Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Software Development Process

RADI601 Health informatics and health information technology / RADI607 Theories in Health Informatics and Health Information Technology

Lect. Anuchate Pattanateepapon, D.Eng.

Section of Data Science for Healthcare

Department of Clinical Epidemiology and Biostatistics

Faculty of Medicine Ramathibodi Hospital, Mahidol University

Wisdom of the Land

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

What is UML?

- A standardized, graphical "modeling language" for communicating software design.
- Allows implementation-independent specification of:
  - user/system interactions (required behaviors)
  - partitioning of responsibility
  - integration with larger or existing systems
  - data flow and dependency
  - operation orderings (algorithms)
  - concurrent operations
- UML is not "process". (That is, it doesn't tell you how to do things, only what you should do.)

Types of UML diagrams

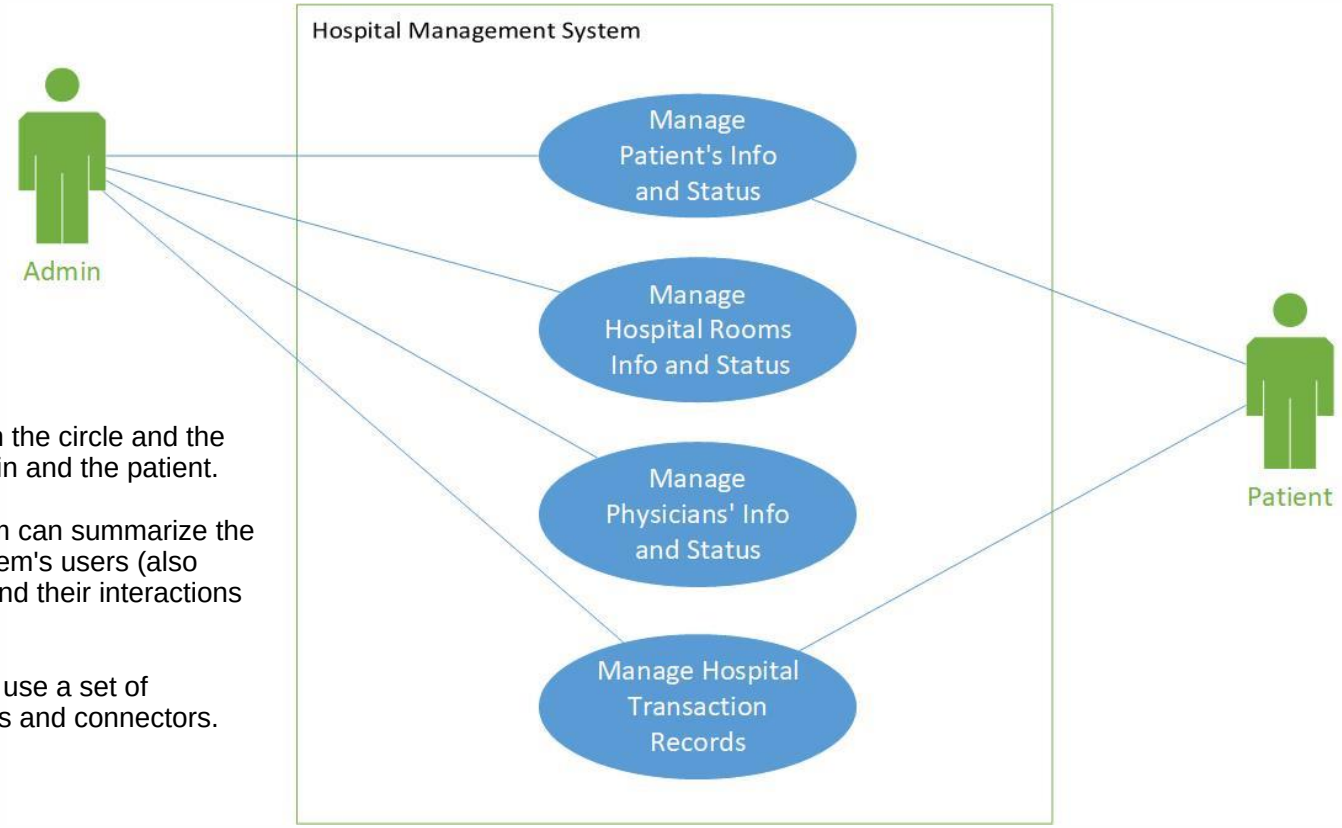
- There are different types of UML diagram, each with slightly different syntax rules:
  - use cases.
  - class diagrams.
  - sequence diagrams.
  - package diagrams.
  - state diagrams
  - activity diagrams
  - deployment diagrams.

use cases

- A use case encodes a typical user interaction with the system. In particular, it:
  - captures some user-visible function.
  - achieves some concrete goal for the user.
- A complete set of use cases largely defines the requirements for your system: everything the user can see, and would like to do.
- The granularity of your use cases determines the number of them (for you system). A clear design depends on showing the right level of detail.
- A use case maps actors to functions. The actors need not be people.

Use case examples, 1

(High-level use case for Hospital Management System)



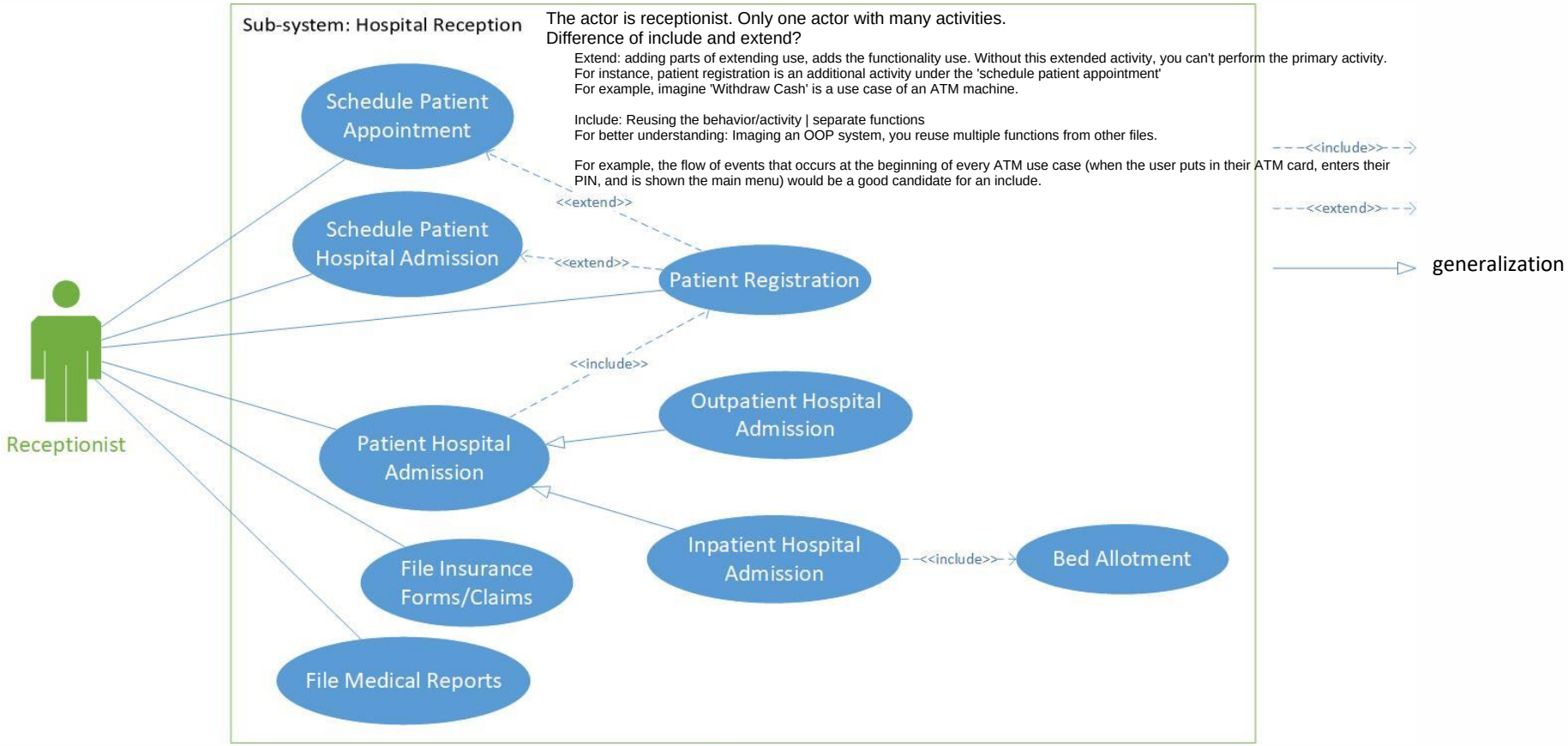
The activities are in the circle and the actors are the admin and the patient.

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.

To build one, you'll use a set of specialized symbols and connectors.

Use case examples, 2

(Finer-grained use cases for Hospital Management System)

