LAB Introduction and goals

- This lab follows a Problem Based Learning (**PBL** approach)
- This lab is intended to **experiment** how a **software engineering process** can be applied to build a software product satisfying a basic set of user requirements.
- Based on an informal specification of user and system requirements, participants will need to apply, analyze and compare both: **a plan-driven** and an **agile-oriented** software engineering process.
- Moreover, participant will work on the various **software engineering activities** (requirements specification, design, validation and evolution) using recommended **IEEE documents** templates

Informal user/system requirements

- The software product : a distributed (p2p) chat system.
- This system will allow users to communicate by sending and receiving text messages using interconnected devices. Optionally, users can also communicate by sending and receiving files (i.e. pictures, documents, programs, etc).
- Every user uses an username (or nickname) to connect to the chat system.
- When a user connects to the system, the list of the other connected users is presented. This list includes connected user names and information about their remote system (i.e. remote host information).
- Only connected users are able to communicate using the chat system functions.
- When any user connect (or log on) or disconnect (or log off), the other users have to be informed about it.
- When an user wants to communicate with another user (send a message or send a file), he has to select the remote user from the connected users' list. The message/file to be sent needs to be indicated. Optionally, a group of connected users could be selected as the destination.
- When the system receives a message or file targeted to the connected local user, the user has to be informed about it (i.e. showing the message or an indication about the received file).
- Technical requirements such as the kind of terminal, the network environment or the user interface need to be specified and refined during the analysis and design process.

Applying

Software Engineering!

Ernesto Exposito
ernesto.exposito@insa-toulouse.fr

Associate professor
INSA-Toulouse
LAAS/CNRS - Team SARA
http://www.laas.fr/~eexposit
http://www.ypbl.net

IEEE Software Document Standards

IEEE Software Document Definitions

- Software Project Management Plan (**SPMP**): specifies the structure of software project management plans that are applicable to any type or size of software project [5].
- Software Requirements Specification (**SRS**): specifies the structure and necessary qualities of software requirements specification documents [6].
- Software Design Description (**SDD**): proposes the necessary information content and recommendations for software design descriptions [7].
- Software Quality Assurance Plan (**SQAP**): specifies the format and content of software quality assurance plans [8].
- Software Configuration Management Plan (**SCMP**): describe the structure and content for a software configuration management applying to the entire life cycle of the software [9].
- Software Test Documentation (**STD**): this document defines the form of a set of documents for use in defined stages of software testing [10].
- Software Validation & Verification Plan (**SVVP**): specifies the structure of the validation and verification plan including analysis, evaluation, review, inspection, assessment, and testing of software products and processes [11].

Plan

Software Project Management Plan (**SPMP**)

This Project Management Plan (PMP) is intended to provide guidance on the management of the project. This template conforms to the Institute of Electrical and Electronics Engineers (IEEE) Standard for Software Project Management Plans, IEEE Std 1058-1998, for format and content. The document structure includes:

a. *Section 1, Project Overview: overview of the scope and objectives of the project, the project's assumptions and constraints, reference to the project deliverables, schedule and budget, and a description of the evolution of the plan.*
b. *Section 2, References: list of all documents, policies, templates, processes, and other sources of information referenced in the plan.*
c. *Section 3, Definitions: abbreviations and acronyms required to properly understand this planning document.*
d. *Section 4, Project Organization: identifies interfaces to organizational entities external to the project, the project's internal organizational structure, and defines roles and responsibilities for the project.*
e. *Section 5, Management Process: describes the planning, measurement, tracking, reporting, risk control mechanisms needed to provide management control over the technical processes and product quality.*
f. *Section 6, Technical Process: describes the technical solution in terms of a process model and implementation methods, tools, and techniques to be used to develop the various work products, plans for establishing and maintaining the project infrastructure, and the product acceptance.*
g. *Section 7, Supporting Processes: describes processes that are employed to facilitate and control the technical processes and the state of the product. These include, but are not limited to, configuration management, verification and validation, documentation, quality assurance, reviews and audits, problem resolution, and methods to ensure continuous process improvement.*

Requirements

Software Requirements Specification (**SRS**)

A SRS provides a description of the software requirements in order to start the design and development process.
The SRS describes the contract that needs to be satisfied by the software product.
This document is intended to be used to collect, translate and classify the requirements. These requirements can be explicitly expressed by the clients (product owner, users or other stakeholders) or can be implicitly deduced as a result of meetings, presentations, interviews, etc.
Requirements need to be "validated" by the clients. In order to guarantee an adequate validation, requirements needs to be represented not only in textual form, but also using standard modeling notations such us UML/SysML (e.g. use cases, sequence, class, activity, statechart diagrams, etc.).

A matrix of well identified/classified/approved requirements needs to be produced during the requirement analysis process. This matrix will be use to trace how the software product satisfies the requirements during the whole process (i.e. the various iterations or releases).
More information about requirements traceability matrix can be found at http://en.wikipedia.org/wiki/Traceability_matrix)

SRS Template (IEEE Std 830)

1 Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions, Acronyms, and Abbreviations
1.4 References
1.5 Overview

2 General Description
2.1 Product Perspective
2.2 Product Functions
2.3 User Characteristics
2.4 General Constraints
2.5 Assumptions and Dependencies

Use case diagram

Sequence, activity, state chart diagrams and textual use cases

3 Specific Requirements
3.1 Functional Requirements
3.2 External Interface Requirements
3.2.1 User Interfaces
3.2.2 Hardware Interfaces
3.2.3 Software Interfaces
3.2.4 Communications Interfaces
3.3 Performance Requirements
3.4 Design Constraints
3.4.1 Standards Compliance
3.4.2 Hardware Limitations
3.5 Attributes
3.5.1 Availability
3.5.2 Security
3.5.3 Maintainability
3.5.4 Transferability/Conversion
3.6 Other Requirements

Design

Software Design Description (**SDD**)

General information/recommendations

A SDD provides a representation of a software system created to facilitate analysis, planning, implementation, and decision making. It is a blueprint or model of the software system. The SDD is used as the primary medium for communicating software design information.

