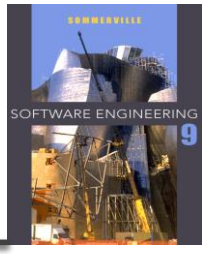
Chapter 1- Introduction

Lecture 1

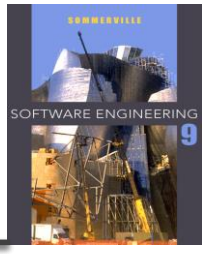
Topics covered



- ✧ Professional software development
  - What is meant by software engineering.
- ✧ Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.
- ✧ Case studies
  - An introduction to three examples that are used in later chapters in the book.

تطوير البرامج المهنية
ما هو المقصود بهندسة البرمجيات.
أخلاقيات هندسة البرمجيات
مقدمة موجزة للقضايا الأخلاقية التي تؤثر على هندسة البرمجيات.
دراسات الحالة
مقدمة لثلاثة أمثلة تم استخدامها في فصول لاحقة من الكتاب.

Software engineering



- The economies of ALL developed nations are dependent on software.

تعتمد اقتصادات جميع الدول المتقدمة على البرامج.

- More and more systems are software controlled

المزيد والمزيد من الأنظمة يتم التحكم فيها بواسطة البرامج

- Software engineering is concerned with theories, **methods and tools for professional software development.**

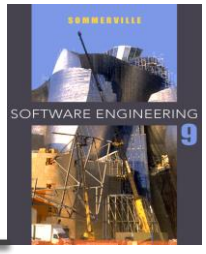
تهتم هندسة البرمجيات بالنظريات والأساليب والأدوات لتطوير البرامج الاحترافية.

.

- Expenditure on software represents a significant fraction of GNP in all developed countries.

يمثل الإنفاق على البرمجيات جزءًا كبيرًا من الناتج القومي الإجمالي في جميع البلدان المتقدمة.

Software costs تكاليف البرمجيات



- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.

غالبًا ما تهيمن تكاليف البرامج على تكاليف أنظمة الكمبيوتر. غالبًا ما تكون تكاليف البرامج على جهاز الكمبيوتر أكبر من تكلفة الأجهزة.

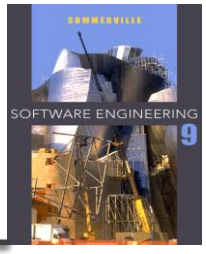
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.

البرمجيات تكلف صيانة أكثر من تطويرها. بالنسبة للأنظمة ذات العمر الطويل ، قد تكون تكاليف الصيانة عدة أضعاف تكاليف التطوير.

- Software engineering is concerned with cost-effective software development.

تهتم هندسة البرمجيات بتطوير البرمجيات الفعالة من حيث التكلفة.

Software products منتجات



1. Generic products

- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
- Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

المنتجات العامة

أنظمة قائمة بذاتها يتم تسويقها وبيعها لأي عميل يرغب في شرائها.

أمثلة - برامج الكمبيوتر مثل برامج الرسومات وأدوات إدارة المشاريع ؛ برنامج CAD برامج لأسواق معينة مثل أنظمة المواعيد لأطباء الأسنان.

2. Customized products

- Software that is commissioned by a specific customer to meet their own needs.
- Examples – embedded control systems, air traffic control software, traffic monitoring systems.

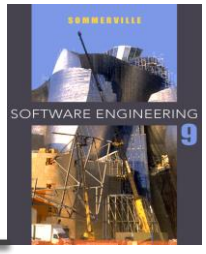
المنتجات حسب الطلب

برنامج تم تكليفه من قبل عميل معين لتلبية احتياجاته الخاصة.

أمثلة - أنظمة التحكم المضمنة ، وبرامج التحكم في الحركة الجوية ، وأنظمة مراقبة حركة المرور.

Product specification

مواصفات المنتج



1. Generic products

- The specification of what the software should do is owned by **the software developer** and decisions on software change **are made by the developer.**

المنتجات العامة

إن مواصفات ما يجب أن يفعله البرنامج مملوكة لمطور البرامج ويتخذ المطور القرارات بشأن تغيير البرنامج.

2. Customized products

- The specification of what the software should do is owned by **the customer for** the software and they make decisions on software changes **that are required**.

المنتجات حسب الطلب

إن مواصفات ما يجب أن يفعله البرنامج مملوكة للعميل للبرنامج ويتخذون القرارات بشأن تغييرات البرامج المطلوبة.

Frequently asked questions about software engineering
الأسئلة المتداولة حول هندسة البرمجيات

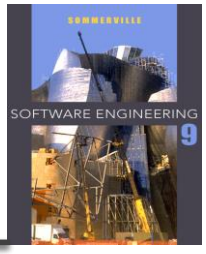
Question	Answer
What is software? ما هي البرمجيات؟	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market. برامج الكمبيوتر والوثائق المرتبطة بها. قد يتم تطوير منتجات البرامج لعميل معين أو يمكن تطويرها لسوق عام.
What are the attributes of good software? ما هي سمات البرمجيات الجيدة؟	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable. يجب أن توفر البرامج الجيدة الوظائف والأداء المطلوبين للمستخدم ويجب أن تكون قابلة للصيانة والاعتماد عليها وقابلة للاستخدام.
What is software engineering? ما هي هندسة البرمجيات؟	Software engineering is an engineering discipline that is concerned with all aspects of software production. هندسة البرمجيات هي تخصص هندسي يهتم بجميع جوانب إنتاج البرمجيات.
What are the fundamental software engineering activities? ما هي الأنشطة الأساسية لهندسة البرمجيات؟	Software specification, software development, software validation and software evolution. مواصفات البرمجيات وتطوير البرمجيات والتحقق من صحة البرمجيات وتطور البرمجيات.
What is the difference between software engineering and computer science? ما هو الفرق بين هندسة البرمجيات وعلوم الكمبيوتر؟	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. يركز علم الحاسوب على النظرية والأساسيات. تهتم هندسة البرمجيات بالجوانب العملية لتطوير وتقديم برامج مفيدة.
What is the difference between software engineering and system engineering? ما الفرق بين هندسة البرمجيات وهندسة النظم؟	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process. تهتم هندسة النظم بجميع جوانب تطوير الأنظمة المستندة إلى الكمبيوتر بما في ذلك هندسة الأجهزة والبرامج والعمليات. هندسة البرمجيات هي جزء من هذه العملية الأكثر عمومية.

Question	Answer
What are the key challenges facing software engineering? ما هي التحديات الرئيسية التي تواجه هندسة البرمجيات؟	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software. التعامل مع التنوع المتزايد والطلبات لتقليل أوقات التسليم وتطوير برامج جديرة بالثقة.
What are the costs of software engineering? ما هي تكاليف هندسة البرمجيات؟	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs. ما يقرب من 60٪ من تكاليف البرامج عبارة عن تكاليف تطوير ، و 40٪ تكاليف اختبار. بالنسبة للبرامج المخصصة ، غالبًا ما تتجاوز تكاليف التطوير تكاليف التطوير.
What are the best software engineering techniques and methods? ما هي أفضل تقنيات وأساليب هندسة البرمجيات؟	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another. بينما يجب إدارة جميع مشاريع البرامج وتطويرها بشكل احترافي ، فإن التقنيات المختلفة مناسبة لأنواع مختلفة من الأنظمة. على سبيل المثال ، يجب دائمًا تطوير الألعاب باستخدام سلسلة من النماذج الأولية بينما تتطلب أنظمة التحكم الحرجة للسلامة مواصفات كاملة وقابلة للتحليل ليتم تطويرها. لذلك لا يمكنك القول إن إحدى الطرق أفضل من الأخرى
What differences has the web made to software engineering? ما هي الاختلافات التي أحدثتها الويب في هندسة البرمجيات؟	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse. أدى الويب إلى توافر خدمات البرمجيات وإمكانية تطوير أنظمة قائمة على الخدمات الموزعةبشكل كبير. أدى تطوير الأنظمة المستندة إلى الويب إلى تطورات مهمة في لغات البرمجة وإعادة استخدام البرامج.

Essential attributes of good software
السمات الأساسية للبرامج الجيدة

Product characteristic	Description
Maintainability قابلية الصيانة	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment. يجب كتابة البرامج بهذه الطريقة بحيث يمكن أن تتطور لتلبية الاحتياجات المتغيرة للعملاء. هذه سمة مهمة لأن تغيير البرامج هو مطلب حتمي لبيئة الأعمال المتغيرة.
Dependability and security الموثوقية والأمان	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system. تتضمن اعتمادية البرامج مجموعة من الخصائص بما في ذلك الموثوقية والأمان والسلامة. يجب ألا تتسبب البرامج التي يمكن الاعتماد عليها في حدوث أضرار مادية أو اقتصادية في حالة فشل النظام. يجب ألا يتمكن المستخدمون الضارون من الوصول إلى النظام أو إتلافه.
Efficiency كفاءة	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc. يجب ألا يهدر البرنامج استخدام موارد النظام مثل الذاكرة ودورات المعالج. وبالتالي ، فإن الكفاءة تشمل الاستجابة ، ووقت المعالجة ، واستخدام الذاكرة ، وما إلى ذلك
Acceptability القبول	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use. يجب أن تكون البرامج مقبولة لنوع المستخدمين الذي تم تصميمها من أجلهم. هذا يعني أنه يجب أن يكون مفهومًا وقابلًا للاستخدام ومتوافقًا مع الأنظمة الأخرى التي يستخدمونها.

Software engineering

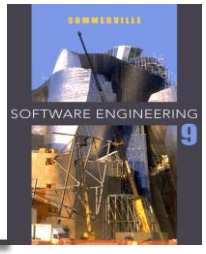


- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

- هندسة البرمجيات هي تخصص هندسي يهتم بجميع جوانب إنتاج البرامج من المراحل الأولى لمواصفات النظام وحتى صيانة النظام بعد استخدامه.

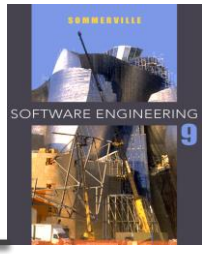
- Engineering discipline الانضباط الهندسة
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
    - استخدام النظريات والأساليب المناسبة لحل المشكلات مع مراعاة القيود التنظيمية والمالية.
- All aspects of software production جميع جوانب إنتاج البرمجيات
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.
    - ليس مجرد عملية تطوير تقنية. أيضًا إدارة المشاريع وتطوير الأدوات والأساليب وما إلى ذلك لدعم إنتاج البرامج.

Importance of software engineering
أهمية هندسة البرمجيات



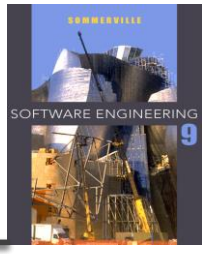
- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- يعتمد الأفراد والمجتمع أكثر فأكثر على أنظمة البرمجيات المتقدمة. نحن بحاجة إلى أن نكون قادرين على إنتاج أنظمة موثوقة وجديرة بالثقة اقتصاديًا وبسرعة.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.
- عادة ما يكون من الأرخص ، على المدى الطويل ، استخدام أساليب وتقنيات هندسة البرمجيات لأنظمة البرمجيات بدلاً من مجرد كتابة البرامج كما لو كانت مشروع برمجة شخصي. بالنسبة لمعظم أنواع الأنظمة ، فإن غالبية التكاليف هي تكاليف تغيير البرنامج بعد أن يدخل حيز الاستخدام

Software process activities
أنشطة عمليات البرمجيات



- **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.
  - مواصفات البرنامج ، حيث يحدد العملاء والمهندسون البرنامج الذي سيتم إنتاجه والقيود المفروضة على تشغيله.
- **Software development**, where the software is designed and programmed.
  - تطوير البرمجيات ، حيث يتم تصميم البرنامج وبرمجته.
- **Software validation**, where the software is checked to ensure that it is what the customer requires.
  - التحقق من صحة البرامج ، حيث يتم فحص البرنامج للتأكد من أنه ما يطلبه العميل.
- **Software evolution,** where the software is modified to reflect changing customer and market requirements.
  - تطور البرامج ، حيث يتم تعديل البرنامج ليعكس متطلبات العملاء والسوق المتغيرة.

General issues that affect most software
المشكلات العامة التي تؤثر على معظم البرامج

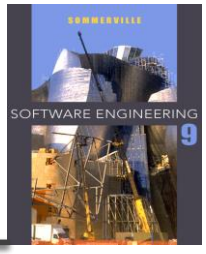


- **Heterogeneity** عدم التجانس
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
  - بشكل متزايد ، هناك حاجة للأنظمة للعمل كنظم موزعة عبر الشبكات التي تتضمن أنواعًا مختلفة من أجهزة الكمبيوتر والأجهزة المحمولة.

- **Business and social change** الأعمال التجارية والتغيير الاجتماعي
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
  - تتغير الأعمال والمجتمع بسرعة مذهلة مع تطور الاقتصادات الناشئة وإتاحة التقنيات الجديدة. يجب أن يكونوا قادرين على تغيير برامجهم الحالية وتطوير برامج جديدة بسرعة.

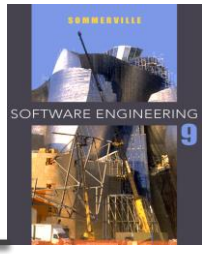
- **Security and trust** الأمان والثقة
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
  - نظرًا لأن البرمجيات متشابكة مع جميع جوانب حياتنا ، فمن الضروري أن نثق في هذا البرنامج.

Key points النقاط الرئيسية



- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- هناك العديد من أنواع الأنظمة المختلفة ويتطلب كل منها أدوات وتقنيات هندسة البرمجيات المناسبة لتطويرها.
- The fundamental ideas of software engineering are applicable to all types of software system.
- تنطبق الأفكار الأساسية لهندسة البرمجيات على جميع أنواع أنظمة البرمجيات.

Software engineering ethics
أخلاقيات هندسة البرمجيات

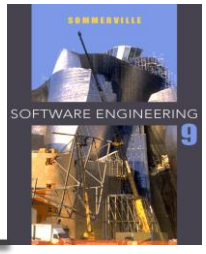


- Software engineering involves wider responsibilities than simply the application of technical skills.
- تنطوي هندسة البرمجيات على مسؤوليات أكبر من مجرد تطبيق المهارات التقنية.

- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- يجب أن يتصرف مهندسو البرمجيات بطريقة صادقة ومسؤولة أخلاقيا إذا كان لهم أن يحترموا كمحترفين.

- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.
- السلوك الأخلاقي هو أكثر من مجرد دعم للقانون ولكنه ينطوي على اتباع مجموعة من المبادئ الصحيحة أخلاقيا.

Issues of professional responsibility
قضايا المسؤولية المهنية



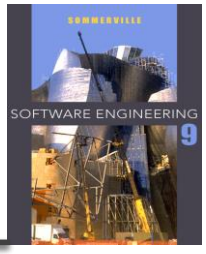
- **Confidentiality** سرية
  - **Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.**

يجب على المهندسين عادة احترام سرية أصحاب العمل أو العملاء بغض النظر عما إذا كان قد تم التوقيع على اتفاقية سرية رسمية أم لا.

- **Competence** كفاءة
  - **Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.**
    - **لا ينبغي للمهندسين أن يحرفوا مستوى كفاءتهم. لا ينبغي أن يقبلوا عملاً عمداً يخرج عن اختصاصهم.**

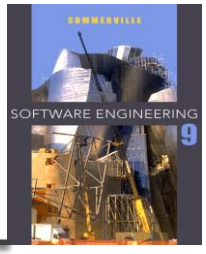
Issues of professional responsibility

قضايا المسؤولية المهنية



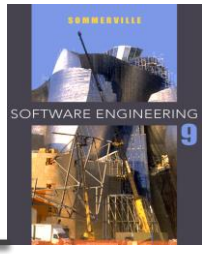
- **Intellectual property rights** حقوق الملكية الفكرية
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
  - يجب أن يكون المهندسون على دراية بالقوانين المحلية التي تحكم استخدام الملكية الفكرية مثل براءات الاختراع وحقوق التأليف والنشر وما إلى ذلك. يجب أن يكونوا حريصين على ضمان حماية الملكية الفكرية لأصحاب العمل والعملاء.
- **Computer misuse** سوء استخدام الكمبيوتر
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).
    - يجب ألا يستخدم مهندسو البرمجيات مهاراتهم التقنية لإساءة استخدام أجهزة الكمبيوتر الخاصة بالآخرين. تتراوح إساءة استخدام الكمبيوتر من التافهة نسبيًا (اللعب على جهاز صاحب العمل ، على سبيل المثال) إلى الخطورة للغاية (انتشار الفيروسات).

Case studies دراسات الحالة



- A personal insulin pump
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- نظام مدمج في مضخة الأنسولين يستخدمه مرضى السكر للحفاظ على التحكم في نسبة الجلوكوز في الدم.
- A mental health case patient management system
  - A system used to maintain records of people receiving care for mental health problems.
    - نظام يستخدم للاحتفاظ بسجلات للأشخاص الذين يتلقون رعاية لمشاكل الصحة العقلية.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas.
    - نظام جمع البيانات الذي يجمع البيانات حول أحوال الطقس في المناطق النائية.

