# مدخل إلى هندسة البرمجيات

د. م. علي ذياب

# 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
- تعريف البرمجيات وأهميتها
- الطريقة الطبقية (**Layered Technology**)
- طريقة هندسة البرمجيات (**Software Engineering Process**)
- تكلفة هندسة البرمجيات (**Software Engineering Costs**)

# مدخل إلى المحاضرة

# المنتجات البرمجية (Software Products)

- منتجات عامة (Generic products)
  - نظم قائمة بذاتها (Stand-alone systems)
    - تُباع لأي مستخدم يريد استخدامها
  - أمثلة
    - البرمجيات الحاسوبية كبرمجيات تحرير النصوص على سبي المثال, البرامج الرسومية, CAD software, الخ.

- نظم مخصصة (Customized products)
  - نظم برمجية مخصصة لتلبي احتياجات مستخدم معين
  - أمثلة
    - الأنظمة المضمنة, أنظمة التحكم الجوي, أنظمة المراقبة, الخ

## 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

# لماذا تُعتبر هندسة البرمجيات مهمة جداً؟

- معنى هندسة البرمجيات


حيث: Process framework = process model + umbrella activities

*Process model: مجموعة الأنشطة التي تهتم بتطوير التطبيق, حيث تكون مقسمة على مراحل(milestones) وكل مرحلة تحوي مجموعة من المهام (tasks), ولها نوعين: plane driven and agile

*umbrella activities: مجموعة إنشطة لإدارة مراحل تنفيذ المشروع ومراقبتها والتحقق من الجودة, وتتبّع المخاطر

# 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

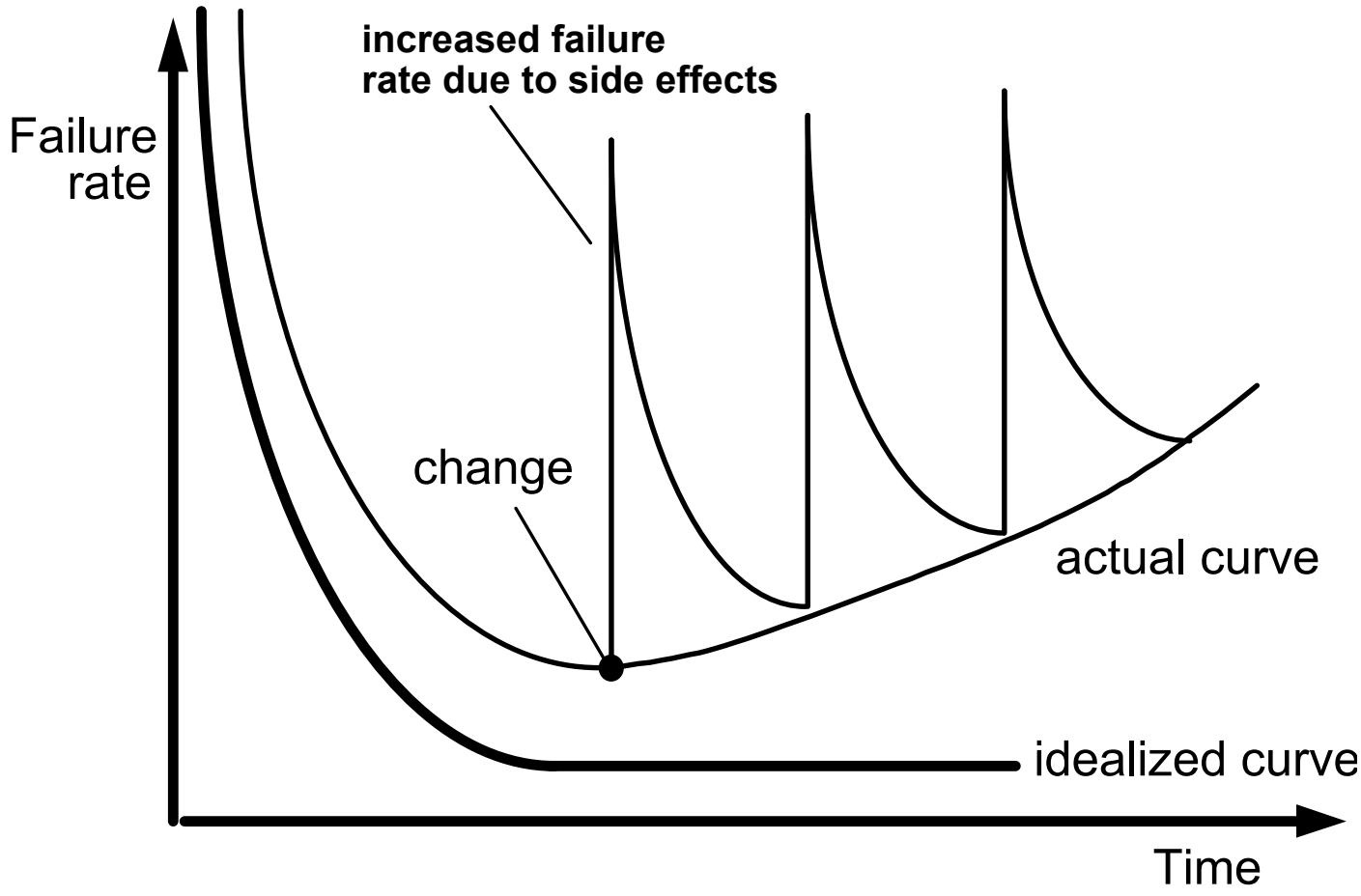
# خصائص البرمجيات (Features of Software?)

- 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

# (Features of Software?) خصائص البرمجيات



# أنواع التطبيقات البرمجية (Software Applications)

- برمجيات النظم (System software)
  - المترجمات, محررات نصوص خاصة, برمجيات الإدارة, الخ
- التطبيقات البرمجية (Application software)
  - برمجيات قائمة بحد ذاتها مخصصة لمجالات معينة
- التطبيقات الهندسية/العلمية (Engineering/scientific software)
  - عبارة عن برمجيات مخصصة لدراسة ظواهر علمية معينة, دراسة الجهد, محاكاة عمل الشبكات اللاسلكية, الخ
- تطبيقات النظم المضمنة (Embedded software)
  - تطبيقات تتوضع في النظم المضمنة, كتطبيقات الأجهزة المنزلية مثلاً

# أنواع التطبيقات البرمجية (Software Applications)

- منتجات برمجية ذات إصدارات متتابعة (Product-line software)
  - تركز على جانب معين ضمن مجال ما, وتصدر بإصدارات متتابعة أكثر تطوراً من سابقتها لتلبي احتياجات الزبائن
    - محررات النصوص (Word processing), البرامج الرسومية, الخ

- تطبيقات الويب (Web applications)
  - عبارة عن تطبيقات معتمدة على الإنترنت, تعتمد على بناء نظم موزعة لمعالجة البيانات معتمدة في بنائها الأساسي على الإنترنت

# أنواع جديدة للبرمجيات (Software - New Categories)

- البرمجيات في العالم المفتوح (Open world computing)
  - عبارة عن برمجيات تعتمد على الشبكات اللاسلكية وتصنف على أنها
    - Pervasive
    - Ubiquitous
    - Distributed computing
  - يُمكّن هذا الصنف من جعل الأجهزة النقالة قادرة على التواصل مع أجهزة أخرى عبر الشبكة

- الإنترنت كوسيلة معالجة (Net-sourcing)
  - يتم استخدام الإنترنت كوحدة معالجة للبرمجيات, ترجمة برامج باستخدام النت على سبيل المثال

- البرمجيات مفتوحة المصدر (Open source)
  - هي برمجيات نكون الشيفرة البرمجية لها متاحة ويمكن أن يتم تعديلها

# تعريف البرمجيات وأهميتها

# تعريف هندسة البرمجيات

- تعريف 1
  - تأسيس واستخدام المبادئ الهندسية للحصول على برمجية اقتصادية توصف على أنها موثوقة وتعمل بفعالية ضمن ظروف العمل الفعلية

- تعريف 1 (تعريف IEEE)
  - تطبيق طريقة منظمة ذات خطوات محددة و قابلة للتتبع والقياس من أجل تطوير و تشغيل و صيانة البرمجيات. بكلام آخر تطبيق الهندسة على البرمجيات

- أهمية هندسة البرمجيات
  - تعتمد المجتمعات والأفراد أكثر فأكثر على البرمجيات, لذا نحن بحاجة لإنتاج برمجيات موثوقة, اقتصادية وسريعة التنفيذ
  - استخدام مبادئ هندسة البرمجيات لبناء برمجية ما يجعلها أرخص عند النظر لفترة حياة البرمجية الطويلة من بنائها دون استخدام مبادئ هندسة البرمجيات

# 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. |

# 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؟

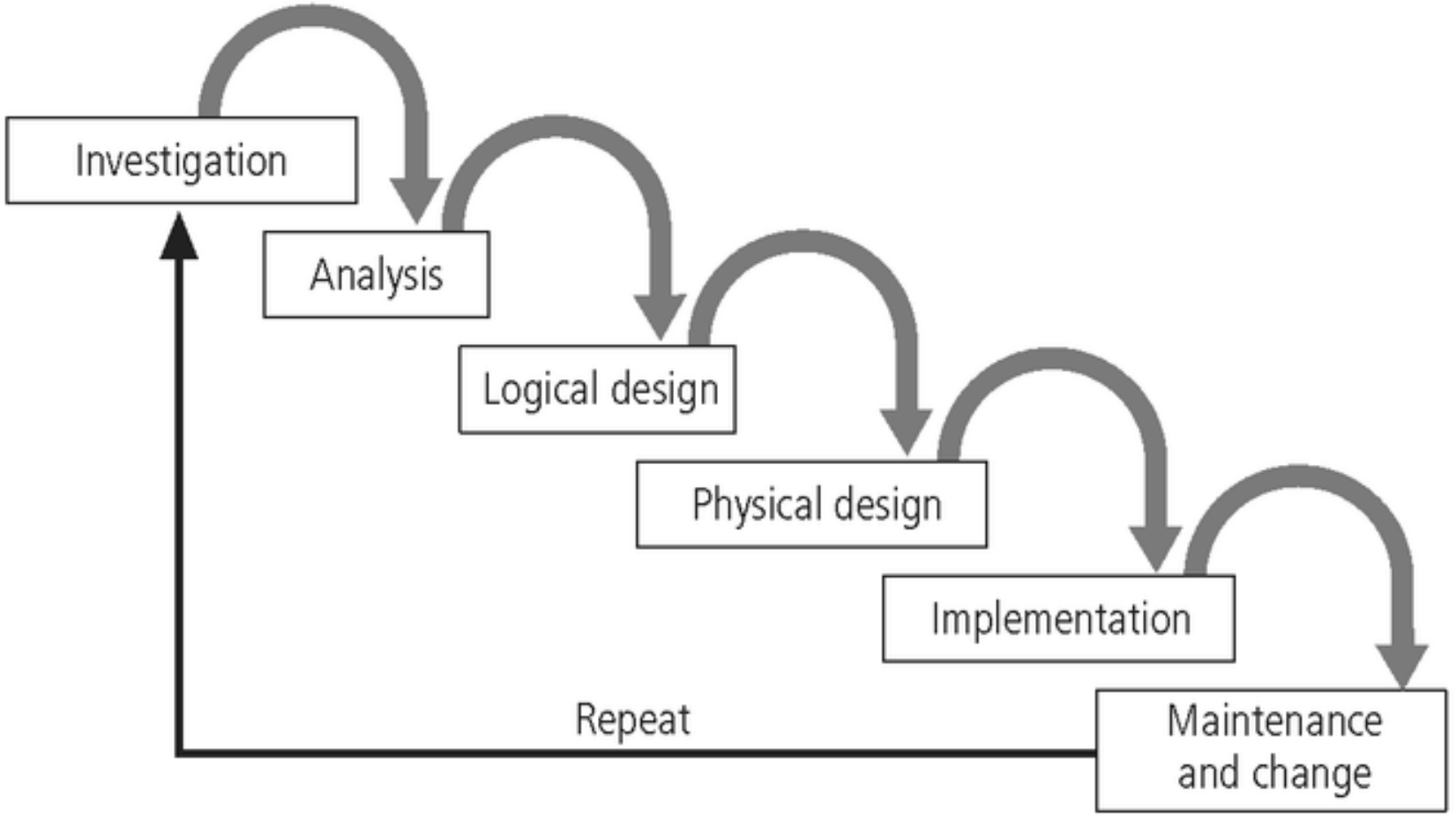
- عبارة عن نظام برمجي مخصص للمساعدة في أتمتة عملية بناء البرمجيات

