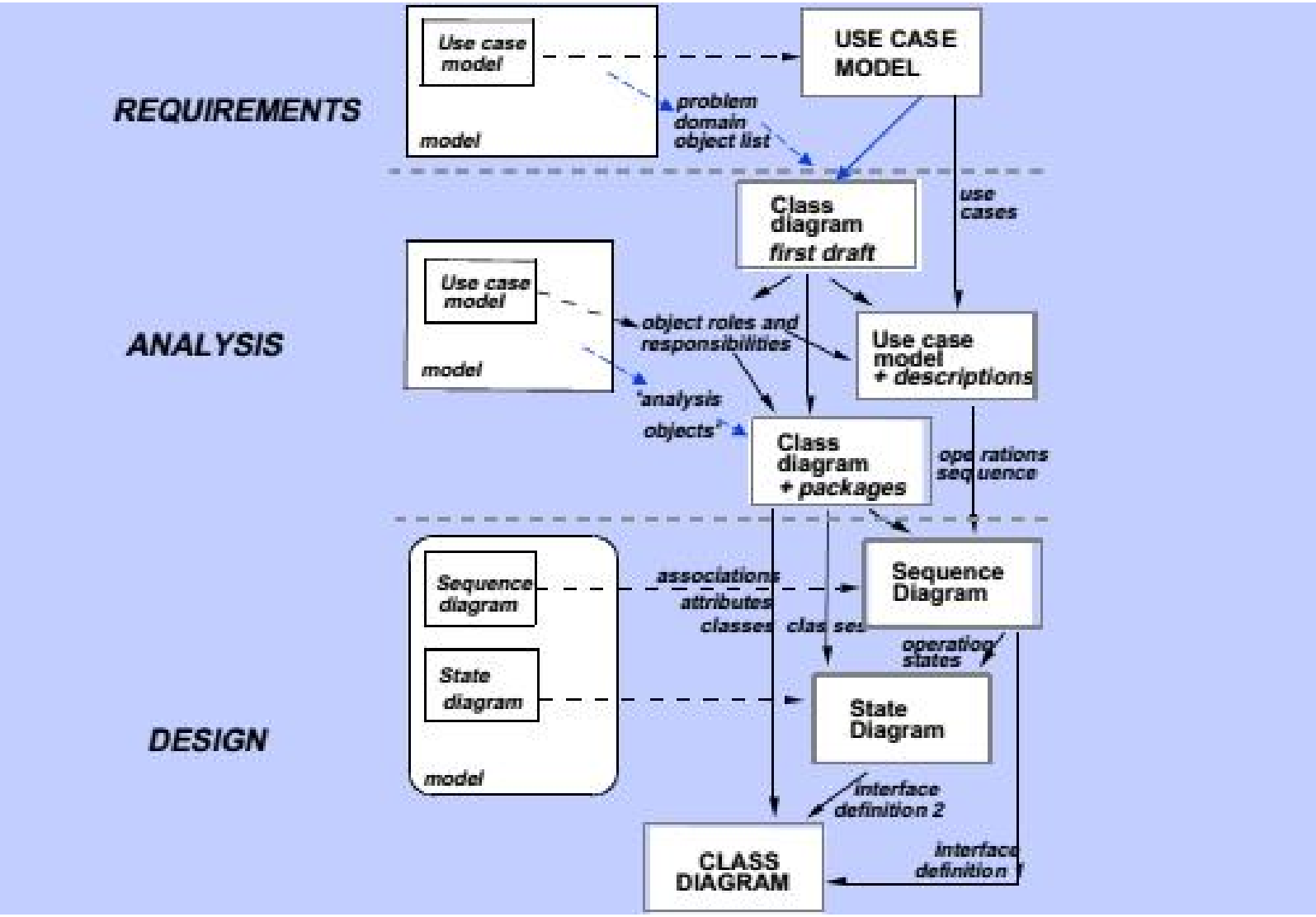
Building Models



Building Models



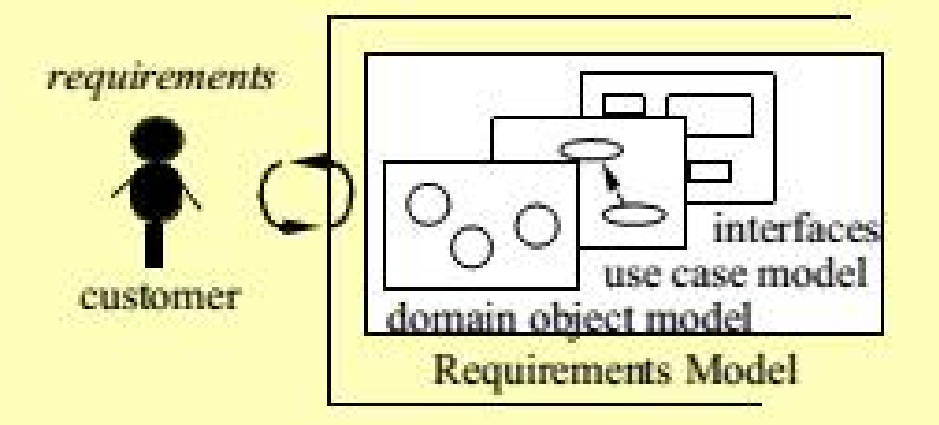
Building Models



مرحلة التحليل

- الأهداف الأولية
  - تحديد ماذا يجب أن يفعل النظام
  - دمج النظام البرمجي في بيئته

- قضيتين أساسيتين ينبغي الانتباه لهما
  - بناء الأجزاء البرمجية الصحيحة
  - بناء الأجزاء البرمجية بشكل صحيح (للاستخدام الحالي و المستقبلي)



- المخرجات
  - Requirement Model
  - Analysis Model

توليد Requirement Model

- إيجاد حالات الاستخدام المحتملة
- التمييز بين حالات الاستخدام الممكنة
- توليد Use case descriptions

- تحديد الارتباطات (Associations) بين حالات الاستخدام
- إعادة مراجعة و معالجة حالات الاستخدام و موديل حالات الاستخدام لإنتاج نسخة أكثر دقة

- توصيف و فحص منافذ الربط مع المستخدم
- توصيف منافذ الربط ضمن النظام

دخل و خرج الـ Requirement Model

- **دخل**
  - ملف توصيف المتطلبات (System Requirements Specification)
  - توثيق الأنظمة الموجودة، التوثيق العملي، الخ و الذي يجب اتباعه.
  - البيانات التي سيتم تبادلها بين المستخدم و النظام

دخل و خرج الـ Requirement Model

- **خرج**
  - موديل حالات الاستخدام
  - توصيف دقيق لحالات الاستخدام (توصيف نصي)
  - توصيف دقيق لمنافذ الربط (توصيف نصي، Prototype)
  - توصيف دقيق لمنافذ الربط ضمن النظام (Protocols)
  - قائمة بالأغراض الواجب إنشاؤها (أسماؤها، خصائصها، الخ)

دخل و خرج الـ Requirement Model

- **ملاحظات (Notations)**

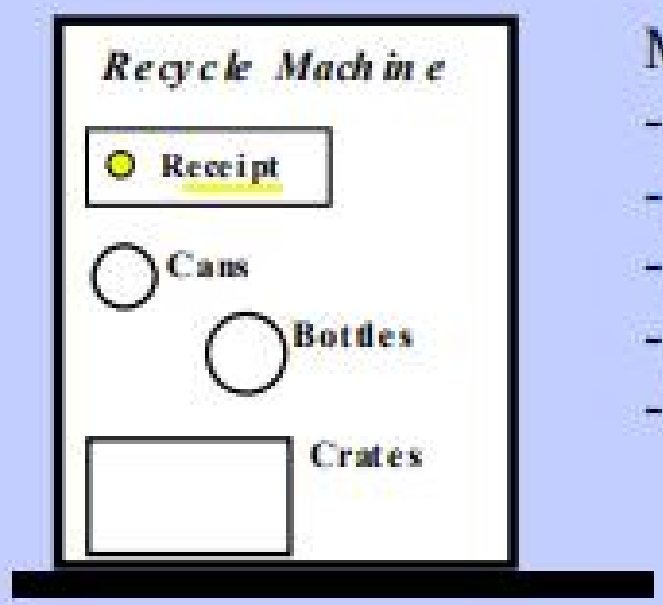
– مخطط حالات الاستخدام (System box, ellipses, names, actor icons, etc.):

– الفاعلين (actors/case links): <users>, <extends>, الخ

- الارتباطات (<users>, <extends>)

Requirement Model - Example

Multi-purpose recycling machine



Machine must:
- receive & check items for customers,
- print out receipt for items received,
- print total received items for operator,
- change system information,
- signal alarm when problems arise.

توليد Use Case Model

- يتم هنا اتباع نفس الخطوات التي تتبع عند تصميم أي منتج برمجي
- يجب تحديد أنواع حالات الاستخدام و توسيعاتها و الارتباطات فيما بينها
- لمزيد من المعلومات: راجع أساسيات هندسة البرمجيات
- يجب أيضاً توصيف و تفصيل واجهات الربط باستخدام الطرق المعروفة

تحديد الأغراض (Problem Domain Objects)