Insulin pump control system
نظام التحكم في مضخة الأنسولين



- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.

يجمع البيانات من جهاز استشعار نسبة السكر في الدم ويحسب كمية الأنسولين المطلوب حقنها.

- Calculation based on the rate of change of blood sugar levels.

يعتمد الحساب على معدل تغير مستويات السكر في الدم.

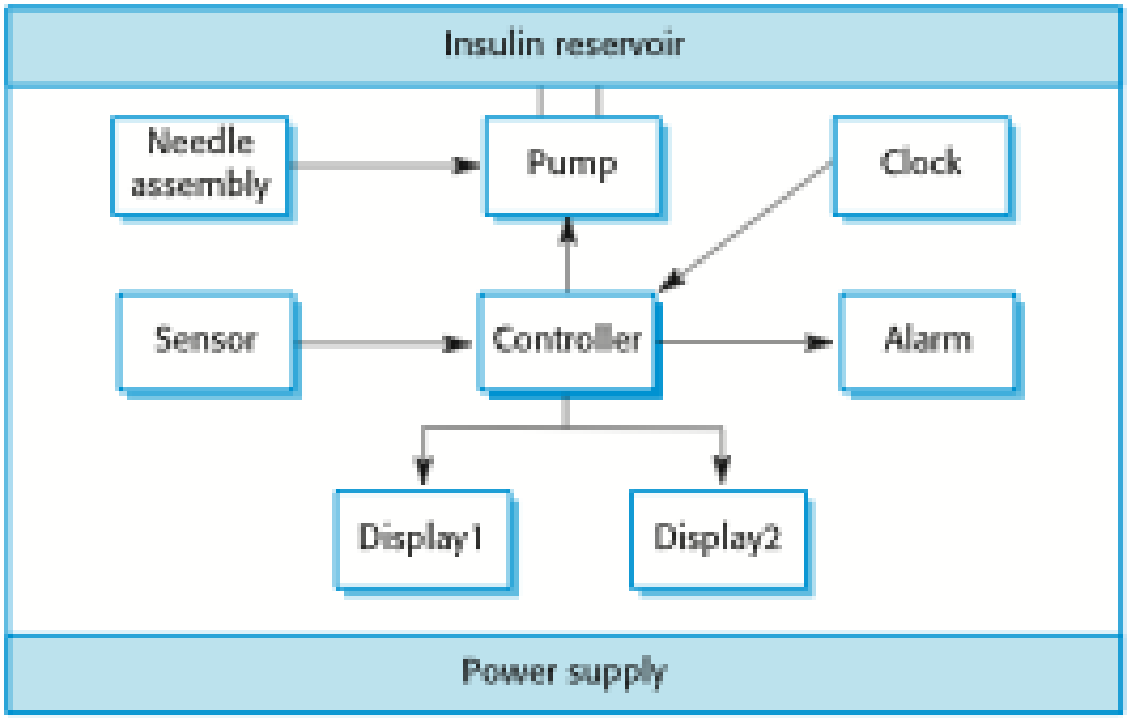
- Sends signals to a micro-pump to deliver the correct dose of insulin.

.
يرسل إشارات إلى مضخة صغيرة لتوصيل الجرعة الصحيحة من الأنسولين.

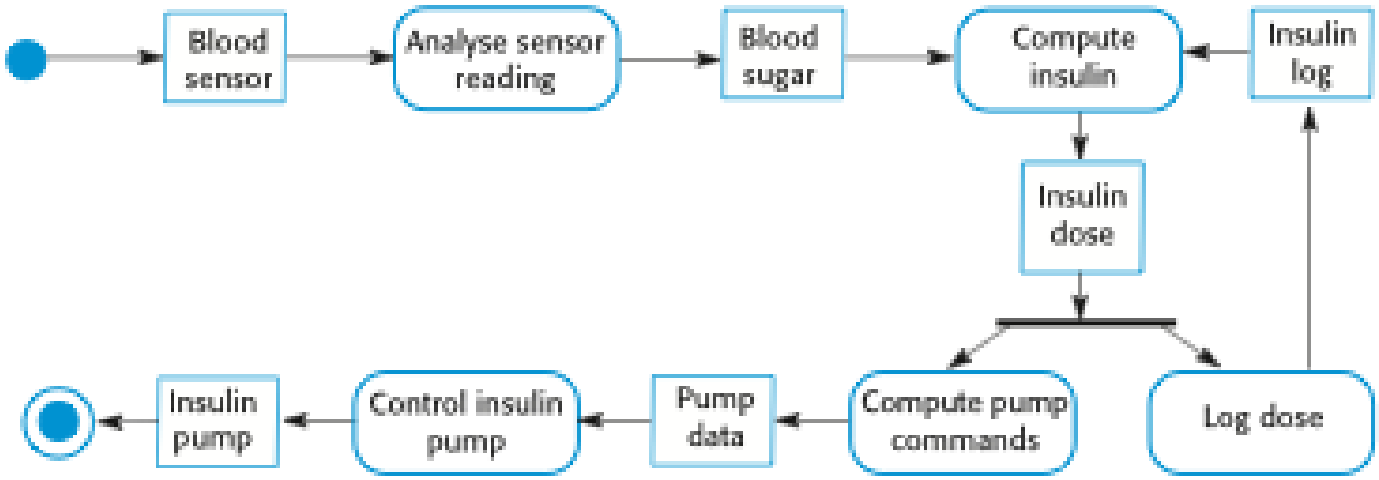
- Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.

نظام سلامة حرج حيث أن انخفاض نسبة السكر في الدم يمكن أن يؤدي إلى خلل في وظائف الدماغ والغيبوبة والموت ؛ مستويات السكر المرتفعة في الدم لها عواقب طويلة المدى مثل تلف العين والكلى.

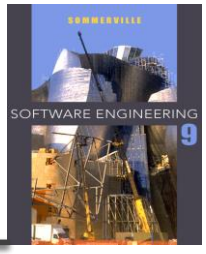
Insulin pump hardware architecture
هندسة أجهزة مضخة الأنسولين



Activity model of the insulin pump
نموذج نشاط لمضخة الأنسولين



Essential high-level requirements
المتطلبات الأساسية عالية المستوى



- The system shall be available to deliver insulin when required.

يجب أن يكون النظام متاحًا لتوصيل الأنسولين عند الحاجة.

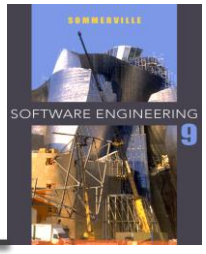
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.

يجب أن يعمل النظام بشكل موثوق ويقدم الكمية الصحيحة من الأنسولين لمواجهة المستوى الحالي لسكر الدم.

- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

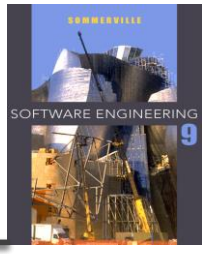
لذلك يجب تصميم النظام وتنفيذه لضمان أن النظام يلبي دائمًا هذه المتطلبات.

A patient information system for mental health care



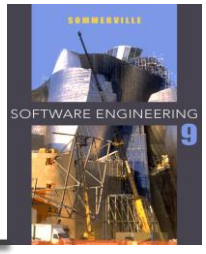
- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

MHC-PMS



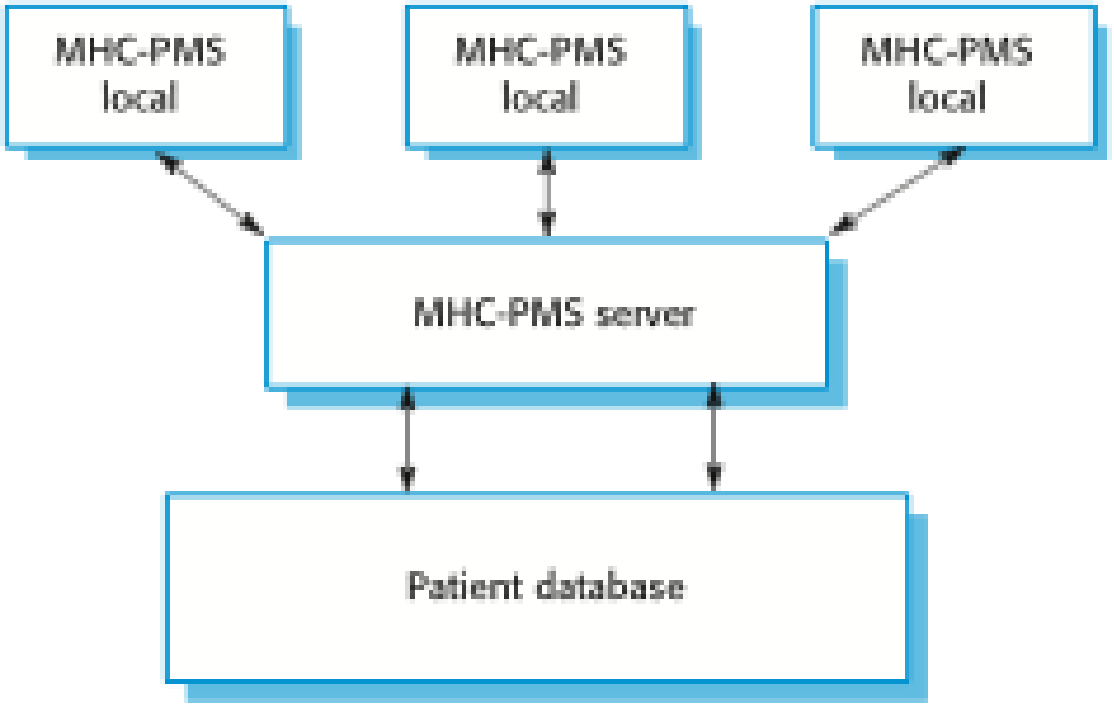
- The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

MHC-PMS goals

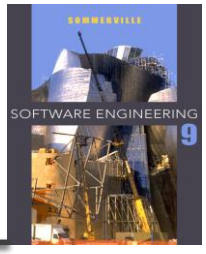


- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients.

The organization of the MHC-PMS

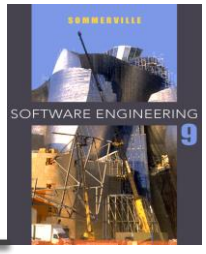


MHC-PMS key features



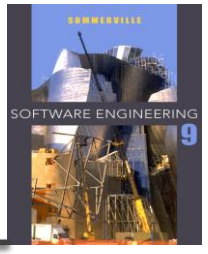
- ✧ Individual care management
  - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- ✧ Patient monitoring
  - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- ✧ Administrative reporting
  - The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

MHC-PMS concerns



- Privacy
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
  - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

Key points



- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.
- Three case studies are used in the book:
  - An embedded insulin pump control system
  - A system for mental health care patient management
  - A wilderness weather station

Chapter 2 – Software Processes

Topics covered

- ✧Software process models
- ✧Process activities
- ✧Coping with change
- ✧The Rational Unified Process
  - An example of a modern software process.

The software process
عملية البرنامج

✧**A structured set of activities required to develop a software system.** مجموعة منظمة من الأنشطة المطلوبة لتطوير نظام برمجي.

✧**Many different software processes but all involve:** العديد من عمليات البرامج المختلفة ولكن جميعها تتضمن:

1. **Specification – defining what the system should do;** المواصفات - تحديد ما يجب أن يفعله النظام ؛
1. **Design and implementation – defining the organization of the system and implementing the system;** التصميم والتنفيذ - تحديد تنظيم النظام وتنفيذ النظام ؛
1. **Validation – checking that it does what the customer wants;** التحقق من الصحة - التحقق من أنه يفعل ما يريده العميل ؛
1. **Evolution – changing the system in response to changing customer needs.** التطور - تغيير النظام استجابة لاحتياجات العملاء المتغيرة.

✧**A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.**

✧نموذج العملية البرمجية هو تمثيل تجريدي للعملية. يقدم وصفًا لعملية من منظور معين.

Software process descriptions
أوصاف عمليات البرامج

✧When we **describe and discuss processes**, we usually talk about **the activities in these processes**

عندما نصف العمليات ونناقشها ، عادة ما نتحدث عن الأنشطة في هذه العمليات.✧

✧ such as specifying a data model, designing a user interface. and the ordering of these activities.

مثل تحديد نموذج بيانات وتصميم واجهة مستخدم وترتيب هذه الأنشطة✧

✧Process descriptions may also include:: **قد تشمل أوصاف العملية أيضًا**

1. Products, which are the outcomes of a process activity;
2. **المنتجات ، وهي نتائج نشاط العملية**
3. Roles, which reflect the responsibilities of the people involved in the process;
   الأدوار ، التي تعكس مسؤوليات الأشخاص المشاركين في العملية ؛
4. Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

الشروط المسبقة واللاحقة ، وهي عبارات صحيحة قبل وبعد تفعيل نشاط العملية أو إنتاج منتج.

Plan-driven and agile processes

أنواع البروسس

✧**Plan-driven processes** are processes where all of the process activities are planned in advance and progress is measured against this plan.

✧ **العمليات التي تعتمد على الخطة** هي العمليات التي يتم فيها تخطيط جميع أنشطة العملية مسبقًا ويتم قياس التقدم وفقًا لهذه الخطة.

✧**In agile processes**, ***planning is incremental*** and it is easier to change the process to reflect changing customer requirements.

✧ في العمليات الرشيقة ، يكون التخطيط تدريجيًا ومن الأسهل تغيير العملية لتعكس متطلبات العملاء المتغيرة.

✧**In practice,** most practical processes include elements of both plan-driven and agile approaches.

✧ من الناحية العملية ، تشتمل معظم العمليات العملية على عناصر من كل من المناهج المستندة إلى الخطة والمرونة.

✧There are no right or wrong software processes.

✧لا توجد عمليات برمجية صحيحة أو خاطئة

Software process models
نماذج العمليات البرمجية

1. The waterfall model
   - Plan-driven model. Separate and distinct phases of specification and development.

نموذج خطة مدفوعة. مراحل منفصلة ومتميزة من المواصفات والتطوير.

1. Incremental development
   - Specification, development and validation are interleaved. May be plan-driven or agile.

المواصفات والتطوير والتحقق متشابكة. قد تكون خطة مدفوعة أو رشيقة.

2. Reuse-oriented software engineering
   - The system is assembled from existing components. **May be plan-driven or agile.**

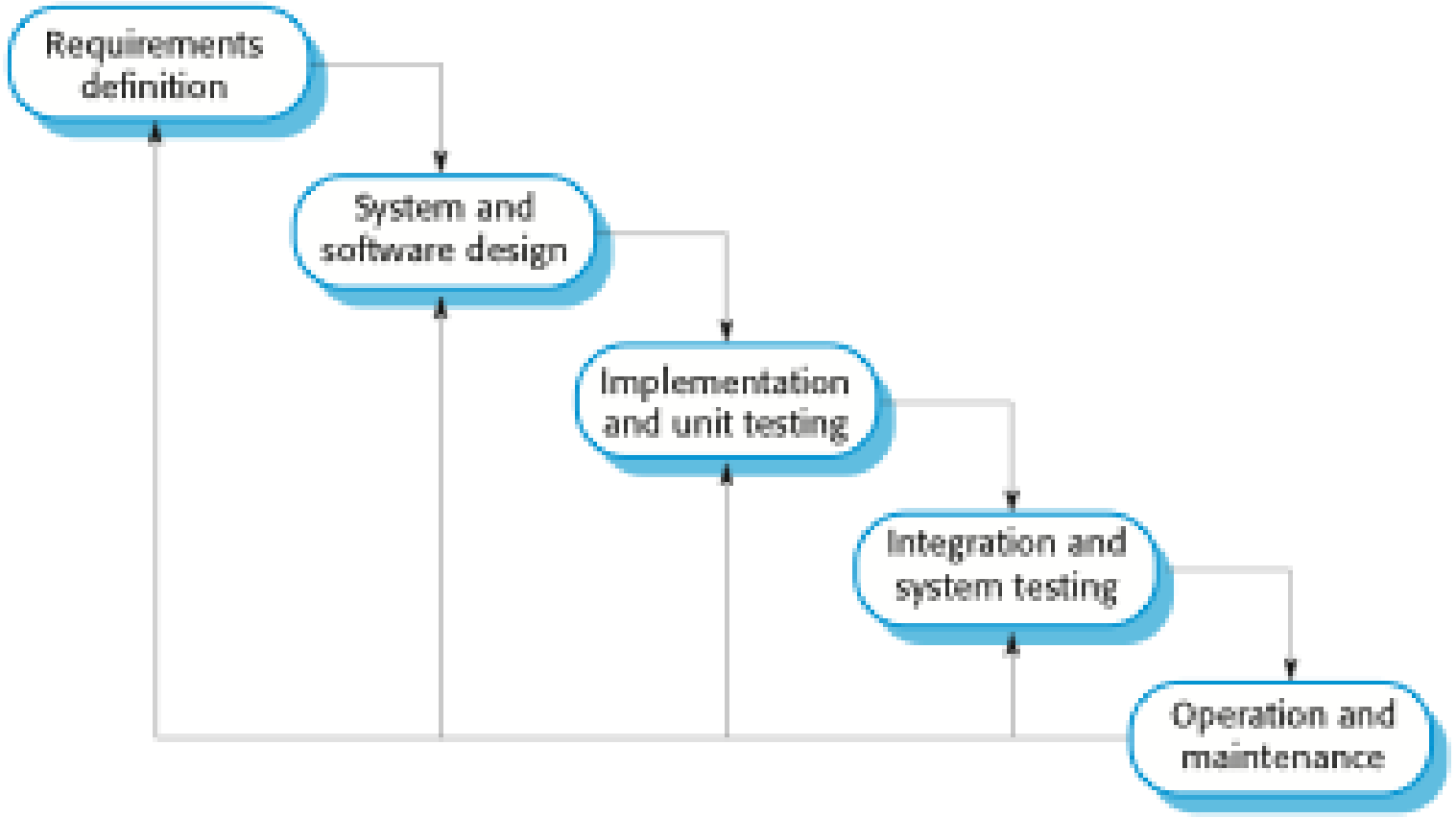
✧ يتم تجميع النظام من المكونات الموجودة. قد تكون خطة مدفوعة أو رشيقة.

