Software Requirements Engineering

Week 3

Asst. Prof. Bilge Kağan DEDETÜRK

Content

- Non-functional requirements (Remaining Parts)
- Process Models
- Process Actors

Non-Functional Requirements Testability

- Testability is whether or not the software and integrated products can be tested.

Non-Functional Requirements Availability

- Availability is the probability that a system will be operational to deliver the requested services at a point in time.

- Availability is expressed as the percentage of time the infrastructure, system, or solution remains operational under normal circumstances to serve its intended purpose.

Non-Functional Requirements Availability

- If a service is at 90% availability,
  - =>876 hours yearly service downtime
  - = 2.4 hours a day yearly service downtime
  - = 21.6 hours a day system availability

- If a service 99.999% availability,
  - 5256 minutes yearly service downtime

Non-Functional Requirements Availability

- We need to measure how many times the software crashes occur. Hence, we measure "main time between failures," which is the difference between crashes.

Non-Functional Requirements
Reliability

- If the system is available, it doesn’t mean that it will give you satisfactory results. The system should serve the intended purpose under varying and unexpected conditions.

- Reliability refers to the probability that the system will meet specific performance standards while providing correct output for the desired time duration.

- Reliability ensures that the software doesn't have many bugs that will slow down the user from accomplishing a task.

Non-Functional Requirements
Robustness

- Robustness is the degree to which the system continues to function correctly when encountered with invalid data.
- The system should continue to work even when presented with invalid data, which we call a robust system.
- If there is invalid data, the system can continue functioning and doesn’t interrupt the user activity with an error message.
- It is not very wise to make some systems robust.

Emergent Properties

- Emergent properties can only show when the entire software is finished.
- Emergent properties cannot be addressed by a single component but depend on how all the software components interoperate.
- Emergent properties are important in software or system design as they help in understanding how all components interact.

Emergent Properties

- **Performance**: Individual components of a software system may perform well on their own, but when combined, the overall system performance may decrease, especially when components must share resources.
- **Security**: Each component may be secure, but the interactions and data flows between components can create vulnerabilities that compromise the security of the entire system.
- **Reliability**: Components may be reliable on their own, but a failure in one component could cause a cascading failure in the entire system, affecting its reliability.

Quantifiable Attributes

- Quantifiable requirements are measurable and verifiable.
- It helps us to ensure that the requirements are clear and unambiguous.

Quantifiable Attributes

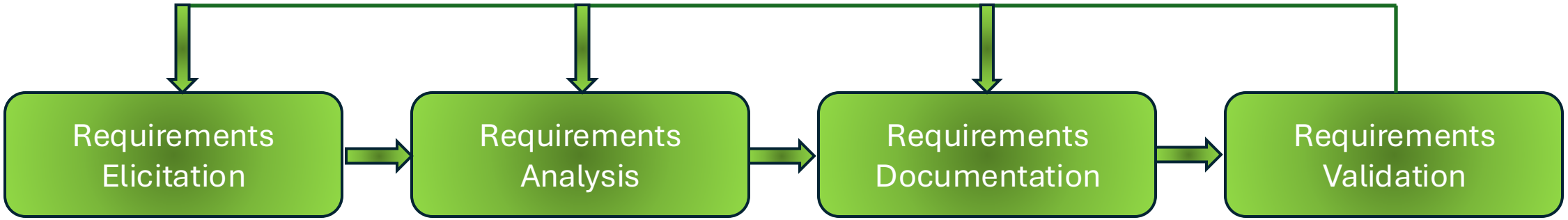
- How do I measure usability?
  - We can measure how many errors we meet during the actual operation of the software
- How to measure good-looking software?
  - We could measure based on the rate of acceptance.

Process Models

- To ensure that you will always produce a good quality product, we must follow a proven successful process.
- Therefore, requirements process models are a subset of the overall software development process model.
- The requirements process takes a business or an engineering problem and creates from it the specifications for a system that will provide a solution to that problem.

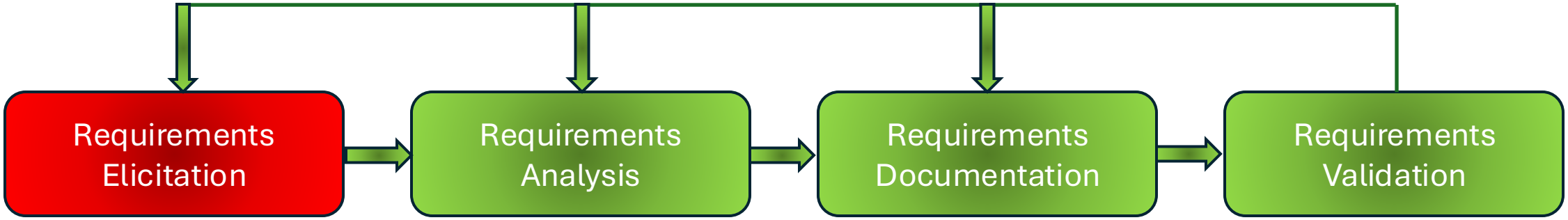
Iteratice Process

- There are four sets of activities that have been shown to produce specifications or requirements.



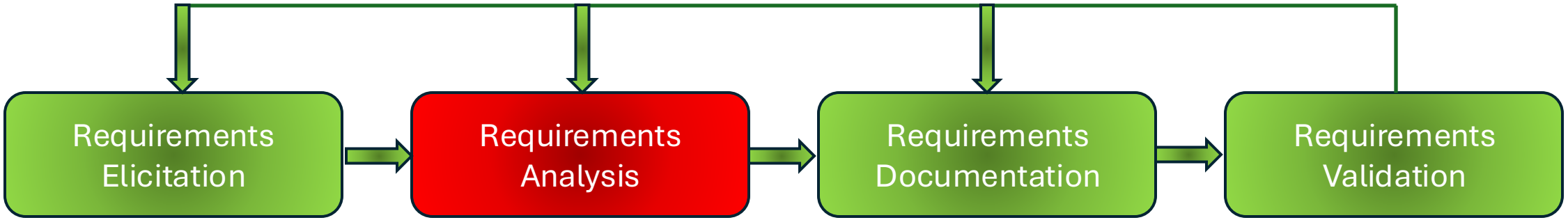
Requirements Elicitation

- Requirements Elicitation - whereby the information needed to develop and document the requirements are collected from the stakeholders.



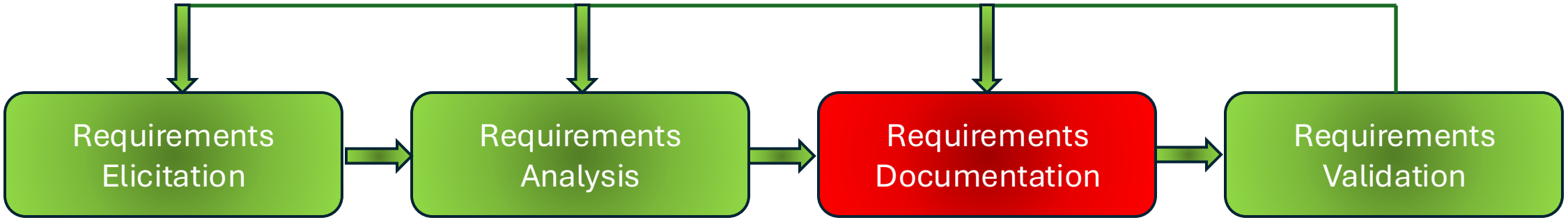
Requirements Analysis

- Requirements Analysis - we refine and improve the collected requirements.



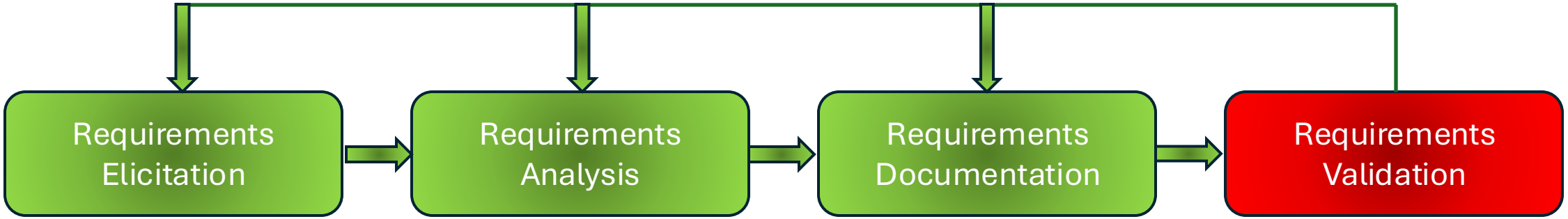
Requirements Documentation

- Requirements Documentation which creates the software and systems specifications needed to document the requirements.



Requirements Analysis

- Requirements Validation - uses modeling, reviews, prototypes, and acceptance testing of the final product to verify and validate the correctness of the requirements.



Aspects of Requirements Process

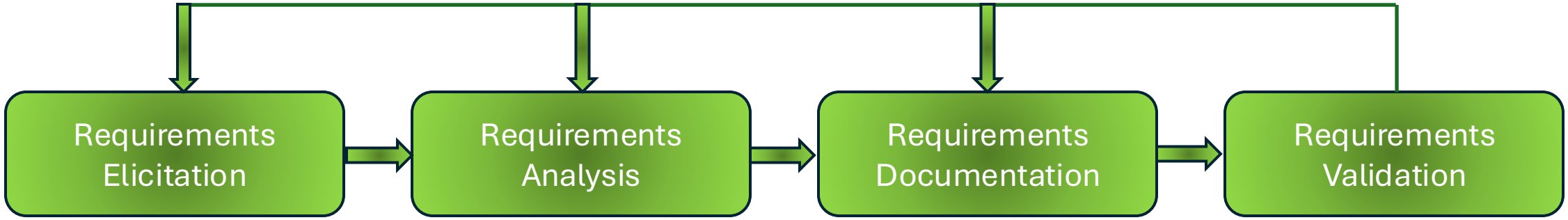
- The Requirements Process is not simply a front-end process

- Such new information usually alters some of the requirements, which leads to executing the requirements process repeatedly.

Aspects of Requirements Process

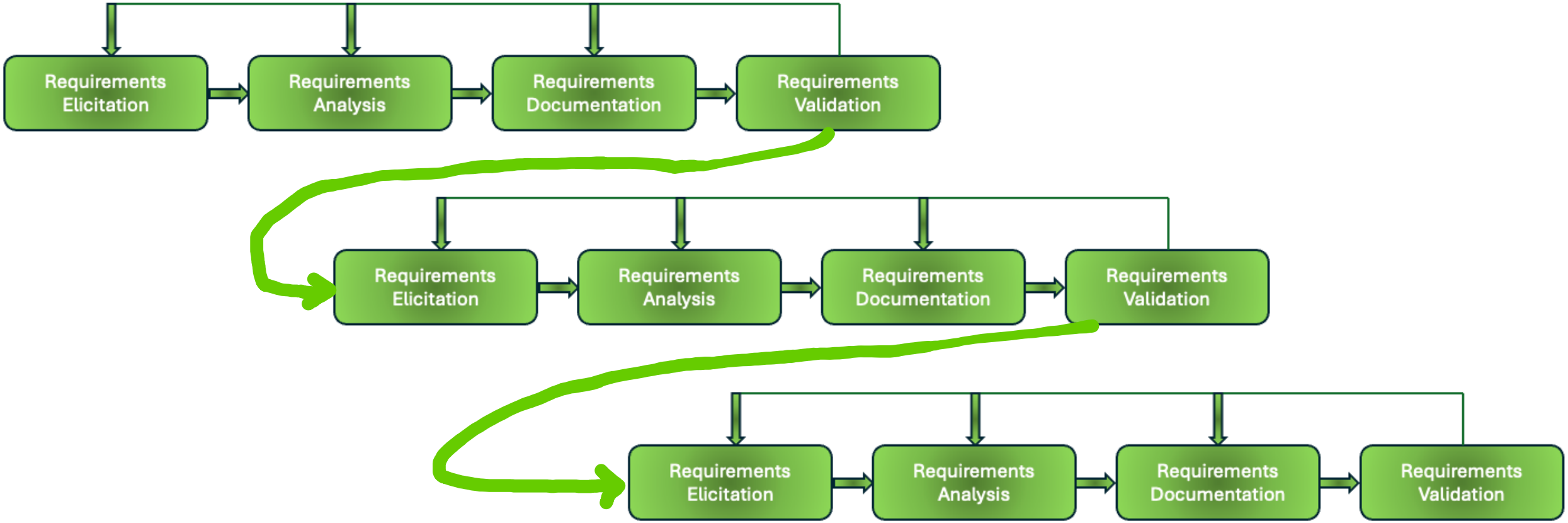
- The activities within the process itself are never strictly linear.

- The requirements process is iterative irrespective of linear or incremental model selection.



Aspects of Requirements Process

- Despite the development life cycle, we will follow the requirement process



Aspects of Requirements Process

- The requirements process needs to be flexible and easily tailored, and adaptable to different organization contexts.

Process Actors

- These are people who have an interest in the software engineering project / product.
- Stakeholders vary across projects.
- Stakeholders can be internal or external to the organization.
- Stakeholders are usually identified by their roles rather than individually.

Process Actors

- **Users:** Who will operate the software.
- **Customers:** Who have requested the software or who represent the target market.
- **Market analysts:** Who determine the external market need and who may act as proxy customers.
- **Regulators:** Which are representatives of third-party regulatory agencies.

Process Actors

- **Project leader:**
  - Determine scope and create project plan.
  - Get agreement from owners.

- **Requirements analyst:**
  - Elicit and decompose the requirements

- **Development team:**
  - Design, code to requirements, and when possible, engineer efficiency and reuse

Process Actors

- **Test team:**
  - Verify conformance to requirements
- **Maintenance team:**
  - Support users in production and ensure changes fit requirements
- **Project sponsor:**
  - Provide motivation and sign off

The requirements engineer’s job

- Mediate between the stakeholder domain and the software engineering team and between the stakeholders themselves.
- Identify all the stakeholders,
  - Their needs,
  - Their interests,
  - Power levels to control the project’s requirements