CMPS1134
Fundamentals of Computing

Software Engineering 2

Computer Science: An Overview
Eleventh Edition
J. Glenn Brookshear
Chapter 7

Chapter 7: Software Engineering

- Tools of the Trade
- Quality Assurance
  - Testing
  - Documentation
- The Human Machine Interface
- Software Ownership and Liability

Tools of the Trade

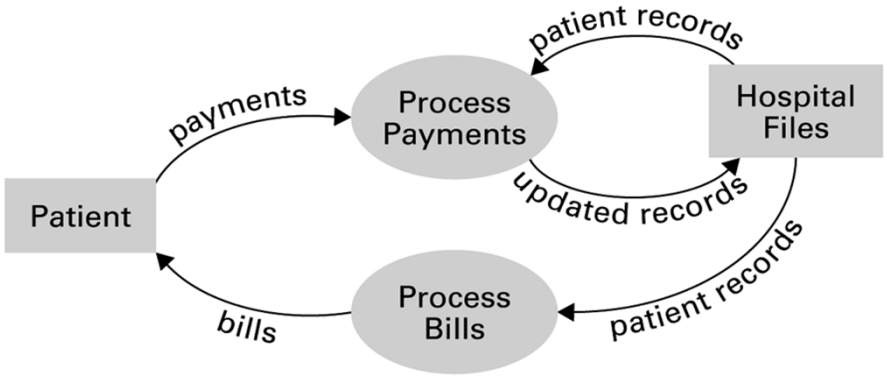
Modelling techniques and Notational Systems used during the analysis and design stages of software development.

- Data Flow Diagram
- Data Dictionary
- Unified Modelling Language (UML)
- Structured Walkthroughs
- Design Patterns

A simple dataflow diagram (Fig 7.8)

Means of representing information gained from data flow studies.

- Arrows represent data paths
- Ovals represent data manipulation (processes)
- Rectangles represent data sources and stores



Tools of the Trade

Data Dictionary

Central repository of information about the data items appearing throughout a software system.

1.1. Data Dictionary

Entity Name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Nullable	Unique
Employee	An employee is someone who work in a company	CompanyID		integer	10	false	false	false
		ID	For the unique identification of employee records.	integer	10	true	false	false
		name	Name of the employee.	varchar	255	false	true	false
		jobtitle	The position of the employee in a company.	varchar	50	false	true	false

 5

Tools of the Trade

Unified Modeling Language

Collection of tools developed with the object-oriented paradigm in mind.

- Use Case Diagram
  - Use cases
  - Actors
- Class Diagram
- Entity-Relationship Diagram
  - One-to-one relation
  - One-to-many relation
  - Many-to-many relation
- Class diagram depicting generalizations
- Interaction Diagrams

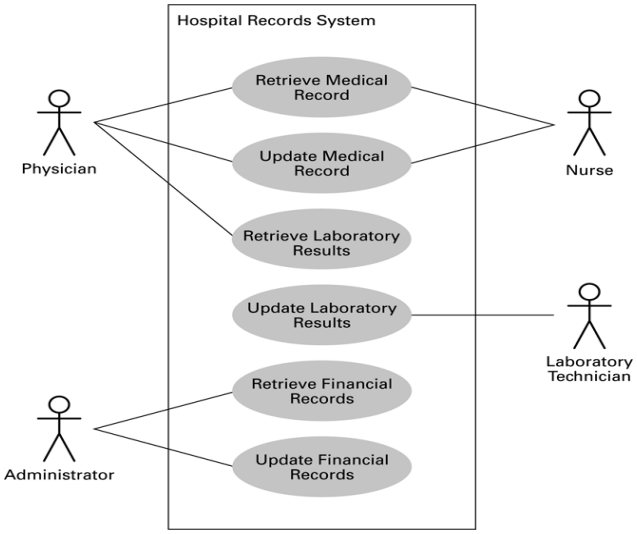
 6

Tools of the Trade: UML

Simple use case diagram (Fig 7.9)

It captures an image of the proposed system (large rectangle) from the user's point of view.

- Interactions (**use cases**) are represented by ovals
- Users of the system (**actors**) are represented by stick figures

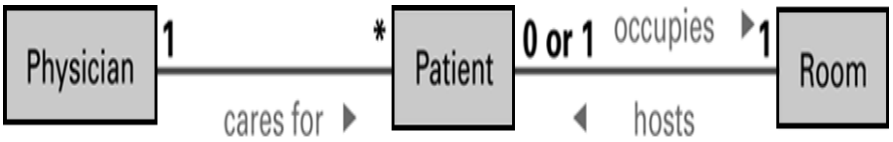


 7

Tools of the Trade: UML

A simple class diagram (Fig 7.10)

Notational system for representing the structure of classes (rectangles) and relationships (lines) between classes (**associations**).