✧ **In practice,** most large systems are developed using a process that incorporates elements from all of these models.

✧ في الممارسة العملية ، يتم تطوير معظم الأنظمة الكبيرة باستخدام عملية تتضمن عناصر من كل هذه النماذج.

The waterfall model
نموذج الشلال

من سيئاته انه لا يتقبل التغيير



Waterfall model phases

- There are separate identified phases in the waterfall model:

1-Requirements analysis and definition: The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

-تحليل وتعريف المتطلبات: يتم تحديد خدمات النظام والقيود والأهداف من خلال التشاور مع مستخدمي النظام. ثم يتم تعريفها بالتفصيل وتكون بمثابة مواصفات النظام.

2-System and software design: Software design involves identifying and describing the fundamental software system abstractions and their relationships.

تصميم النظام والبرمجيات: يتضمن تصميم البرمجيات تحديد ووصف تجريدات أنظمة البرمجيات الأساسية وعلاقاتها.

3-Implementation and unit testing :the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

-التنفيذ واختبار الوحدة: يتحقق تصميم البرنامج كمجموعة من البرامج أو وحدات البرنامج. يتضمن اختبار الوحدة التحقق من أن كل وحدة تفي بمواصفاتها.

4-Integration and system testing: The individual program units or programs are integrated and tested

التكامل واختبار النظام: يتم دمج واختبار وحدات البرامج الفردية أو البرامج

5-Operation and maintenance

The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

العيب الرئيسي لنموذج الشلال هو صعوبة استيعاب التغيير بعد أن تجري العملية. من حيث المبدأ ، يجب أن تكتمل المرحلة قبل الانتقال إلى المرحلة التالية.

Waterfall model problems
مشاكل نموذج الشلال

1. **Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.**
   - **التقسيم غير المرن للمشروع إلى مراحل متميزة يجعل من الصعب الاستجابة لمتطلبات العملاء المتغيرة.**
   - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
   - **لذلك ، يكون هذا النموذج مناسبًا فقط عندما تكون المتطلبات مفهومة جيدًا وستكون التغييرات محدودة إلى حد ما أثناء عملية التصميم.**
   - Few business systems have stable requirements.

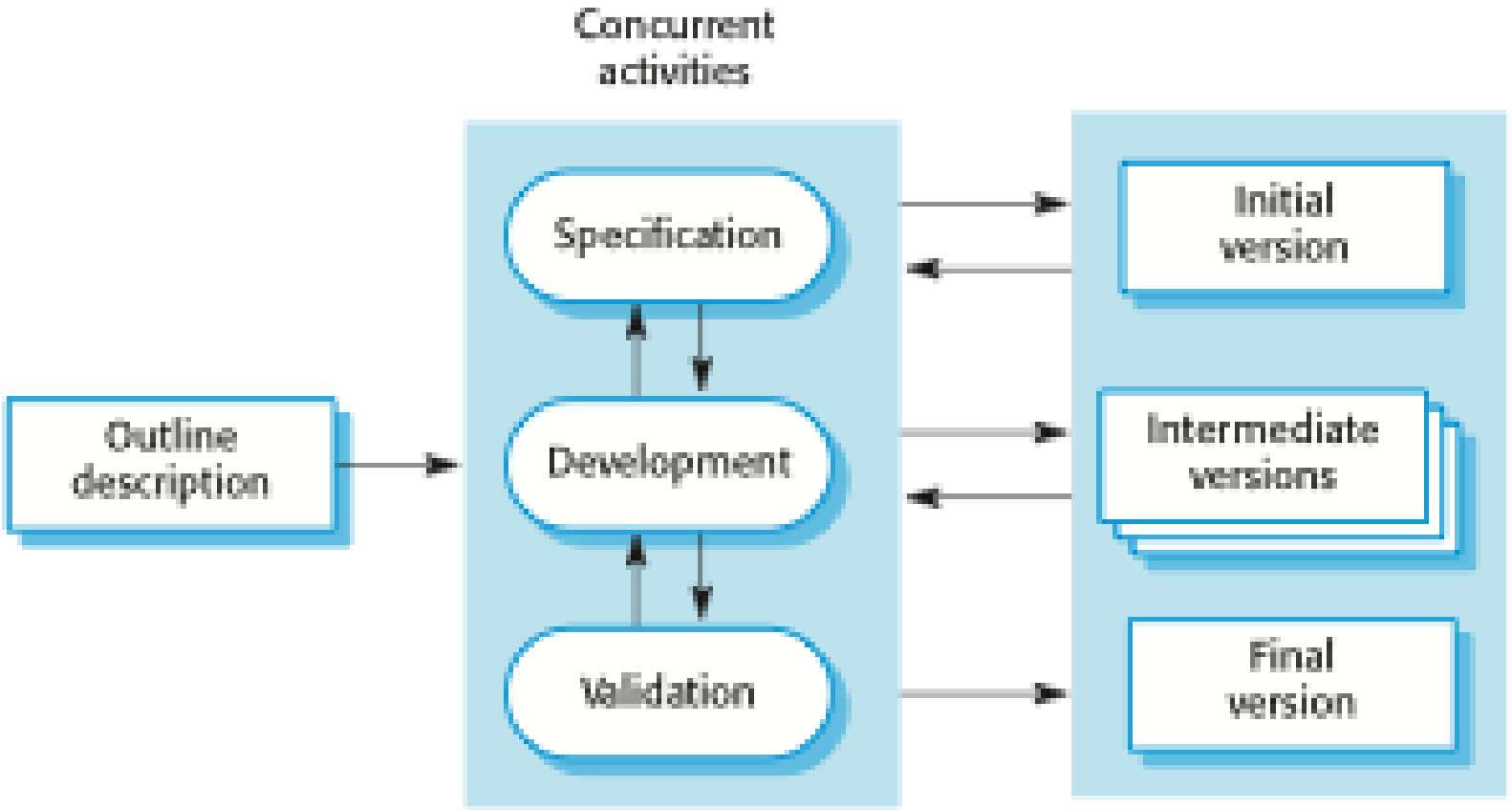
قليل من أنظمة الأعمال لديها متطلبات مستقرة.

1. **The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.**

يستخدم نموذج الشلال في الغالب لمشاريع هندسة الأنظمة الكبيرة حيث يتم تطوير نظام في عدة مواقع.

- In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work. **في هذه الظروف ، تساعد الطبيعة القائمة على الخطة لنموذج الشلال في تنسيق العمل**

Incremental development
التنمية المتزايدة
in parrall



Incremental development benefits
فوائد التنمية المتزايدة

- The cost of accommodating changing customer requirements is reduced.

تقليل تكلفة استيعاب متطلبات العملاء المتغيرة.

  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
  - كمية التحليل والتوثيق التي يجب إعادة بناؤها أقل بكثير مما هو مطلوب مع نموذج الشلال.

- **It is easier to get customer feedback** on the development work that has been done.

من الأسهل الحصول على ملاحظات العملاء حول أعمال التطوير التي تم إجراؤها

  - Customers can comment on demonstrations of the software and see how much has been implemented.

يمكن للعملاء التعليق على العروض التوضيحية للبرنامج ومعرفة مقدار ما تم تنفيذه.

- More rapid delivery and deployment of useful software to the customer is possible.

من الممكن توفير المزيد من البرامج المفيدة ونشرها للعميل بشكل أسرع.

  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

يمكن للعملاء استخدام البرنامج واكتساب قيمة منه في وقت أبكر مما هو ممكن من خلال عملية الانحدار.

Incremental development problems

مشاكل النمو المتزايدة

✧The process is not visible. العملية غير مرئية.

- Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

يحتاج المديرون إلى مخرجات منتظمة لقياس التقدم. إذا تم تطوير الأنظمة بسرعة ، فلن يكون إنتاج المستندات التي تعكس كل إصدار من النظام أمرًا فعالاً من حيث التكلفة.

✧System **structure tends to degrade as new increments are added.**

تميل بنية النظام إلى التدهور مع إضافة زيادات جديدة.

- Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

ما لم يتم إنفاق الوقت والمال على إعادة بناء البرامج لتحسين البرنامج ، فإن التغيير المنتظم يميل إلى إفساد هيكله. يصبح دمج المزيد من التغييرات البرمجية أمرًا صعبًا ومكلفًا بشكل متزايد.

Reuse-oriented software engineering

هندسة البرمجيات الموجهة لإعادة الاستخدام

✧Based on systematic **reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf)** systems.

استنادًا إلى إعادة الاستخدام المنهجية حيث يتم دمج الأنظمة من المكونات الحالية أو () الأنظمة (التجارية الجاهزة).

✧**Process stages** **مراحل العملية**

- **Component analysis**; given the requirement specification, a search is made for components to implement that specification. Usually, there is not an exact match and components may provide some of required function
- **تحليل المكونات نظرًا لمواصفات المتطلبات ، يتم إجراء بحث عن مكونات لتنفيذ تلك المواصفات. عادة ، لا يوجد تطابق تام وقد توفر المكونات بعض الوظائف المطلوبة**
- **Requirements modification**; the requirements are analyzed using information about the components which have been discovered. They modify it to reflect the available components

  تعديل المتطلبات. يتم تحليل المتطلبات باستخدام معلومات حول المكونات التي تم اكتشافها. قاموا بتعديله ليعكس المكونات المتاحة

- **System design with reuse**; the framework of the system is designed or an existing framework is revised.

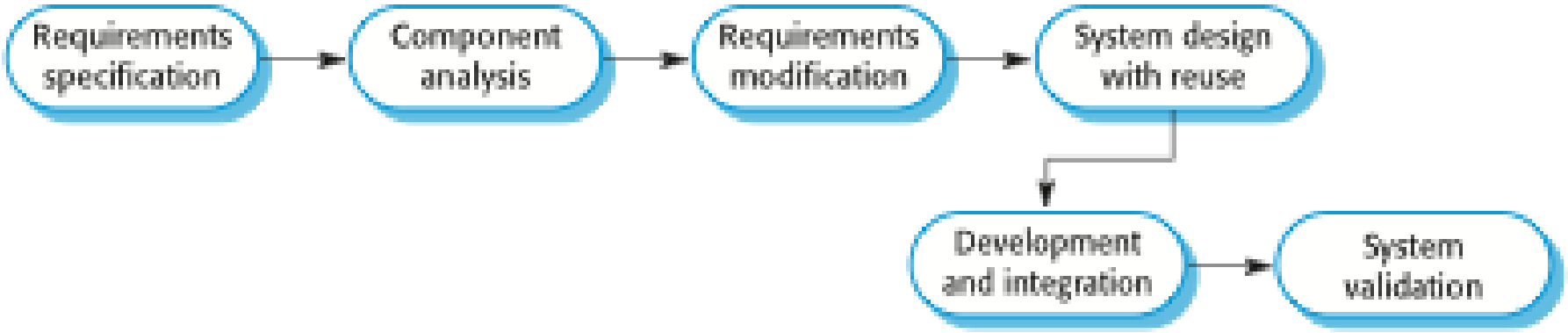
  تصميم النظام مع إعادة الاستخدام. تم تصميم إطار عمل النظام أو مراجعة إطار العمل الحالي.

- **Development and integration**. Software which cannot be bought is developed, the components and COTS are integrated to create the system.

  التطوير والتكامل يتم تطوير البرامج التي لا يمكن شراؤها ، ويتم دمج المكونات و () لإنشاء النظام.

- Reuse is now the standard approach for building many types of business system
  - إعادة الاستخدام هي الآن النهج القياسي لبناء العديد من أنواع أنظمة الأعمال

Reuse-oriented software engineering
هندسة البرمجيات الموجهة لإعادة الاستخدام



The advantages are:

1- reduced the amount of software to be developed
2- reduce the cost and risks.
3- lead to a faster delivery of the software

الإيجابيات:

1- تقليل كمية البرمجيات المراد تطويرها
2- تقليل التكلفة والمخاطر.
3- يؤدي إلى تسليم أسرع للبرنامج

Types of software component
أنواع مكونات البرنامج

- Web services that are developed according to service standards and which are available for remote invocation.
- خدمات الويب التي تم تطويرها وفقًا لمعايير الخدمة والمتاحة للاستدعاء عن بُعد.
- Collections of objects that are developed as a package to be integrated with a component framework such as {.NET or J2EE.}
- مجموعات الكائنات التي تم تطويرها كحزمة ليتم دمجها مع إطار عمل مكون مثل {}
- Stand-alone software systems (COTS) that are configured for use in a particular environment.
- أنظمة البرامج المستقلة () التي تم تكوينها للاستخدام في بيئة معينة.

Process activities
أنشطة العملية

✧Real software processes are interleaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.

عمليات البرمجيات الحقيقية عبارة عن تسلسلات متداخلة من الأنشطة الفنية والتعاونية والإدارية بهدف عام يتمثل في تحديد وتصميم وتنفيذ واختبار نظام برمجي.

✧The four basic process activities of specification, development, validation and evolution are organized differently in different development processes. In the waterfall model, they are organized in sequence, whereas in incremental development they are inter-leaved.

يتم تنظيم أنشطة العملية الأساسية الأربعة المتمثلة في المواصفات والتطوير والتحقق من الصحة والتطور بشكل مختلف في عمليات التطوير المختلفة. في نموذج الشلال ، يتم تنظيمها بالتسلسل ، بينما في التطوير التدريجي تكون متداخلة الأوراق.

Software specification
مواصفات البرنامج

✧**The process of establishing what services are required and the constraints on the system's operation and development.**

عملية تحديد الخدمات المطلوبة والقيود على تشغيل النظام وتطويره.✧

✧**Requirements engineering process** *** متطلبات العملية الهندسية**

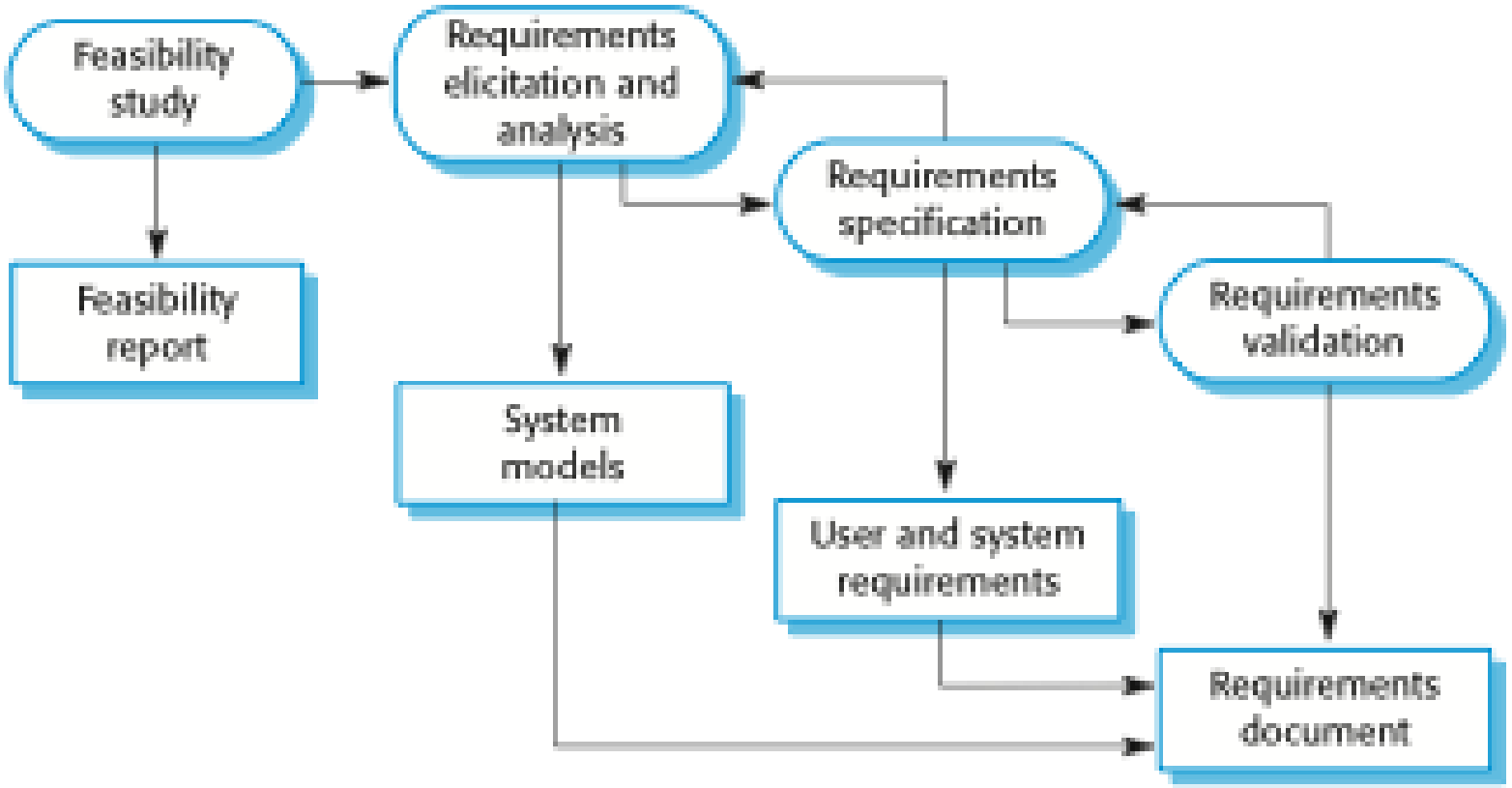
- **Feasibility study** **دراسة الجدوى**
  - **Is it technically and financially feasible to build the system?**
    - **هل من الممكن فنيا وماليا بناء النظام؟**
- **Requirements elicitation and analysis** *** استنباط المتطلبات وتحليلها**
  - **What do the system stakeholders require or expect from the system?**

    ما الذي يطلبه أو يتوقعه أصحاب المصلحة في النظام من النظام؟
- **Requirements specification** **مواصفات المتطلبات**
  - **Defining the requirements in detail**

    تحديد المتطلبات بالتفصيل
- **Requirements validation** **التحقق من المتطلبات**
  - **Checking the validity of the requirements**

    التحقق من صلاحية المتطلبات

The requirements engineering process
عملية المتطلبات الهندسية



The requirements engineering process (Continue)

عملية هندسة المتطلبات (متابعة)

✧System Models helps to understand the system that should be developed

تساعد نماذج النظام على فهم النظام الذي يجب تطويره

✧**User Requirements**: Stating the requirements in a short and high level fashion so that they can be understood by end users and customers

متطلبات المستخدم: تحديد المتطلبات بأسلوب قصير وعالي المستوى بحيث يمكن فهمها من قبل المستخدمين النهائيين والعملاء

✧**System Requirements**: Adding details to user requirements. This is needed by developers

متطلبات النظام: إضافة التفاصيل لمتطلبات المستخدم. هذا مطلوب من قبل المطورين

✧**Requirements Validation**: Making sure the requirements are consistent, complete, and can be realized

التحقق من صحة المتطلبات: التأكد من أن المتطلبات متسقة وكاملة ويمكن تحقيقها

Software design and implementation
تصميم البرامج وتنفيذها

✧**The process of converting the system specification into an executable system.**

عملية تحويل مواصفات النظام إلى نظام قابل للتنفيذ.

✧**Software design** **تصميم البرمجيات**

- **Design a software structure that realises the specification;**

تصميم بنية برمجية تحقق المواصفات ؛

✧**Implementation** **تطبيق**

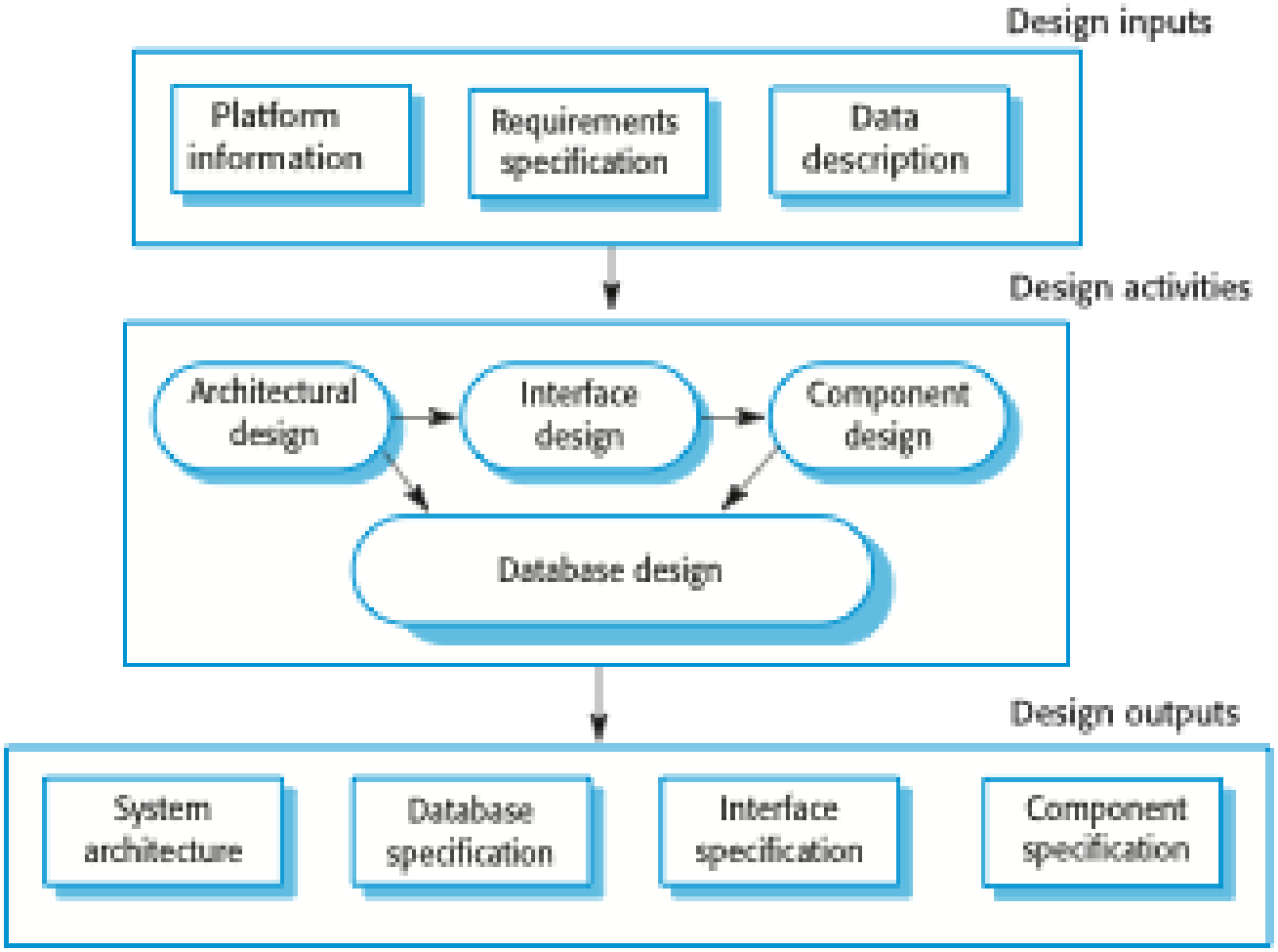
- **Translate this structure into an executable program;**

ترجمة هذا الهيكل إلى برنامج قابل للتنفيذ ؛

✧**The activities of design and implementation are closely related and may be inter-leaved.**

ترتبط أنشطة التصميم والتنفيذ ارتباطًا وثيقًا ويمكن أن تكون متداخلة.

A general model of the design process
نموذج عام لعملية التصميم



Design Inputs
مدخلات التصميم

- ✧Platform Information: Examples: Type of operating system, other systems to interact with, pre-existing data

  معلومات النظام الأساسي: أمثلة: نوع نظام التشغيل والأنظمة الأخرى التي يجب التفاعل معها والبيانات الموجودة مسبقًا

- ✧Requirements Specification **مواصفات المتطلبات**

- ✧Data Description: The data the system should generate and use

  وصف البيانات: البيانات التي يجب أن ينشئها النظام ويستخدمها

Design activities
أنشطة التصميم

- **Architectural design: the sub-systems making up the system and their relationships are identified and documented**

التصميم المعماري: يتم تحديد وتوثيق الأنظمة الفرعية المكونة للنظام وعلاقاتها

- **Abstract specification: for each sub-system, an abstract specification of its services and the constraints under which it must operate is produced**

المواصفات المجردة: لكل نظام فرعي ، يتم إنتاج مواصفات مجردة لخدماته والقيود التي يجب أن يعمل في ظلها

- **Interface design: for each sub-system, its interface with other sub-system is designed and documented. This interface specification must be unambiguous as it allow the sub-system to be used without the knowledge of the other sub-system operation**

تصميم الواجهة: لكل نظام فرعي ، تم تصميم وتوثيق واجهته مع النظام الفرعي الآخر. يجب أن تكون مواصفات الواجهة هذه واضحة لأنها تسمح باستخدام النظام الفرعي دون معرفة عملية النظام الفرعي الأخرى

- **Component design: services are allocated to components and the interface if this components are designed.**

تصميم المكون: يتم تخصيص الخدمات للمكونات والواجهة إذا تم تصميم هذه المكونات.

- **Data structure design: the data structure used in the system implementation are designed in detail and specified (Database Design)**

تصميم هيكل البيانات: تم تصميم بنية البيانات المستخدمة في تنفيذ النظام بالتفصيل وتحديد (تصميم قاعدة البيانات)

- **Algorithm design: the algorithms used to provide services are designed in detail and specified**

تصميم الخوارزمية: تم تصميم الخوارزميات المستخدمة لتقديم الخدمات بالتفصيل وتحديدها

Software validation
التحقق من صحة البرامج

✧Verification and validation **(V & V)** is intended to show that a system conforms to its specification and meets the requirements of the system customer.

يهدف التحقق والتحقق () إلى إظهار أن النظام يتوافق مع مواصفاته ويلبي متطلبات عميل النظام.

✧Involves checking and review processes and system testing.

تتضمن عمليات فحص ومراجعة واختبار النظام.

✧System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

يتضمن اختبار النظام تنفيذ النظام مع حالات الاختبار المشتقة من مواصفات البيانات الحقيقية التي سيعالجها النظام.

✧Testing is the most commonly used **(V & V)** activity.

الاختبار هو النشاط () الأكثر استخدامًا.

Verification vs validation
تصديق{{{{{{ تحقق

✧Verification: **تحقق:**

"Are we building the product right”.."**هل نبني المنتج بشكل صحيح**"

- The software should conform to its specification.

يجب أن يتوافق البرنامج مع المواصفات الخاصة به.

✧Validation: **تصديق:**

"Are we building the right product”..**"هل نبني المنتج المناسب**"

- The software should do what the user really requires.

يجب أن يقوم البرنامج بما يطلبه المستخدم حقًا.

Testing is part of a more general verification and validation process, which also includes static validation techniques

يعد الاختبار جزءًا من عملية تحقق وتحقق أكثر عمومية ، والتي تتضمن أيضًا تقنيات التحقق من الصحة الثابتة

Verification vs validation
مقارنة تصديق{{{{{{ تحقق

✧**Verification: A test of a system to prove that it meets all its specified requirements <u>at a particular stage of its development</u>.**

التحقق: اختبار لنظام لإثبات أنه يفي بجميع متطلباته المحددة في مرحلة معينة من تطوره.

- **Make sure we are <u>currently</u> on the right path toward our goal**
  - تأكد من أننا نسير حاليًا على الطريق الصحيح نحو هدفنا
- **Eg. Comparing requirements against code logic**
- **Eg. Code review**
- **Normally, it does not involve actual testing** في العادة ، لا يتضمن الاختبار الفعلي

✧**Validation: An activity that ensures that an end product stakeholder's true needs and expectations are met**

التحقق من الصحة: نشاط يضمن تلبية الاحتياجات والتوقعات الحقيقية لأصحاب المصلحة في المنتج النهائي

- **Performed upon completion of a given module or the entire system**
  - يتم إجراؤه عند الانتهاء من وحدة معينة أو النظام بأكمله
- **Involves actual testing** يتضمن الاختبار الفعلي

Taken from: Plutora.com

Verification vs validation

مقارنة تصديق{{{{{{ تحقق

- Driving with [GPS] directions: القيادة باستخدام [] الاتجاهات:
  - Verification: If <Google> maps tells you to drive south until you pass a gas station, you can verify this is true while you are driving
    - التحقق: إذا أخبرك <> الخرائط بالقيادة جنوباً حتى تمر بمحطة وقود ، فيمكنك التحقق من صحة ذلك أثناء القيادة
  - Validation: Did I reach my final destination? (eg. A beach). Is beach name the same as the one I wanted to go to?

تصديق : هل وصلت إلى وجهتي النهائية؟ (على سبيل المثال ، شاطئ). هل اسم الشاطئ هو نفس الاسم الذي أردت الذهاب إليه؟

- Designing Excel Spreadsheet تصميم جدول إكسل
  - Verification: Check that each single equation calculation is correct before applying it in more complex calculations.

التحقق: تحقق من صحة حساب كل معادلة قبل تطبيقها في عمليات حسابية أكثر تعقيدًا.

  - Validation: Check that final calculations of the spreadsheet meets what the customer asked for

التصديق: تحقق من أن الحسابات النهائية لجدول البيانات تفي بما يطلبه العميل

Taken from: Plutora.com

Stages of testing
مراحل الاختبار

Testing stages
مراحل الاختبار

1. Development or component testing

- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.

- **التطوير أو اختبار المكونات**
- **يتم اختبار المكونات الفردية بشكل مستقل ؛**
- **قد تكون المكونات وظائف أو كائنات أو مجموعات متماسكة من هذه الكيانات.**

2. System testing

- Testing of the system as a whole. Testing of emergent properties is particularly important.

- **اختبار النظام**
- **اختبار النظام ككل. اختبار الخصائص الناشئة مهم بشكل خاص.**

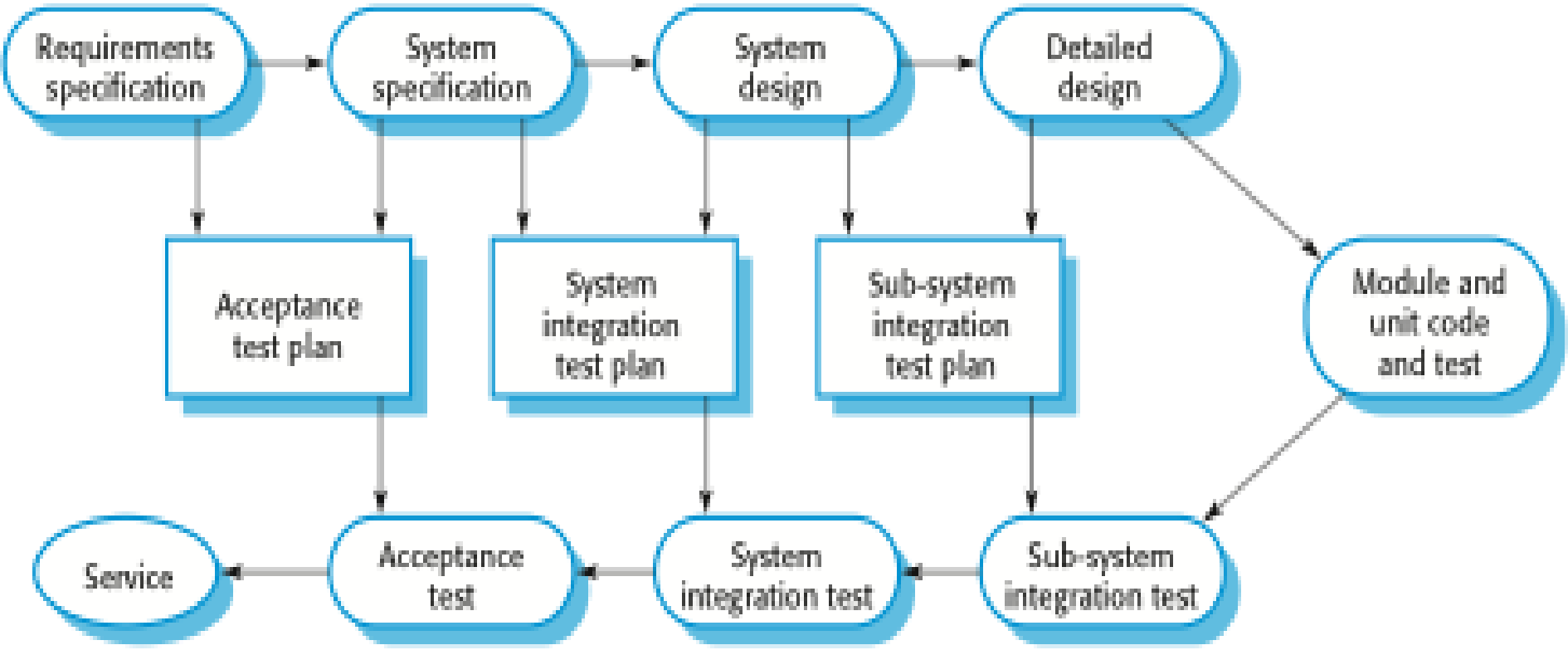
3. Acceptance testing

- Testing with customer data to check that the system meets the customer's needs.

