https://www.halvorsen.blog

Week Assignment

Unified Modeling Language (UML)

Hans-Petter Halvorsen

Week Assignment

1. **Create UML diagrams** for your part of the system and the overall system.
2. Start Implement/Update Code according to the UML Design

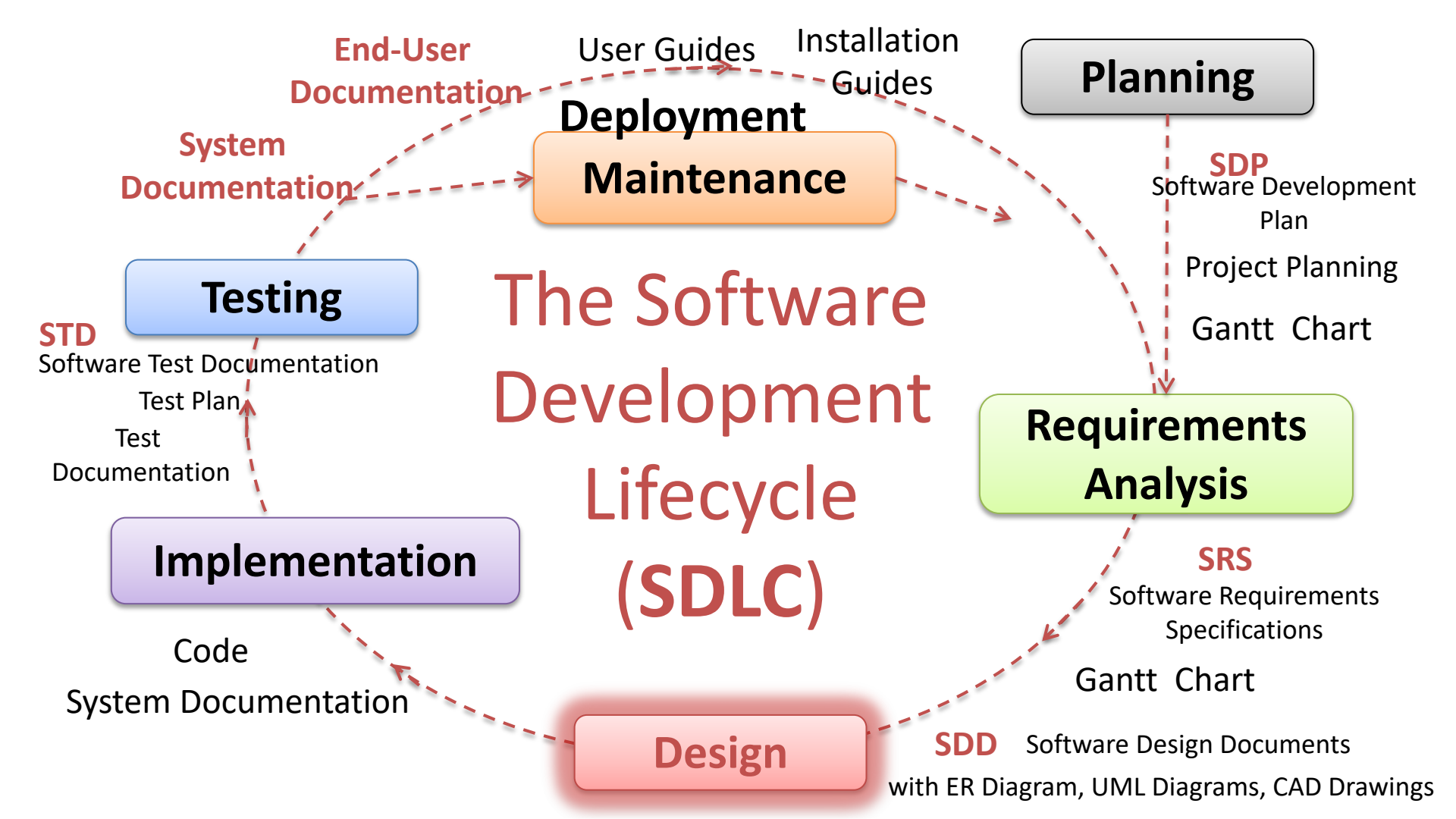
The UML diagrams with descriptions should be part of the Requirements and Design document(s): SRS/SDD →SRD