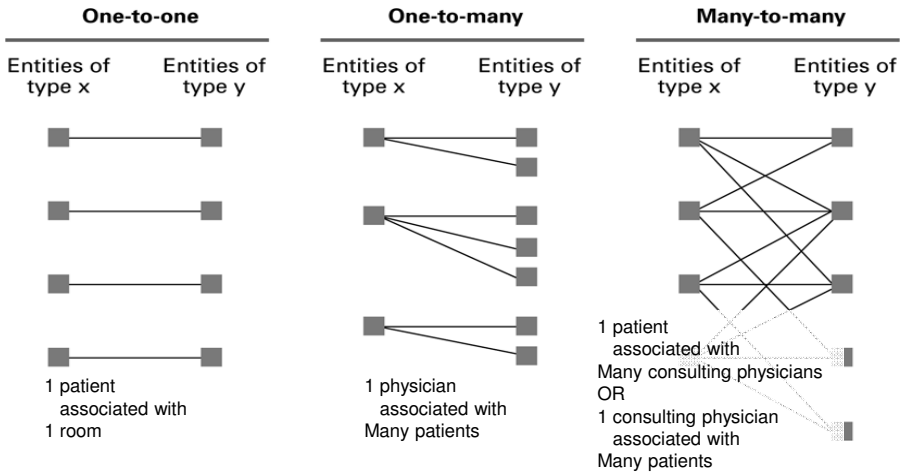


 8

Tools of the Trade: UML

Relationships between entities of types X and Y (Fig 7.11)

Associations occur in three basic forms:

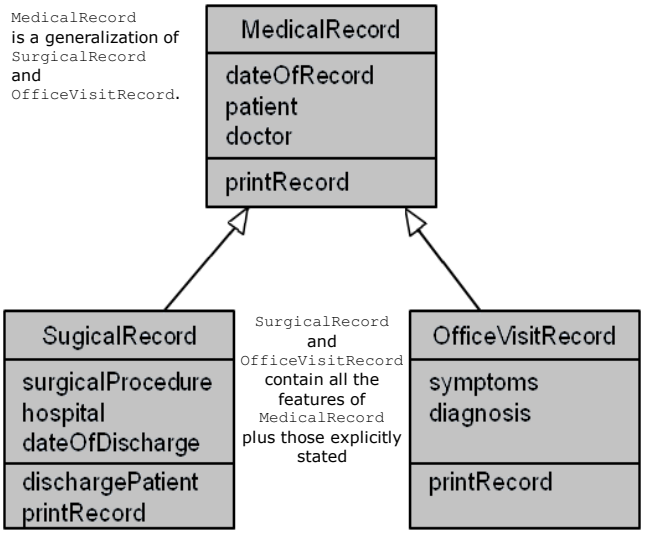


Tools of the Trade: UML

Class diagram depicting generalizations (Fig 7.12)

Depicts the generalizations among classes.

- Hallow arrowheads represents the association among classes
- Classes are represented by rectangles containing the class's:
  - Name
  - Attributes
  - Methods

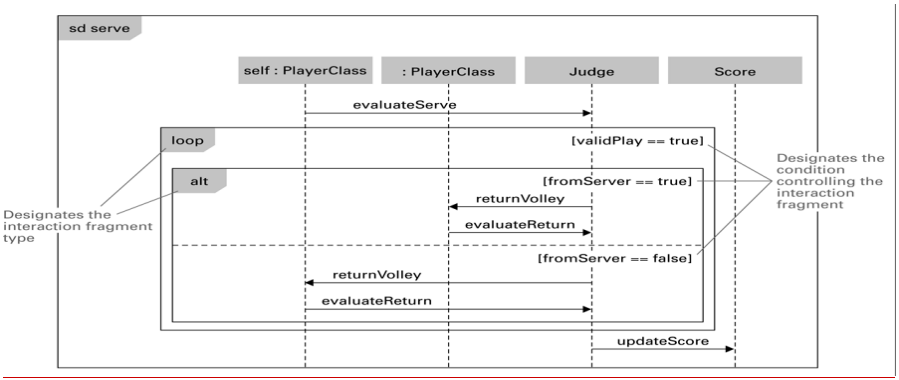


Tools of the Trade: UML – Interaction Diagrams

Sequence diagram depicting a generic volley (Fig 7.13)

Interaction Diagrams represent dynamic sequences of events that occur during execution.

Sequence Diagrams (a type of interaction diagram) depicts the communication between individuals (such as actors, components, or objects) involved in performing a task.



Tools of the Trade

Structured Walkthroughs

Found useful in identifying flaws in a design prior to the design's implementation.

- **Class-responsibility-collaboration** cards (**CRC**)
  - Description of an object
  - A CRC is produced for each object in a proposed system
  - Used in the simulation of the system
- Simulation conducted on a desktop or via a "Theatrical" experiment where design team members play the role of the system object described by their card

Tools of the Trade

Design Patterns

Well designed "templates" for solving recurring problems.

- Examples:
  - **Adapter pattern**: Used to wrap a desired prefabricated module whose interface is incompatible with an application, in another module so it can be used by the application
  - **Decorator pattern**: Used to control the complexity involved when many different combinations of the same activities are required
- Inspired by the work of Christopher Alexander in architecture

Quality Assurance

Software malfunctions, cost overruns, and missed deadlines require the use of software quality control methods.

- Scope

  Includes the debugging process, software engineering procedures, development of certification training programs, and the establishment of standards for software engineering.
- Two key QA activities in Software Engineering:
  - Software Testing
  - Documentation

Quality Assurance

Software Testing Strategies

Testing methodologies that improve the odds of revealing errors in software with a limited number of tests.

- **Glass-box testing**
- **Black-box testing**

Quality Assurance: Software Testing Strategies

Glass-box Testing

Relies on knowledge of the internal composition of the software being tested and uses this knowledge when designing the test.

- **Pareto principle**

  80/20 principle: a small number of modules within a large software system tend to be more problematic than the rest.

  - Therefore more of the system errors can be discovered/corrected by applying efforts in this concentrated number of modules.

- **Basis path testing**

  Impossible to test every path through the software system so a set of test data is developed that insures that each instruction in the software is executed at least once.

Quality Assurance: Software Testing Strategies

Black-box Testing

Does not rely on knowledge of the software's internal composition, it is performed from the user's point of view.

Main concern is not how the software goes about its task but whether the software performs correctly in terms of accuracy and timeliness.

- **Boundary value analysis**
  Identifies ranges of data, called **equivalence classes**, over which the software should perform in a similar manner.
  - Tests the software on data close to the edges of those ranges
  - This improves chances of error identification and minimizes the number of test cases
- **Beta testing**
  A preliminary version of the software is given to a segment of the intended audience. (Testing performed on the developer's site is **alpha testing**)
  - Learn how the software performs in real-life situations before it is released to the market.
  - Helps to discover errors but also aids in refining market strategies and the development of compatible products by other software developers.

 17

Quality Assurance

Documentation

- **User Documentation**
  Explains the features of the software and describes how to use them.
  - Printed book for all customers
  - On-line help modules
- **System Documentation**
  Describe the software's internal composition so that the software can be maintained later in its life cycle.
  - Source code
  - Design documents
- **Technical Documentation**
  Describes how a software system should be installed and serviced.
  - Utilized by users but typically intended for system administrators

 18

The Human-Machine Interface

Design of a system's interface for the convenience of its human users rather than merely the expediency of the software system.

The design of a system's interface can ultimately be the determining factor in the success or failure of a software engineering project.

Two key areas are:

- **Ergonomics**
  Designing systems that harmonize with the physical abilities of humans.
  E.g. static media such as: text, graphics or images, and dynamic media such as: audio, animation, video or media related to other sensory modalities.
- **Cognetics** "the ergonomics of the mind"
  Designing systems that harmonize with the mental abilities of humans.
  E.g. Persistent use of any interface will cause the user to develop habits therefore interfaces must be designed in such a way that they do not allow the user's development of habits to cause problems.
  An example of habit: Deletion confirmation messages

Software Ownership and Liability

Software developers need a level of ownership over the software they produce.

Legal efforts fall under the category of **intellectual property** law.

- **Copyright**
  - Allow a product to be released while retaining ownership of intellectual property
  - Asserted in all works:
    - Specifications
    - Source code
    - Final product
- **Software License**
  - A legal agreement that grants the user certain permissions without transferring ownership
- **Patents**
  - Must demonstrate that it is new, usable, and not obvious to others with similar backgrounds
  - Process is expensive and time-consuming