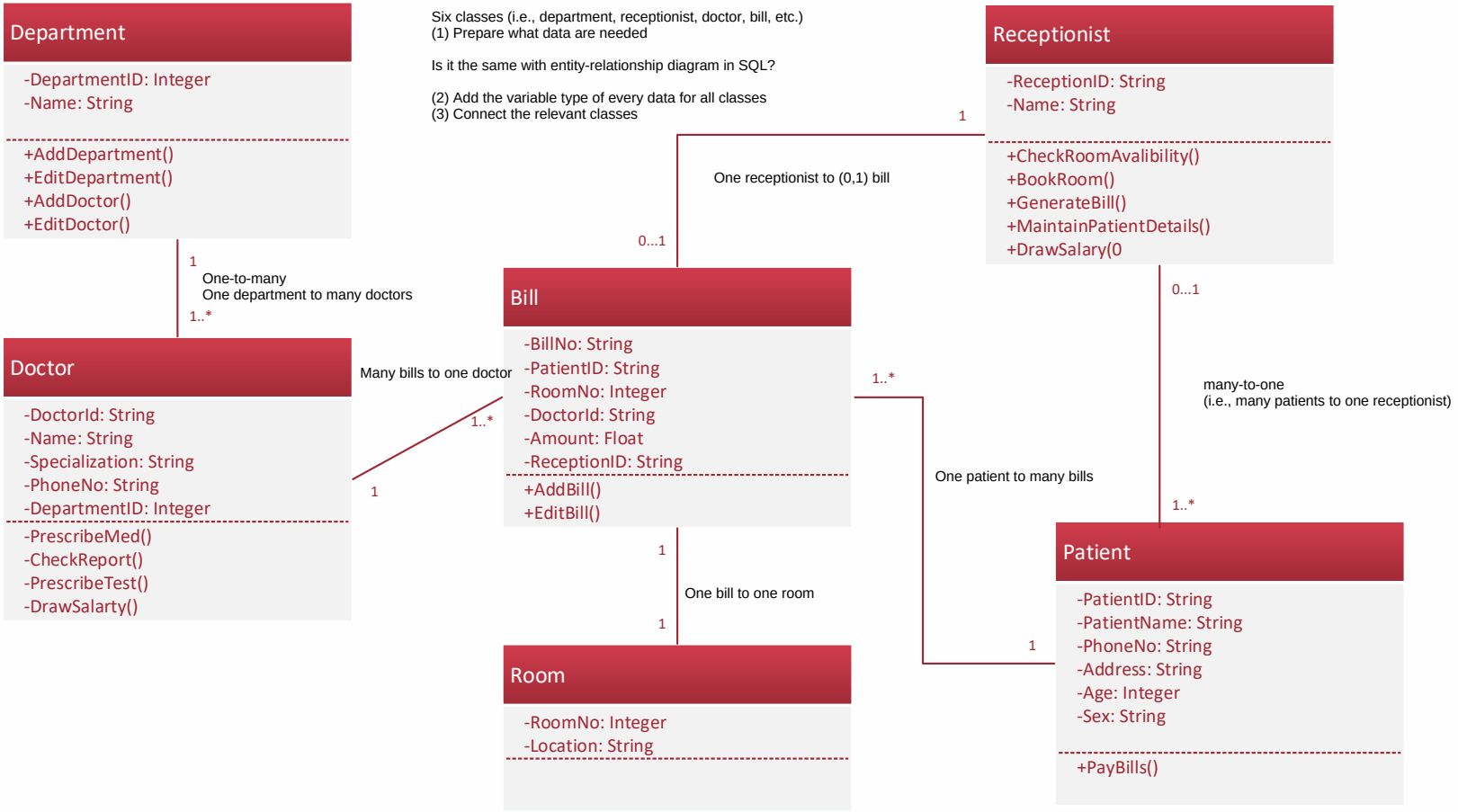


class diagram

- Motivated by Object-Oriented design and programming (OOD, OOP).
- A class diagram partitions the system into areas of responsibility (classes), and shows “associations” (dependencies) between them.
- Attributes (data), operations (methods), constraints, part-of (navigability) and type-of (inheritance) relationships, access, and cardinality (1 to many) may all be noted.

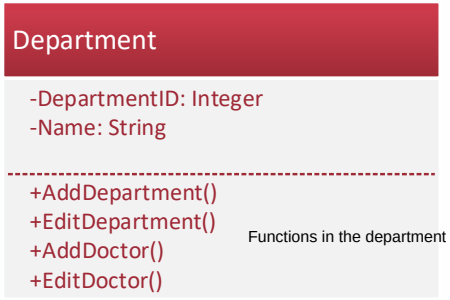
Class diagram examples

(A Hospital System: Room Management)



Class diagram examples

(A Hospital System: Room Management)



Attribute	Example Data
DepartmentID	1 should be integer
Name	Otolaryngology string

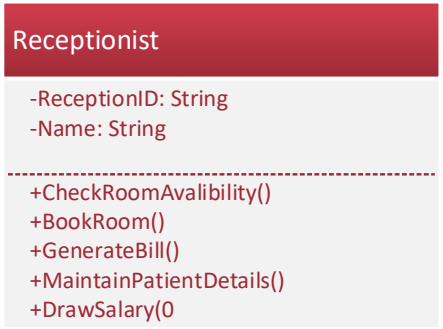
Class diagram examples

(A Hospital System: Room Management)

Attribute	Example Data
DoctorId	D00001 string: add characters + integers
Name	Dr. Thor Asgard
Specialization	Otolaryngologists
PhoneNo	0918888888
DepartmentID	1 integer: should be the same data type from the Department class

Class diagram examples

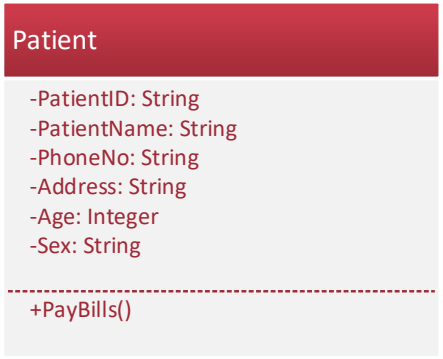
(A Hospital System: Room Management)



Attribute	Example Data
ReceptionID	R00001
Name	Jane Foster

Class diagram examples

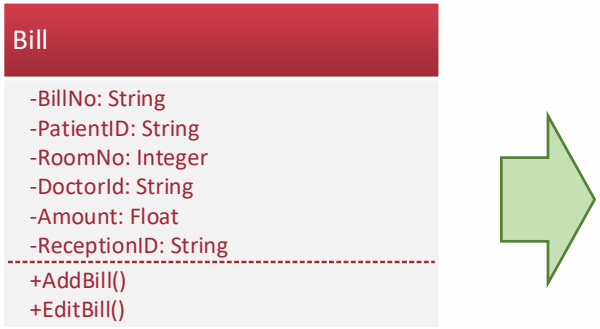
(A Hospital System: Room Management)



Attribute	Example Data
PatientID	4585999333
PatientName	Loki Asgard
PhoneNo	0899999999
Address	270 RAMA VI Road. Rachathevi, Bangkok 10400, Thailand.
Age	35 integer
Sex	Male can also be binary: 0 or 1

Class diagram examples

(A Hospital System: Room Management)

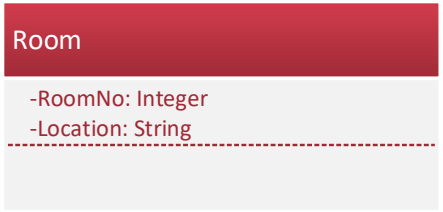


Note: if you will use zero (0), you must choose string as variable's data type

Attribute	Example Data
BillNo	20225500822564
PatientID	4585999333
RoomNo	13
DoctorId	D0001
Amount	15,456.00 float (decimal)
ReceptionID	R00001

Class diagram examples

(A Hospital System: Room Management)

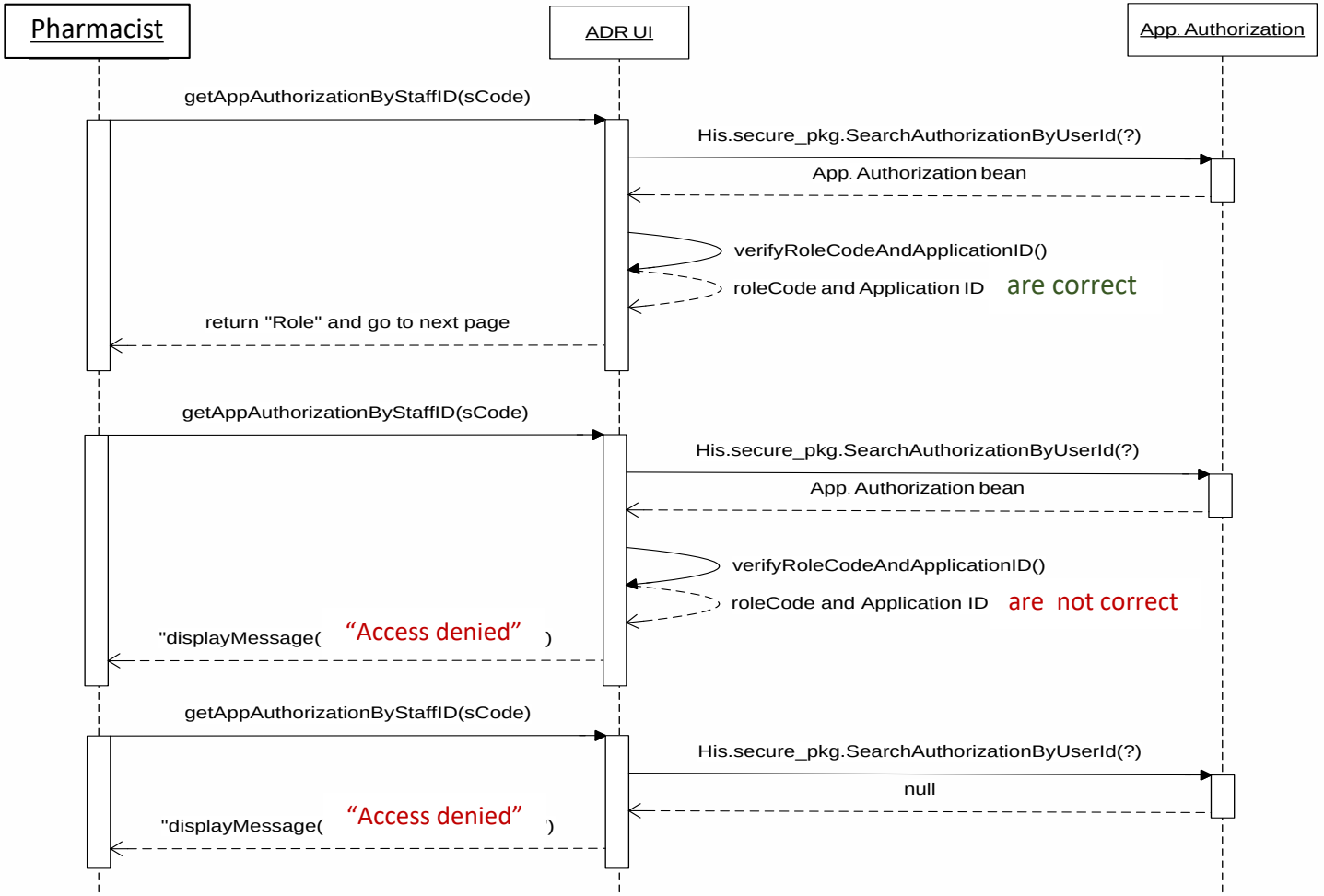


Attribute	Example Data
RoomNo	13
Location	ENT M1

sequence diagram

- Sequence diagram describe algorithms, though usually at a high level: the operations in a useful sequence diagram specify the "message passing" (method invocation) between objects (classes, roles) in the system.
- The notation is based on each object's life span, with message passing marked in time-order between the objects. Iteration and conditional operations may be specified.
- May in principle be used at the same three levels as class diagrams, though the specification level will usually be most useful. (At the implementation level, you might better use pseudocode.)