https://www.halvorsen.blog

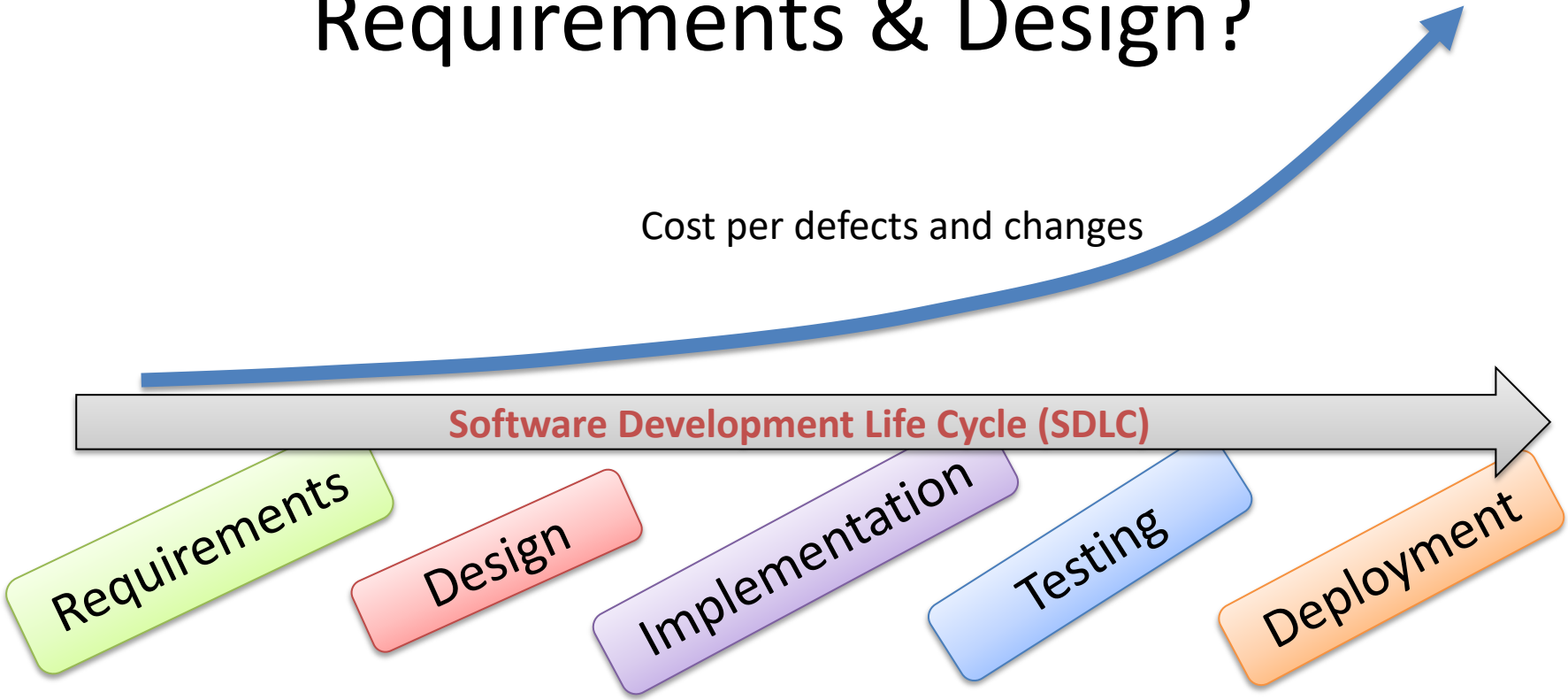
UML Design & Modelling

Hans-Petter Halvorsen

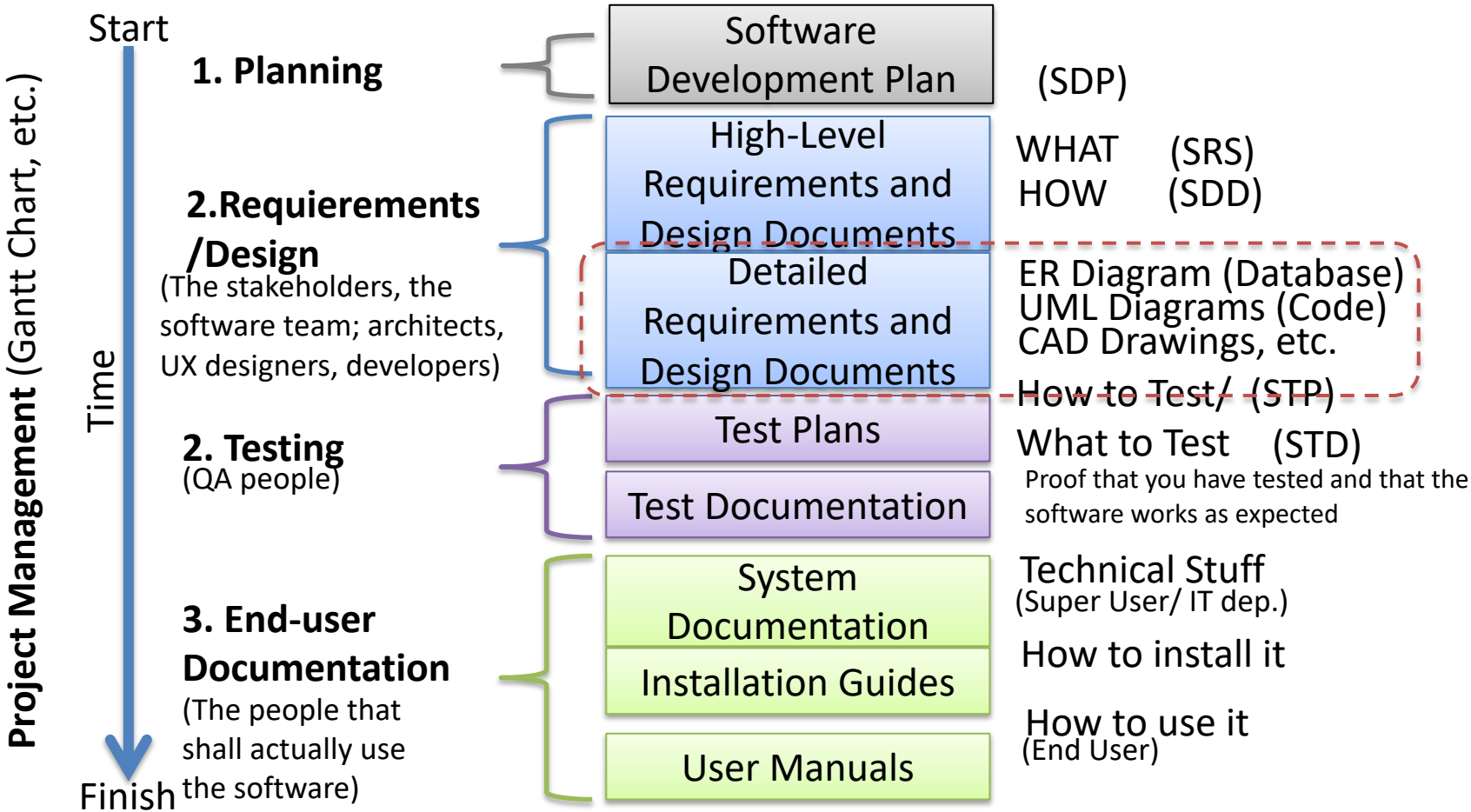
Table of Contents


End-User Documentation
User Guides
Installation Guides
Planning
Deployment
System Documentation
Maintenance
SDP
Software Development Plan
Project Planning
Gantt Chart
Testing
The Software Development Lifecycle (SDLC)
STD
Software Test Documentation
Test Plan
Test Documentation
Requirements Analysis
SRS