- 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 الطبقية (Layered Technology)

# Systems Development Life ) دورة حياة تطوير البرمجية (Cycle (SDLC)



# طريقة هندسة البرمجيات ( Software Engineering Process)

# 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)

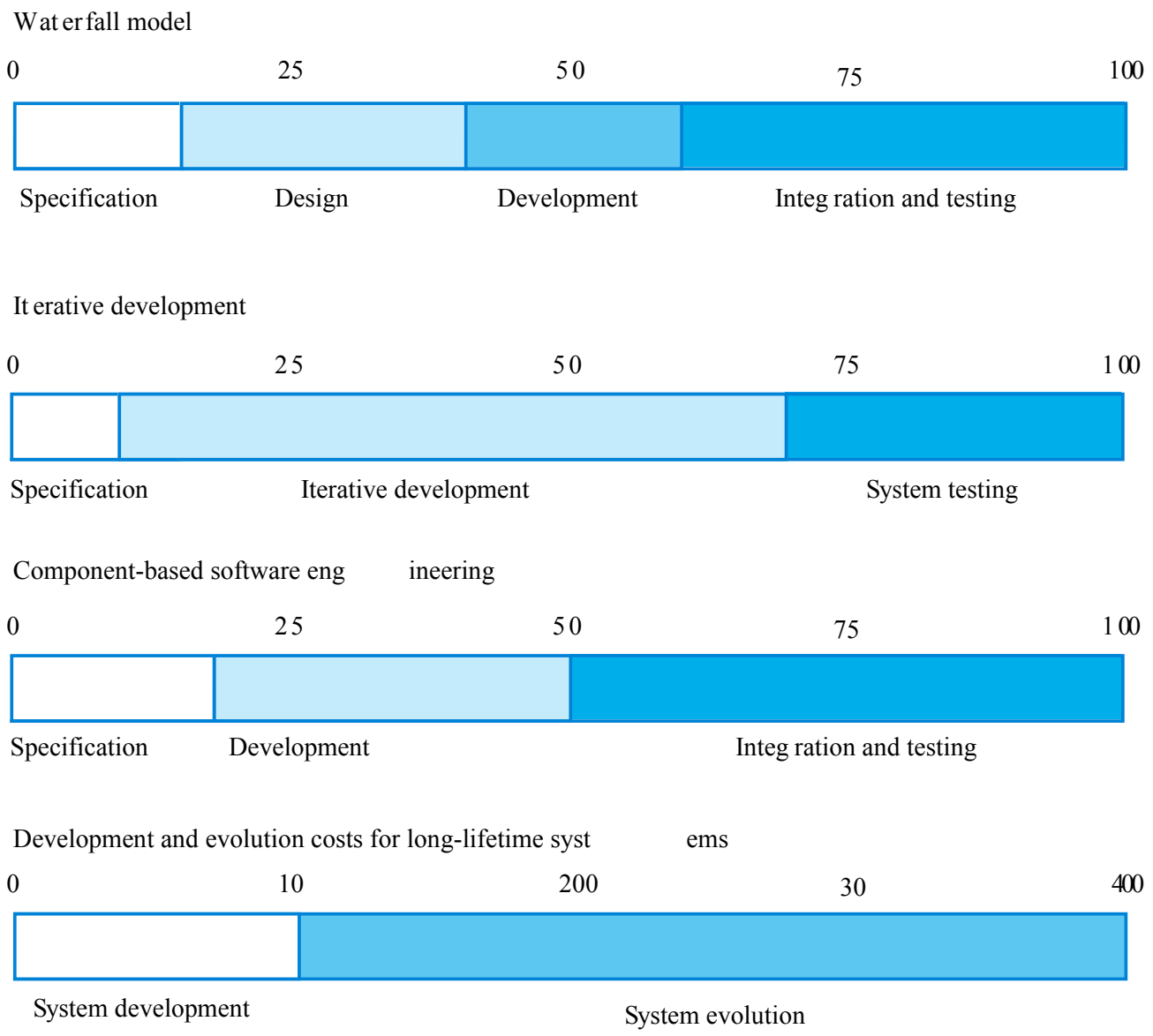
# ماهي كلفة هندسة البرمجيات؟

- 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



# دورة حياة تطوير البرمجيات ( Software
# (Developement Life Cycles (SDLC

## محتويات المحاضرة

- تعريف موديل SDLC
- موديل Bug & Fix
- موديل شلال الماء (Waterfall Model)
- موديل النموذج السريع (Rapid Prototyping Model)
- الموديل التزايدي (Incremental Model)
- الموديل الحلزوني (Spiral Model)

# 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
  - ....

# 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)

# النموذج الإجرائي العام ( Generic Process Model)

# النموذج الإجرائي العام

- يمكن أن يكون Plan-driven أو Agile
- يتألف من الخطوات التالية

1- الإتصالات(communication): التواصل مع أصحاب المصلحة

2- التخطيط(planning): تتضمن خريطة توجه فريق التطوير, وتحوي على الموارد التي يجب توفرها والجداول الزمنية والمخاطر الممكنة الخ..

3- النمذجة(modeling): خلق نماذج تحليلية من أجل فهم المتطلبات, وتطوير التطبيق الذي يلبي هذه المتطلبات

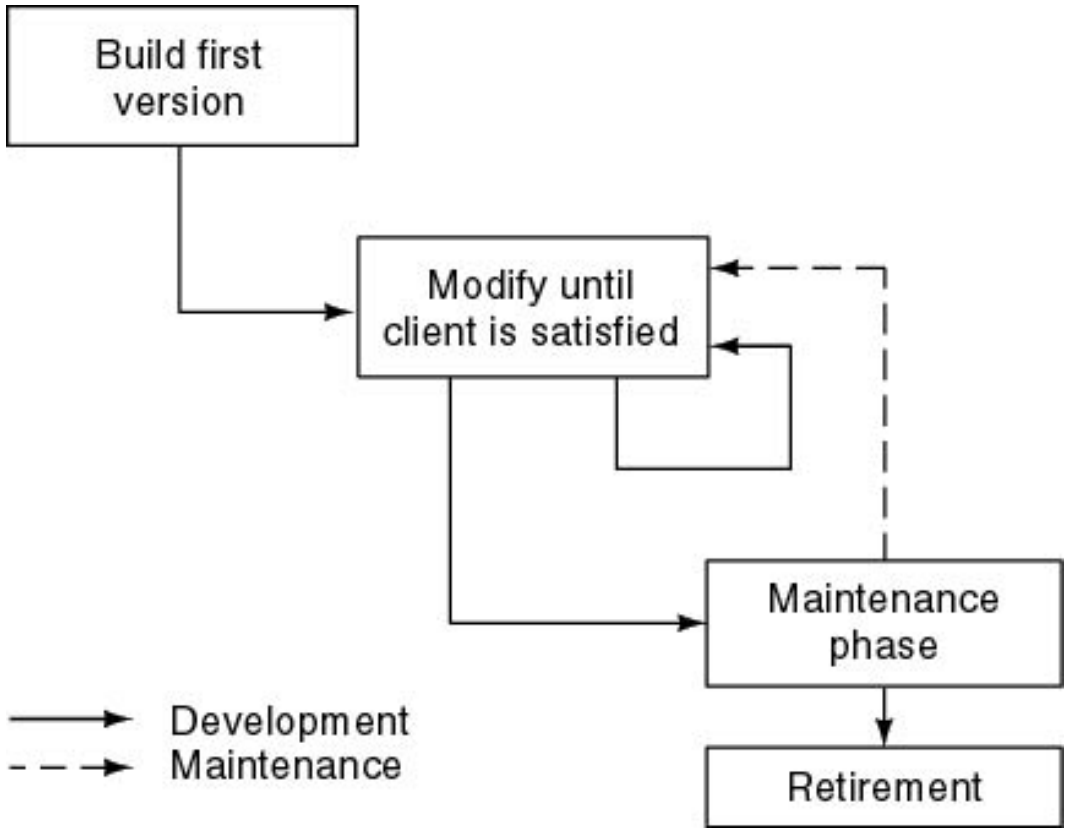
4- البناء(construction): تتضمن توليد الكود وإختباره

5- النشر(deployment): تسليم المنتج البرمجي سواء كامل أو على مراحل, وأخذ تغذية راجعة(feedback) من الزبون لتقييم المنتج البرمجي

# Bug & Fix موديل

# موديل Build & Fix

- يتم تنفيذ المشروع البرمجي بدون وجود توصيف تفصيلي له



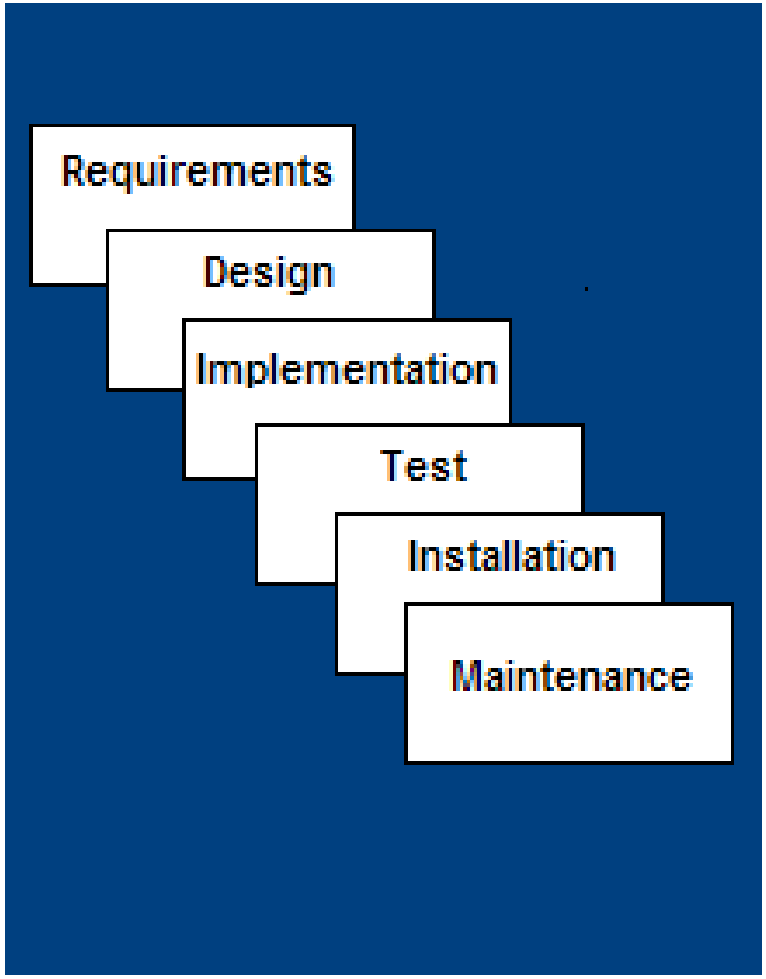
## السلبيات

- المشاكل التي يتم مواجهتها
  - لا يوجد توصيف
  - لا يوجد تصميم
- غير مرضي على الإطلاق
- يحتاج لدورة حياة
- المساوئ
  - يتم إجراء التغييرات في في مراحل متأخرة أثناء تطوير البرمجيات, و هذا يؤدي لتكلفة عالية
  - الصيانة صعبة