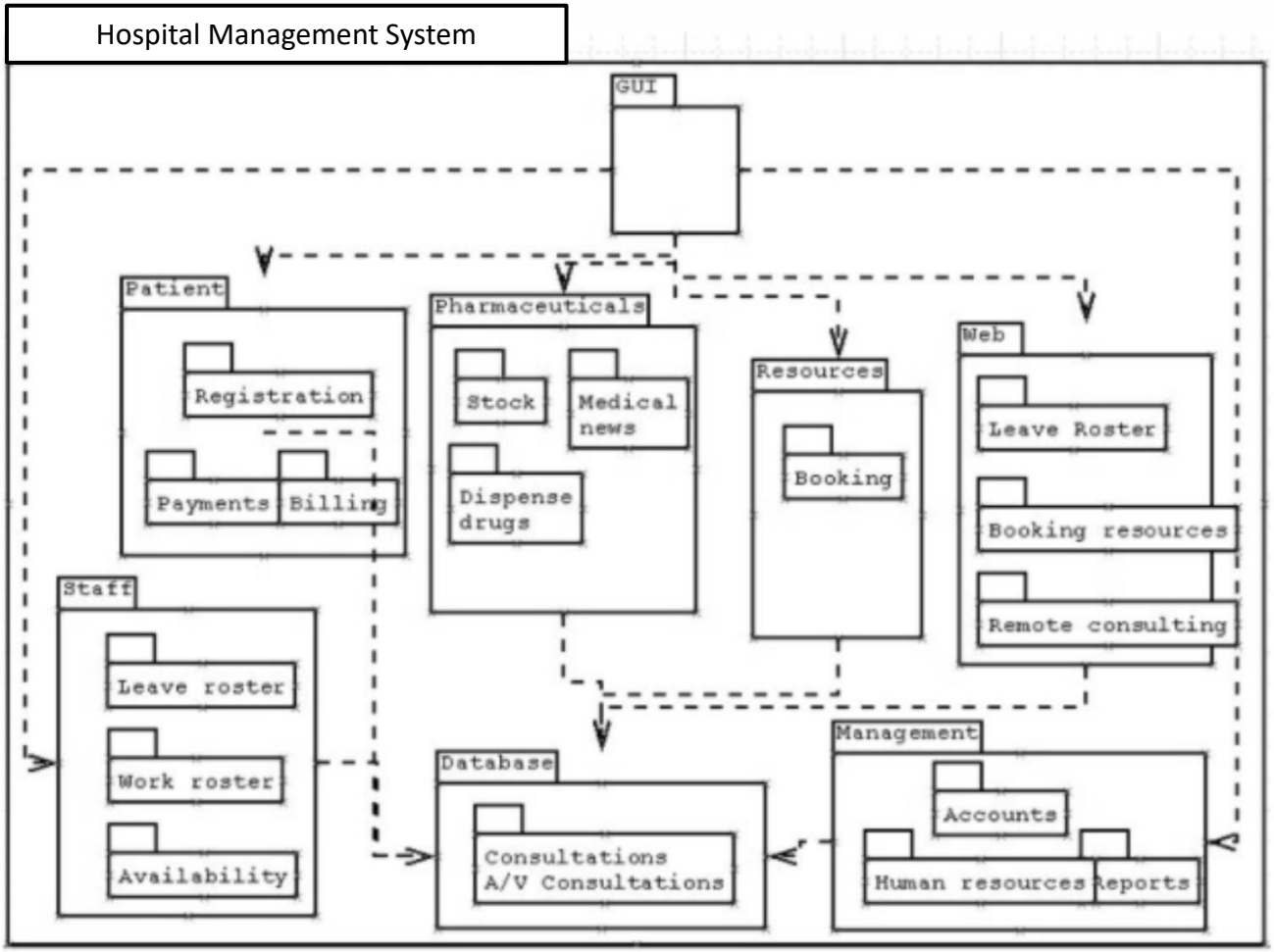
Sequence diagram example: Adverse Drug Reaction Sys.



Package diagram

- A type of class diagram, package diagrams show dependencies between high-level system component.
- A “package” is usually a collection of related classes, and will usually be specified by it’s own class diagram.
- The software in two distinct packages is separate; packages only interact through well-defined interfaces, there is no direct sharing of data or code.
- Not all packages in a system’s package diagram are new software; many packages (components) in a complex system are often already available as existing or off-the-shelf software.

Package diagram example

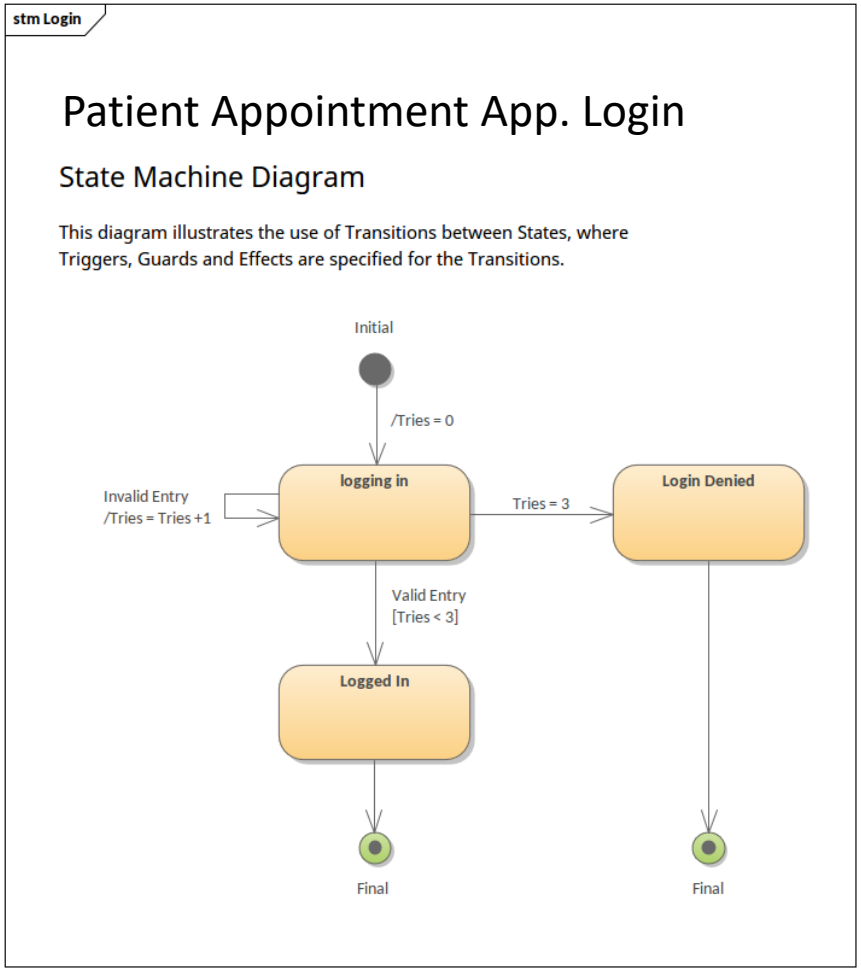


state diagrams

- state diagram: Depicts data and behavior of a single object throughout its lifetime.
  - set of states (including an initial start state)
  - transitions between states
  - entire diagram is drawn from that object's perspective

- What objects are best used with state diagrams?
  - large, complex objects with a long lifespan
  - domain ("model") objects
  - not useful to do state diagrams for every class in the system!

State diagram example



https://sparxsystems.com/resources/gallery/diagrams/software/sw-state_machine_diagram_-transition_guards_and_effects.html

Activity Diagram

- Activity diagrams describe the workflow behavior of a system.
  - Activity diagrams are used in process modeling and analysis of during requirements engineering.
  - A typical business process which synchronizes several external incoming events can be represented by activity diagrams.

- They are most useful for understanding work flow analysis of synchronous behaviors across a process.

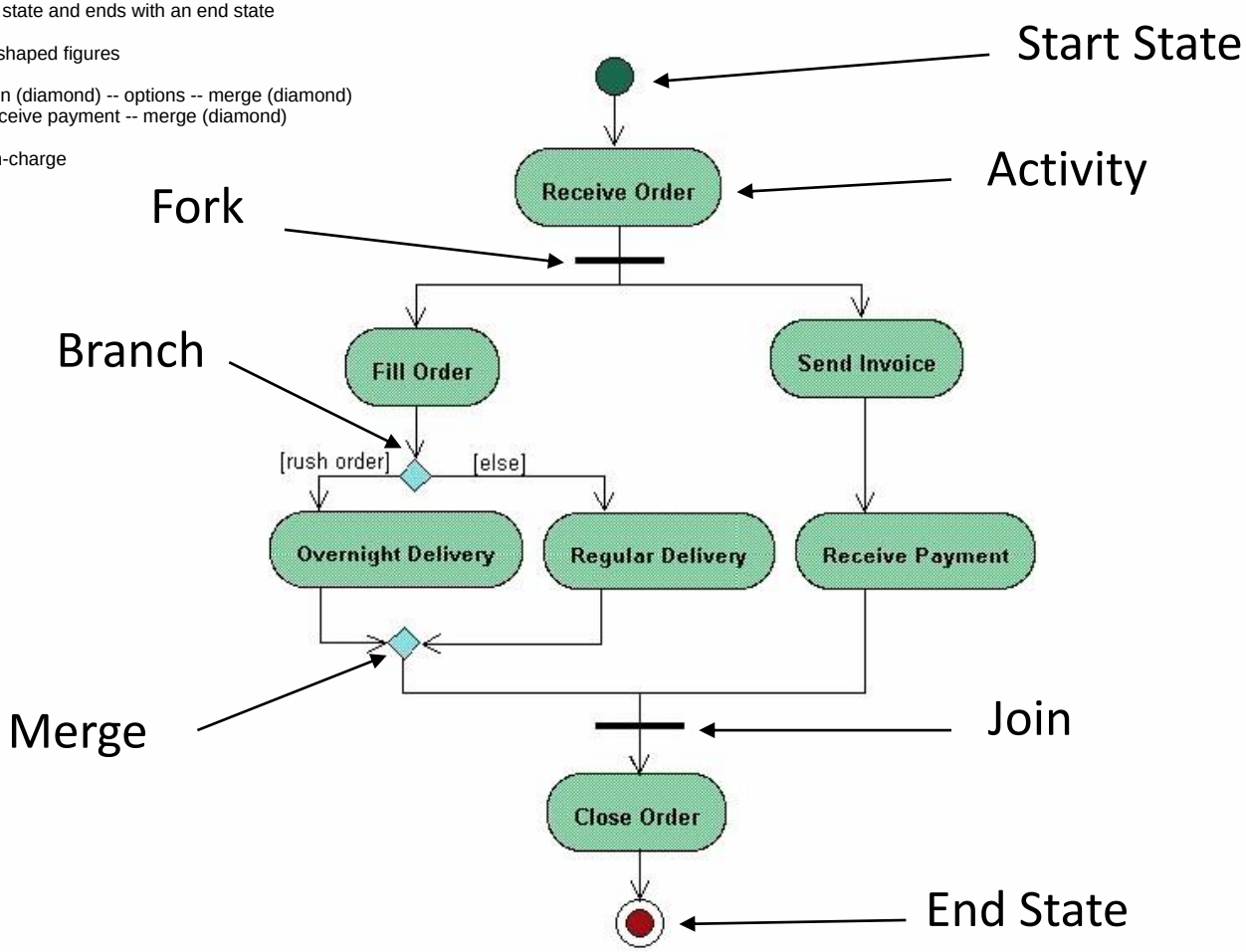
Activity Diagram Example

Begins with a start state and ends with an end state

Activities are oval-shaped figures

Fill order -- question (diamond) -- options -- merge (diamond)
Send invoice -- Receive payment -- merge (diamond)

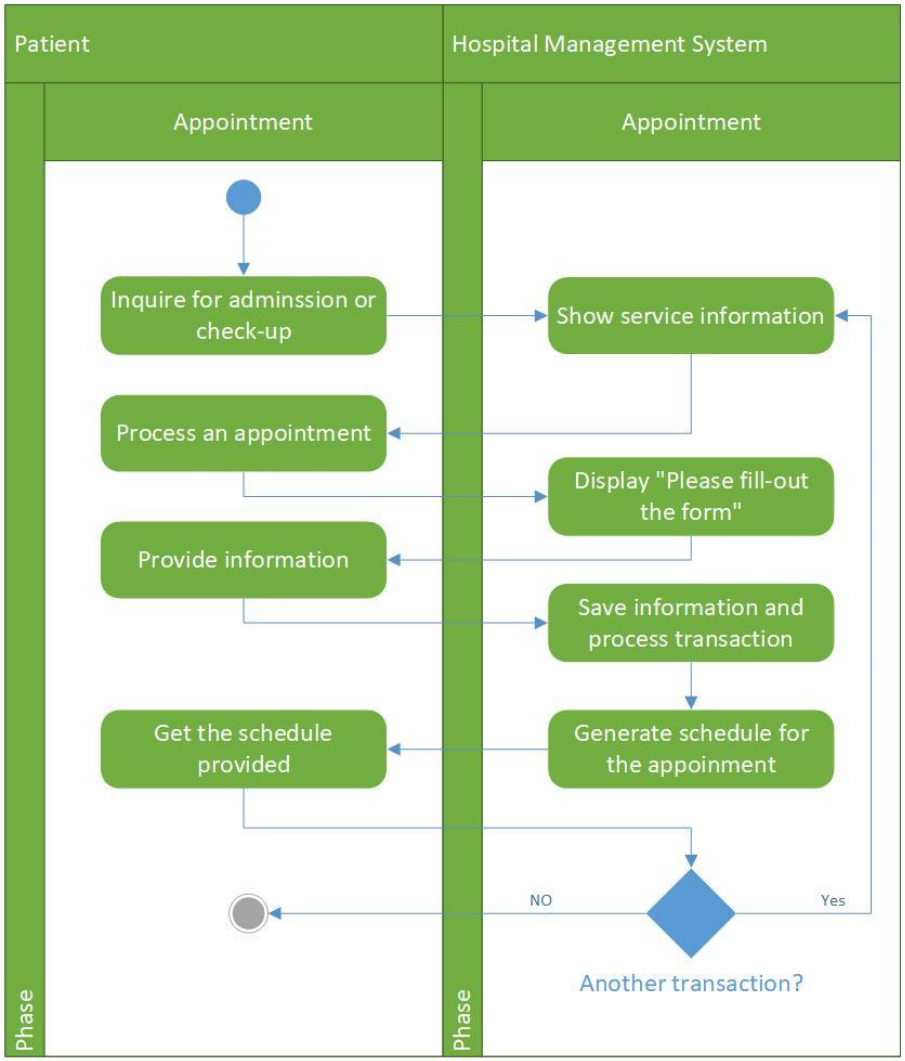
Only one person-in-charge



Activity Diagram Example

If there are more than one PIC, split the diagram into this form.

This diagram is sequential.

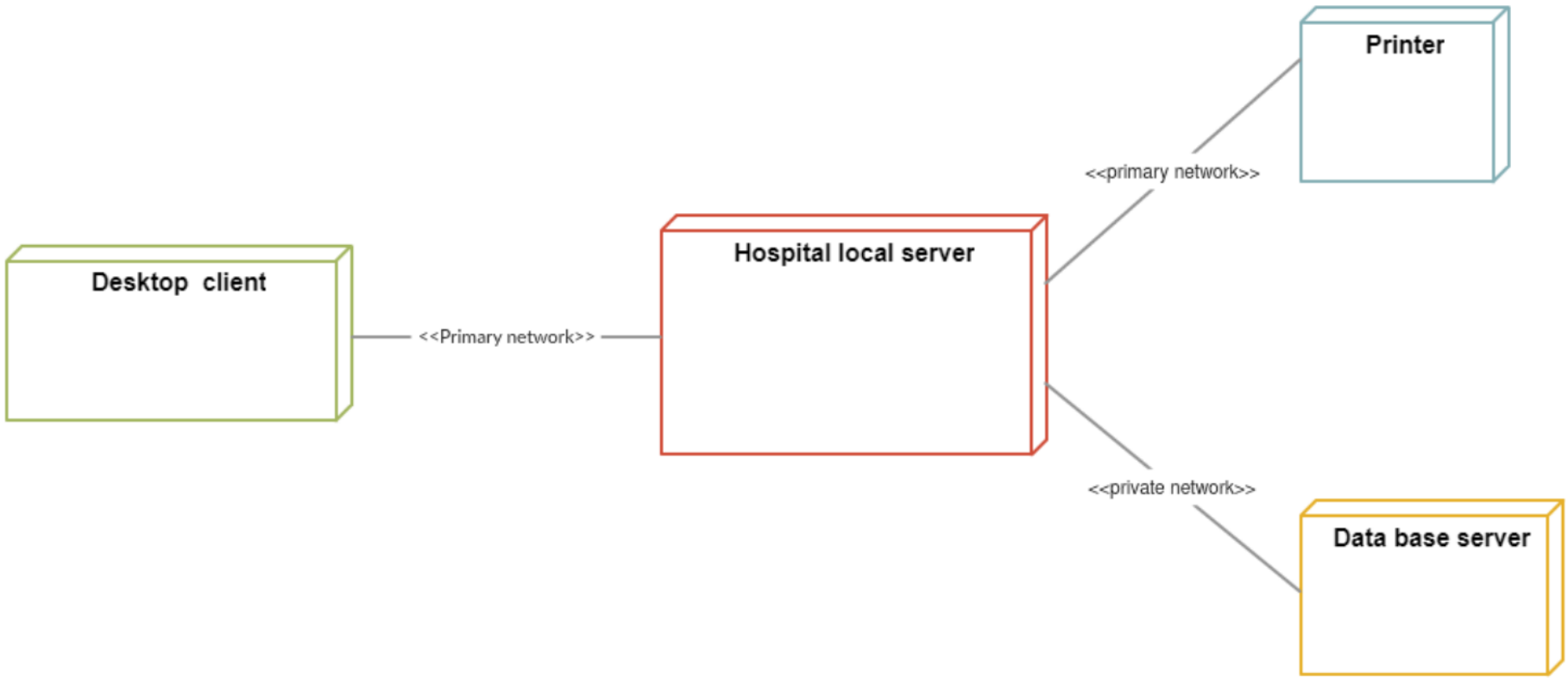


Deployment Diagram

- A deployment diagram is useful for showing how your software will be deployed on hardware. It may show how your system will integrate with existing systems in the domain.

Deployment Diagram

(A Hospital Management System)



Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

The Software Development Process

- A structured set of activities required to develop a software system
  - Specification
  - Analysis, design and implementation.
  - Validation
  - Evolution

SDP: SAVE (Specification, Analysis, Validation, Evolution)

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

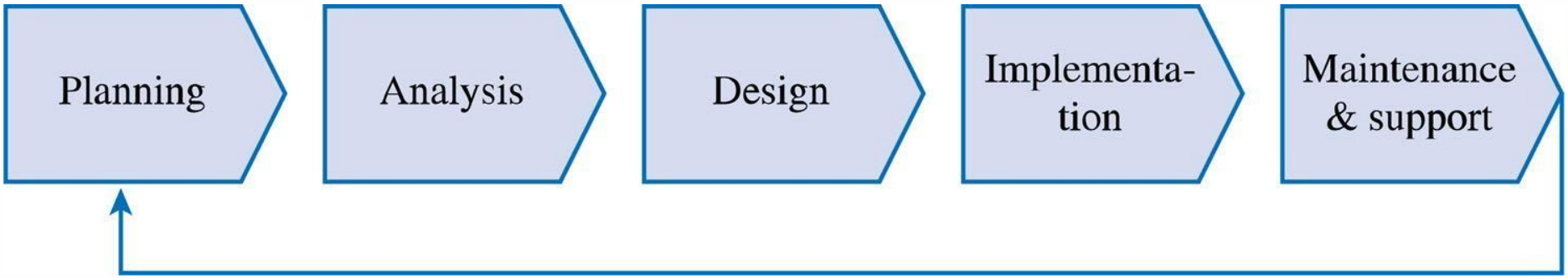
Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Software Development Life Cycle(SDLC)

- Represents the sequential phases or stages an information system follows throughout its useful life
- Useful for understanding the development of the project's largest work product – the application system
- Phases/Stages
  - Planning
  - Analysis
  - Design
  - Implementation
  - Maintenance and Support

Software Development Life Cycle(SDLC)



Marchewka JT (2006)

Guideline

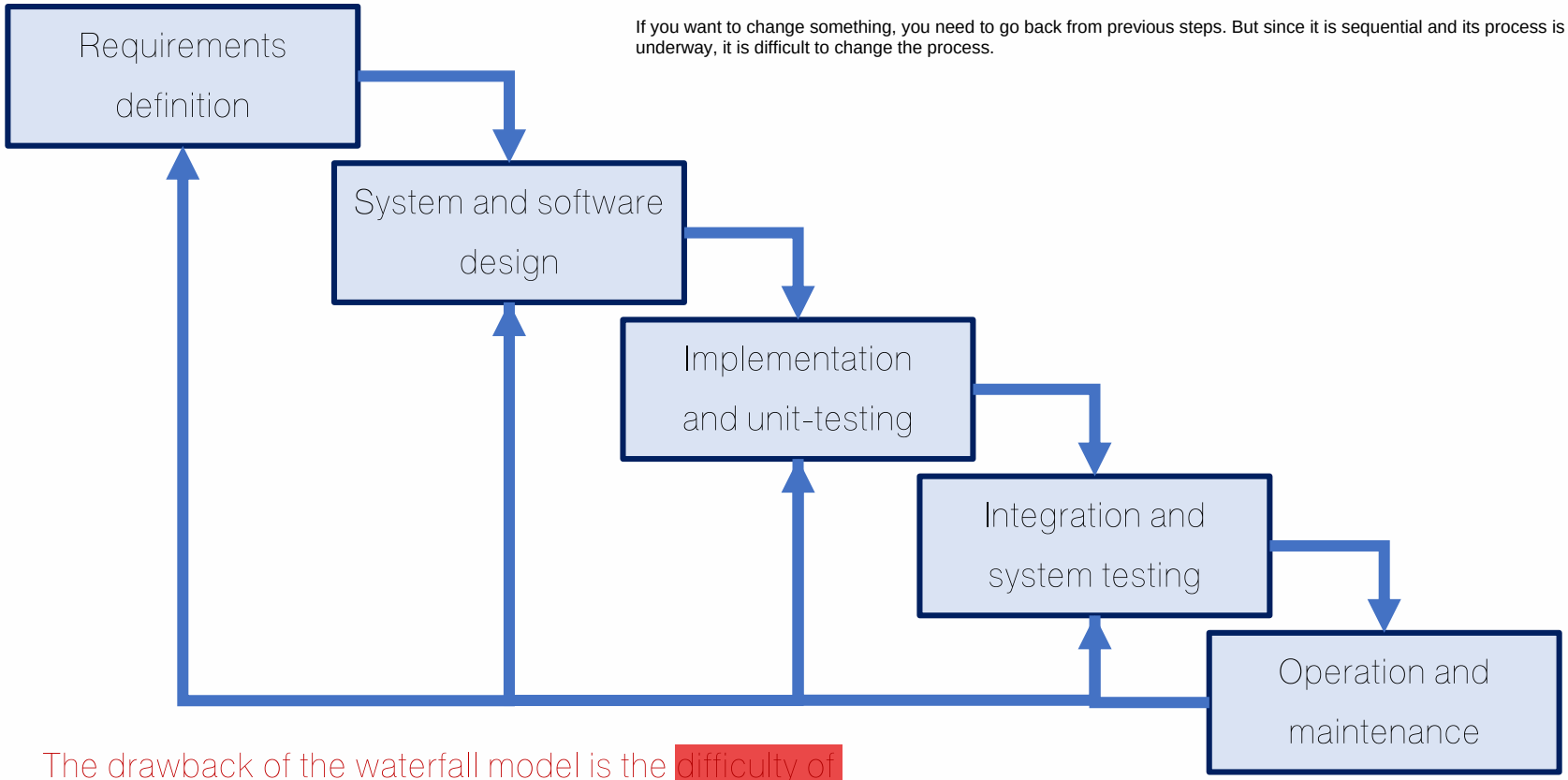
- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

Waterfall Model



If you want to change something, you need to go back from previous steps. But since it is sequential and its process is underway, it is difficult to change the process.

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

http://images.slideplayer.in.th/8/2082141/slides/slide_14.jpg

Waterfall model Requirement and Design

Artefacts produced in the requirements and Design phases

- SRS -Software Requirements specification document
- SRS might include:
  - User usage stories (scenarios) – Use cases.
  - Static analysis – class diagrams.
  - Behavioural analysis – sequence diagrams, statecharts.

The specification and design activities are heavily time consuming.

The requirements document

- The requirements document is the official statement of what is required of the system developers.

Remember the README.md file or requirements.txt in GitHub or Model Deployment

- Should include both a definition of user requirements and a specification of the system requirements.

- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Users of a requirements document

System customers	Specify the requirement and read them to check that they meet their needs. They specify changes to the requirements
Managers	Use the requirements document to plan a bid for the system and to plan the system development process
System engineers	Use the requirements to understand what system is to be developed
System test engineers	Use the requirements to develop validation tests for the system
System maintenance engineers	Use the requirements to help understand the system and the relationships between its parts

The Waterfall process characterization:

- Figure out what
- Figure out how
- Then do it
- Verify was done right
- Hand product to customer
- Perfect requirement definition?

Figure out what are the necessary requirements to avoid redo processes.

Waterfall model problems

Waterfall model is only suitable for freezing. It may not be suitable for changes since it is difficult to respond to changing customer requirements.

- Inflexible partitioning of the project into distinct stages
- Difficult to respond to changing customer requirements

This model is only appropriate when the requirements are well-understood

Waterfall model describes a staged development process

- Based on hardware engineering models
- Change during development is unlikely
- Widely used in large systems: military and aerospace industries

Why Not a Waterfall

Because software is different :

- No fabrication step
  - Program code is another design level
  - Hence, no "commit" step – software can always be changed…!
- No body of experience for design analysis (yet)
  - Most analysis (testing) is done on program code
  - Hence, problems not detected until late in the process
- Waterfall model takes a static view of requirements
  - Ignore changing needs
  - Lack of user involvement once specification is written
- Unrealistic separation of specification from the design
- Doesn't accommodate prototyping, reuse, etc

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

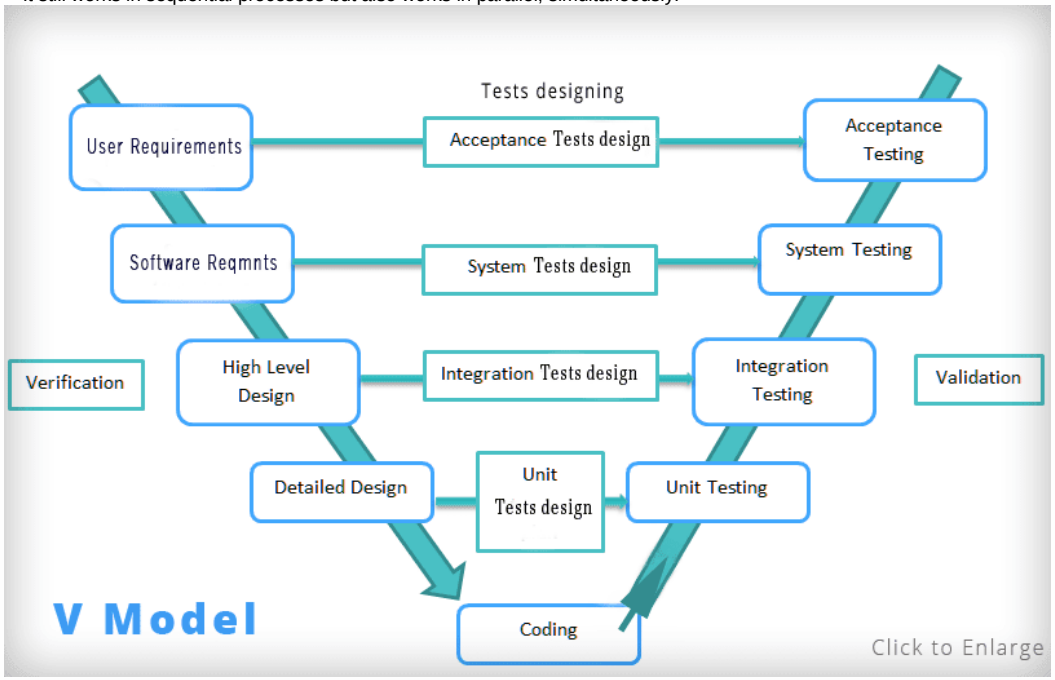
Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Log-in Email: Email System
- Create email, send email, read inbox
- Unit test: focus on functions, log-in function test
- For instance, log-in test needs to be tested on username and password. If your requirement: Password should be in email format. Password should also be in a specific format: numbers, characters. If it fails, then you cannot pass the testing unit.

Modified from the waterfall: V-Shaped SDLC model

V-Shaped SDLC Model

Every activity has testing designs:
For instance, user requirements need to be tested under acceptance testing
It still works in sequential processes but also works in parallel, simultaneously.



- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development

V-Shaped is more focused on testing. For every phase some testing activity are done

http://www.testnbug.com/wp-content/uploads/2014/12/v-model.png

V-Shaped Steps

- User Requirements and Planning – allocate resources
- Software Requirements and Specification Analysis – complete specification of the software system
- Architecture or High-Level Design – defines how software functions fulfill the design
- Detailed Design – develop algorithms for each architectural component
- Coding – transform algorithms into software
- Unit testing – check that each module acts as expected
- Integration and Testing – check that modules interconnect correctly
- System testing – check the entire software system in its environment
- Acceptance testing – provide for enhancement and corrections

V-Shaped Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

V-Shaped Weaknesses

- Does not easily handle concurrent events does not handle the simultaneous (overload) events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

When to use the V-Shaped Model

- Excellent choice for systems requiring high reliability – hospital patient control applications Risk Analysis
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

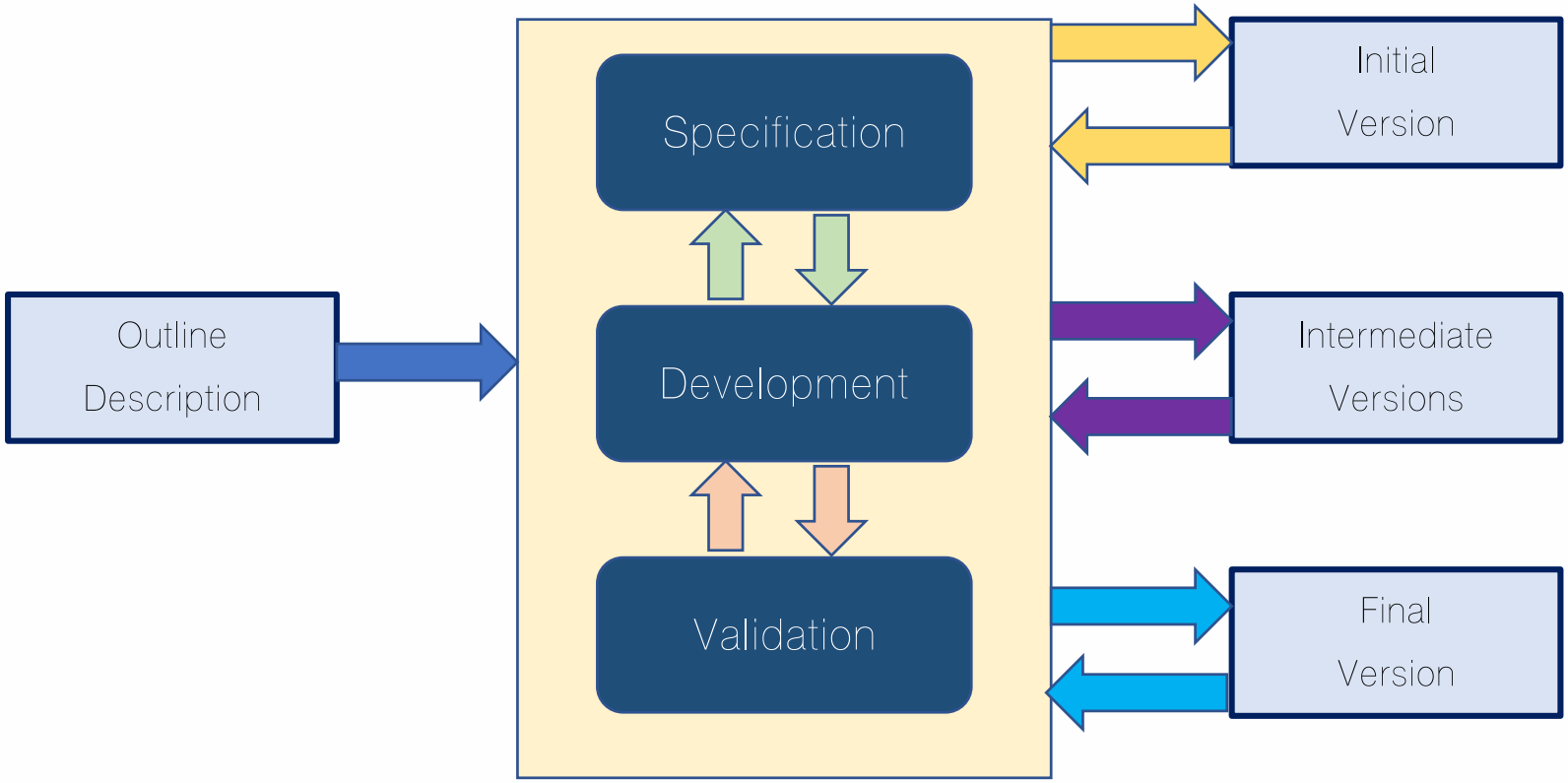
Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Evolutionary Development

- Modern development processes take *evolution* as fundamental, and try to provide ways of managing, rather than ignoring the risk.
- System requirements always *evolve* in the course of a project.
- Specification is *evolved* in conjunction with the software – No complete specification in the system development contract. Difficult for large customers.
- Two (related) process models:
  - Incremental development
  - Spiral development

Evolutionary development



Evolutionary is a customer focused model. The software is divided in small units which is delivered earlier to the customers

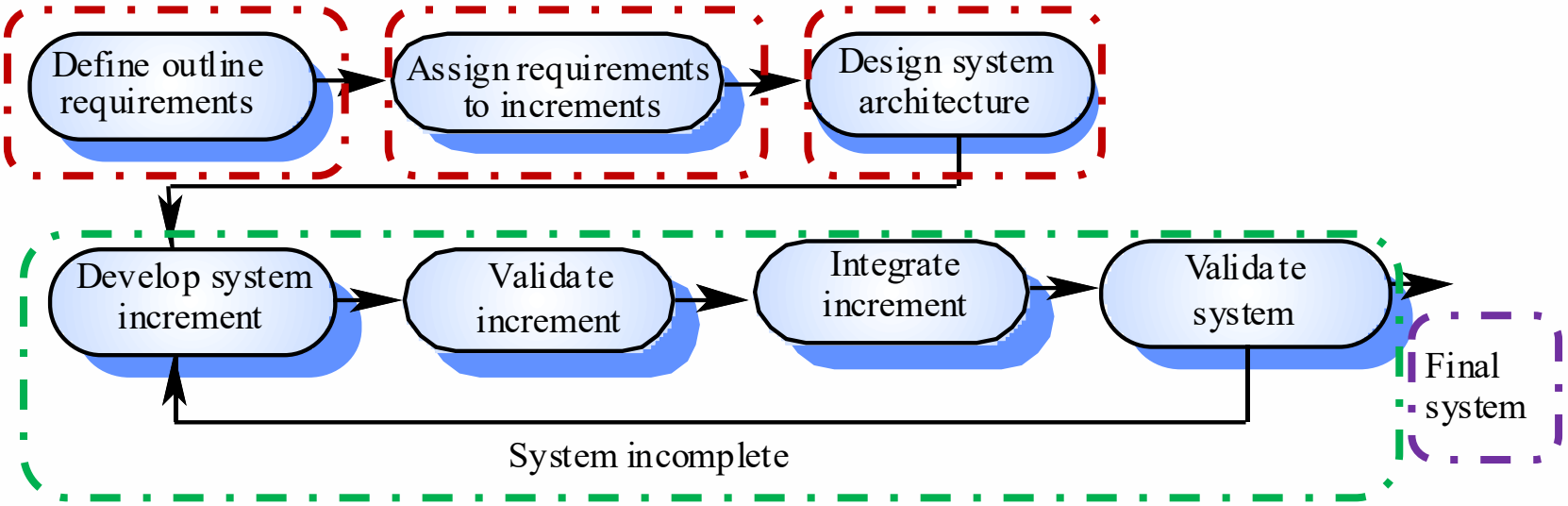
Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Incremental Development

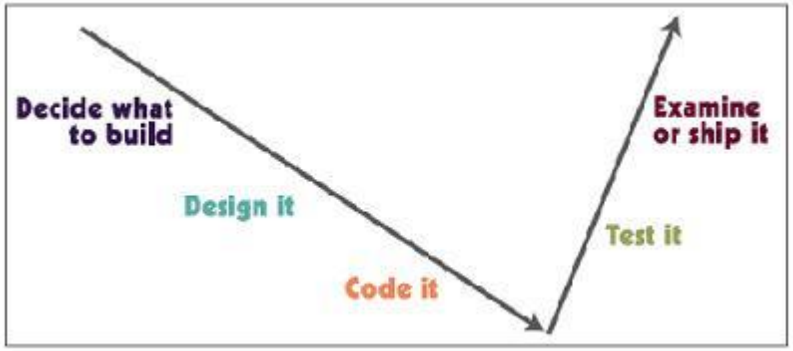
- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

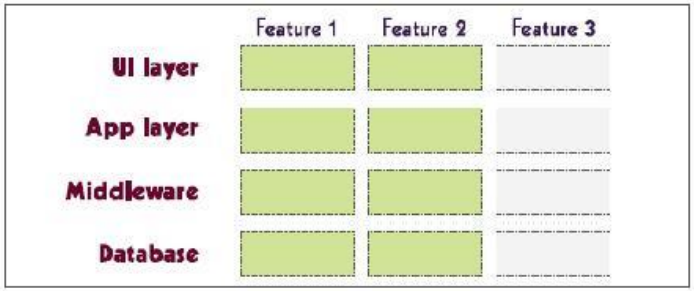
Incremental Development

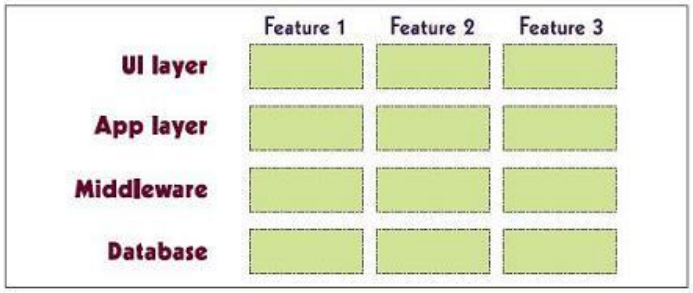


Incremental is a non integrated development model. This model divides work in chunks and one team can work on many chunks. It is more flexible

Incremental Development – (Ex. 3 features to develop)







- The first increment delivers one slice of functionality through the whole system.
- The second increment delivers another slice of functionality through the whole system.
- The third increment delivers the rest of the system

Incremental Development –Advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services

  tend to receive the most testing.

Problems with incremental development

- Lack of process visibility
- Systems are often poorly structured
- Management problems
  - Progress can be hard to judge and problems hard to find because there is no documentation to demonstrate what has been done.
- Contractual problems

  When you don't specify in the documentation/contract, problems may arise between the client and the company. Therefore, proper contract specification is vital in project management.
  - The normal contract may include a specification; without a specification, different forms of contract have to be used.
- Validation problems
  - Without a specification, what is the system being tested against?
- Maintenance problems
  - Continual change tends to corrupt software structure making it more expensive to change and evolve to meet new requirements.

When to use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

For instance, you have limitations on number of staff. You need to optimize the labor resources to meet the organizational goals.

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

iterative Development

- Iterative development is a rework scheduling strategy in which time is set aside to revise and improve parts of the system.
- The alternative development is to get it right the first time (or at least declare that it is right!).

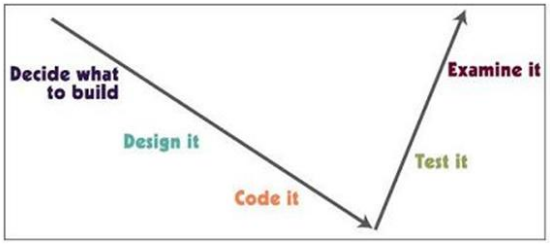
iterative means reworking, Incremental means adding

(by Alistair Cockburn)

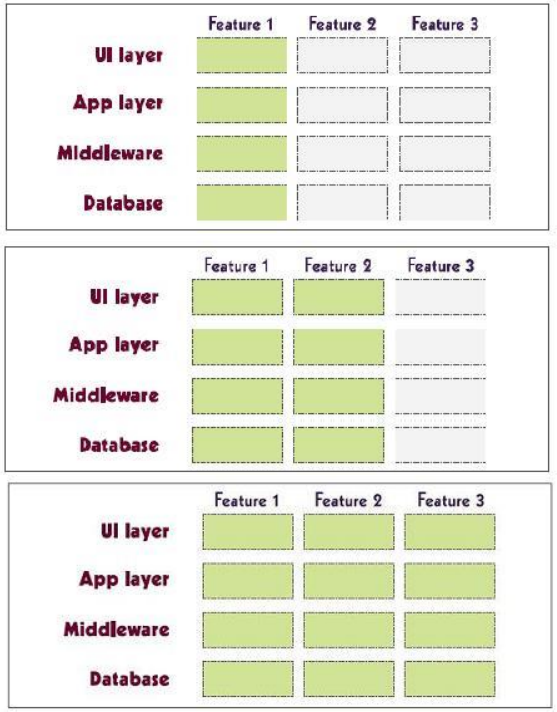
Iterate	Increment
fundamentally means *"change"*.	fundamentally means *"add onto"*.
repeating the process on the *same* section of work	repeating the process on a *new* section of work.
repeat the process (design, implement, evaluate),	repeat the process (design, implement, evaluate),