Software Requirements Specifications
Gantt Chart
Implementation
Code
System Documentation
Design
SDD Software Design Documents
with ER Diagram, UML Diagrams, CAD Drawings

Why spend time on Requirements & Design?



Typical Software Documentation



Software Requirements & Design

Requirements (WHAT):

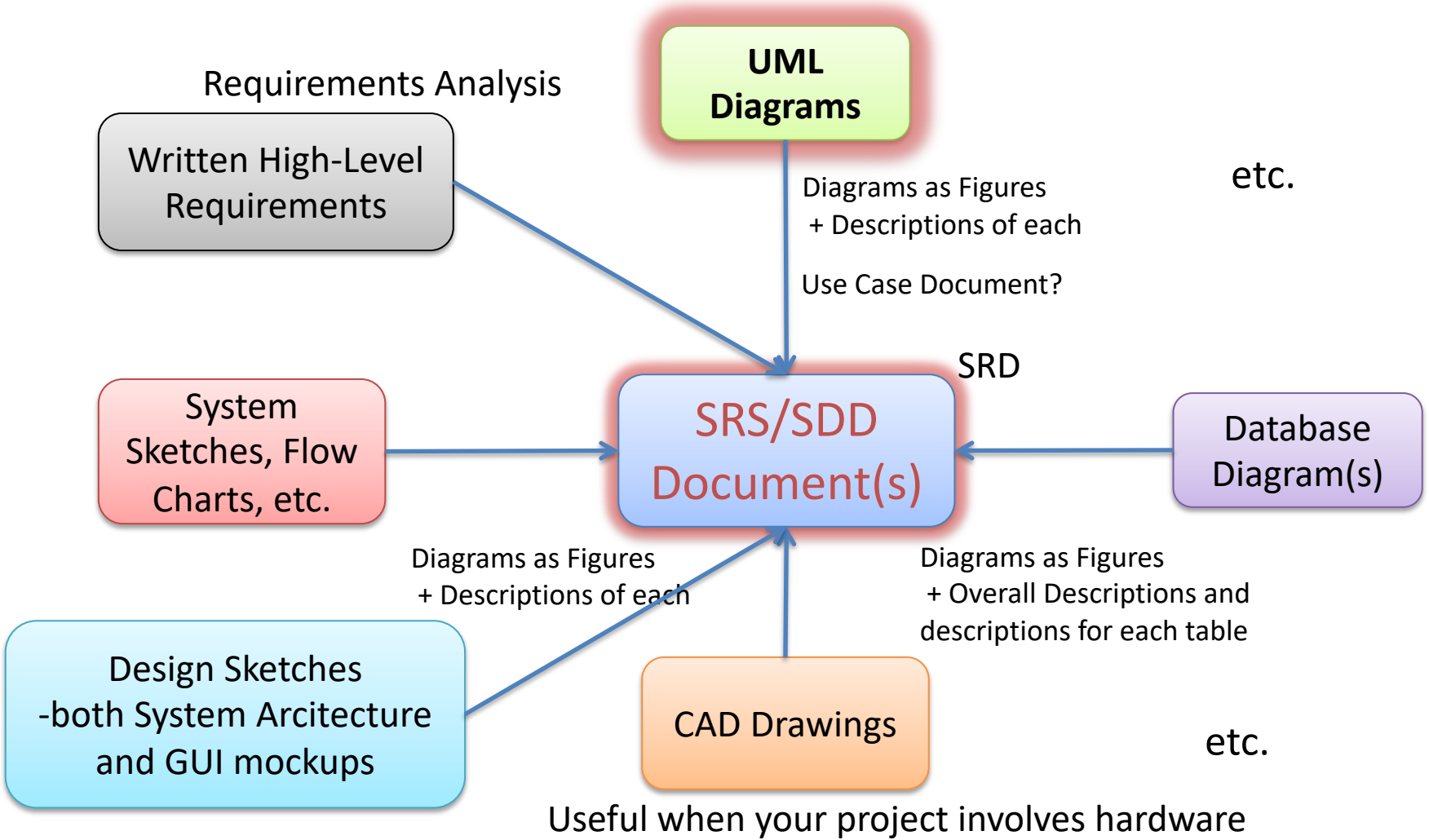
- **WHAT** the system should do
- Describe what the system should do with Words and Figures, etc.
- **SRS** – Software Requirements Specification