- **اختبار القبول**
- **الاختبار باستخدام بيانات العميل للتحقق من أن النظام يلبي احتياجات العميل.**

Testing phases in a plan-driven software process

مراحل الاختبار في عملية برمجية مدفوعة بالخطة



Software evolution
تطور البرمجيات

✧Software is inherently flexible and can change.

❖ البرمجيات مرنة بطبيعتها ويمكن أن تتغير.

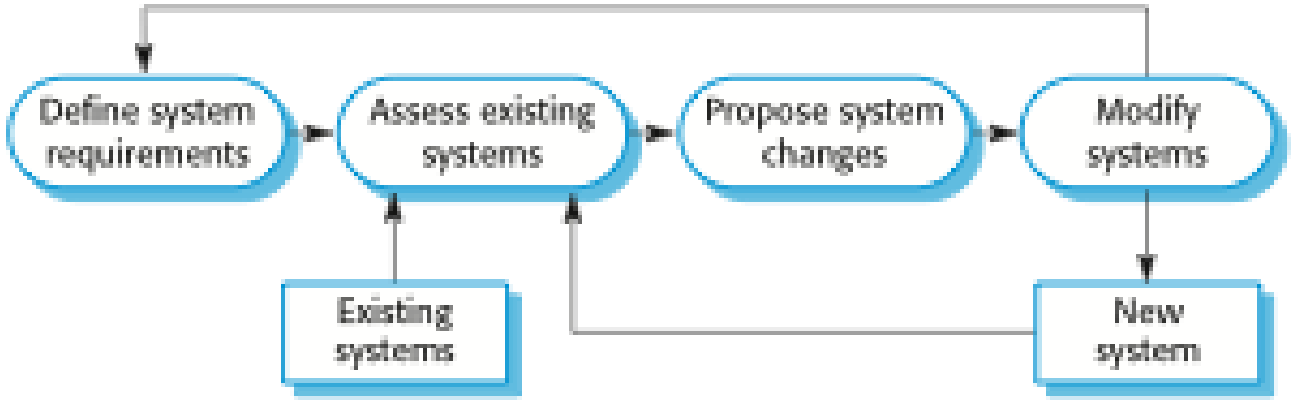
✧As requirements change through changing business circumstances, the software that supports the business must also evolve and change.

❖ مع تغير المتطلبات من خلال ظروف العمل المتغيرة ، يجب أيضًا أن يتطور البرنامج الذي يدعم العمل ويتغير.

✧Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

❖ على الرغم من وجود ترسيم بين التطوير والتطور (الصيانة) ، فإن هذا الأمر غير ذي صلة بشكل متزايد حيث أن عددًا أقل من الأنظمة الجديدة تمامًا.

System evolution
تطور النظام



Key points
النقاط الرئيسية

- ✧Requirements engineering is the process of developing a software specification.

هندسة المتطلبات هي عملية تطوير مواصفات البرنامج.

- ✧Design and implementation processes are concerned with transforming a requirements specification into an executable software system.

تهتم عمليات التصميم والتنفيذ بتحويل مواصفات المتطلبات إلى نظام برمجيات قابل للتنفيذ.

- ✧Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.

التحقق من صحة البرامج هو عملية التحقق من أن النظام يتوافق مع مواصفاته وأنه يلبي الاحتياجات الحقيقية لمستخدمي النظام.

- ✧Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.

يحدث تطور البرامج عندما تقوم بتغيير أنظمة البرامج الحالية لتلبية المتطلبات الجديدة. يجب أن يتطور البرنامج ليظل مفيدًا.

Coping with change
التعامل مع التغيير

✧Change is inevitable in all large software projects.

التغيير أمر لا مفر منه في جميع مشاريع البرامج الكبيرة.

- Business changes lead to new and changed system requirements

تؤدي تغييرات الأعمال إلى متطلبات نظام جديدة ومتغيرة

- New technologies open up new possibilities for improving implementations

تفتح التقنيات الجديدة إمكانيات جديدة لتحسين عمليات التنفيذ

- Changing platforms require application changes

تغيير الأنظمة الأساسية يتطلب تغييرات في التطبيق

✧Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

يؤدي التغيير إلى إعادة العمل ، لذا فإن تكاليف التغيير تشمل إعادة العمل (على سبيل المثال ، متطلبات إعادة التحليل) بالإضافة إلى تكاليف تنفيذ وظائف جديدة

Reducing the costs of rework
تقليل تكاليف إعادة العمل

- **Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.**

  تجنب التغيير ، حيث تتضمن عملية البرنامج أنشطة يمكنها توقع التغييرات المحتملة قبل الحاجة إلى إعادة صياغة مهمة.

  - **For example, a prototype system may be developed to show some key features of the system to customers.**

    على سبيل المثال ، قد يتم تطوير نظام نموذج أولي لإظهار بعض الميزات الرئيسية للنظام للعملاء.

- **Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.**

  تحمل التغيير ، حيث تم تصميم العملية بحيث يمكن استيعاب التغييرات بتكلفة منخفضة نسبيًا.

  - **This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.**

    هذا عادة ما ينطوي على شكل من أشكال التنمية الإضافية. يمكن تنفيذ التغييرات المقترحة في زيادات لم يتم تطويرها بعد. إذا كان هذا مستحيلًا ، فقد يتم تعديل زيادة واحدة فقط (جزء صغير من النظام) لتضمين التغيير.

Software prototyping
نماذج البرمجيات

✧A prototype is an initial version of a system used to demonstrate concepts and try out design options.

النموذج الأولي هو نسخة أولية من نظام يستخدم لشرح المفاهيم وتجربة خيارات التصميم.

✧A prototype can be used in:

يمكن استخدام النموذج الأولي في:

- The requirements engineering process to help with requirements elicitation and validation;

عملية هندسة المتطلبات للمساعدة في استنباط المتطلبات والتحقق من صحتها ؛

- In design processes to explore options and develop a UI design;

في عمليات التصميم لاستكشاف الخيارات وتطوير تصميم واجهة المستخدم ؛

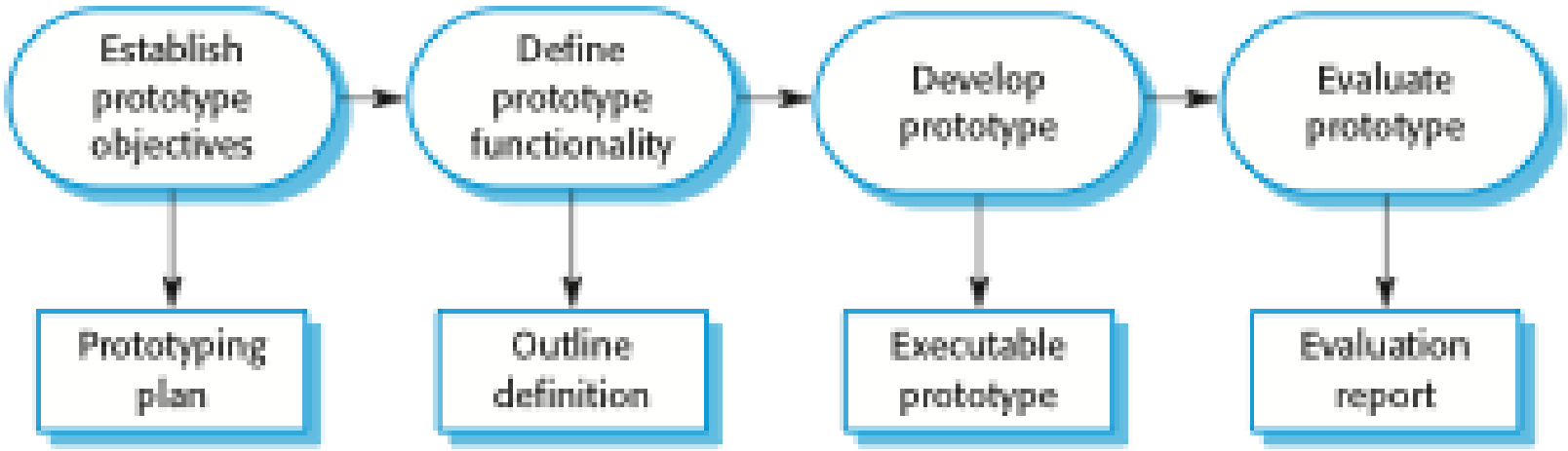
- In the testing process to run back-to-back tests.

في عملية الاختبار لإجراء اختبارات متتالية.

Benefits of prototyping
فوائد النماذج الأولية

- ✧Improved system usability. تحسين قابلية استخدام النظام.
- ✧A closer match to users' real needs. تطابق أوثق مع احتياجات المستخدمين الحقيقية.
- ✧Improved design quality. تحسين جودة التصميم.
- ✧Improved maintainability. تحسين قابلية الصيانة.
- ✧Reduced development effort. انخفاض جهود التنمية.

The process of prototype development
عملية تطوير النموذج الأولي



Prototype development
تطوير النموذج الأولي

✧May be based on rapid prototyping languages or tools

قد يعتمد على لغات أو أدوات النماذج الأولية السريعة

May involve leaving out functionality

قد تنطوي على إهمال الوظيفة

- Prototype should focus on areas of the product that are not well-understood;

يجب أن يركز النموذج الأولي على مناطق المنتج غير المفهومة جيدًا ؛

- Error checking and recovery may not be included in the prototype;

قد لا يتم تضمين التحقق من الأخطاء والاسترداد في النموذج الأولي ؛

- Focus on functional rather than non-functional requirements such as reliability and security

ركز على المتطلبات الوظيفية بدلاً من المتطلبات غير الوظيفية مثل الموثوقية والأمان

Throw-away prototypes
النماذج الأولية التي يمكن التخلص منها

✧Prototypes should be discarded after development as they are not a good basis for a production system:

يجب التخلص من النماذج الأولية بعد التطوير لأنها ليست أساسًا جيدًا لنظام الإنتاج:

- It may be impossible to tune the system to meet non-functional requirements;

قد يكون من المستحيل ضبط النظام لتلبية المتطلبات غير الوظيفية ؛

- Prototypes are normally undocumented;

▪ النماذج الأولية عادة ما تكون غير موثقة ؛

- The prototype structure is usually degraded through rapid change;

عادة ما يتدهور هيكل النموذج الأولي من خلال التغيير السريع ؛

- The prototype probably will not meet normal organisational quality standards.

▪ ربما لن يلبي النموذج الأولي معايير الجودة التنظيمية العادية

Incremental delivery
تسليم تزايدي

✧Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.

بدلاً من تقديم النظام كتسليم منفرد ، يتم تقسيم التطوير والتسليم إلى زيادات مع كل زيادة تقدم جزءًا من الوظيفة المطلوبة.

✧User requirements are prioritised and the highest priority requirements are included in early increments.

يتم ترتيب متطلبات المستخدم حسب الأولوية ويتم تضمين المتطلبات ذات الأولوية القصوى في الزيادات المبكرة.

✧Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

بمجرد بدء تطوير الزيادة ، يتم تجميد المتطلبات على الرغم من أن متطلبات الزيادات اللاحقة يمكن أن تستمر في التطور.

Incremental development and delivery
التطوير والتسليم التدريجي

✧Incremental development التنمية المتزايدة

- Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;

تطوير النظام بزيادات وتقييم كل زيادة قبل الشروع في تطوير الزيادة التالية ؛

- Normal approach used in agile methods;

- النهج العادي المستخدم في الأساليب الرشيقة ؛

- Evaluation done by user/customer proxy.

تم التقييم بواسطة وكيل المستخدم / العميل.

✧Incremental delivery تسليم تزايدي

- Deploy an increment for use by end-users;

نشر زيادة للاستخدام من قبل المستخدمين النهائيين ؛

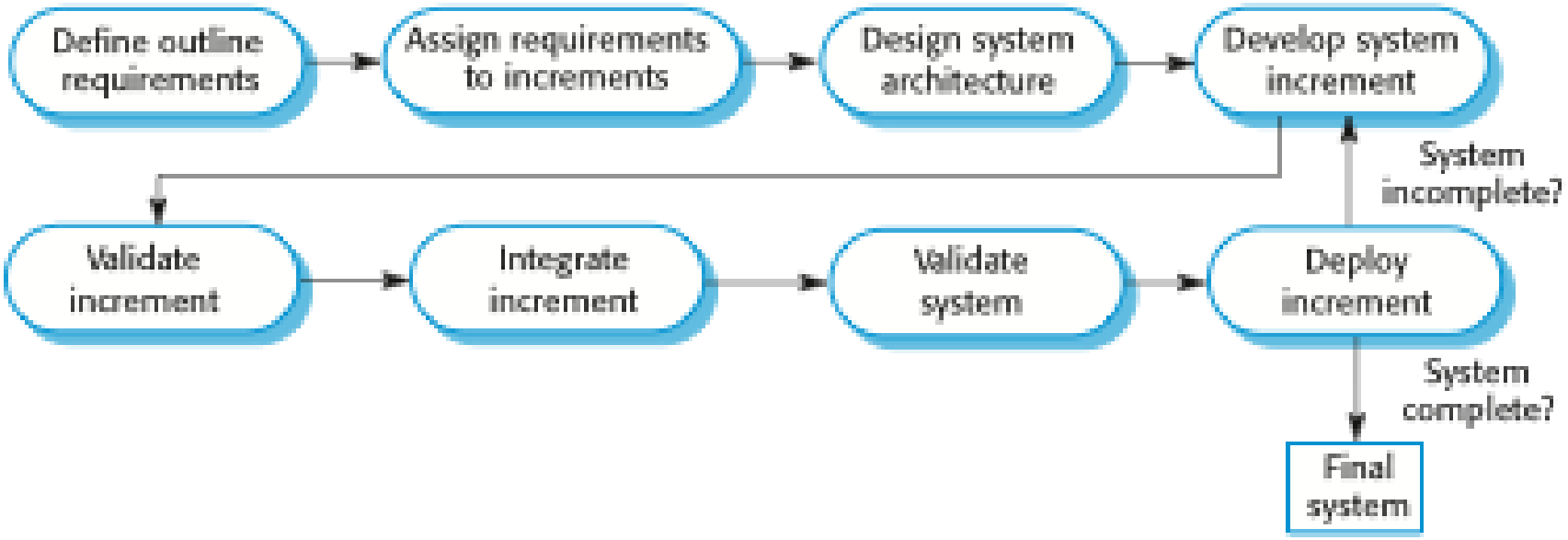
- More realistic evaluation about practical use of software;

تقييم أكثر واقعية حول الاستخدام العملي للبرمجيات ؛

- Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

من الصعب التنفيذ لأنظمة الاستبدال لأن الزيادات لها وظائف أقل من النظام الذي يتم استبداله.

Incremental delivery
تسليم تزايدي



Incremental delivery advantages
مزايا التسليم الإضافية

- ✧Customer value can be delivered with each increment so system functionality is available earlier.

يمكن تسليم قيمة العميل مع كل زيادة حتى تتوفر وظائف النظام مسبقًا.

- ✧Early increments act as a prototype to help elicit requirements for later increments.

تعمل الزيادات المبكرة كنموذج أولي للمساعدة في تحديد متطلبات الزيادات اللاحقة.

- ✧Lower risk of overall project failure.

انخفاض مخاطر فشل المشروع بشكل عام.

- ✧The highest priority system services tend to receive the most testing.

تميل خدمات النظام ذات الأولوية القصوى إلى تلقي معظم الاختبارات.

Incremental delivery problems
مشاكل التسليم المتزايدة

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

تتطلب معظم الأنظمة مجموعة من المرافق الأساسية التي تستخدمها أجزاء مختلفة من النظام.

نظرًا لعدم تحديد المتطلبات بالتفصيل حتى يتم تنفيذ الزيادة ، فقد يكون من الصعب تحديد المرافق المشتركة التي تحتاجها جميع الزيادات.

جوهر العمليات التكرارية هو أن المواصفات تم تطويرها بالتزامن مع البرنامج.

ومع ذلك ، يتعارض هذا مع نموذج الشراء للعديد من المؤسسات ، حيث تكون مواصفات النظام الكاملة جزءًا من عقد تطوير النظام.

Boehm’s spiral model
نموذج بوم الحلزوني

- ✧Process is represented as a spiral rather than as a sequence of activities with backtracking.