http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea&id=108

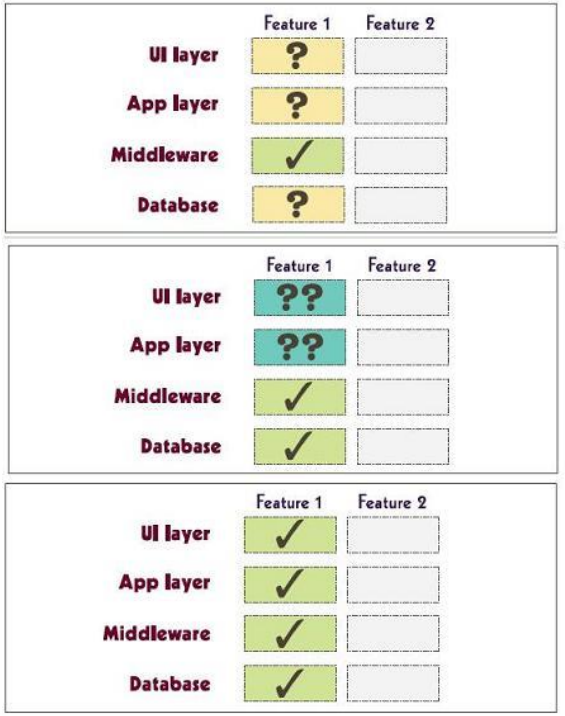
Incremental vs Iterative Development



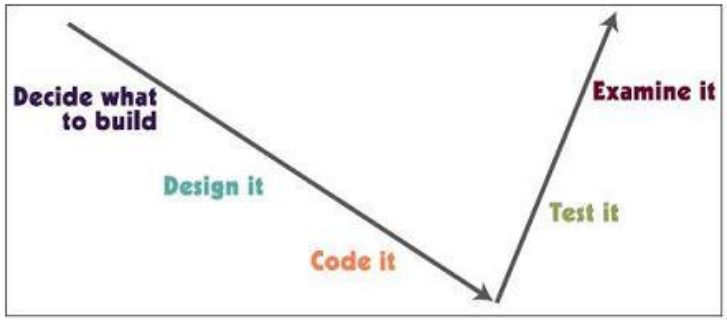
incremental Development



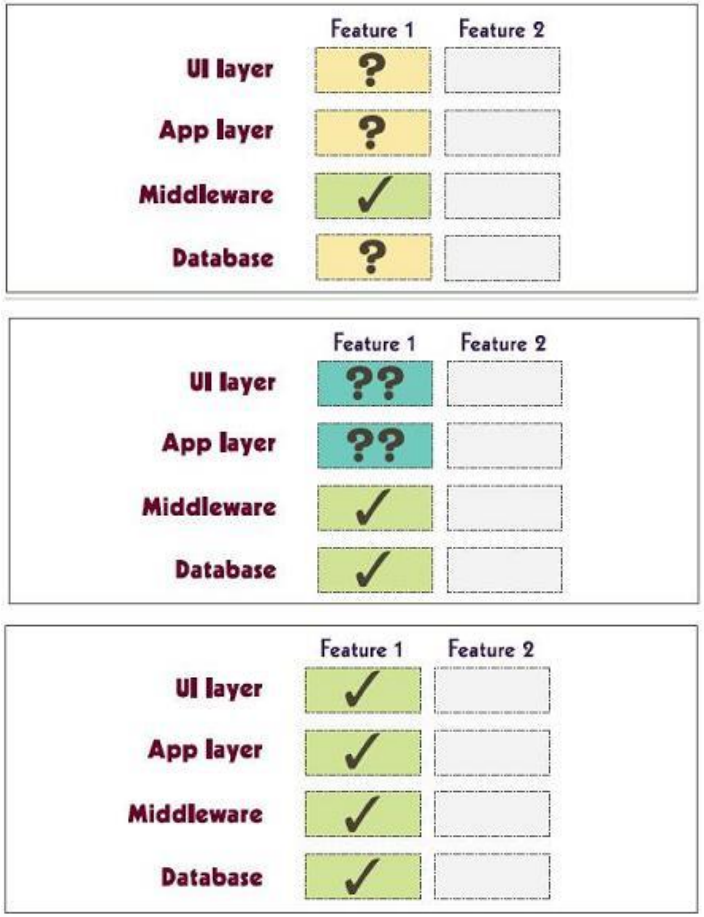
iterative Development



Iterative Development



- The first iteration delivers enough of feature 1 to evaluate what is correct and what needs revision.
- After the second iteration, some revised parts still need improvement.
- The third iteration produces the final and stable feature



Advantage(Pros) of Iterative Model:

- Testing and debugging during smaller iteration is easy.
- A Parallel development can plan.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.
- Limited time spent on documentation and extra time on designing.

Disadvantage(Cons) of Iterative Model:

- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

When to use iterative development

- When requirements are defined clearly and easy to understand.
- When the software application is large.
- When there is a requirement of changes in future.

Guideline

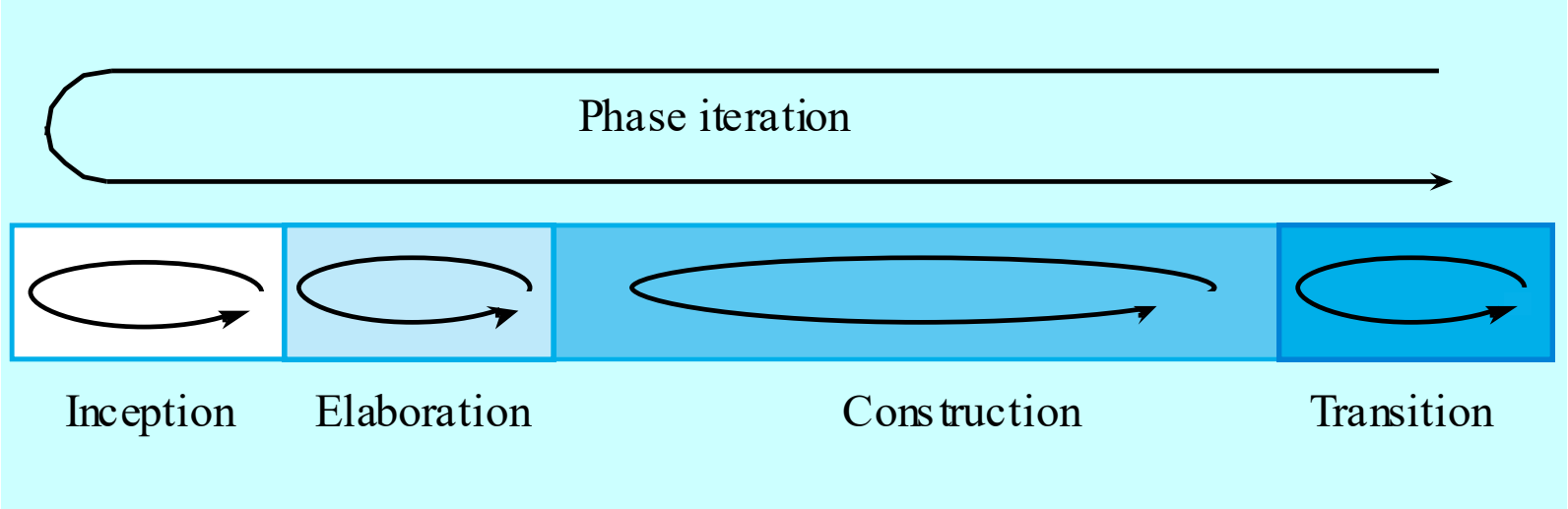
- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

The (Rational) Unified Process

- A modern process model derived from the work on the UML.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practice perspective that suggests good practice.

The (Rational) Unified Process

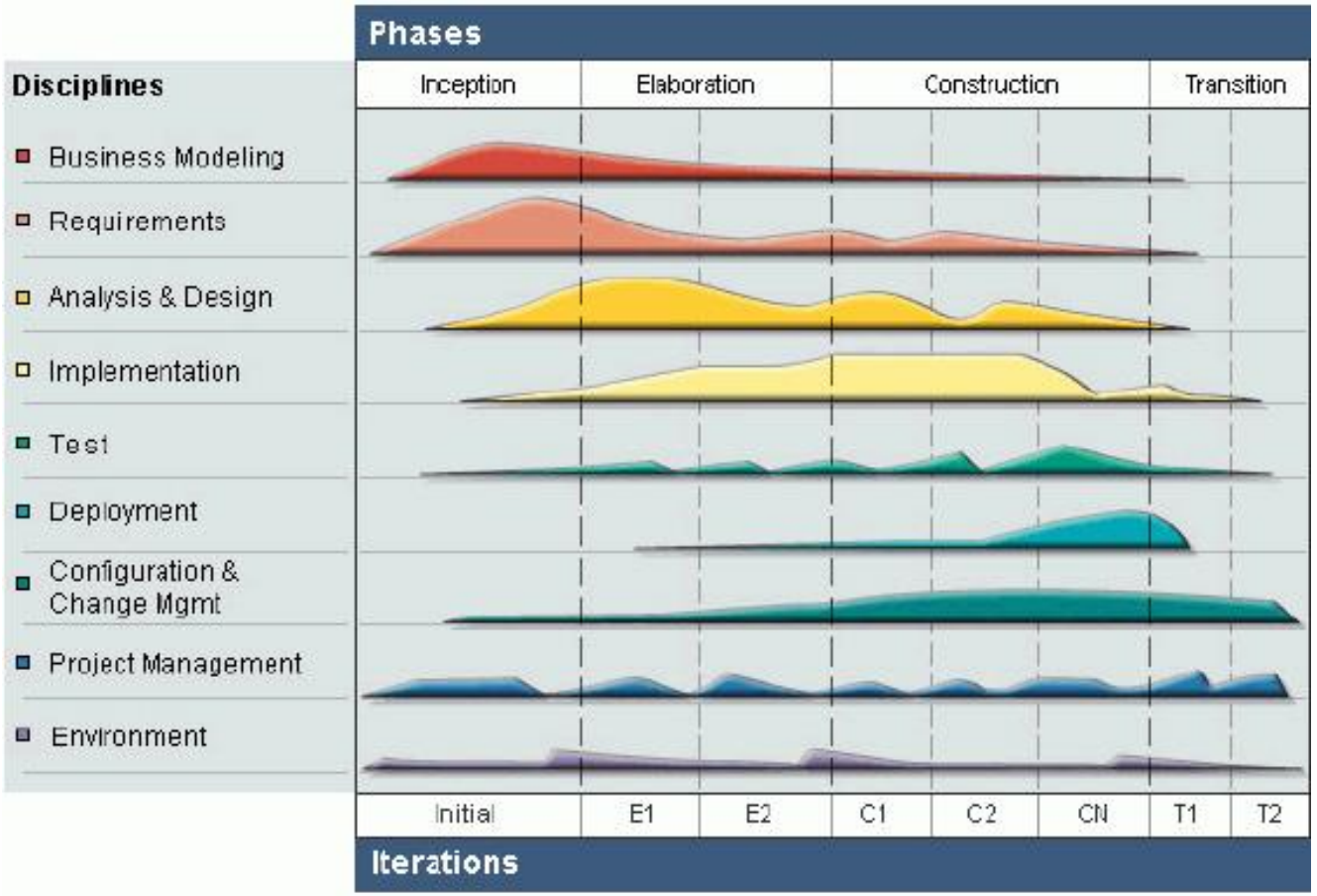
One cycle consists of four phases



The (Rational) Unified Process

- One cycle consists of four phases:
  - Inception
    - Establish the business case for the system.
  - Elaboration
    - Develop an understanding of the problem domain and the system architecture.
  - Construction
    - System design, programming and testing.
  - Transition
    - Deploy the system in its operating environment.

(R)UP phases and iterations



Picture taken from: http://www.ibm.com/developerworks/webservices/library/ws-soa-term2/

The benefits of RUP

- It allows you to deal with changing requirements regardless of whether they are coming from the customer or from the project itself.
- It emphasizes the need for accurate documentation.
- It forces integration to happen throughout the software development, more specifically in the construction phase.

The disadvantages of RUP

- It mostly relies on the ability of experts and professionals to assign the activities to individuals who should then produce pre-planned results in the form of artifacts.
- The integration in development process can also have an adverse impact on some more fundamental activities during the stages of testing
- Although RUP has delivered excellent results, especially in software development, it is a rather complex method which makes its implementation challenging, particularly for smaller businesses, teams or projects.

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

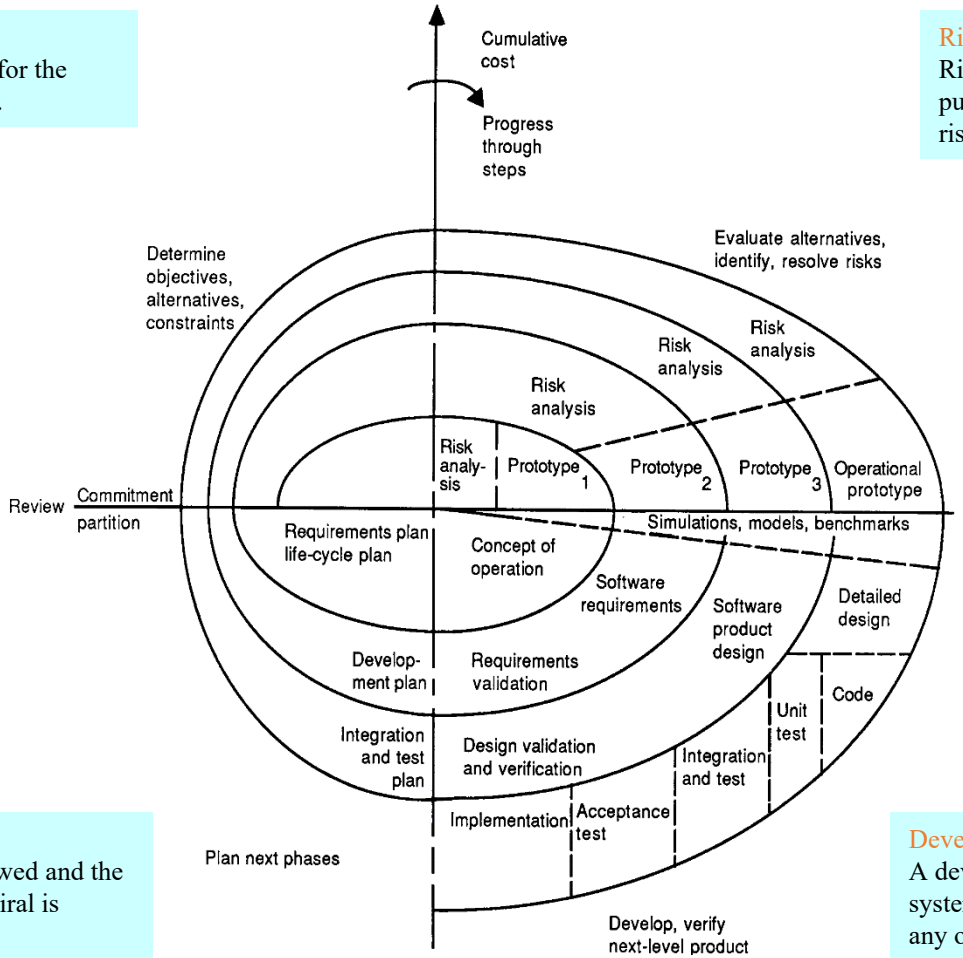
Spiral Development

- Process is conceived as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

Spiral model (Boehm 87)

Objective setting
Specific objectives for the phase are identified.

Risk assessment and reduction
Risks are assessed and activities put in place to reduce the key risks.



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

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

Advantages of Spiral Development

- Additional functionality or changes can be done at a later stage
- Cost estimation becomes easy as the prototype building is done in small fragments
- Continuous or repeated development helps in risk management
- Development is fast and features are added in a systematic way in Spiral development
- There is always a space for customer feedback

Disadvantages of Spiral Development

- Risk of not meeting the schedule or budget
- Spiral development works best for large projects only also demands risk assessment expertise
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases
- Spiral software development is not advisable for smaller project, it might cost them a lot

When to Use Spiral Development

- A Spiral model in software engineering is used when project is large
- When releases are required to be frequent, spiral methodology is used
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- Spiral methodology is useful for medium to high-risk projects
- When requirements are unclear and complex, Spiral model in SDLC is useful
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Guideline

- Introduction to Unified Modelling language (UML)
- Software Development Process
  - Software Development Life Cycle
  - Waterfall Model
  - V-Shaped SDLC Model
  - Evolutionary development
  - Incremental Development
  - Iterative Development
  - The (Rational) Unified Process
  - Spiral Development
  - Agile Development

Agile Methods

Result from dissatisfaction with the overheads involved in design methods.

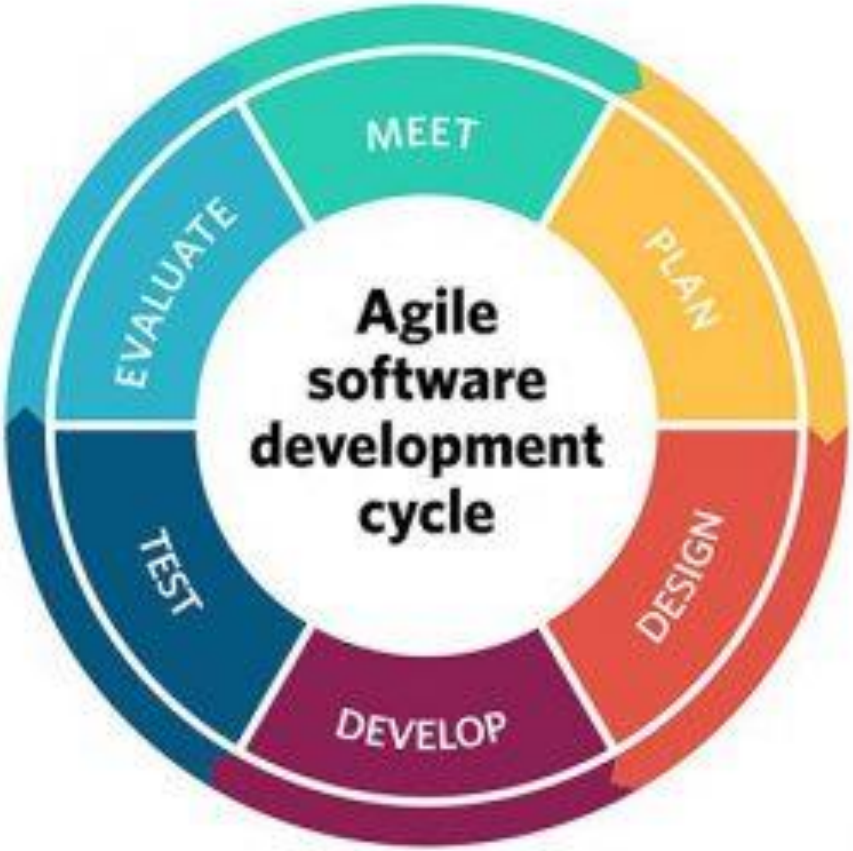
Software Development History:

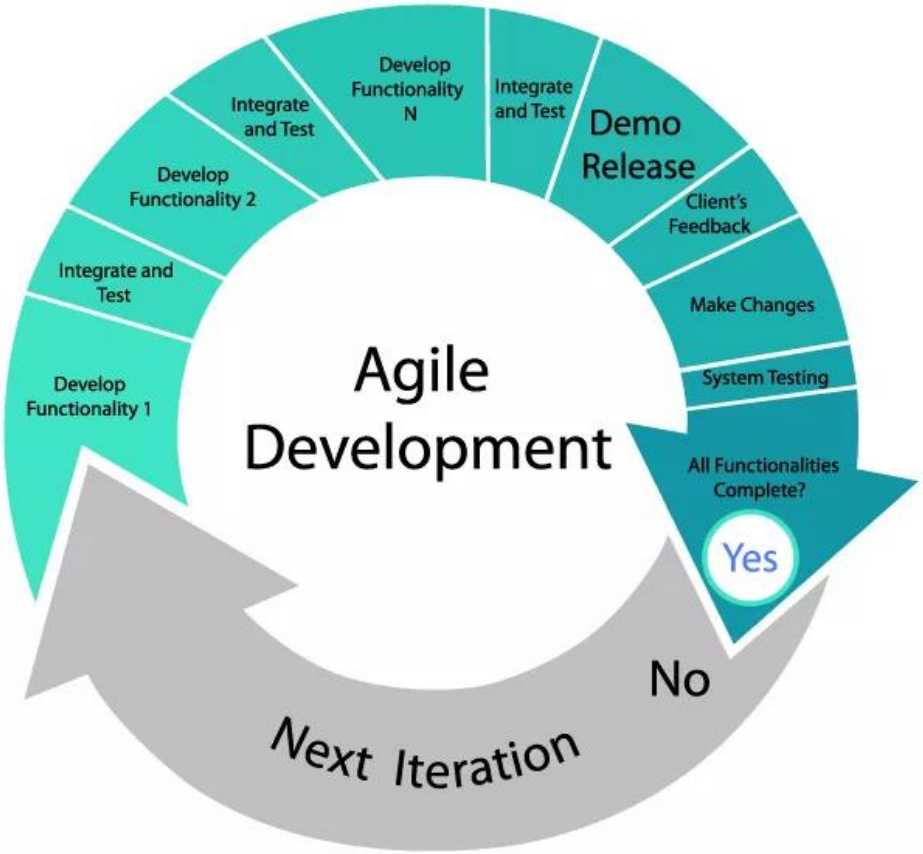
- During the 1970s, it was discovered that most large software development projects failed.
- During the 1980s, many of the reasons for those failures began to be recognized.
- In the 1990s, experiments and measurements were used to validate individual methods to prevent failure.
- The current decade is characterized by complete processes to improve success.

Project Failure – the trigger for Agility

- One of the primary causes of project failure was the extended period of time it took to develop a system.
- Costs escalated and requirements changed.
- Agile methods intend to develop systems more quickly with limited time spent on analysis and design.

Agile Development





What is an Agile method? (1)

- Focus on the code rather than the design.
- Based on an iterative approach to software development.
- Intended to deliver working software quickly.
- Evolve quickly to meet changing requirements.
- There are claims that agile methods are probably best suited to small/medium-sized business systems or PC products.

What is an Agile method? (2)

- Agile proponents believe:
  - Current software development processes are too heavyweight or cumbersome
    - Too many things are done that are not directly related to software product being produced
  - Current software development is too rigid
    - Difficulty with incomplete or changing requirements
    - Short development cycles (Internet applications)
  - More active customer involvement needed

What is an Agile method? (3)

- Agile methods are considered
  - Lightweight
  - People-based rather than Plan-based
- Several agile methods
  - No single agile method
  - Extreme Programming (XP) most popular
- No single definition
- Agile Manifesto closest to a definition
  - Set of principles
  - Developed by Agile Alliance

Summary of Principles of agile methods

Principle	Description
Customer involvement	The customer should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and design the system so that it can accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process used. Wherever possible, actively work to eliminate complexity from the system.

Claimed Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterizes agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

eXtreme Programming

- Development and delivery of very small increments of functionality.
- Relies on constant code improvement, user involvement in the development team and pair wise programming.
- Emphasizes Test Driven Development (TDD) as part of the small development iterations.

Summary

- Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time.
- Waterfall Model is a sequential design process in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.
- V-Shaped SDLC Model is more focused on testing. For every phase some testing activity are done.
- Evolutionary model is a customer focused model. In this model the software is divided in small units which is delivered earlier to the customers.

Summary

- Incremental model is a non integrated development model. This model divides work in chunks and one team can work on many chunks. It is more flexible.
- Iterative Development iterates requirements, design, build and test phases again and again for each requirement and builds up a system iteratively till the system is completely build.
- The (Rational) Unified Process is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases that each involve business modeling, analysis and design, implementation, testing, and deployment.

Summary

- Spiral Development - This model uses series of prototypes in stages, the development ends when the prototypes are developed into functional system. It is flexible model and used for large and complicated projects.
- Agile Development - refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. It allows you to break large projects down into more manageable tasks.