Software Design (HOW):

- **HOW** it should do it
- Examples: GUI Design, UML, ER diagram, CAD, etc.
- **SDD** – Software Design Document

Note! Many don't separate SRS and SDD documents but include everything in a Requirements & Design Document (SRD).

➔In practice, Requirements and Design are inseparable.


Requirements Analysis
Written High-Level Requirements
UML Diagrams
Diagrams as Figures
+ Descriptions of each
Use Case Document?
etc.
SRD
System Sketches, Flow Charts, etc.
SRS/SDD Document(s)
Database Diagram(s)
Diagrams as Figures
+ Descriptions of each
Diagrams as Figures
+ Overall Descriptions and descriptions for each table
Design Sketches
-both System Arcitecture and GUI mockups
CAD Drawings
etc.
Useful when your project involves hardware

Documents

The following Documents should be “finished”* within this Week Assignment!

- Software Development Plan (SDP)
- Software Requirements and Design (SRD)
  - Including Database and UML modelling with textual descriptions and explanations

*But they can and should be continuously updated throughout the project when changes occur

UML Diagrams

Hans-Petter Halvorsen

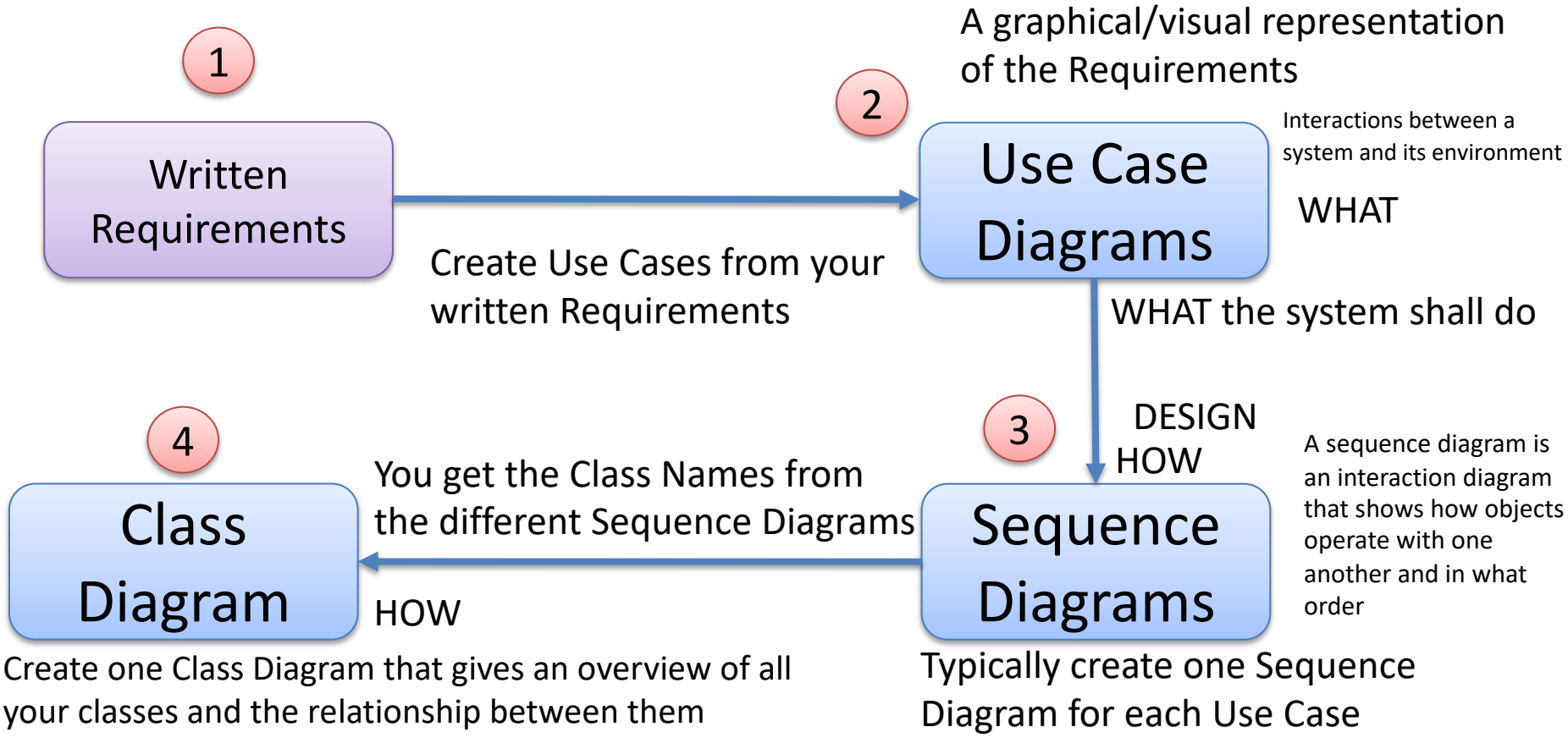
Table of Contents

UML Diagrams

- The following UML diagrams for your system should be created:
  1. **Use Case Diagrams**
  2. **Sequence Diagrams**
  3. **Class Diagrams**
- **UML Tools**: We will use **StarUML** as our UML Tool
- Include the UML diagrams (with detailed descriptions!) as part of your Requirements and Design Document (SRD)

See Next Slides for more details...

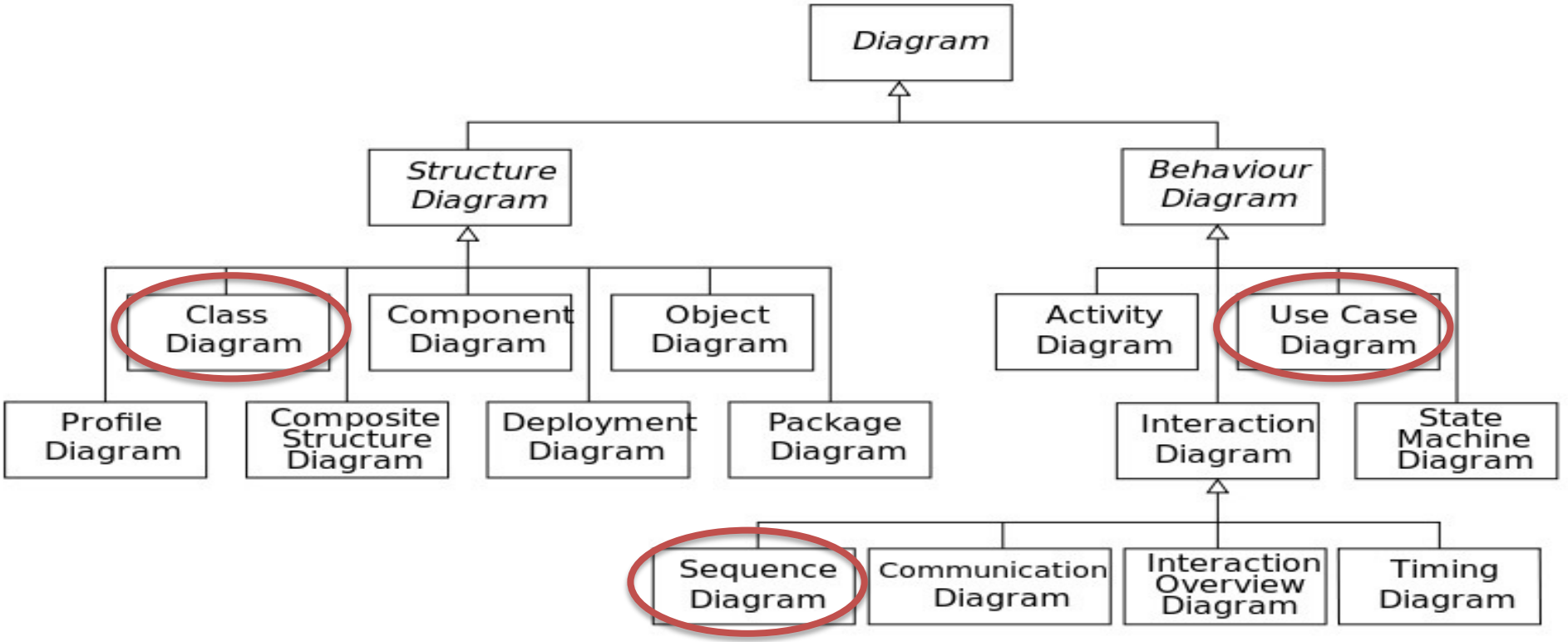
Creating UML - A practical Approach



Why use UML?