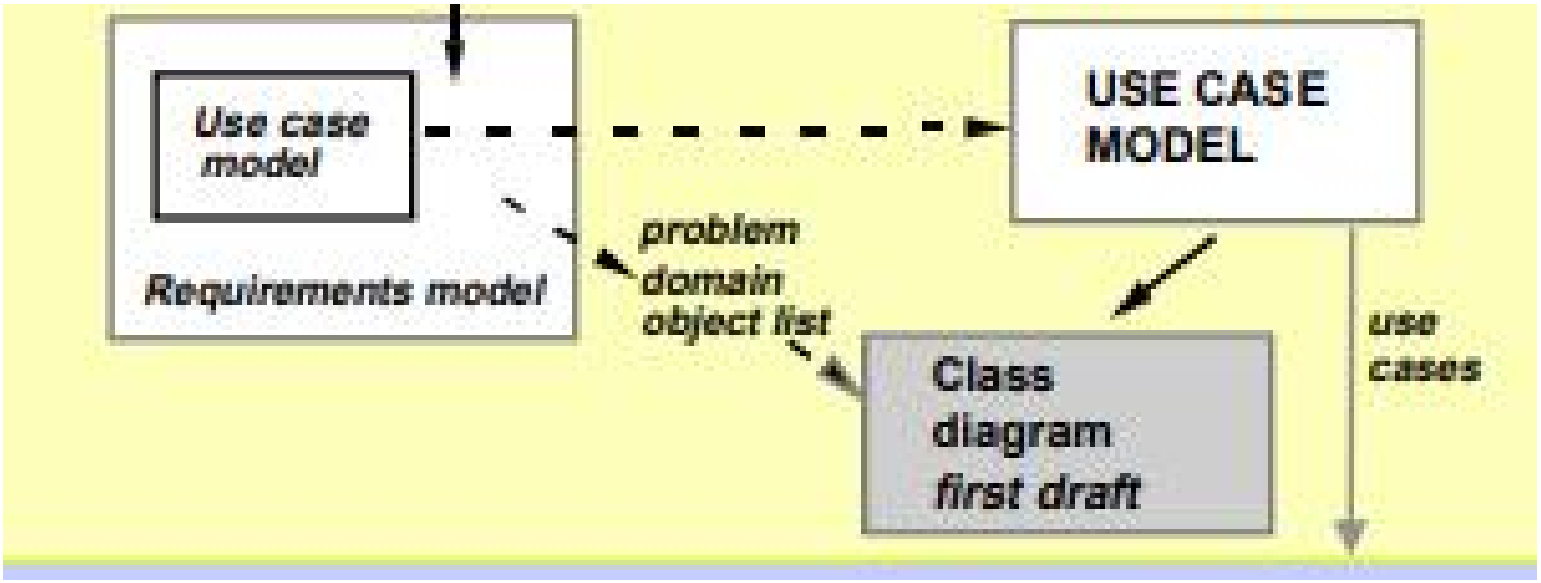
- تحديد الأغراض أثناء التوصيف

- تحديد كيف سيتم استخدامها ضمن التطبيق و ماذا سيتم تخصيصي و اشتقاق منها
- مثال

تحديد الأغراض (Problem Domain Objects)

OBJECT	ATTRIBUTES
name	*characteristic / information : type*
Depoelt item	name: string, total: integer, value: ECU
Can	width: cm, height: cm
Bottle	width: cm, height: cm, bottom: cm
Crate	width: cm, height: cm, lenght: cm
Receipt	total cans: int, total bottles: int, ...
Customer panel	receipt button: button
Operator panel	bottle data: cm, ...

تحديد الأغراض (Problem Domain Objects)



Building on the Experience of Others

- Software engineers should avoid re-developing software already developed
- Types of reuse
  - Reuse of expertise
  - Reuse of standard designs and algorithms
  - Reuse of libraries of classes or procedures
  - Reuse of powerful commands built into languages and operating systems
  - Reuse of frameworks
  - Reuse of complete applications

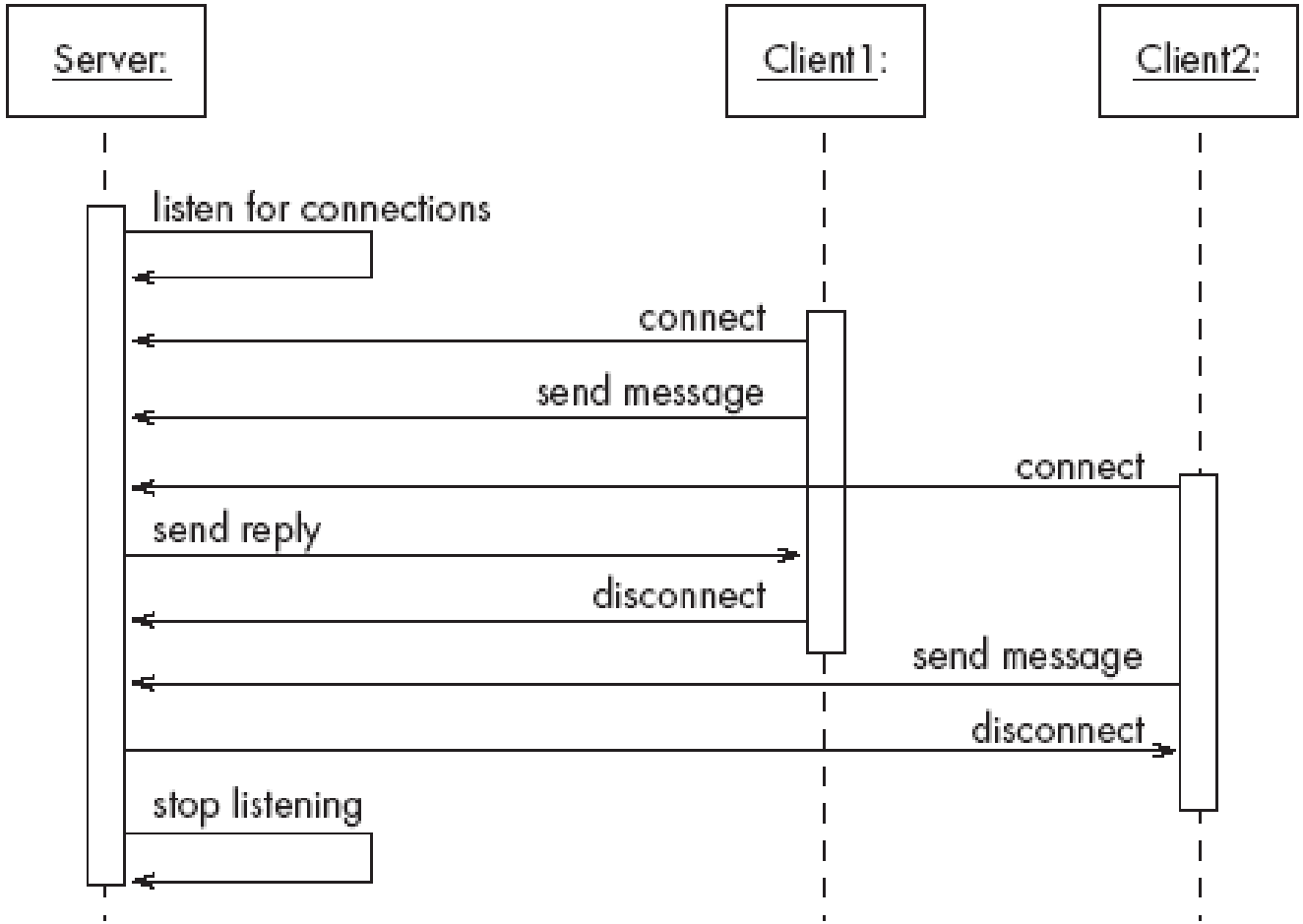
Frameworks: Reusable Subsystems

- A framework is reusable software that implements a generic solution to a generalized problem
  - It provides common facilities applicable to different application programs.
- **Principle**
  - Applications that do different, but related, things tend to have quite similar designs
- Examples
  - A framework for payroll management
  - A framework for frequent buyer clubs
  - A framework for university registration
  - A framework for e-commerce web sites

Client-Server Architecture

- A distributed system is a system in which
  - computations are performed by separate programs
  - normally running on separate pieces of hardware, that co-operate to perform the task of the system
- Server
  - A program that provides a service for other programs that connect to it using a communication channel
- Client
  - A program that accesses a server (or several servers) to obtain services
  - A server may be accessed by many clients simultaneously

Client-Server Architecture



Client-Server Architecture

- The work can be *distributed* among different machines
- The clients can access the server's functionality from a *distance*
- The client and server can be *designed separately*
  - *Competing clients can be written* to communicate with the same server, and vice-versa
- They can both be *simpler*
- All the *data can be kept centrally* at the server
- Conversely, *data can be distributed* among many different geographically-distributed clients or servers
- The server can be accessed *simultaneously* by many clients

Alternatives to the Client-Server Architecture

- Have a *single program* on one computer that does everything
- Have *no communication*
  - Each computer performs the work separately
- Have some mechanism other than client-server communication for exchanging information
  - E.g. one program writes to a database; the other reads from the database

Communications Protocols

- The messages the client sends to the server form a language
  - The server has to be programmed to understand that language –
- The messages the server sends to the client also form a language
  - The client has to be programmed to understand that language –
- When a client and server are communicating, they are in effect having a conversation using these two languages
- The two languages and the rules of the conversation, taken together, are called the protocol

Developing Client-Server Applications

1. Design the primary work to be performed by both client and server
2. Design how the work will be distributed
3. Design the details of the set of messages that will be sent
4. Design the mechanism for
   - Initializing
   - Handling connections
   - Sending and receiving messages
   - Terminating