The SDD shows how the software system will be structured to satisfy the requirements identified in the software requirements specification (e.g. SRS IEEE 830). It is a translation of requirements into a description of the structure and behaviour of the software product/system, the software components, the interfaces, and the data necessary for implementing the software solution.

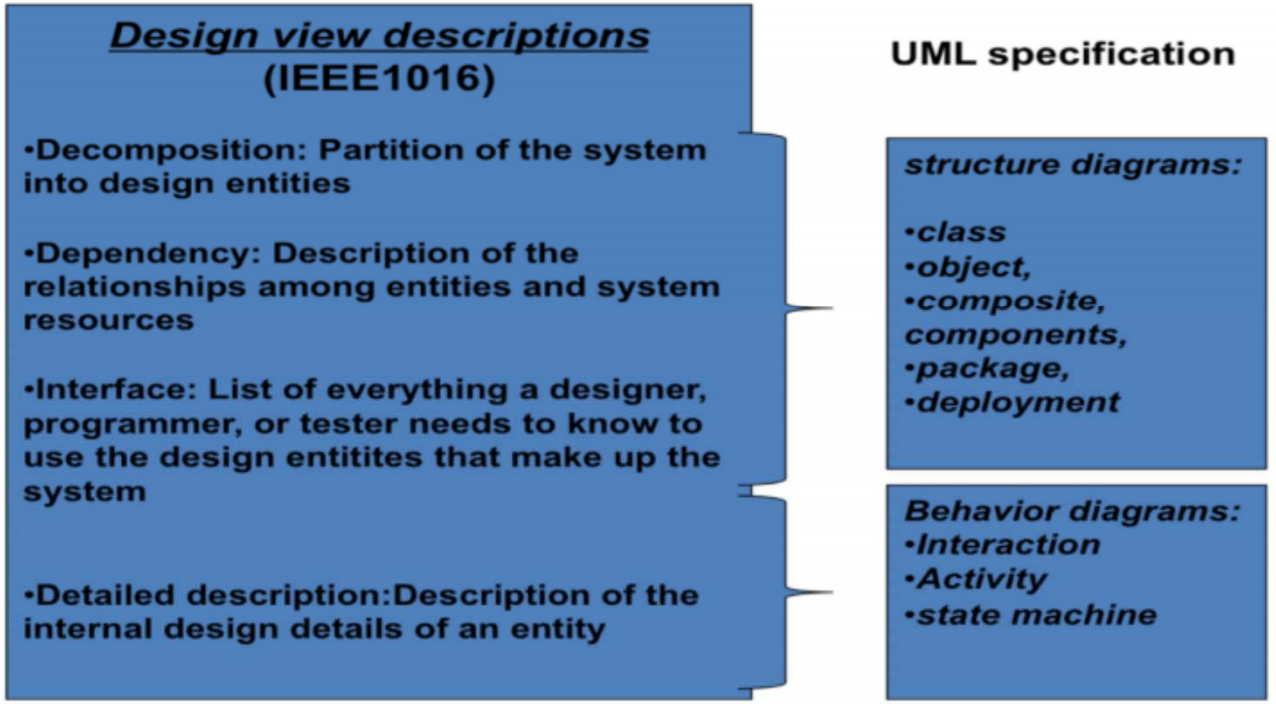
In a complete SDD, each requirement must be traceable to one or more design entities.

This template can be used directly or it can be adapted in order to better fit the followed software design methodology.

SDD Template (IEEE 1016)

- INTRODUCTION
  - Design Overview
- SYSTEM ARCHITECTURAL DESIGN
  - Description of System
  - System Architecture (Stand-alone, C/S, P2P, Multitiered, etc)
  - Identification of potential design patterns
  - System Interface Description
- COMPONENTS DESCRIPTION (**structural** system decomposition)
  - Component n
  - Component n+1
- EXTERNAL INTERFACES DESCRIPTION
  - Description external human or other systems interfaces
- DETAILED DESIGN (**behavioral** specification)
  - Component n
  - Component n+1
- ANNEXES - ADDITIONAL MATERIAL

SDD Template (IEEE 1016)



Tests

Software Test Document (**STD**)

A STD provides the basic strategies and plan defining the tests to be applied to evaluate the developed system.

This document contains the following information:

Test plan: A high level detail of tests including the subject/system to be tested, who is responsible, how and when is going to be tested.
Test design specification: details of the tests including expected outputs.
Test case specification: specification of the data required for the tests.
Test procedure specification: description of procedures to be performed by the testers.
Test Item Transmittal: description of tested items and the test stages.
Test log: log of performed tests
Test Incident Summary: details of failed tests.

Test Summary: general overview of the tests allowing to evaluate the overall state and quality of the system.

This template can be used directly or it can be adapted in order to better fit the followed software design methodology.