يتم تمثيل العملية على أنها دوامة وليس كسلسلة من الأنشطة مع التراجع.

- ✧Each loop in the spiral represents a phase in the process.

تمثل كل حلقة في اللولب مرحلة في العملية.

- ✧No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.

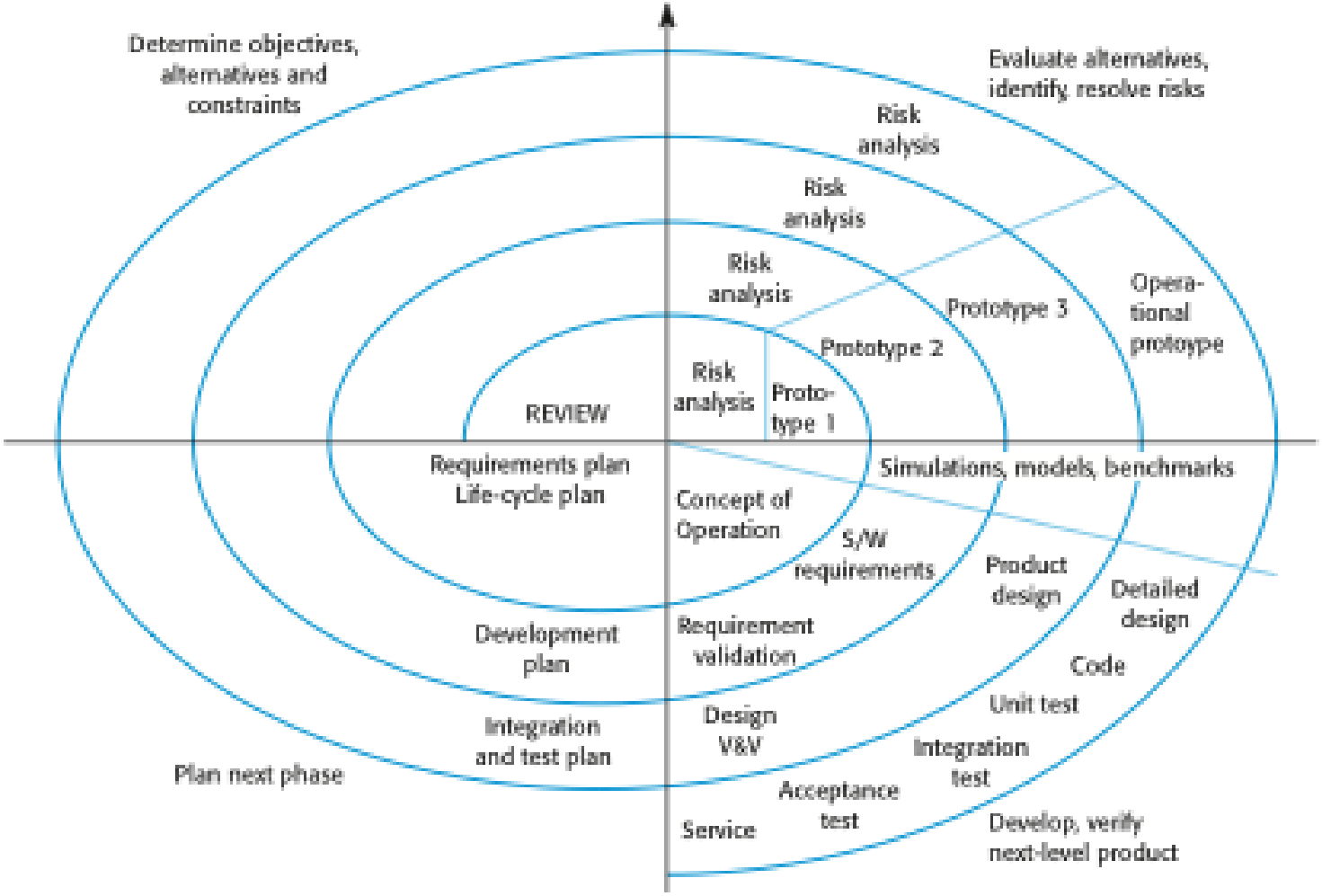
لا توجد مراحل ثابتة مثل المواصفات أو التصميم - يتم اختيار الحلقات في اللولب اعتمادًا على ما هو مطلوب.

- ✧Risks are explicitly assessed and resolved throughout the process.

يتم تقييم المخاطر بشكل صريح وحلها طوال العملية.

Boehm's spiral model of the software process

<> نموذج حلزوني لعملية البرنامج



Spiral model sectors
قطاعات النموذج الحلزوني

✧Objective setting

- Specific objectives for the phase are identified. Constraints on the process and the product are identified ,A detailed management plan is drawn up. Projects risks are identified

تحديد الأهداف

تم تحديد الأهداف المحددة للمرحلة. يتم تحديد القيود على العملية والمنتج ، يتم وضع خطة إدارة مفصلة. تحديد مخاطر المشاريع

✧Risk assessment and reduction

- Risks are assessed and activities put in place to reduce the key risks.

تقييم المخاطر والحد منها

يتم تقييم المخاطر وتنفيذ الأنشطة للحد من المخاطر الرئيسية.

✧Development and validation

- A development model for the system is chosen which can be any of the generic models.

التطوير والتحقق

يتم اختيار نموذج تطوير للنظام يمكن أن يكون أيًا من النماذج العامة.

✧Planning

- The project is reviewed and the next phase of the spiral is planned.

تخطيط

تتم مراجعة المشروع ويتم التخطيط للمرحلة التالية من الحلزون.

Spiral model usage
استخدام النموذج الحلزوني

✧Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.

كان للنموذج الحلزوني تأثير كبير في مساعدة الناس على التفكير في التكرار في عمليات البرمجيات وإدخال نهج يحركه المخاطر في التنمية.

✧In practice, however, the model is rarely used as published for practical software development.

في الممارسة العملية ، ومع ذلك ، نادرًا ما يتم استخدام النموذج كما هو منشور لتطوير البرامج العملية.

The Rational Unified Process
العملية الموحدة العقلانية

✧A modern generic process derived from the work on the **{UML}** and associated process.

عملية عامة حديثة مشتقة من العمل على {} والعملية المرتبطة بها.

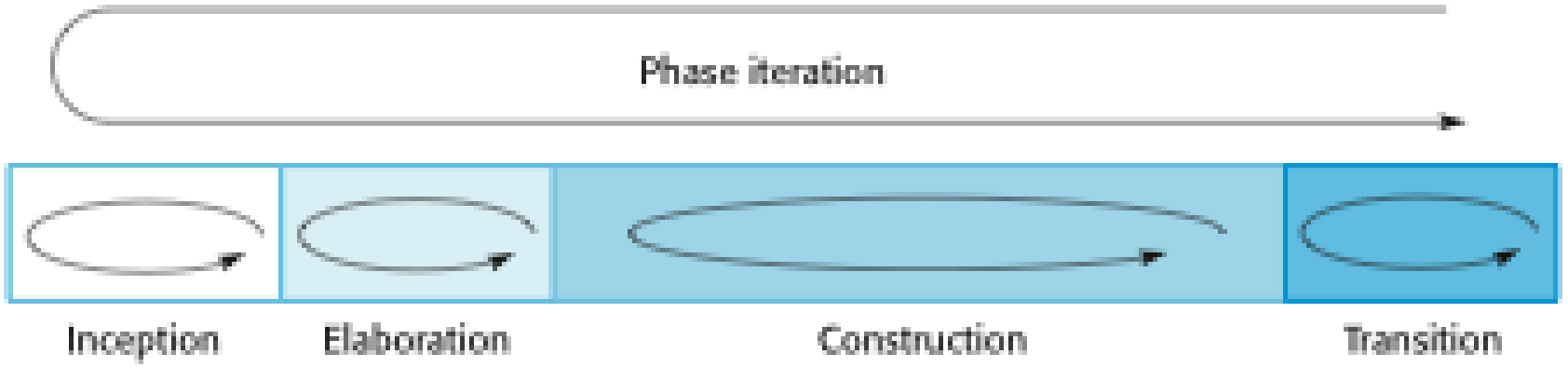
✧Brings together aspects of the 3 generic process models discussed previously.

يجمع بين جوانب نماذج العمليات العامة الثلاثة التي تمت مناقشتها سابقًا.

✧**Normally described from 3 perspectives** **يوصف عادة من 3 وجهات نظر**

- **A dynamic perspective** that shows **phases over time;**
  منظور ديناميكي يظهر مراحل بمرور الوقت ؛
- **A static perspective** that shows **process activities**;
  منظور ثابت يوضح أنشطة العملية ؛
- **A proactive** perspective that **suggests good practice**.
  منظور استباقي يقترح الممارسة الجيدة.

Phases in the Rational Unified Process
مراحل في العملية الموحدة العقلانية



RUP phases
مراحلRUP

✧Inception بداية

- Establish the business case for the system. identify all external entities (people and systems) that will interact with the system and define these interactions.

إنشاء دراسة الجدوى للنظام. تحديد جميع الكيانات الخارجية (الأشخاص والأنظمة) التي ستتفاعل مع النظام وتحدد هذه التفاعلات.

✧Elaboration التفصيل

- Develop an understanding of the problem domain and the system architecture.

تطوير فهم لمجال المشكلة وبنية النظام.

✧Construction بناء

- System design, programming and testing.

تصميم النظام وبرمجته واختباره.

✧Transition انتقال

- Deploy the system in its operating environment.

- نشر النظام في بيئة التشغيل الخاصة به.

RUP iteration
تكرارRUP

✧In-phase iteration التكرار في المرحلة

- Each phase is iterative with results developed incrementally.

كل مرحلة تكرارية مع تطوير النتائج تدريجياً.

✧Cross-phase iteration التكرار عبر المراحل

- As shown by the loop in the {} model, the whole set of phases may be enacted incrementally.

كما هو موضح من خلال الحلقة في نموذج {} ، قد يتم تفعيل مجموعة المراحل بالكامل بشكل تدريجي.

Static workflows in the Rational Unified Process
سير العمل الثابت في العملية الموحدة العقلانية

Workflow سير العمل	Description وصف
Business modelling نمذجة الأعمال	The business processes are modelled using business use cases..يتم تصميم عمليات الأعمال باستخدام حالات استخدام الأعمال
Requirements متطلبات	Actors who interact with the system are identified and use cases are developed to model the system requirements. يتم تحديد الجهات الفاعلة التي تتفاعل مع النظام ويتم تطوير حالات الاستخدام لنمذجة متطلبات النظام.
Analysis and design التحليل والتصميم	A design model is created and documented using architectural models, component models, object models and sequence models. يتم إنشاء وتوثيق نموذج التصميم باستخدام النماذج المعمارية ونماذج المكونات ونماذج الكائنات ونماذج التسلسل.
Implementation تطبيق	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process. يتم تنفيذ مكونات النظام وتنظيمها في أنظمة فرعية للتنفيذ. يساعد إنشاء الكود التلقائي من نماذج التصميم في تسريع هذه العملية.

Static workflows in the Rational Unified Process

Workflow	Description
Testing اختبارات	**Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.** الاختبار هو عملية تكرارية يتم تنفيذها بالتزامن مع التنفيذ. يتبع اختبار النظام الانتهاء من التنفيذ.
Deployment تطوير	**A product release is created, distributed to users and installed in their workplace.** يتم إنشاء إصدار منتج وتوزيعه على المستخدمين وتثبيته في أماكن عملهم.
Configuration and change management التكوين وإدارة التغيير	**This supporting workflow managed changes to the system (see Chapter 25).** يدير سير العمل الداعم هذا تغييرات على النظام (انظر الفصل 25).
Project management ادارة مشروع	**This supporting workflow manages the system development (see Chapters 22 and 23).** يدير سير العمل الداعم هذا تطوير النظام (انظر الفصلين 22 و 23).
Environment بيئة	**This workflow is concerned with making appropriate software tools available to the software development team.** يهتم سير العمل هذا بإتاحة أدوات البرامج المناسبة لفريق تطوير البرامج

RUP good practice
الممارسات الجيدةRUP

✧**Develop software iteratively** تطوير البرامج بشكل متكرر

- **Plan increments based on customer priorities and deliver highest priority increments first.**
- قم بتخطيط الزيادات بناءً على أولويات العميل وتقديم الزيادات ذات الأولوية القصوى أولاً.

✧**Manage requirements** إدارة المتطلبات

- **Explicitly document customer requirements and keep track of changes to these requirements.**

✧ قم بتوثيق متطلبات العملاء بشكل صريح وتتبع التغييرات التي تطرأ على هذه المتطلبات.

✧**Use component-based architectures** استخدم البنى القائمة على المكونات

- **Organize the system architecture as a set of reusable components.**
- تنظيم بنيةالنظام كمجموعة من المكونات التي يمكن إعادة استخدامها.

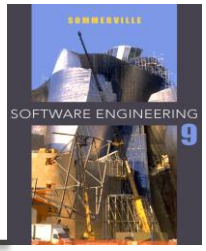
RUP good practice
الممارسات الجيدةRUP

- **Visually model software** برمجيات النموذج البصري
  - **Use graphical UML models to present static and dynamic views of the software.** () استخدم نماذج رسومية لتقديم عروض ثابتة وديناميكية للبرنامج.
- **Verify software quality** تحقق من جودة البرنامج
  - **Ensure that the software meet's organizational quality standards.**
    - تأكد من أن البرنامج يلبي معايير الجودة التنظيمية.
- **Control changes to software** تغييرات التحكم في البرامج
  - **Manage software changes using a change management system and configuration management tools.**
    - إدارة تغييرات البرامج باستخدام نظام إدارة التغيير وأدوات إدارة التكوين.

Key points

- ✧Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design.
- ✧ يجب أن تتضمن العمليات أنشطة للتعامل مع التغيير. قد يتضمن ذلك مرحلة النماذج الأولية التي تساعد على تجنب القرارات السيئة بشأن المتطلبات والتصميم.
- ✧Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
- ✧ قد يتم تنظيم العمليات للتطوير والتسليم التكراري بحيث يمكن إجراء التغييرات دون تعطيل النظام ككل.
- ✧The Rational Unified Process is a modern generic process model that is organized into phases (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.

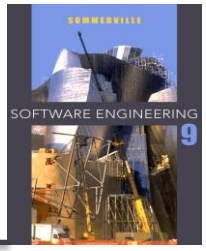
العملية الموحدة العقلانية هي نموذج عملية عام حديث يتم تنظيمه في مراحل (البداية ، والتوضيح ، والبناء ، والانتقال) ولكنه يفصل الأنشطة (المتطلبات والتحليل والتصميم ، وما إلى ذلك) عن هذه المراحل



Chapter 3 – Agile Software Development

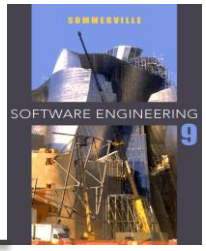
Lecture 1

Topics covered



- ✧ Agile methods
- ✧ Plan-driven and agile development
- ✧ Extreme programming
- ✧ Agile project management
- ✧ Scaling agile methods

Rapid software development



- Rapid development and delivery is now often the most important requirement for software systems

  - يعد التطوير والتسليم السريع الآن أهم متطلبات أنظمة البرامج

  - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
    - تعمل الشركات في متطلبات سريعة التغير ومن المستحيل عمليا إنتاج مجموعة من متطلبات البرامج الثابتة
  - Software has to evolve quickly to reflect changing business needs.
    - يجب أن يتطور البرنامج بسرعة ليعكس احتياجات العمل المتغيرة.

- Rapid software development التطوير السريع للبرامج

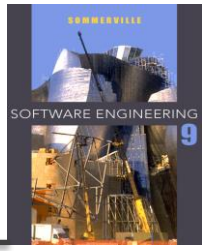
  - Specification, design and implementation are inter-leaved

    المواصفات والتصميم والتنفيذ متداخلة

  - System is developed as a series of versions with stakeholders involved in version evaluation
    - تم تطوير النظام كسلسلة من الإصدارات مع أصحاب المصلحة المشاركين في تقييم الإصدار
  - User interfaces are often developed using an IDE and graphical toolset.

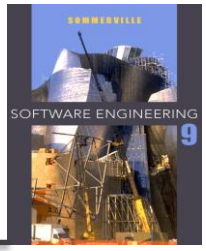
    غالبًا ما يتم تطوير واجهات المستخدم باستخدام <> ومجموعة أدوات رسومية.

Agile methods



- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s
- **led to the creation of agile methods. These methods:**
  - **Focus on the code rather than the design** ركز على الكود بدلاً من التصميم
  - **Are based on an iterative approach to software development**
    - تستند إلى نهج تكراري لتطوير البرمجيات
  - **Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.**
    - تهدف إلى تقديم برامج العمل بسرعة وتطويرها بسرعة لتلبية المتطلبات المتغيرة.
- **The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without too much rework.**
- الهدف من الأساليب الرشيقة هو تقليل النفقات العامة في عملية البرنامج (على سبيل المثال عن طريق الحد من التوثيق) والقدرة على الاستجابة بسرعة للمتطلبات المتغيرة دون الحاجة إلى الكثير من إعادة العمل.

Agile manifesto

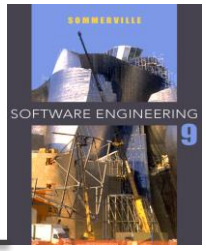


- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
  - *Individuals and interactions over processes and tools*
    Working software over comprehensive documentation
    Customer collaboration over contract negotiation
    Responding to change over following a plan
- *That is, while there is value in the items on the right, we value the items on the left more.*

The principles of agile methods
مبادئ الأساليب الرشيقة

Principle مبدأ	Description وصف
Customer involvement مشاركة العملاء	**Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.** يجب أن يشارك العملاء عن كثب في جميع مراحل عملية التطوير. دورهم هو توفير وتحديد أولويات متطلبات النظام الجديدة وتقييم التكرارات للنظام.
Incremental delivery تسليم تزايدي	**The software is developed in increments with the customer specifying the requirements to be included in each increment.** تم تطوير البرنامج بزيادات مع تحديد العميل للمتطلبات التي سيتم تضمينها في كل زيادة.
People not process	**The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.** يجب التعرف على مهارات فريق التطوير واستغلالها. يجب ترك أعضاء الفريق لتطوير طرقهم الخاصة في العمل دون عمليات إلزامية.
Embrace change	**Expect the system requirements to change and so design the system to accommodate these changes.** توقع أن تتغير متطلبات النظام ، ولذا صمم النظام ليتوافق مع هذه التغييرات.
Maintain simplicity حافظ على البساطة	**Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.** ركز على البساطة في كل من البرامج التي يتم تطويرها وفي عملية التطوير. حيثما أمكن ، اعمل

Agile method applicability
قابلية تطبيق الطريقة الرشيقة

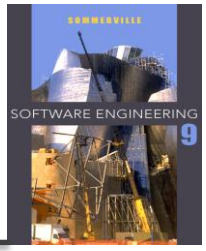


- Product development where a software company is developing a small or medium-sized product for sale.
- تطوير المنتج حيث تقوم شركة برمجيات بتطوير منتج صغير أو متوسط الحجم للبيع.

- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- تطوير نظام مخصص داخل منظمة ، حيث يوجد التزام واضح من العميل بالمشاركة في عملية التطوير وحيث لا يوجد الكثير من القواعد واللوائح الخارجية التي تؤثر على البرنامج.

- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.
- بسبب تركيزهم على الفرق الصغيرة والمتكاملة بإحكام ، هناك مشاكل في توسيع نطاق الأساليب الرشيقة للأنظمة الكبيرة.

Problems with agile methods
مشاكل مع الأساليب الرشيقة



- It can be difficult to keep the interest of customers who are involved in the process.

قد يكون من الصعب الحفاظ على اهتمام العملاء المشاركين في العملية.

- Team members may be unsuited to the intense involvement that characterises agile methods.

- قد يكون أعضاء الفريق غير مناسبين للمشاركة المكثفة التي تميز الأساليب الرشيقة.

- Prioritising changes can be difficult where there are multiple stakeholders.

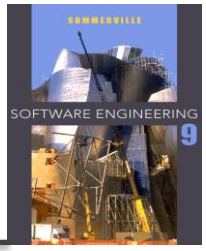
- قد يكون تحديد أولويات التغييرات أمرًا صعبًا في حالة وجود العديد من أصحاب المصلحة.

- Maintaining simplicity requires extra work.

يتطلب الحفاظ على البساطة عملاً إضافيًا.

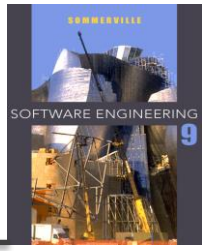
- Contracts may be a problem as with other approaches to iterative development. قد تكون العقود مشكلة كما هو الحال مع الأساليب الأخرى للتطوير التكراري.

Agile methods and software maintenance
الأساليب الرشيقة وصيانة البرامج



- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues: قضيتان رئيسيتان:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
    - هل الأنظمة التي تم تطويرها باستخدام نهج رشيق قابلة للصيانة ، بالنظر إلى التركيز في عملية التطوير لتقليل التوثيق الرسمي؟
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
    - هل يمكن استخدام الأساليب الرشيقة بشكل فعال لتطوير نظام استجابة لطلبات تغيير العملاء؟
- Problems may arise if original development team cannot be maintained
  قد تنشأ مشاكل إذا تعذر الحفاظ على فريق التطوير الاصلي

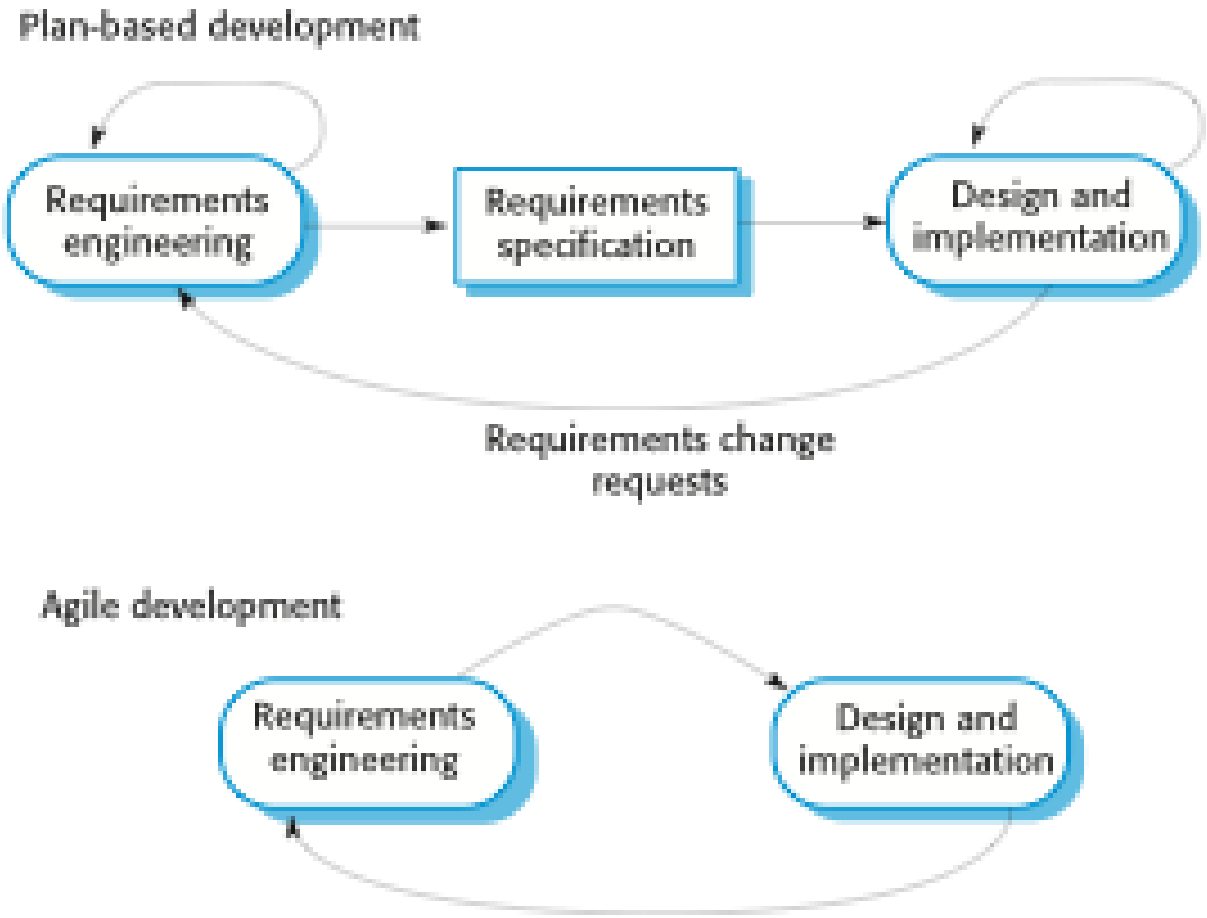
Plan-driven and agile development
التنمية المدفوعة بالخطة ورشيقة



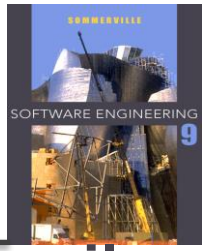
- **Plan-driven development** التنمية المدفوعة بالخطة
  - **A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.**

    يعتمد النهج القائم على الخطة في هندسة البرمجيات على مراحل تطوير منفصلة مع إنتاج المخرجات في كل مرحلة من هذه المراحل المخطط لها مسبقًا.
  - **Not necessarily waterfall model – plan-driven, incremental development is possible** ليس بالضرورة أن يكون نموذج الشلال - التطوير التدريجي المبني على الخطة ممكن
  - **Iteration occurs within activities.** يحدث التكرار في الأنشطة.
- **Agile development** تطوير البرامج بتقنية أجيل
  - **Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.**
    - المواصفات والتصميم والتنفيذ والاختبار متداخلة ويتم تحديد مخرجات عملية التطوير من خلال عملية التفاوض أثناء عملية تطوير البرمجيات.

Plan-driven and agile specification



Technical, human, organizational issues



- **Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:**

- تتضمن معظم المشاريع عناصر من عمليات مدعومة بخطة ورشيقة. يعتمد تحديد الرصيد على:

  - **Is it important to have a very detailed specification and design before moving to implementation?**

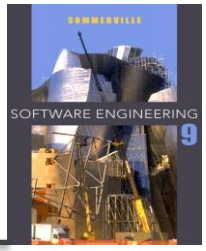
If so, you probably need to use a plan-driven approach.

  - هل من المهم أن يكون لديك مواصفات وتصميم مفصلان للغاية قبل الانتقال إلى التنفيذ؟ إذا كان الأمر كذلك ، فربما تحتاج إلى استخدام نهج مبني على الخطة.

  - **Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic?**

If so, consider using agile methods.

  - هل استراتيجية التسليم التدريجي ، حيث تقدم البرنامج للعملاء وتحصل على ردود فعل سريعة منهم ، واقعية؟ إذا كان الأمر كذلك ، ففكر في استخدام الأساليب الرشيقة.

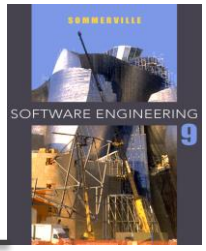


- **How large is the system that is being developed?**

- **Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.**

- **ما هو حجم النظام الذي يتم تطويره؟ تكون الأساليب الرشيقة أكثر فاعلية عندما يمكن تطوير النظام مع فريق صغير مشترك في الموقع يمكنه التواصل بشكل غير رسمي. قد لا يكون هذا ممكنًا للأنظمة الكبيرة التي تتطلب فرق تطوير أكبر ، لذا قد يتعين استخدام نهج مبني على الخطة.**

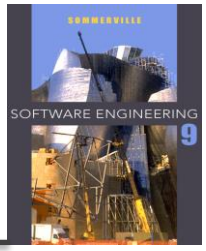
Technical, human, organizational issues



- What type of system is being developed? ما هو نوع النظام الذي يتم تطويره؟
  - Plan-driven approaches may be required for systems that require a lot of analysis before implementation (e.g. real-time system with complex timing requirements).
    - قد تكون المناهج التي تعتمد على الخطة مطلوبة للأنظمة التي تتطلب الكثير من التحليل قبل التنفيذ (مثل نظام الوقت الحقيقي بمتطلبات توقيت معقدة).
- What is the expected system lifetime? ما هو العمر المتوقع للنظام؟
  - Long-lifetime systems may require more design documentation.

    قد تتطلب أنظمة العمر الطويل المزيد من وثائق التصميم.
- What technologies are available to support system development?

  - ما هي التقنيات المتاحة لدعم تطوير النظام؟
  - Agile methods rely on good tools to keep track of an evolving design
    - تعتمد الأساليب الرشيقة على أدوات جيدة لتتبع التصميم المتطور



- **How is the development team organized?**
- **كيف يتم تنظيم فريق التطوير؟**

  If the development team is distributed or if part of the development is being outsourced, then you may need to develop design documents to communicate across the development teams.

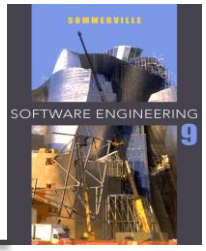
  - **إذا تم توزيع فريق التطوير أو إذا تم الاستعانة بمصادر خارجية لجزء من التطوير ، فقد تحتاج إلى تطوير مستندات التصميم للتواصل عبر فرق التطوير.**
- **Is the system subject to external regulation?**

  هل النظام خاضع للتنظيم الخارجي؟

  If a system has to be approved by an external regulator (e.g. the FAA approve software that is critical to the operation of an aircraft) then you will probably be required to produce detailed documentation as part of the system safety case.

  إذا كان لا بد من اعتماد نظام من قبل منظم خارجي (مثل) (الموافقة على البرنامج الذي يعد أمرًا بالغ الأهمية لتشغيل الطائرة) ، فمن المحتمل أن يُطلب منك تقديم وثائق مفصلة كجزء من حالة سلامة النظام.

Technical, human, organizational issues



- Are there cultural or organizational issues that may affect the system development?
  - هل هناك قضايا ثقافية أو تنظيمية قد تؤثر على تطوير النظام؟
    - Traditional engineering organizations have a culture of plan-based development, as this is the norm in engineering.

تتمتع المنظمات الهندسية التقليدية بثقافة التطوير المستند إلى الخطة ، حيث أن هذا هو المعيار في الهندسة.

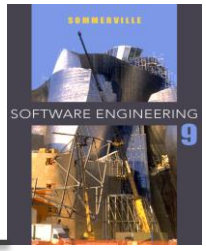
- How good are the designers and programmers in the development team?

ما مدى جودة المصممين والمبرمجين في فريق التطوير؟

- It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code

- يُقال أحيانًا أن الأساليب الرشيقة تتطلب مستويات مهارة أعلى من الأساليب القائمة على الخطة والتي يقوم فيها المبرمجون ببساطة بترجمة التصميم التفصيلي إلى رمز

Extreme programming



- Perhaps the best-known and most widely used agile method..ربما يكون الأسلوب الرشيق الأكثر شهرة والأكثر استخدامًا
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.

تأخذ البرمجة المتطرفة () نهجًا "متطرفًا" في التطوير التكراري.

  - New versions may be built several times per day;

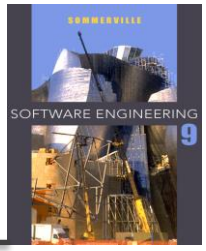
يمكن إنشاء إصدارات جديدة عدة مرات في اليوم ؛

  - Increments are delivered to customers every 2 weeks;

يتم تسليم الزيادات للعملاء كل أسبوعين ؛

  - All tests must be run for every build and the build is only accepted if tests run successfully.
    - يجب تشغيل جميع الاختبارات لكل بناء ولا يتم قبول الإصدار إلا إذا تم إجراء الاختبارات بنجاح.

XP and agile principles



- Incremental development is supported through small, frequent system releases.

يتم دعم التطوير المتزايد من خلال إصدارات النظام الصغيرة والمتكررة.

- Customer involvement means full-time customer engagement with the team.

- يعني إشراك العملاء مشاركة العملاء بدوام كامل مع الفريق.

- People not process through pair programming, collective ownership and a process that avoids long working hours.

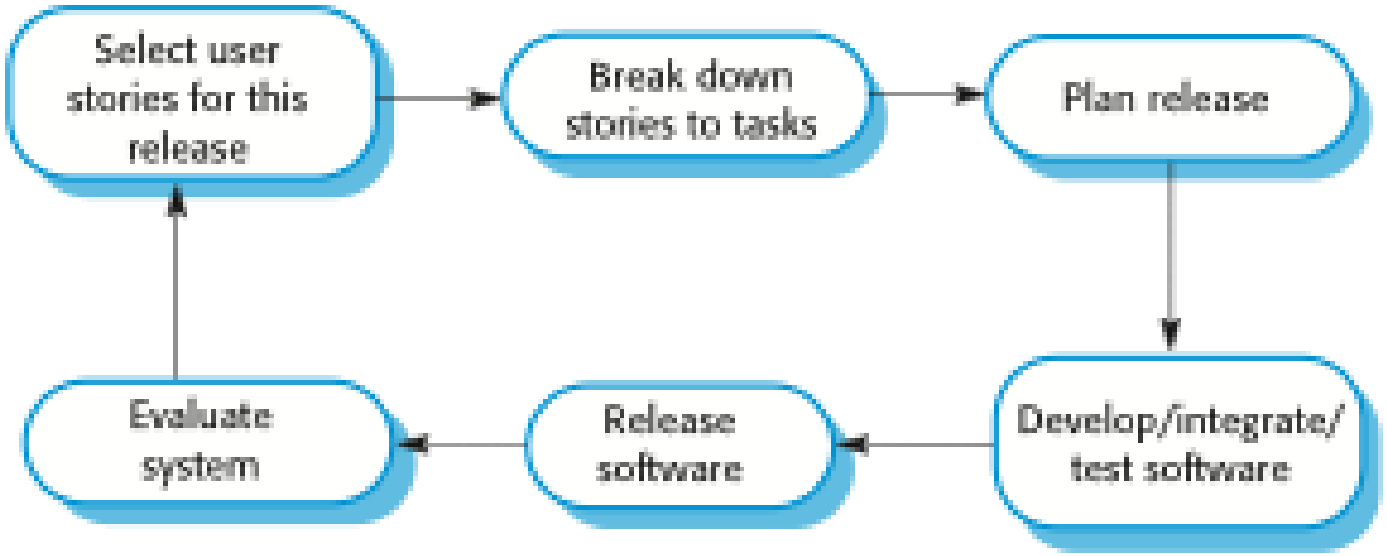
لا يعالج الأشخاص من خلال البرمجة الزوجية والملكية الجماعية والعملية التي تتجنب ساعات العمل الطويلة.

- Change supported through regular system releases.

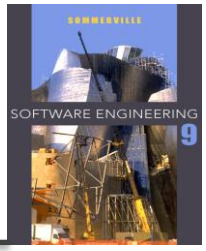
دعم التغيير من خلال إصدارات النظام العادية.

- Maintaining simplicity through constant refactoring of code. الحفاظ على البساطة من خلال إعادة هيكلة الكود باستمرار.

The extreme programming release cycle



Requirements scenarios



- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.

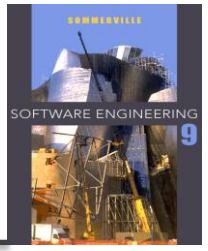
في () يكون العميل أو المستخدم جزءًا من () الفريق ومسؤول عن اتخاذ القرارات بشأن المتطلبات.

- User requirements are expressed as scenarios or user stories.
- يتم التعبير عن متطلبات المستخدم في صورة سيناريوهات أو قصص مستخدم.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.

هذه مكتوبة على بطاقات ويقوم فريق التطوير بتقسيمها إلى مهام تنفيذ. هذه المهام هي أساس الجدول الزمني وتقديرات التكلفة.

- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.
- يختار العميل القصص لإدراجها في الإصدار التالي بناءً على أولوياته وتقديرات الجدول الزمني.

A ‘prescribing medication’ story



Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either ‘current medication’, ‘new medication’ or ‘formulary’.

If you select ‘current medication’, you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

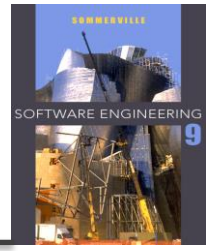
If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose ‘formulary’, you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click ‘OK’ or ‘Change’. If you click ‘OK’, your prescription will be recorded on the audit database. If you click ‘Change’, you reenter the ‘Prescribing medication’ process.

Examples of task cards for prescribing medication



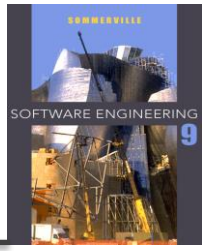
Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Refactoring إعادة بناء التعليمات البرمجية



- Programming team look for possible software improvements and make these improvements even where there is no immediate need for them.
- يبحث فريق البرمجة عن تحسينات البرامج الممكنة وإجراء هذه التحسينات حتى في حالة عدم وجود حاجة فورية لها.
- This improves the understandability of the software and so reduces the need for documentation.

يؤدي ذلك إلى تحسين إمكانية فهم البرنامج وبالتالي تقليل الحاجة إلى التوثيق.

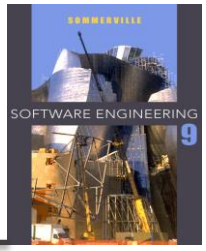
- Changes are easier to make because the code is well-structured and clear.

من الأسهل إجراء التغييرات لأن الكود منظم جيدًا وواضح.

- However, some changes requires architecture refactoring and this is much more expensive.

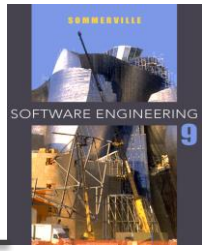
ومع ذلك ، تتطلب بعض التغييرات إعادة هيكلة العمارة وهذا أغلى بكثير.

Examples of refactoring



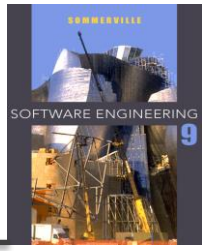
- Re-organization of a class hierarchy to remove duplicate code.
  - إعادة تنظيم التسلسل الهرمي للفئة لإزالة التعليمات البرمجية المكررة.
- Tidying up and renaming attributes and methods to make them easier to understand.
  - ترتيب السمات والطرق وإعادة تسميتها لتسهيل فهمها.
- The replacement of inline code with calls to methods that have been included in a program library.
  - استبدال الكود المضمن باستدعاءات للطرق التي تم تضمينها في مكتبة البرنامج.

Pair programming



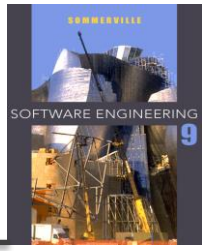
- In XP, programmers work in pairs, sitting together to develop code.
- في [] ، يعمل المبرمجون في أزواج ، ويجلسون معًا لتطوير التعليمات البرمجية.
- This helps develop common ownership of code and spreads knowledge across the team.
- يساعد هذا في تطوير الملكية المشتركة للكود ونشر المعرفة عبر الفريق.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- إنها بمثابة عملية مراجعة غير رسمية حيث يتم النظر إلى كل سطر من التعليمات البرمجية من قبل أكثر من شخص واحد.
- It encourages refactoring as the whole team can benefit from this.
- يشجع إعادة البناء حيث يمكن للفريق بأكمله الاستفادة من ذلك.
- Measurements suggest that development productivity with pair programming is similar to that of two people working independently.
- تشير القياسات إلى أن إنتاجية التطوير مع البرمجة الزوجية تشبه إنتاجية شخصين يعملان بشكل مستقل.

Pair programming



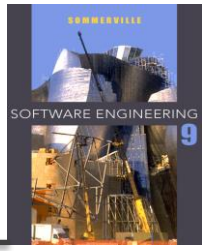
- In pair programming, programmers sit together at the same workstation to develop the software.
- في البرمجة الزوجية ، يجلس المبرمجون معًا في نفس محطة العمل لتطوير البرنامج.
- Pairs are created dynamically so that all team members work with each other during the development process.
- يتم إنشاء الأزواج بشكل ديناميكي بحيث يعمل جميع أعضاء الفريق مع بعضهم البعض أثناء عملية التطوير.
- The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.
- تعد مشاركة المعرفة التي تحدث أثناء البرمجة الزوجية مهمة للغاية لأنها تقلل من المخاطر الكلية للمشروع عندما يغادر أعضاء الفريق.
- Pair programming is not necessarily inefficient and there is evidence that a pair working together is more efficient than 2 programmers working separately.
- البرمجة الزوجية ليست بالضرورة غير فعالة وهناك دليل على أن عمل الزوج معًا يكون أكثر كفاءة من عمل مبرمجين منفصلين.

Advantages of pair programming



- It supports the idea of collective ownership and responsibility for the system.
- وهو يدعم فكرة الملكية الجماعية والمسؤولية عن النظام.
  - Individuals are not held responsible for problems with the code. Instead, the team has collective responsibility for resolving these problems.
    - لا يتحمل الأفراد المسؤولية عن مشاكل الكود. بدلاً من ذلك ، يتحمل الفريق مسؤولية جماعية لحل هذه المشكلات.
- It acts as an informal review process because each line of code is looked at by at least two people.
- إنها بمثابة عملية مراجعة غير رسمية لأن كل سطر من التعليمات البرمجية يتم النظر إليه من قبل شخصين على الأقل.
- It helps support refactoring, which is a process of software improvement.
  - يساعد في دعم إعادة البناء ، وهي عملية تحسين البرامج.
  - Where pair programming and collective ownership are used, others benefit immediately from the refactoring so they are likely to support the process.
  - عند استخدام البرمجة الزوجية والملكية الجماعية ، يستفيد الآخرون على الفور من إعادة البناء ، لذا من المرجح أن يدعموا العملية.

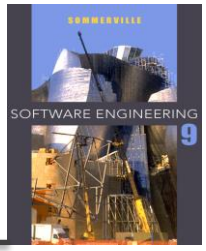
Scaling agile methods



- Agile methods have proved to be successful for small and medium sized projects that can be developed by a small co-located team.
- أثبتت الأساليب الرشيقة نجاحها للمشاريع الصغيرة والمتوسطة الحجم التي يمكن تطويرها بواسطة فريق صغير مشترك في الموقع.
- success of Agile methods comes because of improved communications which is possible when everyone is working together.
- يأتي نجاح أساليب أجايل بسبب تحسين الاتصالات وهو أمر ممكن عندما يعمل الجميع معًا.
- Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations.

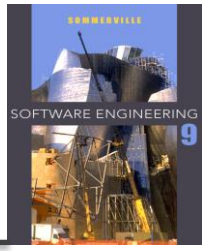
يتضمن توسيع نطاق الأساليب الرشيقة تغييرها للتعامل مع المشاريع الأكبر والأطول حيث توجد فرق تطوير متعددة ، وربما تعمل في مواقع مختلفة.

Large systems development



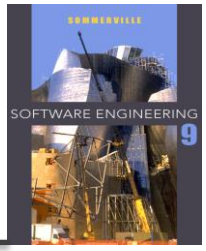
- Large systems are usually collections of separate, communicating systems, where separate teams develop each system. Frequently, these teams are working in different places, sometimes in different time zones.
- عادة ما تكون الأنظمة الكبيرة عبارة عن مجموعات من أنظمة اتصال منفصلة ، حيث تقوم فرق منفصلة بتطوير كل نظام. في كثير من الأحيان ، تعمل هذه الفرق في أماكن مختلفة ، وأحيانًا في مناطق زمنية مختلفة.
- Large systems are (**Brownfield systems**)', that is they include and interact with a number of existing systems. Many of the system requirements are concerned with this interaction and so don't really lend themselves to flexibility and incremental development.
- الأنظمة الكبيرة هي () "، أي أنها تشمل وتتفاعل مع عدد من الأنظمة الحالية. تهتم العديد من متطلبات النظام بهذا التفاعل ولذلك لا تخضع للمرونة والتطور التدريجي حقًا.
- Where several systems are integrated to create a system, a significant fraction of the development is concerned with system configuration rather than original code development.
- عندما يتم دمج العديد من الأنظمة لإنشاء نظام ، فإن جزءًا كبيرًا من التطوير يهتم بتكوين النظام بدلاً من تطوير الكود الأصلي.

Large system development



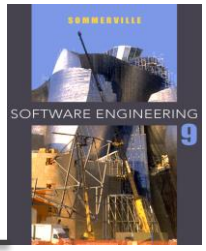
- Large systems and their development processes are often constrained by external rules and regulations limiting the way that they can be developed.
- غالبًا ما تكون الأنظمة الكبيرة وعمليات تطويرها مقيدة بالقواعد واللوائح الخارجية التي تحد من الطريقة التي يمكن بها تطويرها.
- Large systems have a long procurement and development time. It is difficult to maintain coherent teams who know about the system over that period as, inevitably, people move on to other jobs and projects.
- تتمتع الأنظمة الكبيرة بوقت شراء وتطوير طويل. من الصعب الحفاظ على فرق متماسكة على دراية بالنظام خلال تلك الفترة ، حيث ينتقل الأشخاص حتمًا إلى وظائف ومشاريع أخرى.
- Large systems usually have a diverse set of stakeholders. It is practically impossible to involve all of these different stakeholders in the development process.
- عادة ما يكون للأنظمة الكبيرة مجموعة متنوعة من أصحاب المصلحة. من المستحيل عمليا إشراك جميع أصحاب المصلحة المختلفين في عملية التنمية.

Scaling out and scaling up



- ‘Scaling up’ is concerned with using agile methods for developing large software systems that cannot be developed by a small team.
- توسيع نطاق "يهتم باستخدام أساليب رشيقة لتطوير أنظمة البرمجيات الكبيرة التي لا يمكن تطويرها من قبل فريق صغير.
- ‘Scaling out’ is concerned with how agile methods can be introduced across a large organization with many years of software development experience.
- يهتم "التوسع" بكيفية تقديم الأساليب الرشيقة عبر مؤسسة كبيرة تتمتع بسنوات عديدة من الخبرة في تطوير البرامج.
- When scaling agile methods it is essential to maintain agile fundamentals
- عند توسيع نطاق الأساليب الرشيقة ، من الضروري الحفاظ على الأساسيات الرشيقة
  - Flexible planning, frequent system releases, continuous integration, test-driven development and good team communications.
  - التخطيط المرن ، وإصدارات النظام المتكررة ، والتكامل المستمر ، والتطوير القائم على الاختبار ، واتصالات الفريق الجيدة.

Scaling up to large systems
التوسع في الأنظمة الكبيرة



- For large systems development, it is not possible to focus only on the code of the system. You need to do more up-front design and system documentation

لتطوير الأنظمة الكبيرة ، لا يمكن التركيز فقط على كود النظام. تحتاج إلى القيام بالمزيد من وثائق النظام والتصميم المسبق

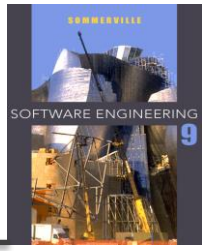
- Cross-team communication mechanisms have to be designed and used. This should involve regular phone and video conferences between team members and frequent, short electronic meetings where teams update each other on progress.

يجب تصميم آليات الاتصال بين الفرق واستخدامها. يجب أن يتضمن ذلك مؤتمرات هاتفية وفيديو منتظمة بين أعضاء الفريق واجتماعات إلكترونية متكررة وقصيرة حيث تُطلع الفرق بعضها البعض على التقدم المحرز.

- Continuous integration, where the whole system is built every time any developer checks in a change, is practically impossible. However, it is essential to maintain frequent system builds and regular releases of the system.

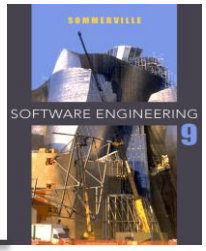
- التكامل المستمر ، حيث يتم بناء النظام بأكمله في كل مرة يتحقق فيها أي مطور من التغيير ، أمر مستحيل عمليًا. ومع ذلك ، فمن الضروري الحفاظ على عمليات إنشاء النظام المتكررة والإصدارات المنتظمة للنظام.

Scaling out to large companies



- Project managers who do not have experience of agile methods may be reluctant to accept the risk of a new approach.
- قد يحجم مديرو المشاريع الذين ليس لديهم خبرة في الأساليب الرشيقة عن قبول مخاطر اتباع نهج جديد.
- Large organizations often have quality procedures and standards that all projects are expected to follow and, because of their bureaucratic nature, these are likely to be incompatible with agile methods.
- غالبًا ما يكون لدى المنظمات الكبيرة إجراءات ومعايير الجودة التي من المتوقع أن تتبعها جميع المشاريع ، وبسبب طبيعتها البيروقراطية ، فمن المحتمل أن تكون غير متوافقة مع الأساليب الرشيقة.
- Agile methods seem to work best when team members have a relatively high skill level. However, within large organizations, there are likely to be a wide range of skills and abilities.
- يبدو أن الأساليب الرشيقة تعمل بشكل أفضل عندما يكون لدى أعضاء الفريق مستوى مهارة مرتفع نسبيًا. ومع ذلك ، في المنظمات الكبيرة ، من المحتمل أن يكون هناك مجموعة واسعة من المهارات والقدرات.
- There may be cultural resistance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes.
- قد تكون هناك مقاومة ثقافية للأساليب الرشيقة ، خاصة في تلك المنظمات التي لها تاريخ طويل في استخدام عمليات هندسة النظم التقليدية.

Key points



- A particular strength of extreme programming is the development of automated tests before a program feature is created. All tests must successfully execute when an increment is integrated into a system.
- تتمثل إحدى نقاط القوة الخاصة في البرمجة الشديدة في تطوير الاختبارات الآلية قبل إنشاء ميزة البرنامج. يجب تنفيذ جميع الاختبارات بنجاح عند دمج الزيادة في النظام.
- The Scrum method is an agile method that provides a project management framework. It is centred round a set of sprints, which are fixed time periods when a system increment is developed.
- طريقة سكروم هي طريقة رشيقة توفر إطار عمل لإدارة المشروع. تتمحور حول مجموعة من سباقات السرعة ، وهي فترات زمنية ثابتة عند تطوير زيادة في النظام.
- Scaling agile methods for large systems is difficult. Large systems need up-front design and some documentation.
- من الصعب تحجيم الأساليب الرشيقة للأنظمة الكبيرة. تحتاج الأنظمة الكبيرة إلى تصميم مسبق وبعض الوثائق.