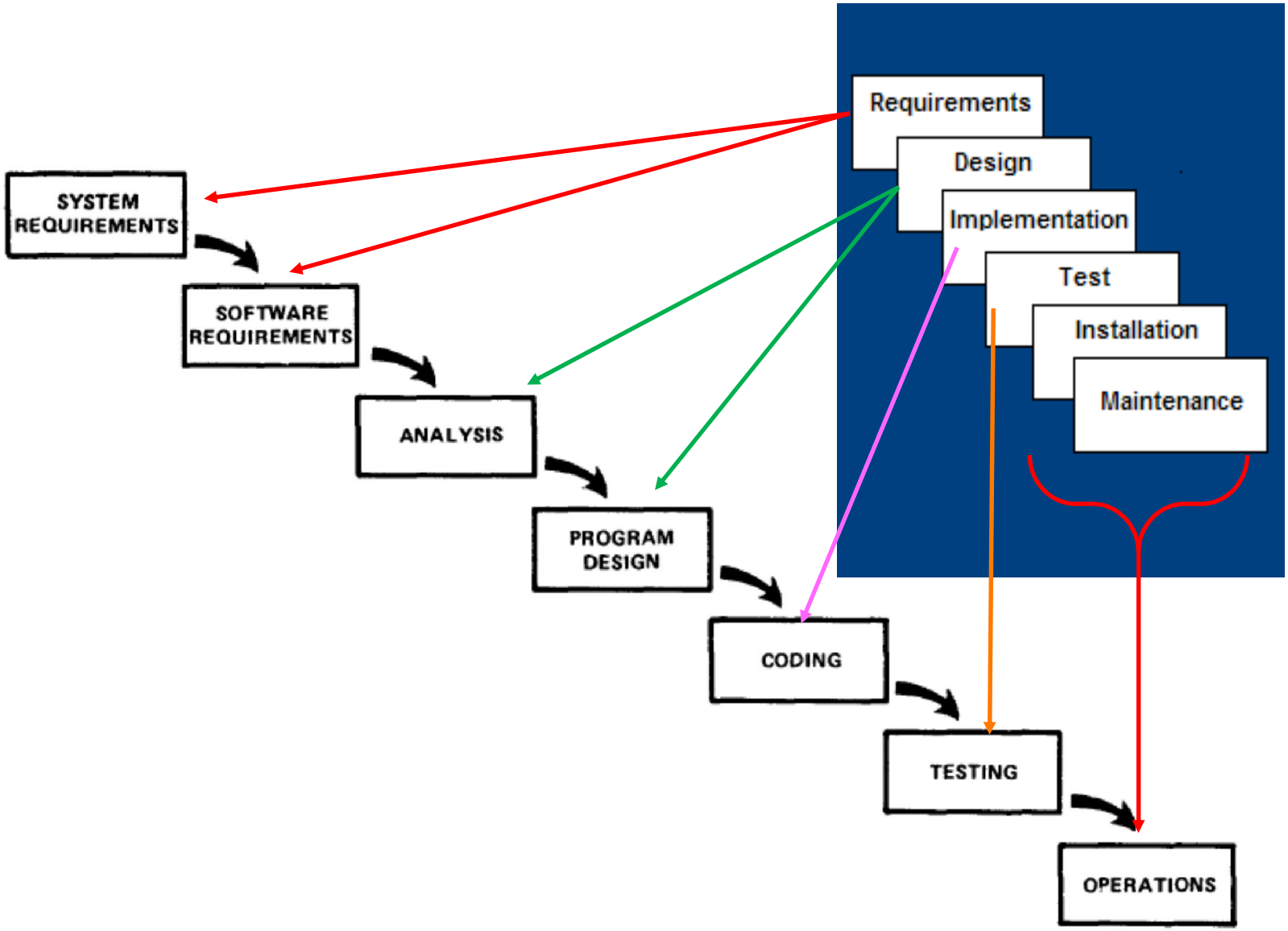
# موديل شلال الماء (Waterfall Model)

# موديل شلال الماء

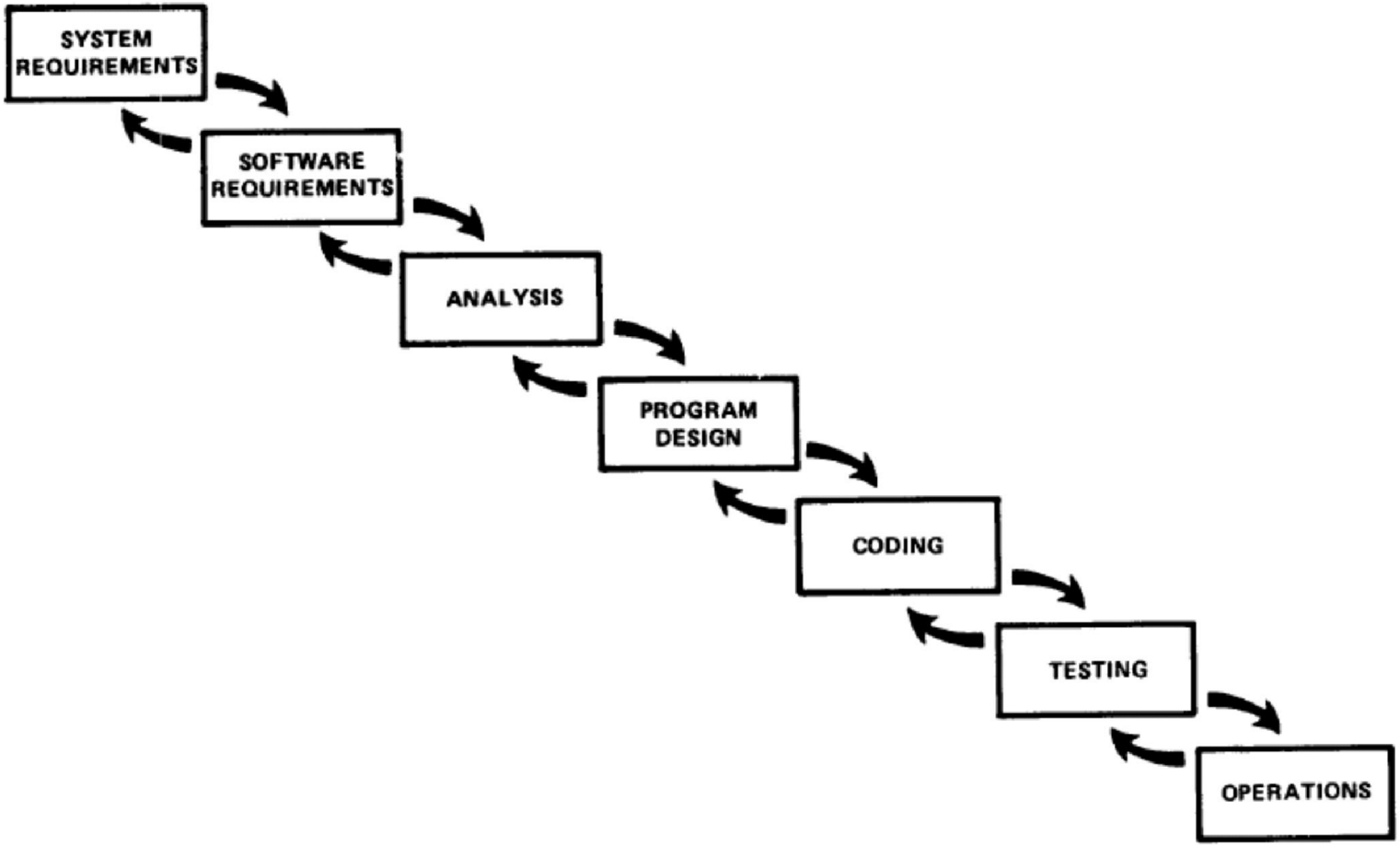
- تحدد المتطلبات (Requirements)
  - المعلومات الضرورية
  - التوابع والسلوك
  - الأداء وواجهات الربط
- التصميم (Design)
  - بنية البيانات
  - بنية البرمجيات
  - تمثيل واجهات الربط
  - تفاصيل الخوارزميات
- التنفيذ الفعلي (Implementation)
  - كتابة الشيفرة البرمجية وبناء قاعدة المعطيات
  - توثيق البرمجية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