- Design
  - Forward Design: doing UML before coding. Makes it easier to create the code in a structured manner
  - Backward Design: doing UML after coding as documentation
  - When doing changes in the Design, make sure to update the Code according to the new Design
- Code
  - Some Tools can Auto-generate Code from UML diagrams
  - When doing changes in the Code, make sure to update the UML Diagrams

Types of UML Diagrams



http://en.wikipedia.org/wiki/Unified_Modeling_Language

UML Diagrams

Main Diagrams:

- Requirements Analysis Phase (WHAT):
  1. **Use Case Diagrams**
- Design Phase (HOW):
  2. **Sequence Diagrams** (Typically one Sequence diagram for each Use Case)
  3. **Class Diagrams** (just one Class diagram in total, for larger systems you may want to create several)

Use Case Diagram

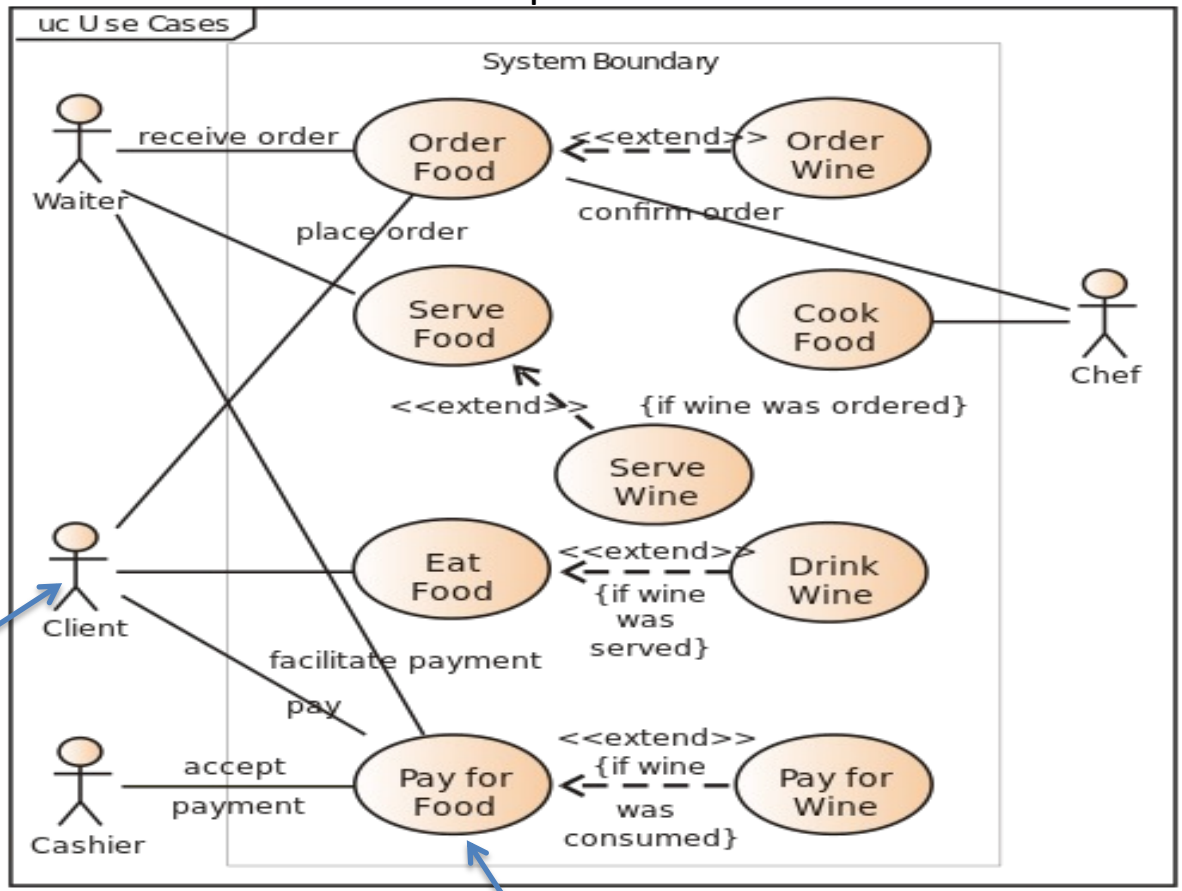
- A Use Case diagram is a representation of a user's interaction with the system
- It shows the relationship between the user and the different use cases in which the user is involved.
- Think as an End User not a Programmer: How should the user use the system?
- Use your list of Requirements (as stated in the SRS/SRD) as a starting point for your Use Cases
- The Use Cases should start with a verb, e.g., ReadSensor, RegisterPatient, ShowPatientData

Use Case Diagram

A use case is a list of steps, typically defining interactions between a role (known in UML as an "**actor**") and a system, to achieve a goal.

The actor can be a human or an external system.

