FACOLTÀ DI INGEGNERIA
UNIVERSITÀ DEGLI
STUDI DI PALERMO

Corso di Laurea Specialistica
Ingegneria Gestionale

Sistemi ICT per il Business Networking

Software Development Processes

Docente: **Vito Morreale** (vito.morreale@eng.it)

The essence of software development

- Only **one out of three** software projects complete on-time and on-budget (The Standish Group report, 2003)
- The **essence** of software development:
  - defined by the issues inherent in the software itself → software is a product of a **creative act** (not a result of a repetitive act of manufacturing)
  - **issues** inherent in the software
    - complexity
    - changeability
    - invisibility

The accidents of software development

- '**Accidental difficulties**' due to software production practices and human intervention
  - an information system is a social system
- Difficulties are **related to**:
  - Stakeholders
  - Process
  - Modeling language and tools

Software development invariants

- Software in inherently **complex**
  - function of interdependencies between components of which the software product is composed → complexity in the wires
  - Data intensive applications (enterprise information systems) are particularly complex
- Software must **conform** to:
  - hardware/software platform on which it is built
  - pre-existing information systems
- Software must be built to accommodate **change**
- Software is buried deeply in "**invisible**" programming statements, binary library code, and surrounding system software

Stakeholders

- People who have a role in a software project: whoever is affected by the system or affects the system
- Important groups of stakeholders:
  - Customers (users and system owners)
  - Developers (analysts, designers, programmers, etc.)
- Information systems are **social systems** → developed by people (developers) for people (customers)
- The main causes of software failure can be traced to the stakeholder factor
  - **on the customer end** (e.g. user needs not fully understood, changes of requirements, customers not fully involved, etc.)
  - **on the developer end** (e.g. development team members without required skills)

Software process

- Defines activities and organizational procedures used in software production and maintenance
- A **process model**:
  - states an **order** for carrying out activities;
  - specifies what development **artifacts** are to be delivered and when;
  - **assigns** activities and artifacts to people/roles;
  - offers criteria for **monitoring** a project's progress, for **measuring** the outcomes, and for **planning** future projects.
- Not susceptible to standardization: every organization should define its own process models or specialize an existing models (e.g. UP, RUP, etc.):
  - **Different contexts** require different (perhaps similar) processes

Software development lifecycle

- Macro phases:
  - **Analysis**: requirements (functional and non-functional) ← WHAT
  - **Design**: architectural and detailed ← HOW
  - **Implementation**: coding
  - **Testing**
  - **Deployment**

Analysis vs. design

- **Analysis** emphasizes an investigation of the problem and requirements, rather than a solution
  - For example, if a new computerized library information system is desired, how will it be used?
- **Design** emphasizes a conceptual solution that fulfils the requirements, rather than its implementation
  - For example, a description of a database schema and software objects.

Analysis: do the right thing

Design: do the thing right

Iterative and incremental process

- An **iterative** process is one that involves managing a stream of executable releases
- An **incremental** process is one that involves the continuous integration of the system's architecture to produce these releases, with each new release embodying incremental improvements over the other.
- Some examples:
  - Unified Process
  - Rational Unified Process (RUP)
  - the agile development processes
- Iterative and incremental development
  - must be **planned and controlled**

Modeling language and tools

- Software development artifacts have to be
  - **communicated** (visual or non-visual language) and
  - **documented** (tools)
- The **Unified Modeling Language (UML)** is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system.
- **Computer-Assisted Software Engineering (CASE)** tool enables storage and retrieval of models in a central repository and graphical and textual manipulation of models on a computer screen

Unified Modeling Language (UML)

- UML is **independent** of
  - any **software development process**
    - A process that adopts UML should support an object-oriented approach to software production, even though ...
  - **implementation technologies** (as long as they are object-oriented)

Lifecycle activities

- Business Analysis
- Requirements determination and specification
  - functional and non-functional requirements
- System Design
  - architectural design
  - detailed design
- Implementation
- Integration and Deployment
- Maintenance
- Testing
- Planning

Business analysis

- Also **requirements analysis**
- Activity of determining and specifying **customer needs**
  - business analyst determines requirements
  - system analyst specifies (or models) requirements
- Business analysis is linked to **business process reengineering** (BPR)
  - aim of BPR is to propose new ways of conducting business and gaining competitive advantage
- Business analysis becomes increasingly an act of **requirements engineering**