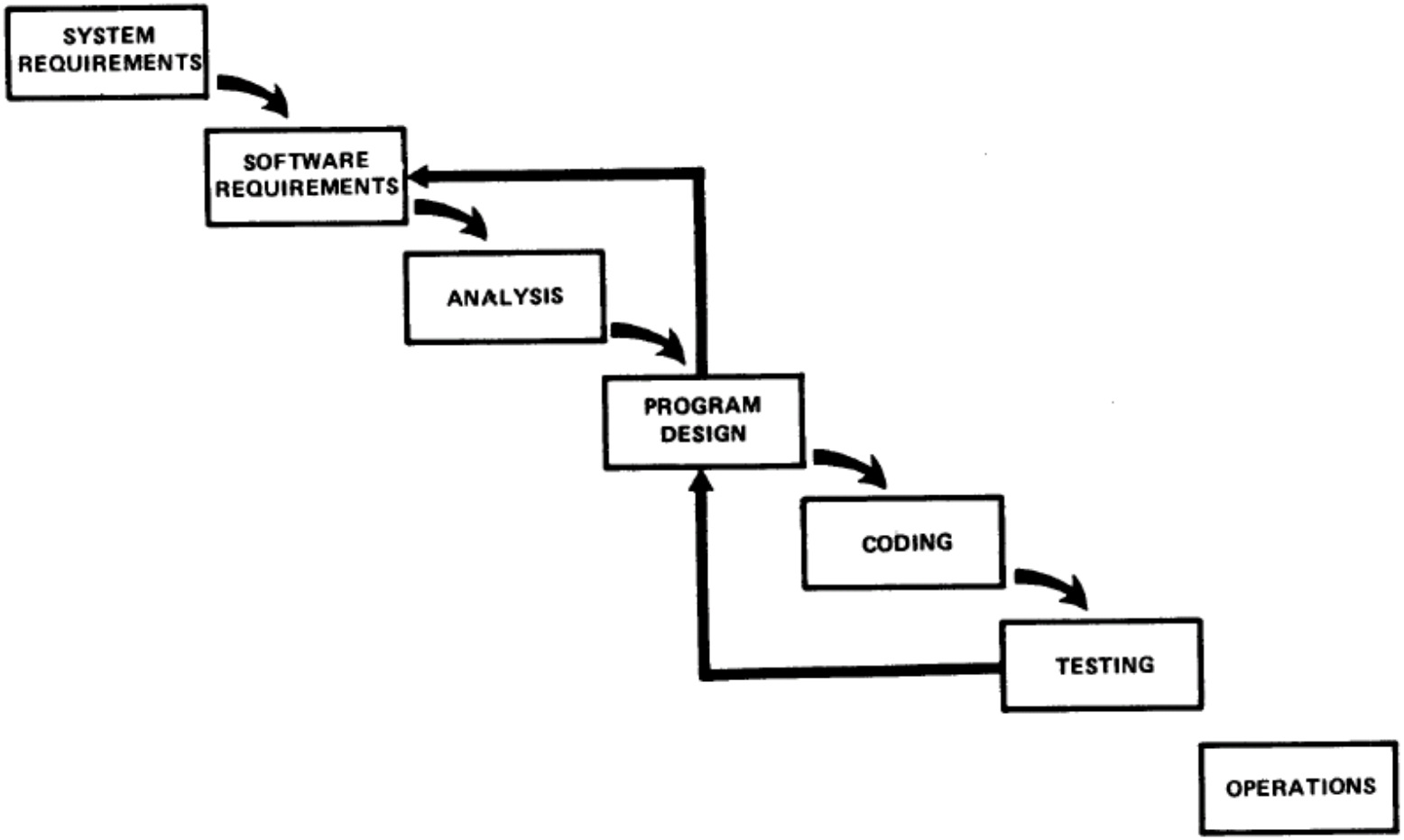
# موديل شلال الماء



# موديل شلال الماء



# موديل شلال الماء

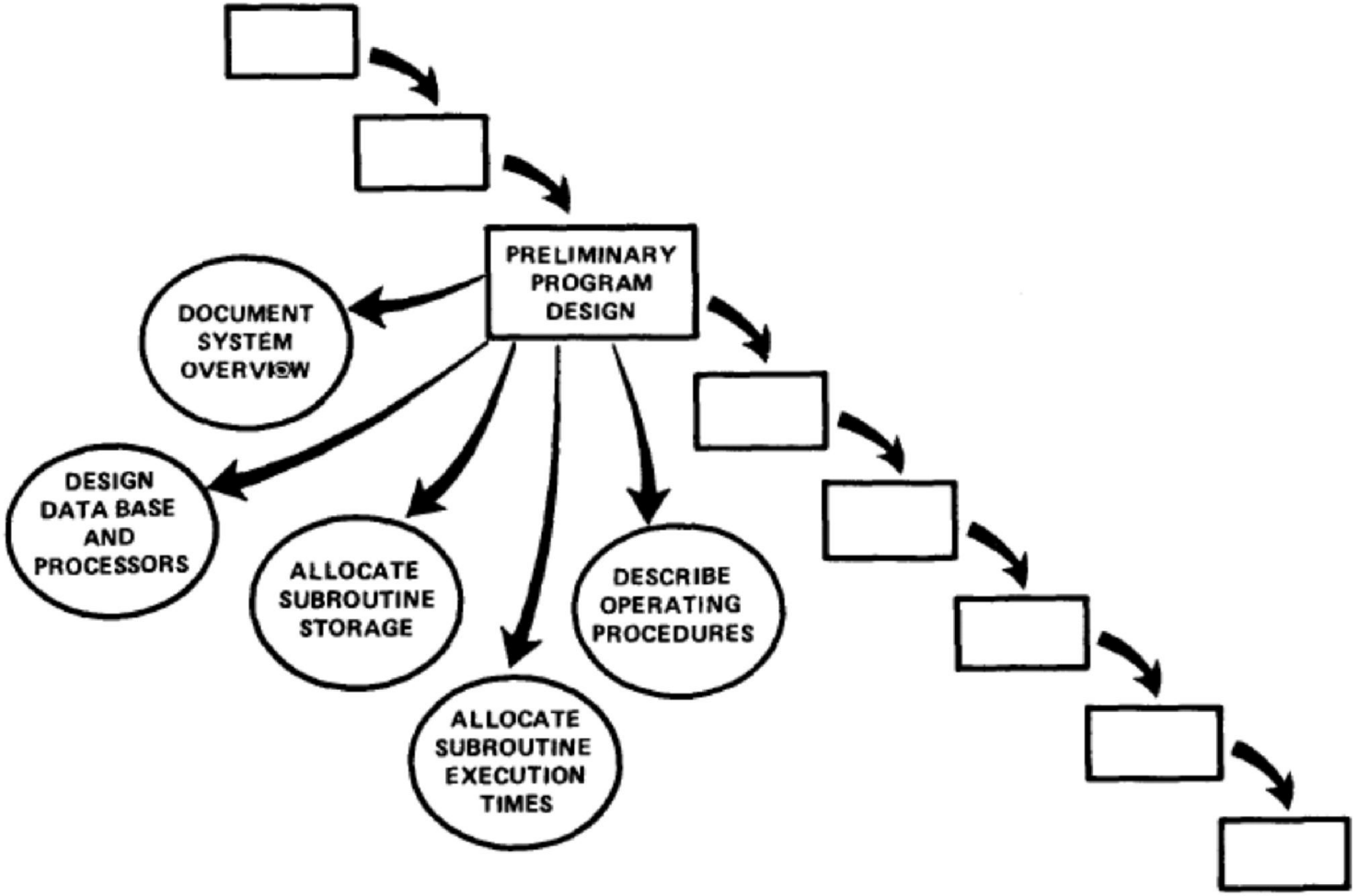


# التغلب على صعوبات التطوير

- يجب إضافة بعض الخصائص والخطوات للموديل الأساسي للتغلب على صعوبات التطوير
  - الخطوة 1: تصميم البرنامج في المراحل الأولى
  - الخطوة 2: توثيق التصميم
  - الخطوة 3: قم بكل خطوة مرتين
  - الخطوة 4: تخطيط و التحكم بـ ومراقبة عملية الفحص
  - الخطوة 5: إدخال المستخدم في عملية التطوير