Use Case Example for a Restaurant:



Actor

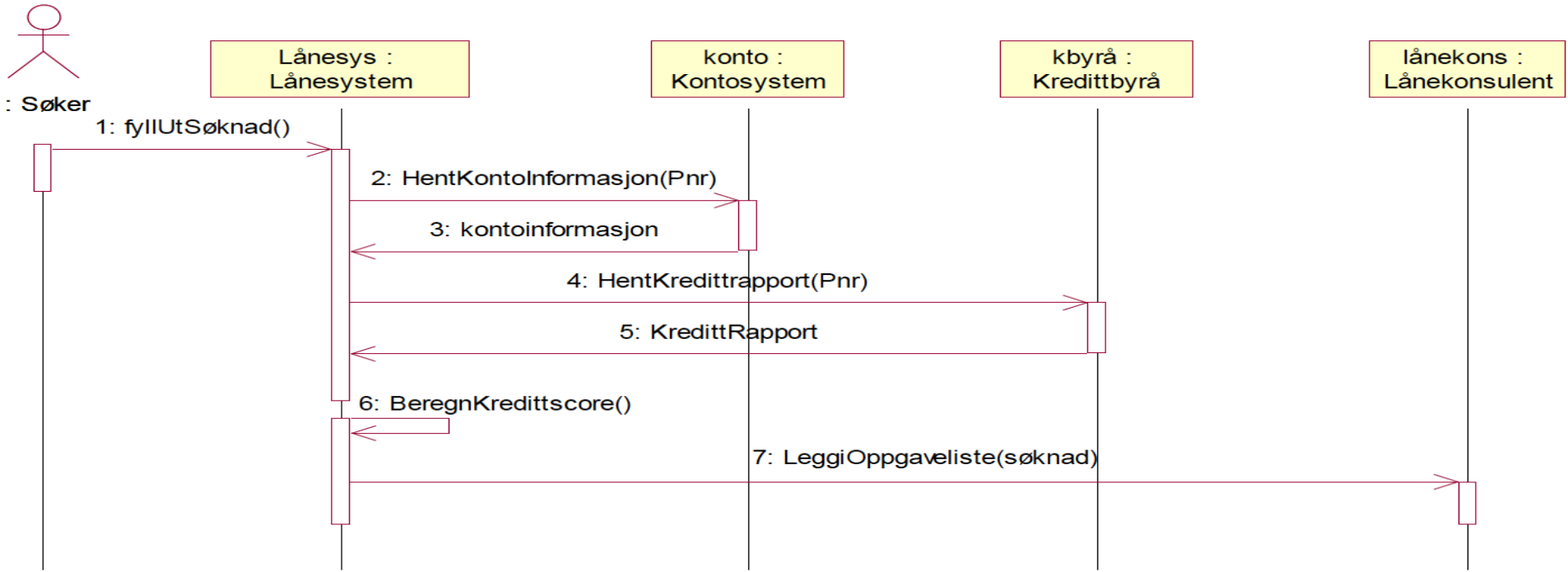
Use Case

http://en.wikipedia.org/wiki/Use_case

Sequence Diagram

- A Sequence diagram is an Interaction Diagram that shows how objects operate with one another and in what order
- A Sequence diagram shows object interactions arranged in time sequence.
- Focus on main Structure, not Details. The details are done in Code