Requirements

- **Requirement**: 'a statement of a system service or constraint'. Capabilities and conditions to which the system must conform.
- **Service statement**
  - a business rule that must be obeyed at all times (e.g. 'salaries are paid on Wednesdays')
  - a computation that the system must carry out (e.g. 'calculate salesperson commission based on the sales in the last month using a particular formula')
- **Constraint statement**
  - a restriction on the system's behavior ('only direct managers can see the salary information of their staff')
  - a restriction on the system's development ('we must use Java development tools')

Requirements determination

- To **identify**, **analyze** and **negotiate** requirements with the customers
- Several **techniques**:
  - Structured and non-structured interviews
  - Questionnaires
  - Analysis of existing documents
  - Rapid prototyping
  - ...
- Attention to overlapping and contradicting requirements
- **Output**: requirements document (text, diagrams, tables, no formal models) ← a **contract** between developers and customers

Requirements specification

- Begins when the developers start **modeling the requirements** using a particular technique (such as UML)
  - **CASE tool** is used to enter, analyze and document the models
  - **Requirements Document** is enriched with graphical models and CASE-generated reports
- Most important OO specification techniques
  - **class diagrams**: to represent data
  - **use case diagrams**: to represent functions
- Ideally, the specification models **should** (if possible) be independent from the hardware/software platform on which the system is to be deployed ... Remember: requirements concern with **WHAT** !!!

Architectural design

- The **description of the system** in terms of its **modules** (components)
- Concerned with modularization of the system
  - relatively independent from a solution strategy but the detailed design of components must conform to a selected client/server solution

Detailed design

- Description of the internal workings of each software component
- Develops detailed algorithms and data structures for each component
- Dependent on the underlying implementation platform
  - Client (Windows, Unix, Web, stand-alone application, etc.)
  - Server (DBMS, legacy systems, etc.)
  - middleware (application server, etc.)

Implementation

- Involves
  - **installation** of purchased software
  - **coding** of custom-written software
  - other important activities, such as loading of test and production databases, testing, user training, hardware issues, etc.

Integration and deployment

- **Module integration** can take more time and effort than any one of the earlier lifecycle phases, including implementation
  - '**The whole is more than the sum of the parts**' (Aristotle)
  - must be carefully planned from the very beginning of the software lifecycle
- **Deployment** must be carefully managed and allow, if at all possible, fallback to the old solution, if problems encountered

Maintenance

- **Maintenance** is not only an inherent part of the software lifecycle – it accounts for most of it as far as IT personnel **time and effort** is concerned

Activities spanning the lifecycle

- Project planning
- Metrics
- Testing

Project planning

- If you can't plan it, you can't do it
- Activity of estimating the project's deliverables, costs, time, risks, milestones, and resource requirements
- Includes the selection of development methods, processes, tools, standards, team organization, etc.
- A moving target
- Typical constraints are time (deadline) and money (budget)

Metrics

- Measuring development **time** and **effort**
- **Without measuring the past, the organization is not able to plan accurately for the future**
- Metrics are usually discussed in the context of software quality and complexity – they apply to the quality and complexity of the software product
- Equally important application of metrics is measuring the development models (development products) at different phases of the lifecycle → to assess the effectiveness of the process and to improve the quality of work at various lifecycle phases

Metrics

- **Typical metrics**:
  - Stability of requirements
  - Number of components, modules, classes, etc.
  - Statistics on errors
  - ...

Testing

- **During the whole life-cycle** ... NOT after developing
- **Test cases** should be defined for each functional module described in the requirements document
- Desktop testing by developers not sufficient: test should be made by third party
- Methodical testing by Software Quality Assurance (SQA) group necessary
- Execution-based testing:
  - Testing to specs (black-box testing)
  - Testing to code (white-box or glass-box testing)

Summary

- The nature of software development is that of a **craft** or even **art**.
- The triangle for success includes the **stakeholders**' factor, a sound **process** and the support of a modeling **language and tools**.
- **System planning** precedes software development and determines which products can be most effective to the organization.
- Software development follows a **lifecycle**.
- Modern development lifecycles and processes are **iterative and incremental**.

References

- **Il processo di sviluppo del software** (Capitolo)