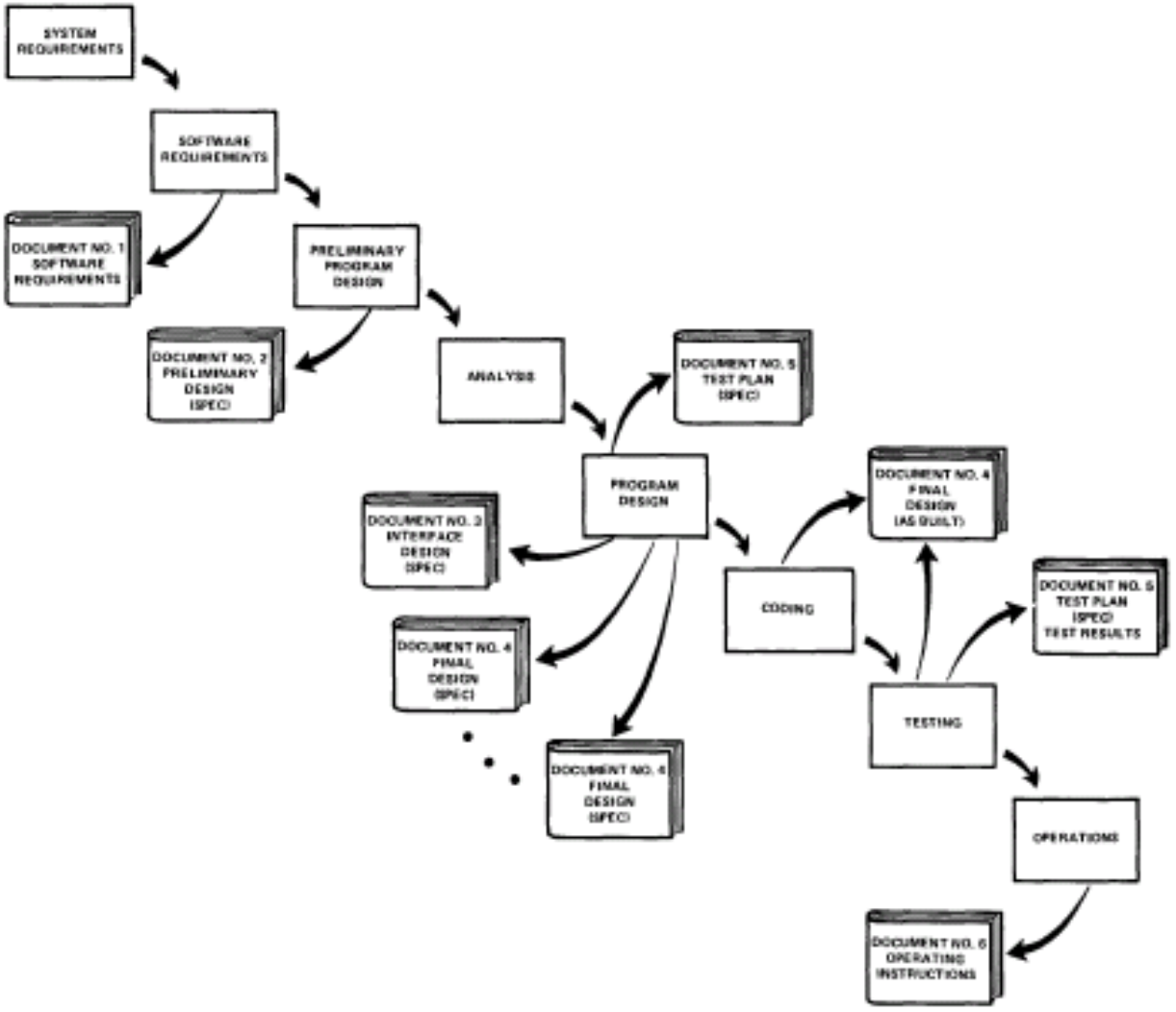
# التغلب على صعوبات التطوير

- الخطوة 1



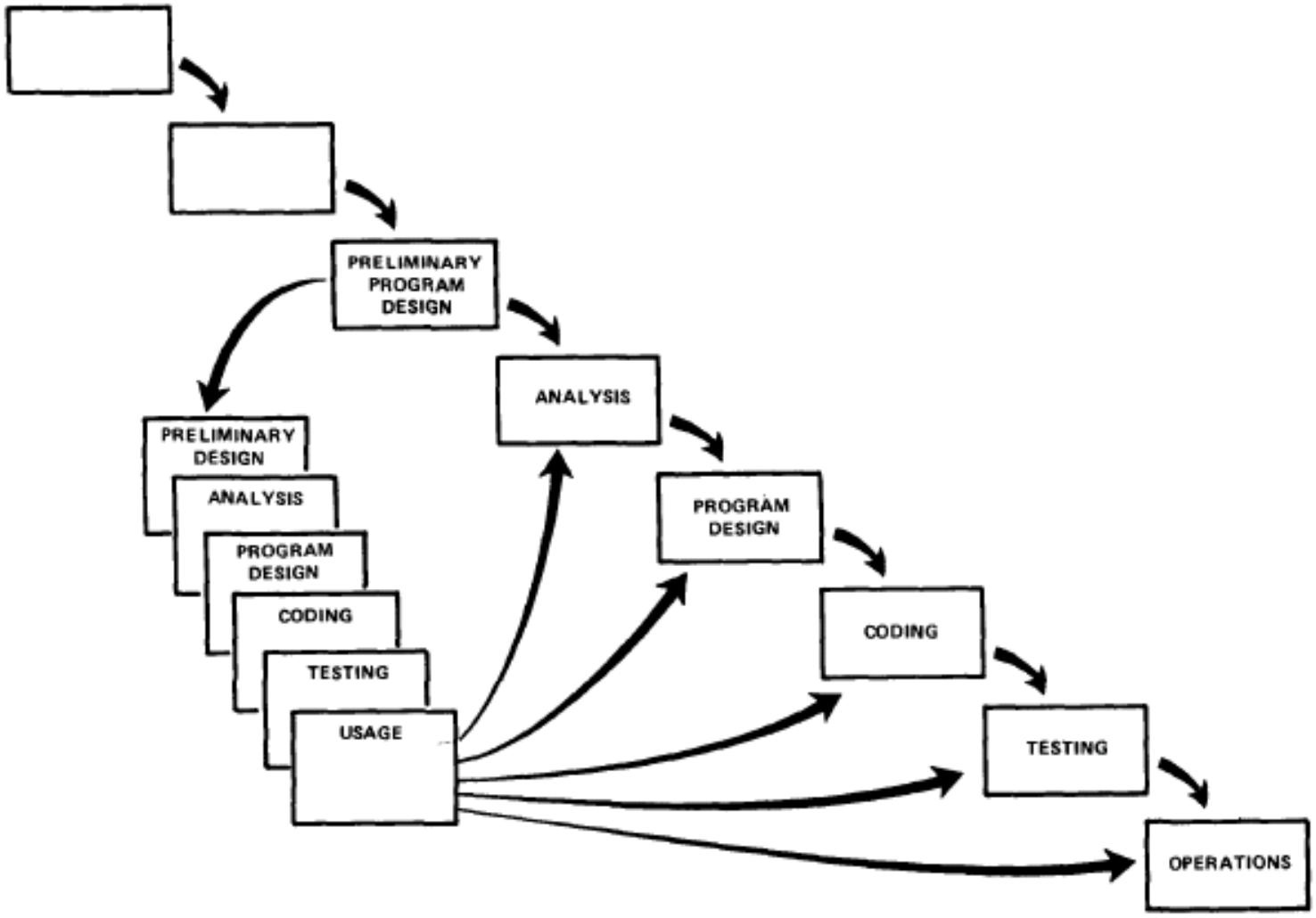
# التغلب على صعوبات التطوير

- الخطوة 2



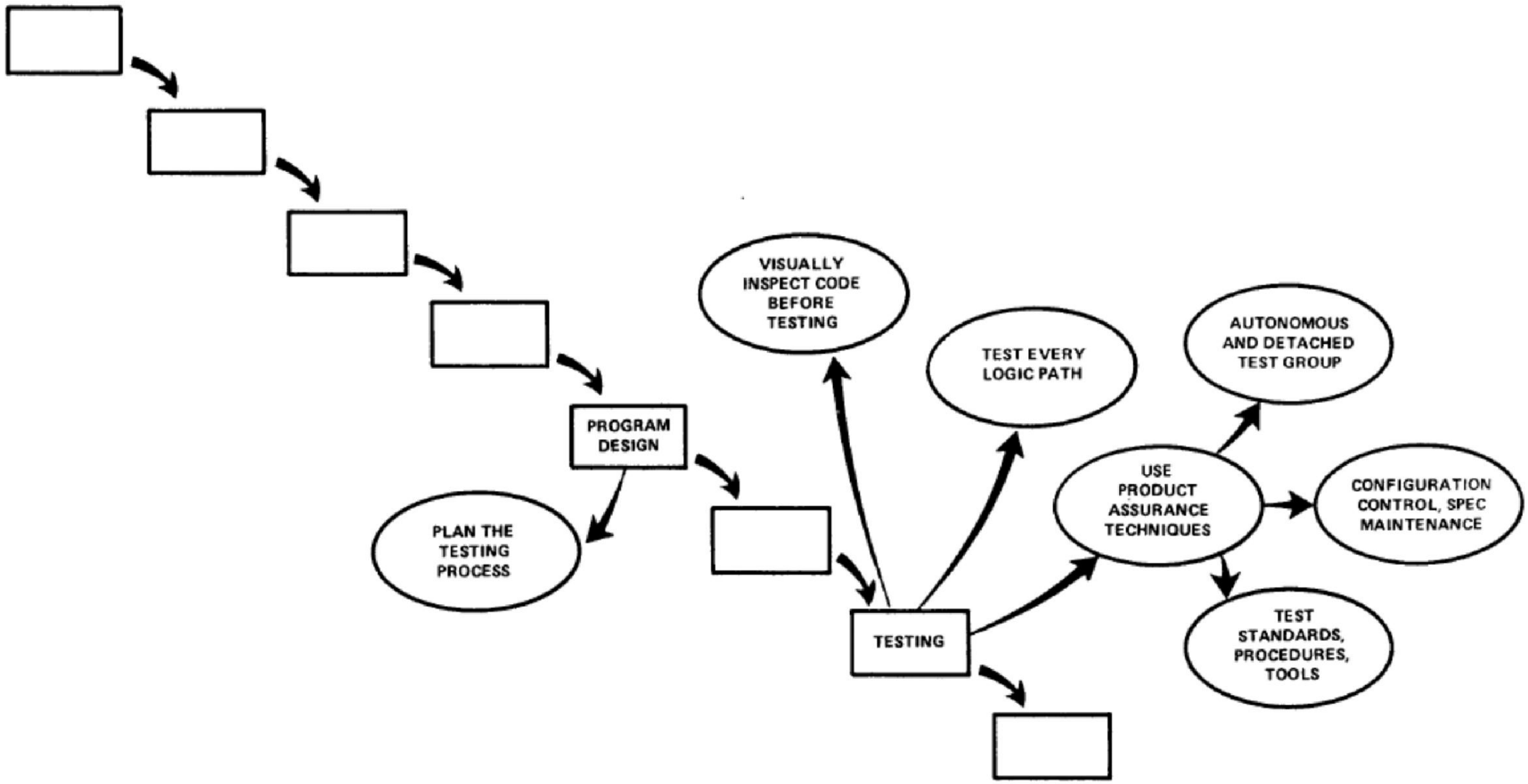
# التغلب على صعوبات التطوير

- الخطوة 3



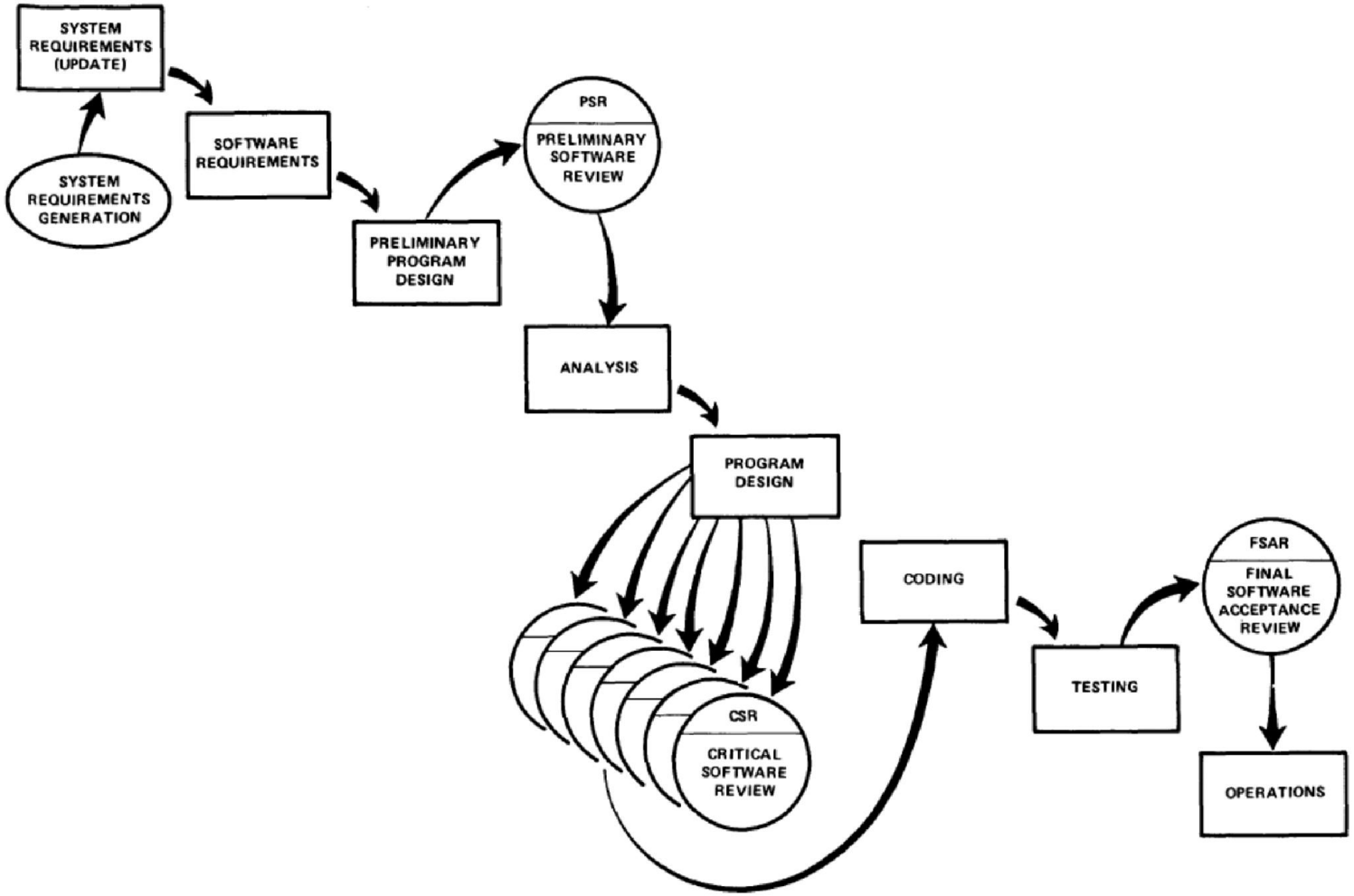
# التغلب على صعوبات التطوير

- الخطوة 4



# التغلب على صعوبات التطوير

- الخطوة 5



## 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

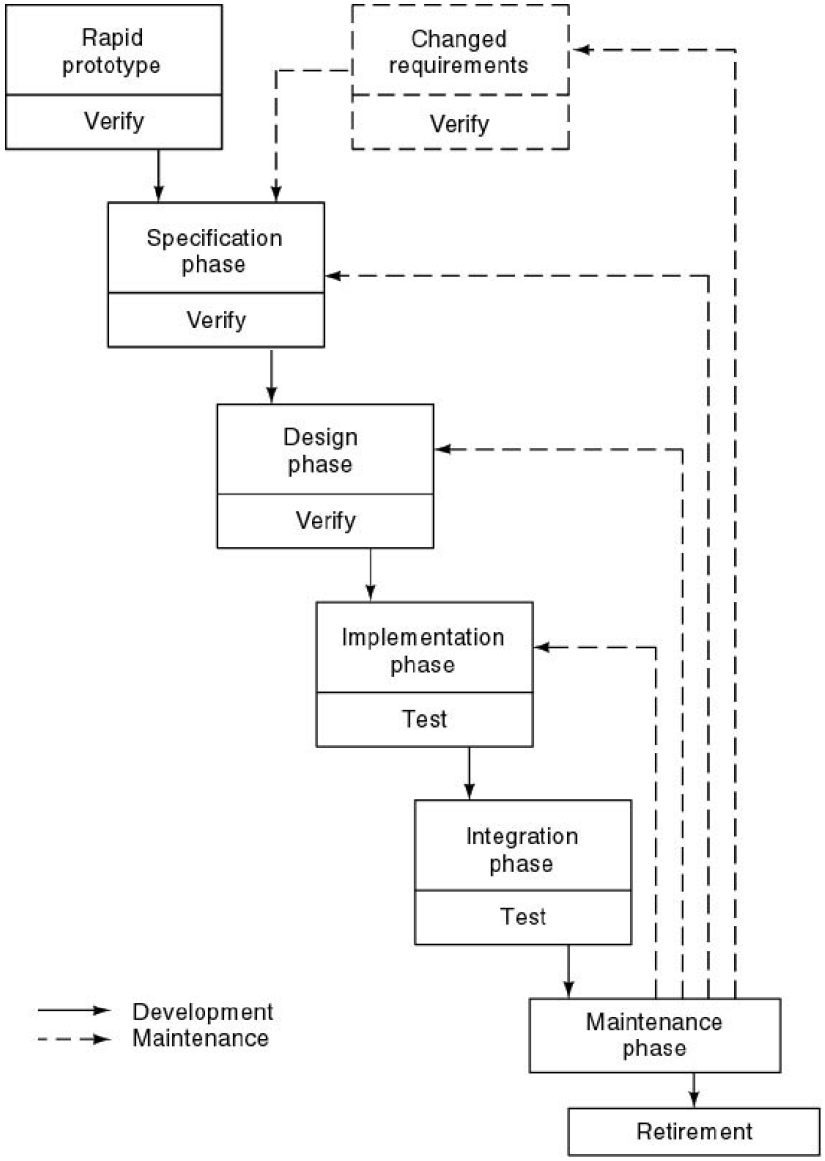
- دمج البرمجية في البيئة أو مع برمجيات أخرى تعتبر مشكلة كبيرة في نهاية تطوير المنتج
- يتواجد فرص قليلة للمستخدم للتعرف عل المنتج البرمجي في مراحله الأولى (قبل وصول المنتج النهائي)

## 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)

# Rapid Prototyping ) موديل النموذج السريع (Model

# 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

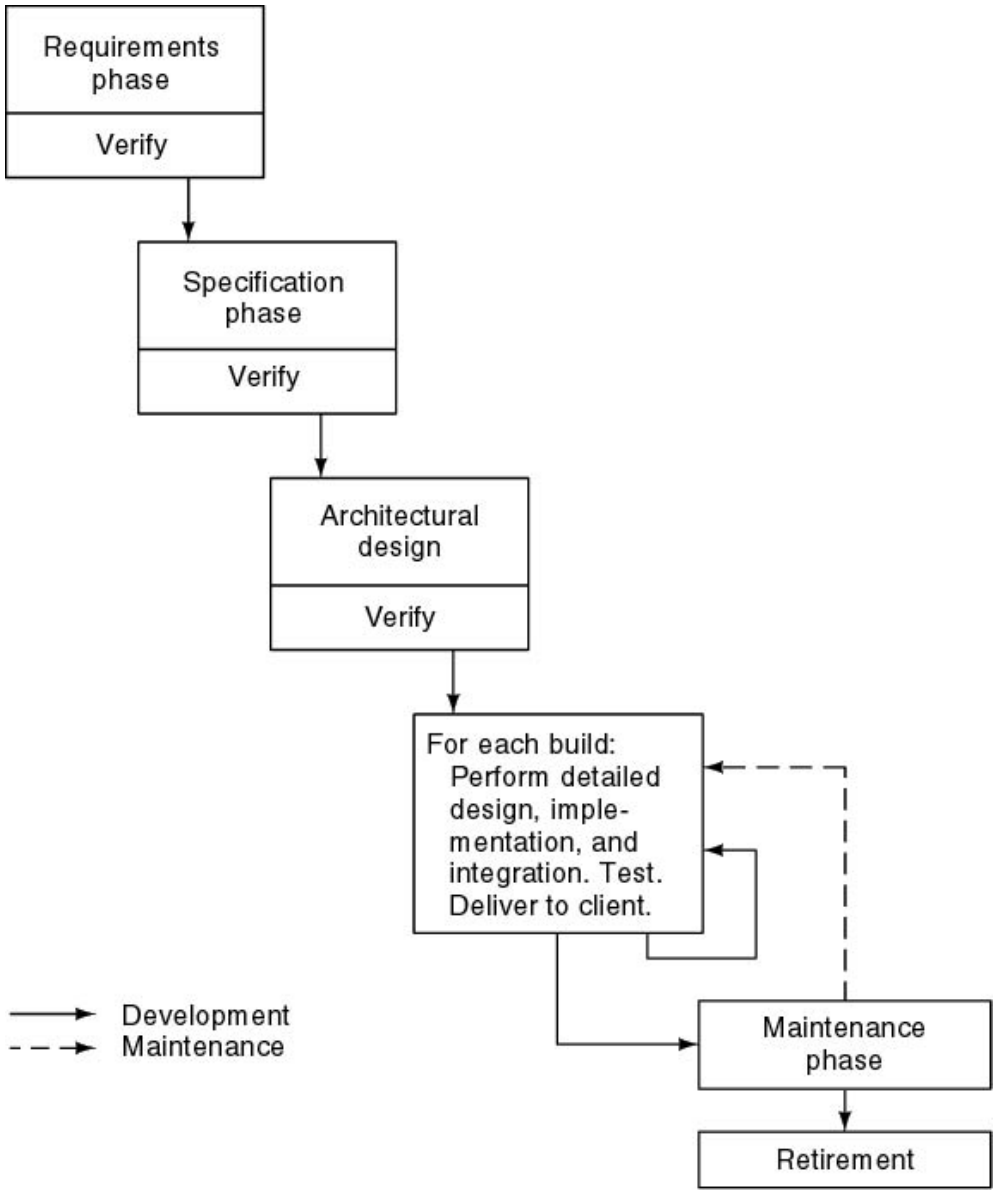
    - يتطلب مبرمجين ذوي مهارة عالية
    - الزبائن يتوقعون أن يتم تلبية رغباتهم والتغييرات التي طلبوها بسرعة كبيرة كسرعة إنجاز الموديل التجريبي

# 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

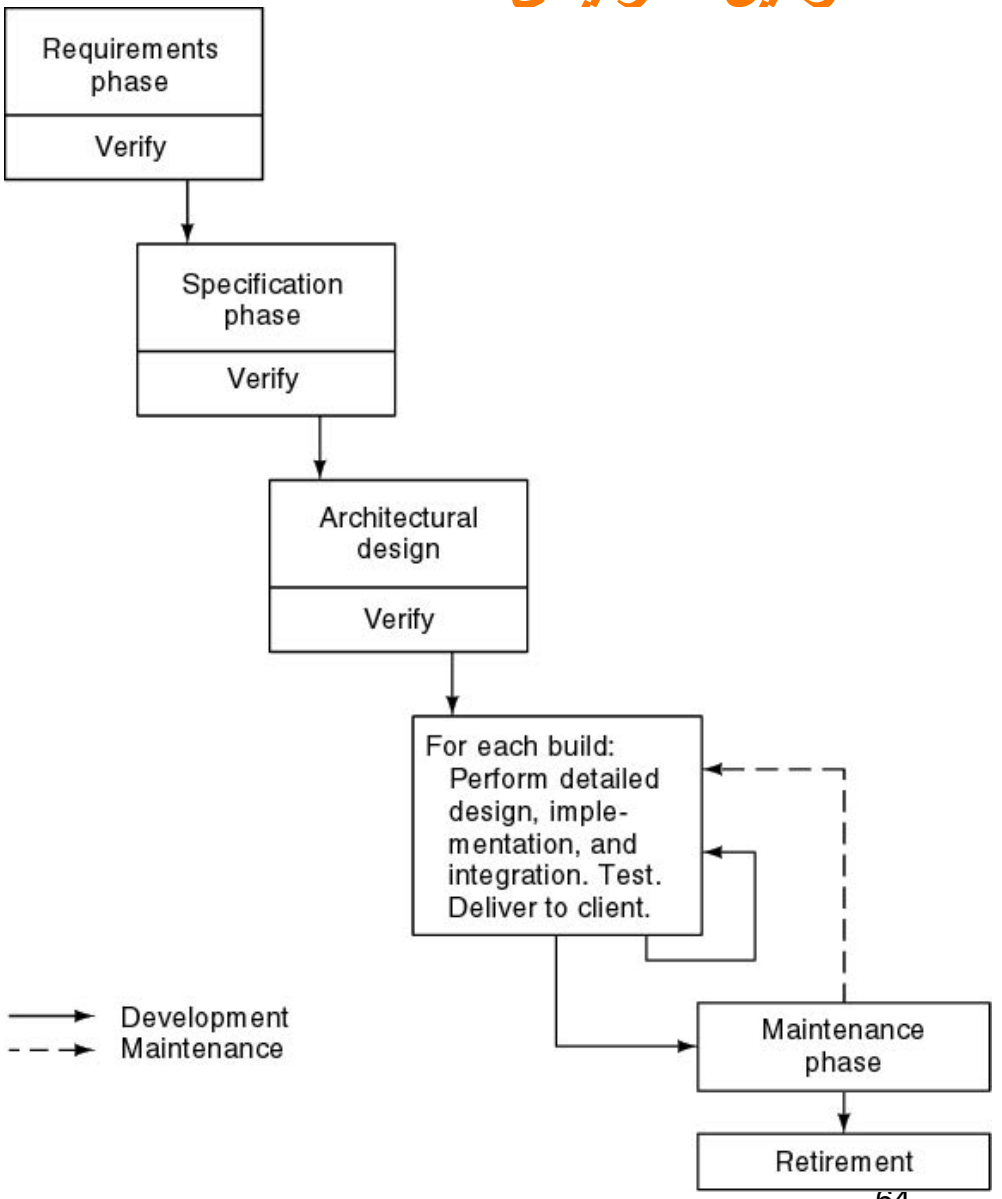
# الموديل التزايدي (Incremental Model)

# الموديل التزايدي



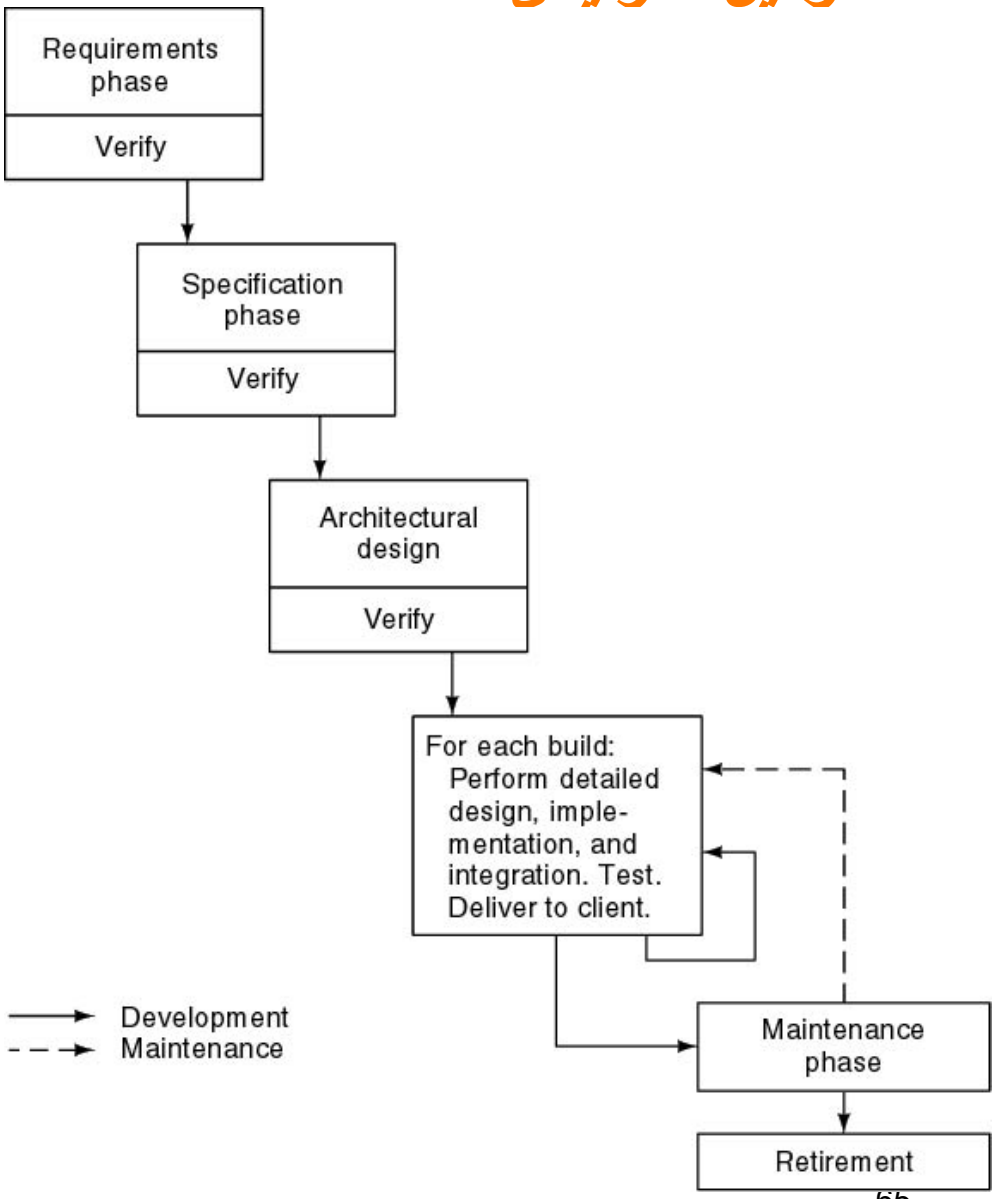
# الموديل التزايدى

- 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



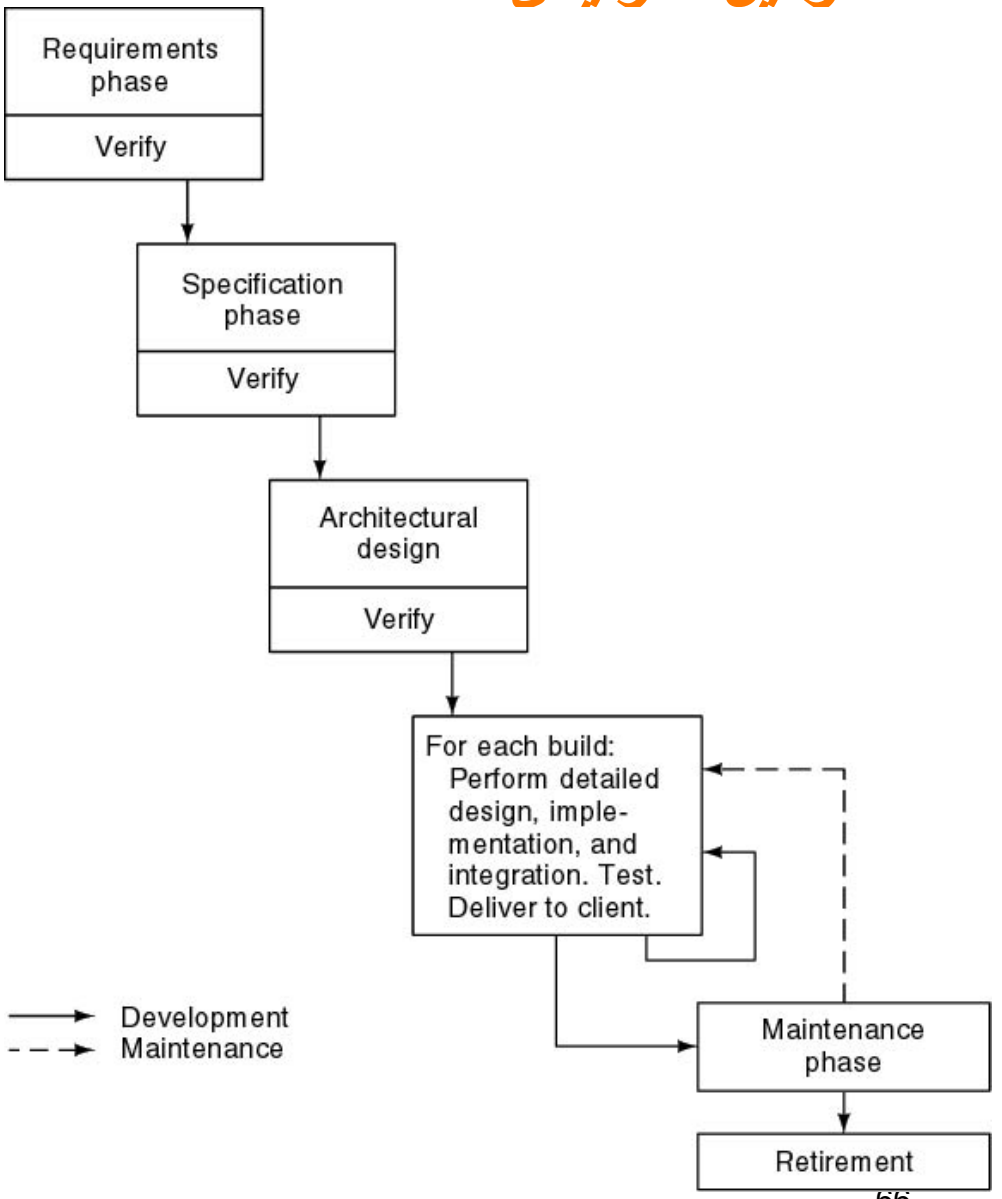
# الموديل التزايدى

- 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



# مقارنة الموديل التزايدي مع الموديلات السابقة

- موديل شلال الماء و موديل النموذج السريع
  - يُعطي منتج كامل للزبون
  - يتواجد تاريخ محدد لتسليم المنتج البرمجي
- في الموديل التزايدي
  - يتم إعطاء منتج قابل للتشغيل و ذو جودة جيدة للزبون في كل مرحلة يتم بناء build فيه

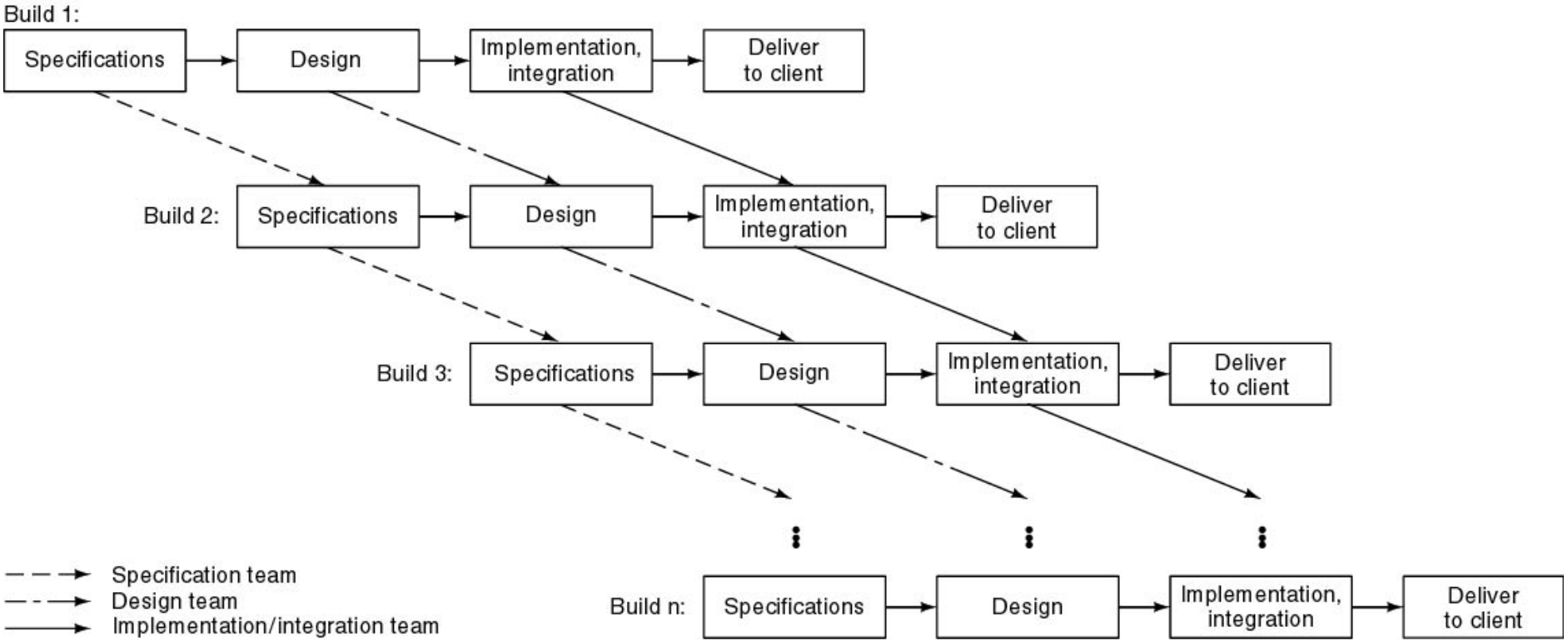
# نقاط القوة ونقاط الضعف

- 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

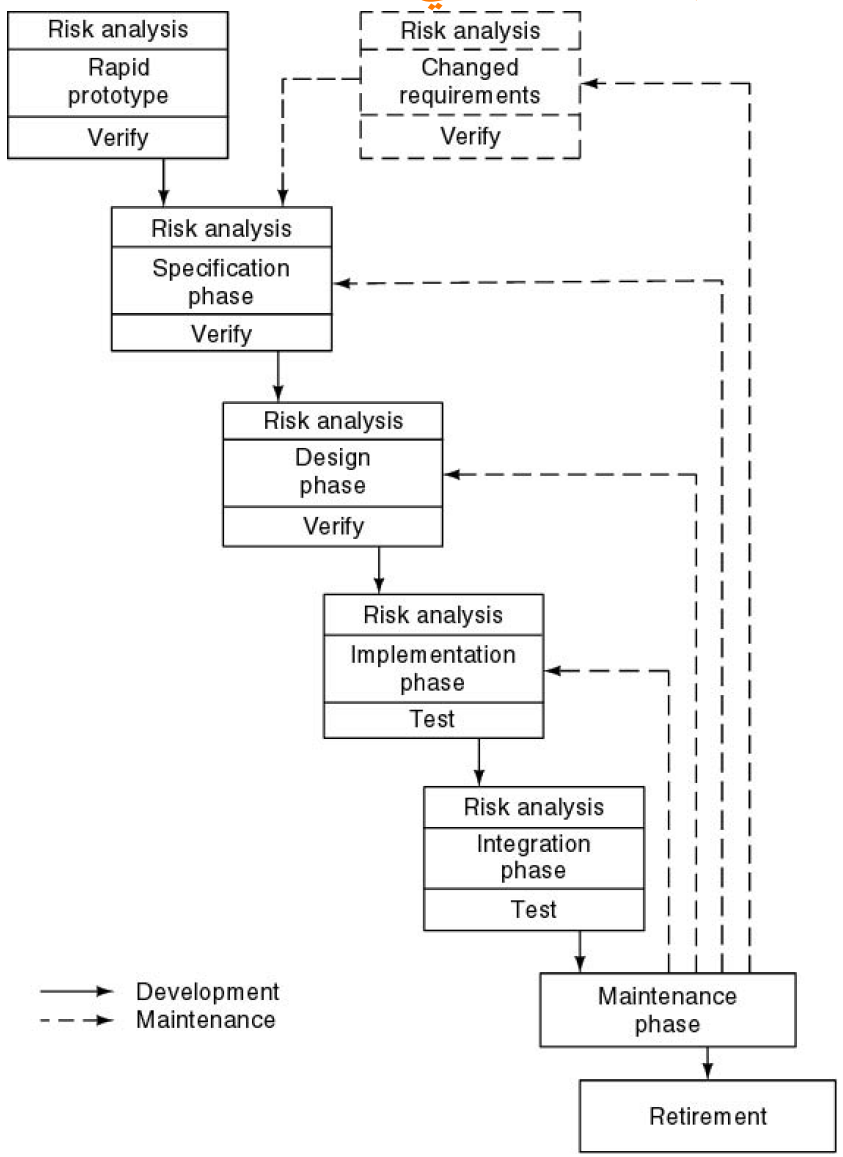
# المخاطر!!


Build 1:
Specifications
Design
Implementation, integration
Deliver to client
Build 2:
Specifications
Design
Implementation, integration
Deliver to client
Build 3:
Specifications
Design
Implementation, integration
Deliver to client
Build n:
Specifications
Design
Implementation, integration
Deliver to client
Specification team
Design team
Implementation/integration team

# الموديل الحلزوني (Spiral Model)

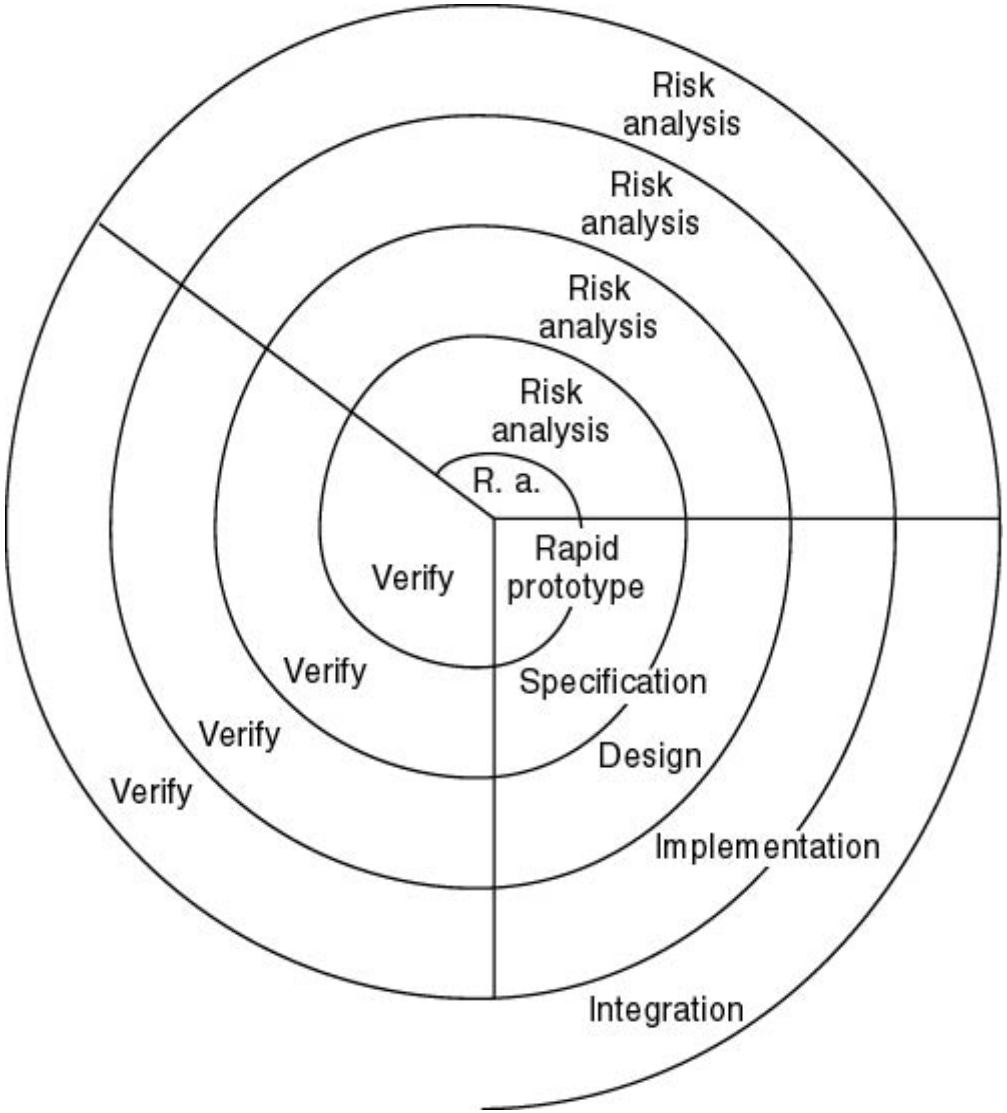
# الموديل الحلزوني

- نمط مبسط
  - موديل شلال الماء مضافاً له تحليل للمخاطر
- كل طور يُسبق بـ
  - فحص البدائل
  - تحليل المخاطر
- كل طور يُتبع بـ
  - تقييم وفحص
  - تخطيط للخطوة التالية

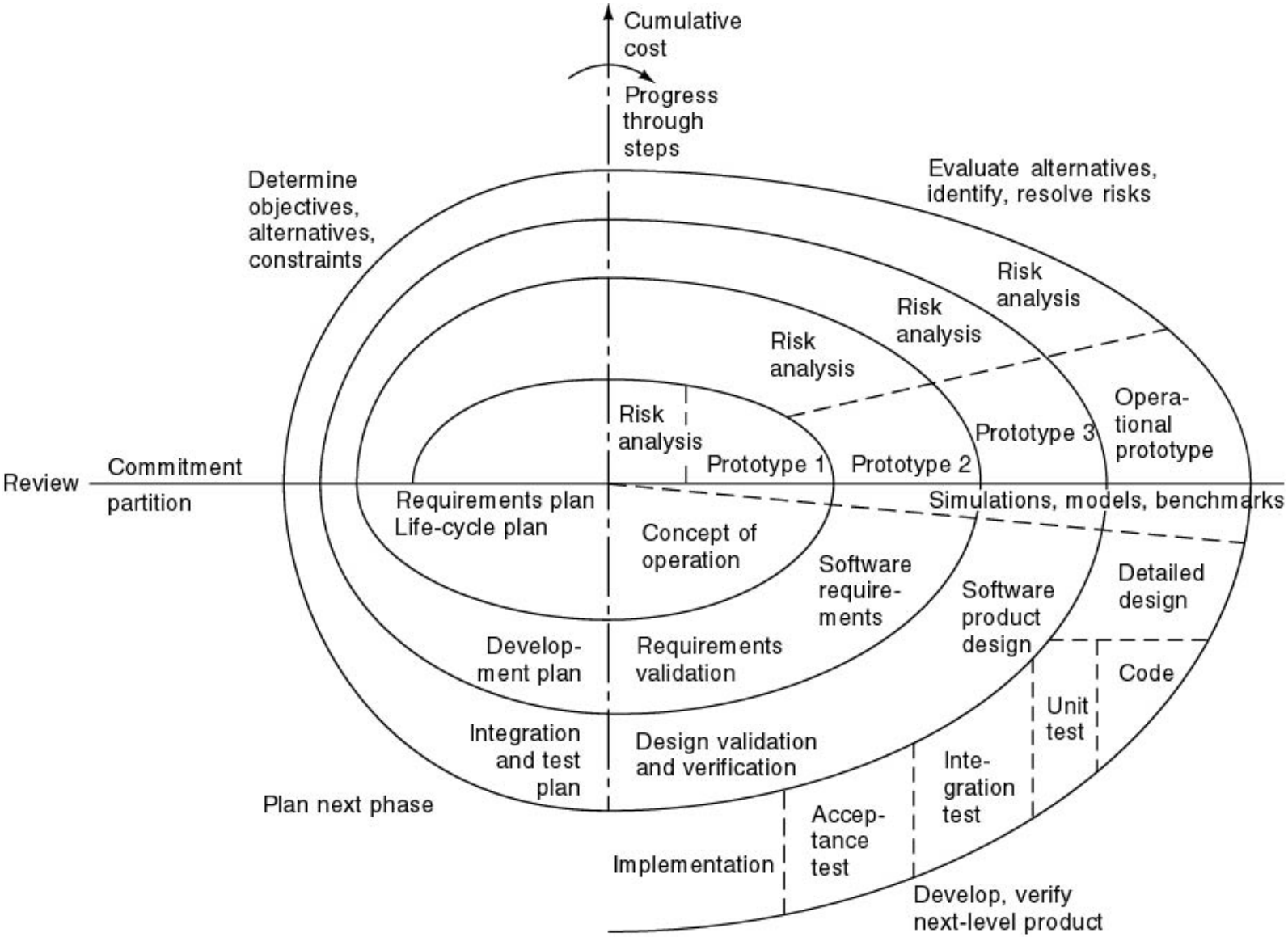


# الموديل الحلزوني البسيط

- إذا لم يكن من الممكن أن يتم حل المخاطر التي تم تحليلها, يتم إيقاف المشروع مباشرةً
- النموذج التجريبي يمكن أن يستخدم بفعالية للحصول على معلومات حول المخاطر المحترمة



# الموديل الحلزوني الموسع



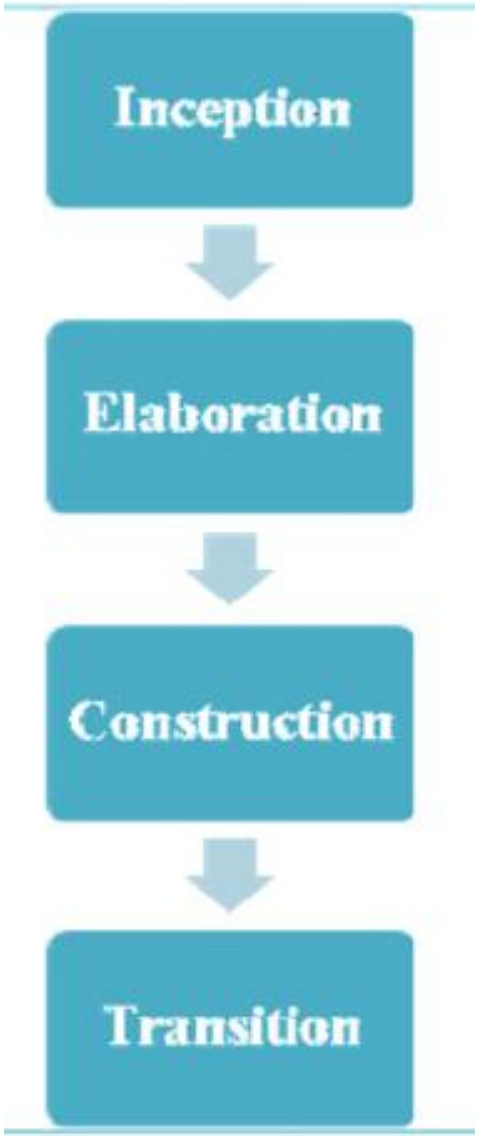
# تحليل الموديل الحلزوني

- نقاط القوة
  - ضمان كبير لجودة البرمجية
  - من السهل تقدير الحد الذي سيتم الفحص لعنده بعد تحليل المخاطر
  - لا تمييز بين التطوير و الصيانة
- نقاط الضعف
  - في المشاريع البرمجية الكبيرة, كل المخاطر ينبغي أن يتم تحليلها بشكل مسبق قبل توقيع العقد
  - هذا الموديل يحتاج مبرمجين ماهرين
  - ناجح للبرمجيات المصممة للعمل في بيئات داخلية خاصة

# النموذج الإجرائي الموحد ( Unified Process Model)

# الموديل الحلزوني

- Admired iterative and incremental software development process
- The well-known and generally documented enrichment of the unified process is the Rational Unified Process (RUP)
- not plainly a process, but rather an extensible framework which should be adapted for specific organizations or projects



# الموديل الحلزوني

- **Unified Process phases**
  - **Inception**: Requirements capture and analysis
  - **Elaboration**: System and class-level design
  - **Construction**: Implementation and testing
  - **Transition**: Deployment