Sequence Diagram

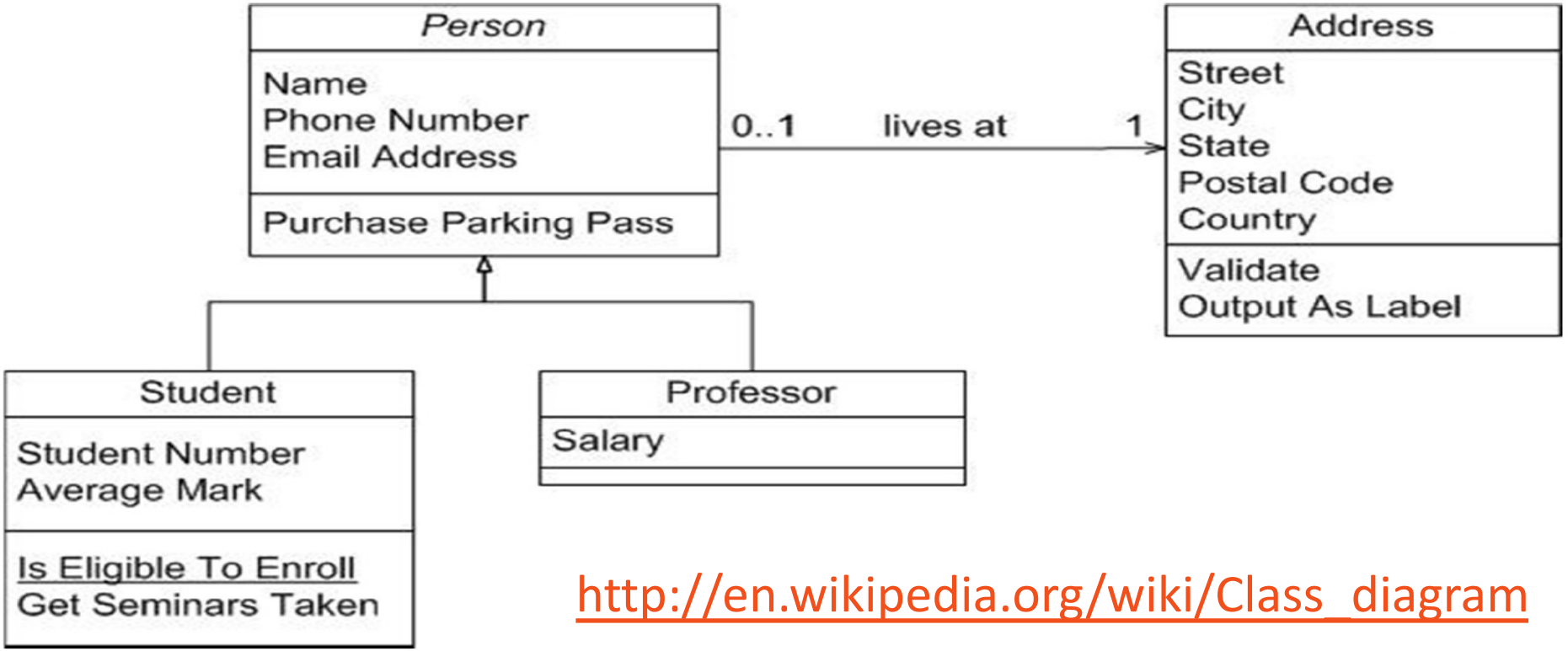


http://en.wikipedia.org/wiki/Sequence_diagram

Class Diagram

- A diagram that shows the structure of the different Classes in a system
- It shows the relationships between the Classes
- It shows Methods and Properties for each Class

Class Diagram



http://en.wikipedia.org/wiki/Class_diagram

UML Software

- MS Visio
- StarUML
- ...hundreds of different tools

https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

FURPS+

FURPS is an acronym representing a model for classifying software quality attributes (functional and non-functional requirements):

- **Functionality** - Capability (Size & Generality of Feature Set), Reusability (Compatibility, Interoperability, Portability), Security (Safety & Exploitability)
- **Usability (UX)** - Human Factors, Aesthetics, Consistency, Documentation, Responsiveness
- **Reliability** - Availability (Failure Frequency (Robustness/Durability/Resilience), Failure Extent & Time-Length (Recoverability/Survivability)), Predictability (Stability), Accuracy (Frequency/Severity of Error)
- **Performance** - Speed, Efficiency, Resource Consumption (power, ram, cache, etc.), Throughput, Capacity, Scalability
- **Supportability** -(Serviceability, Maintainability, Sustainability, Repair Speed) - Testability, Flexibility (Modifiability, Configurability, Adaptability, Extensibility, Modularity), Installability, Localizability
- **+** (extra) - Implementation, licenses, administration, interface to external systems, etc.

https://en.wikipedia.org/wiki/FURPS

FURPS+ Examples of Questions you may want to ask yourself

- **Functionality** - What the customer wants! Note that this includes security-related needs.
- **Usability** - How effective is the product from the standpoint of the person who must use it? Is it aesthetically acceptable? Is the documentation accurate and complete?
- **Reliability** - What is the maximum acceptable system downtime? Are failures predictable? Can we demonstrate the accuracy of results? How is the system recovered?
- **Performance** - How fast must it be? What's the maximum response time? What's the throughput? What's the memory consumption?
- **Supportability** - Is it testable, extensible, serviceable, installable, and configurable? Can it be monitored?

http://agileinaflash.blogspot.no/2009/04/furps.html

FURPS+ in our Project

- You don't need to follow FURPS+ in your project
- We can use it as a tool/aid when creating the Software Requirements Specifications in our SRD document
- If you want, and find it useful, you can also structure the requirements according to FURPS+

“Use Case Document”

- Purpose: Documenting Use Cases
- Textual documentation and explanations of your Use Cases
- **In our Project**:
  - We include UML diagrams and textual descriptions of these diagrams in our SRD document
  - “Fully dressed use case document”: We can use it as a tool/aid when creating the diagrams, but it is not needed

Use Case and Scrum (Agile)

- In Agile, we don't refer to requirements; instead, we talk about stories. Stories are really reminders for customer needs (requirements, in the general sense).
- The Team works closely together with the Product Owner
- Less need for detailed descriptions and requirements
- Agile/Scrum uses **User Stories** instead (which could be considered as a light version of Use Case)
- The User Stories are the base for the **Product Backlog** and the Sprint Backlog

UML Summary

- You should create Design and Specifications (including UML) before you start Coding
- But UML can also be used to document your code afterwards (Reverse Engineering)
- UML diagrams is a general method/standard to do just that
- This makes it easier to create structured code
- A good way to document your code properly.
- Code refactoring: Use UML as part of the continuous code improvements process
- **Note!** If you update the code, make sure to update the UML and vice versa!
- An (non-graphical) alternative to UML Use Cases is User Stories

Coding

Hans-Petter Halvorsen

Table of Contents

Coding

- Make the overall Code Structure according to the UML Design
- Start creating Classes, etc. according to your UML design
- It may be a good idea to create a Class Library (or another form for API) that can be shared between your applications/modules.
- Make sure that you code reflects the UML design regarding classes, etc. If you update your code, you need to update the UML diagrams and vice versa.
- It is important that we have a working software at all times (so it can be reviewed, tested, etc.)! That is a basic requirement in Scrum!

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog