

CSC 4181 – Compiler Construction Software Engineering Lectures

Part 2

Software Requirements

Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
 - May be the basis for a bid for a contract - therefore must be open to interpretation;
 - May be the basis for the contract itself - therefore must be defined in detail;
 - Both these statements may be called requirements.

Types of requirement

- User requirements
 - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
 - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Functional and non-functional requirements

- Functional requirements
 - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements
 - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
 - Requirements that come from the application domain of the system and that reflect characteristics of that domain.

Functional requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

Requirements imprecision

- Problems arise when requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term 'appropriate viewers'
 - User intention - special purpose viewer for each different document type;
 - Developer interpretation - Provide a text viewer that shows the contents of the document.

Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
 - They should include descriptions of all facilities required.
- Consistent
 - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Non-functional classifications

- Product requirements
 - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
 - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
 - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Problems with natural language

- Lack of clarity
 - Precision is difficult without making the document difficult to read.
- Requirements confusion
 - Functional and non-functional requirements tend to be mixed-up.
- Requirements amalgamation
 - Several different requirements may be expressed together.

Dr. Tom Way

CSC 4181

Slide 13

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.

Dr. Tom Way

CSC 4181

Slide 14

Object-oriented Design

Dr. Tom Way

CSC 4181

Slide 15

Objects and object classes

- Objects are entities in a software system which represent instances of real-world and system entities.
- Object classes are templates for objects. They may be used to create objects.
- Object classes may inherit attributes and services from other object classes.

Dr. Tom Way

CSC 4181

Slide 16

The Unified Modeling Language

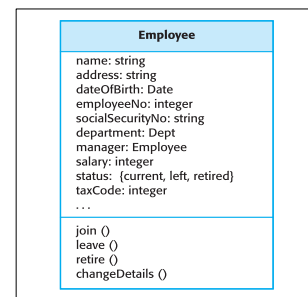
- Several different notations for describing object-oriented designs were proposed in the 1980s and 1990s.
- The Unified Modeling Language is an integration of these notations.
- It describes notations for a number of different models that may be produced during OO analysis and design.
- It is now a *de facto* standard for OO modelling.

Dr. Tom Way

CSC 4181

Slide 17

Employee object class (UML)



Dr. Tom Way

CSC 4181

Slide 18

Object communication

- Conceptually, objects communicate by message passing.
- Messages
 - The name of the service requested by the calling object;
 - Copies of the information required to execute the service and the name of a holder for the result of the service.
- In practice, messages are often implemented by procedure calls
 - Name = procedure name;
 - Information = parameter list.

Dr. Tom Way

CSC 4181

Slide 19

Message examples

```
// Call a method associated with a buffer
// object that returns the next value
// in the buffer
v = circularBuffer.Get () ;

// Call the method associated with a
// thermostat object that sets the
// temperature to be maintained
thermostat.setTemp (20) ;
```

Dr. Tom Way

CSC 4181

Slide 20

Generalisation and inheritance

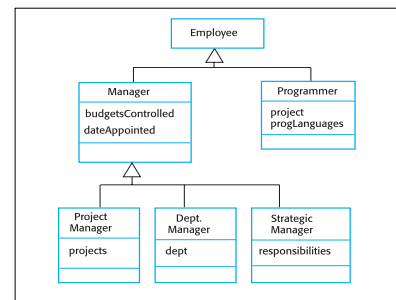
- Objects are members of classes that define attribute types and operations.
- Classes may be arranged in a class hierarchy where one class (a super-class) is a generalisation of one or more other classes (sub-classes).
- A sub-class inherits the attributes and operations from its super class and may add new methods or attributes of its own.
- Generalisation in the UML is implemented as inheritance in OO programming languages.

Dr. Tom Way

CSC 4181

Slide 21

A generalisation hierarchy



Dr. Tom Way

CSC 4181

Slide 22

UML associations

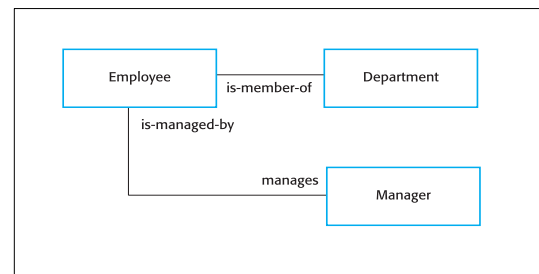
- Objects and object classes participate in relationships with other objects and object classes.
- In the UML, a generalised relationship is indicated by an association.
- Associations may be annotated with information that describes the association.
- Associations are general but may indicate that an attribute of an object is an associated object or that a method relies on an associated object.

Dr. Tom Way

CSC 4181

Slide 23

An association model



Dr. Tom Way

CSC 4181

Slide 24

Use-case models

- Use-case models are used to represent each interaction with the system.
- A use-case model shows the system features as ellipses and the interacting entity as a stick figure.

Use-cases for the weather station

