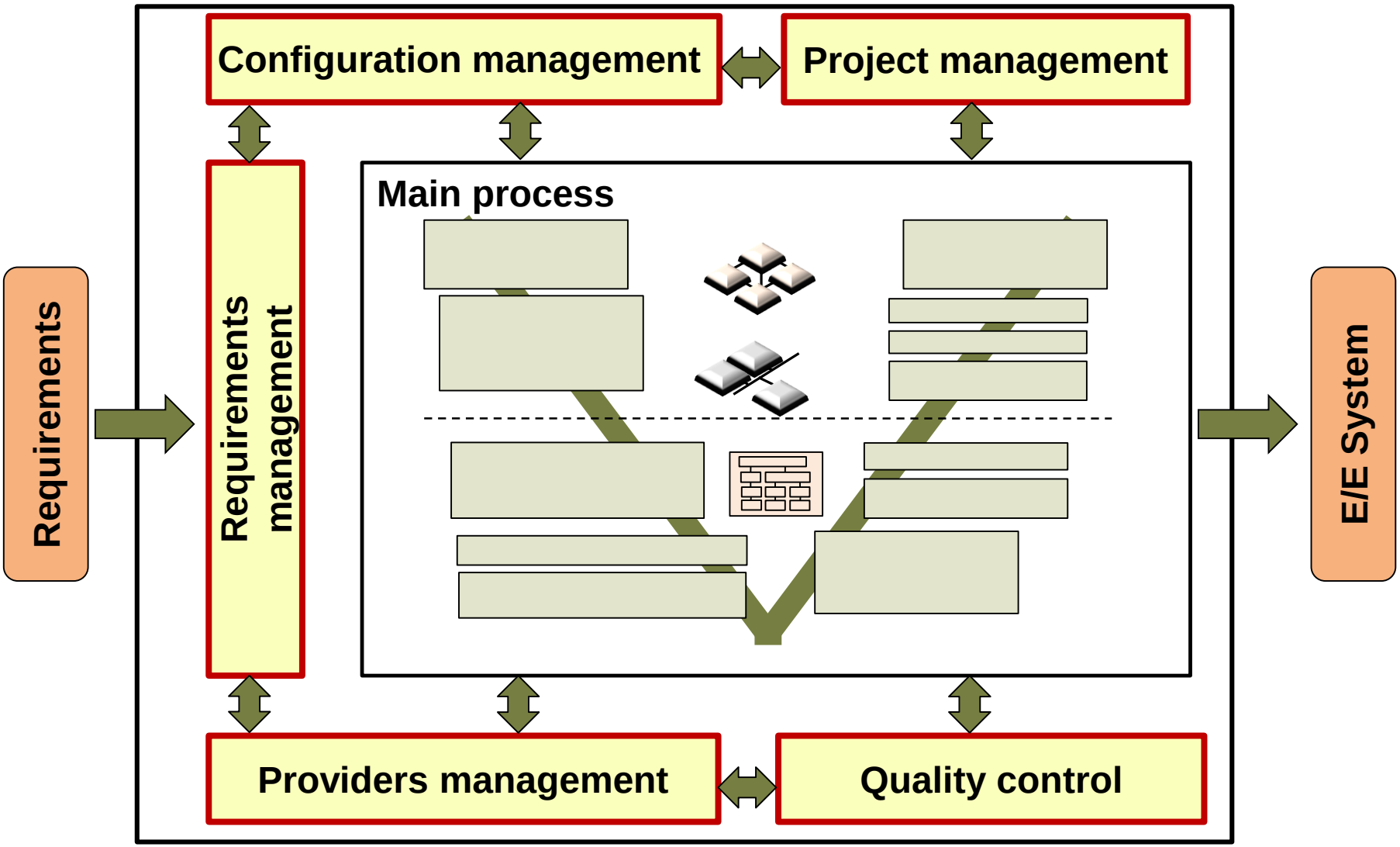
Software Platforms for Automotive Systems

Lecture 5: Supportive Development Processes

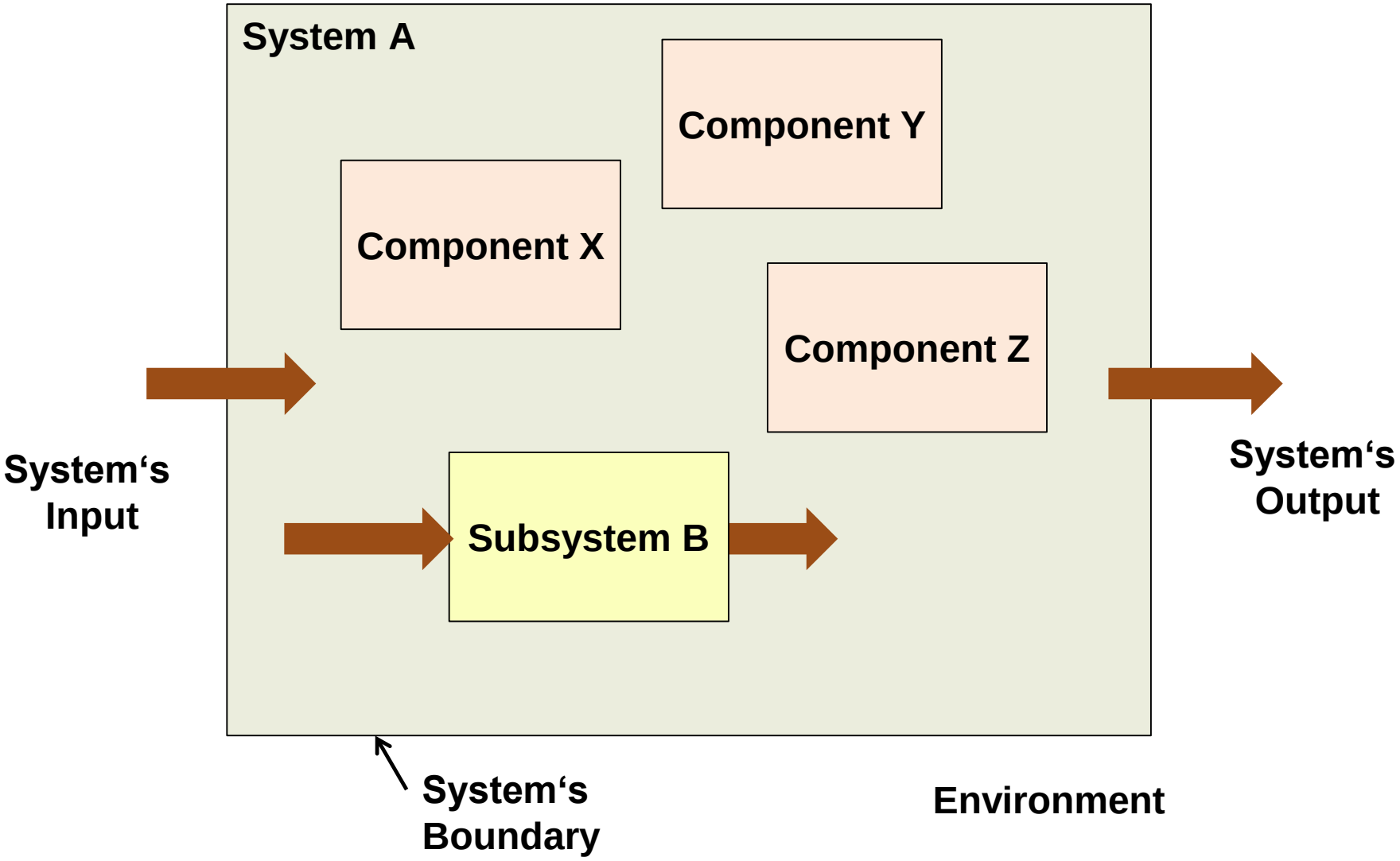
Alejandro Masrur

12th November 2015, TU Chemnitz

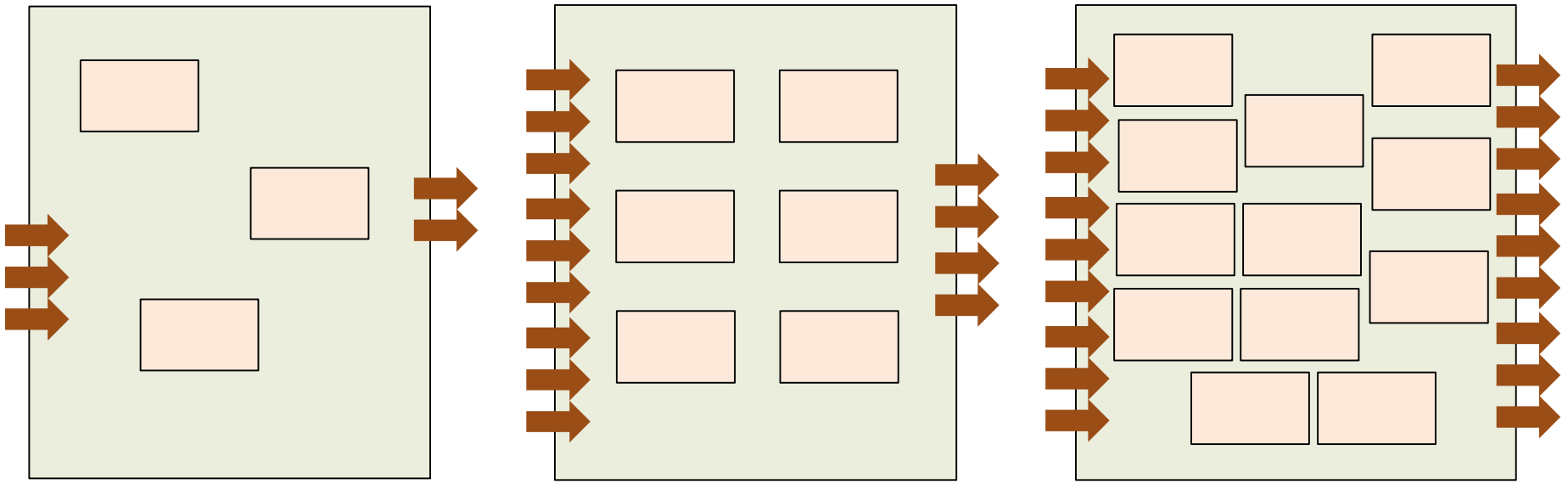
Supportive Development Processes



Systems Theory

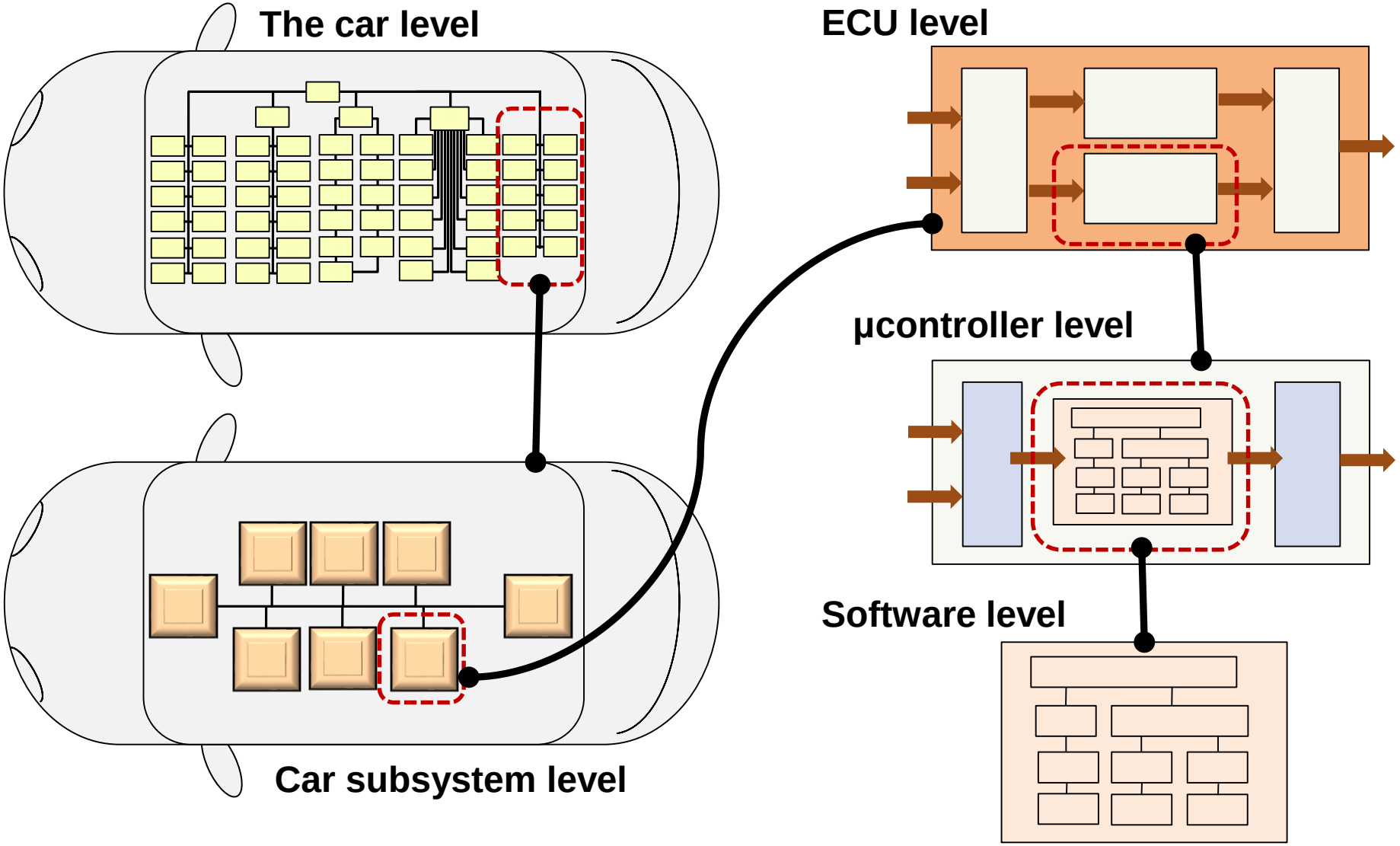


Perception of Complexity



- **The 5±2 rule states**
  - **More than 5+2 components are regarded as complex**
  - **Less than 5-2 components are considered trivial**

Levels of Abstraction in a Car

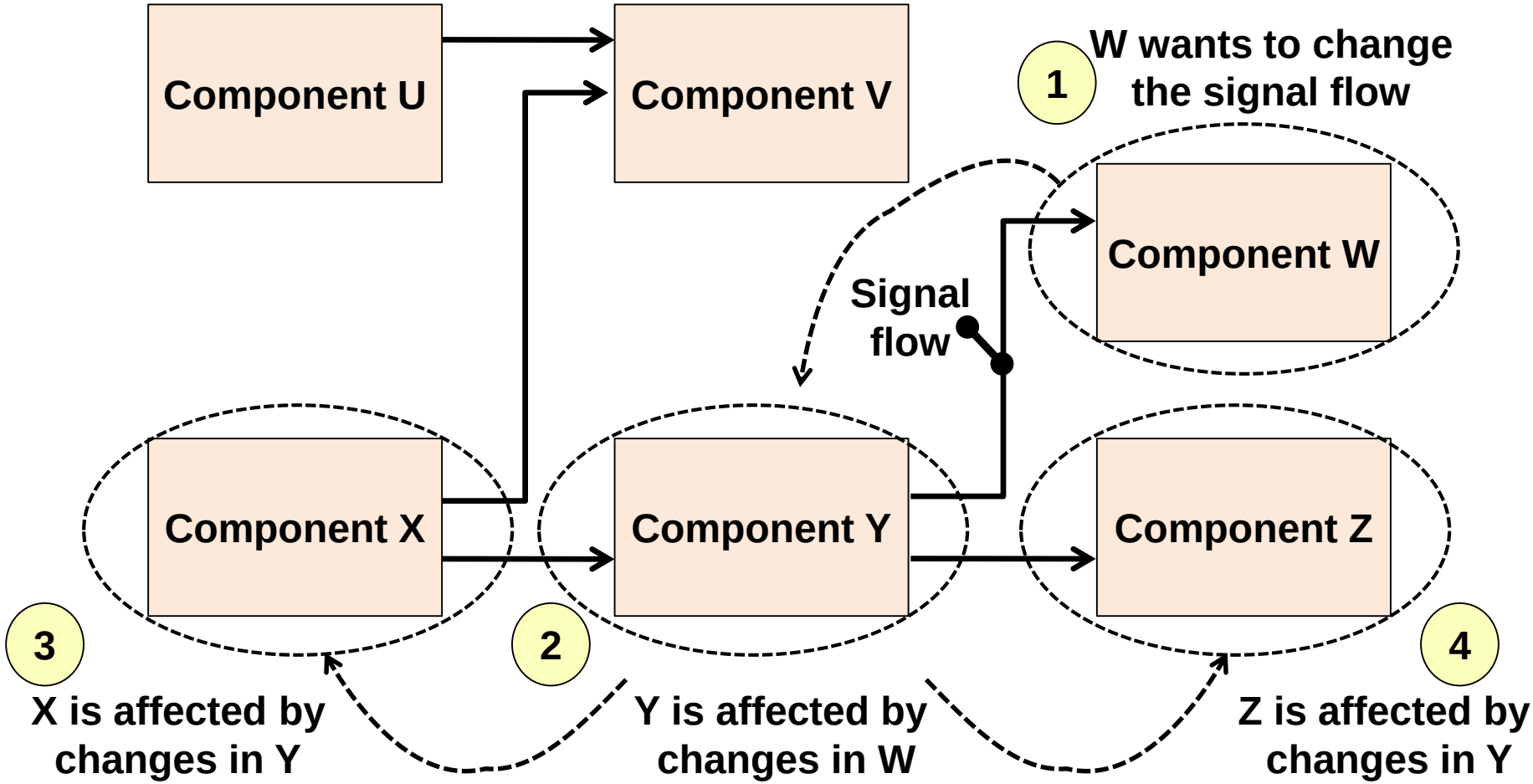


Process Models and Standards

- Process models for the development of software
  - CMMI (Capability-Maturity Models Integration)
  - SPICE (SW Process Improvement & Capability dEtermination)
  - Scrum
  - V-Model
    - Mainly adopted in the automotive domain
      - Focus of this course
- Such models should be adapted to the specific use case
- They all deal with: Requirements, configuration, project, providers management and quality of control

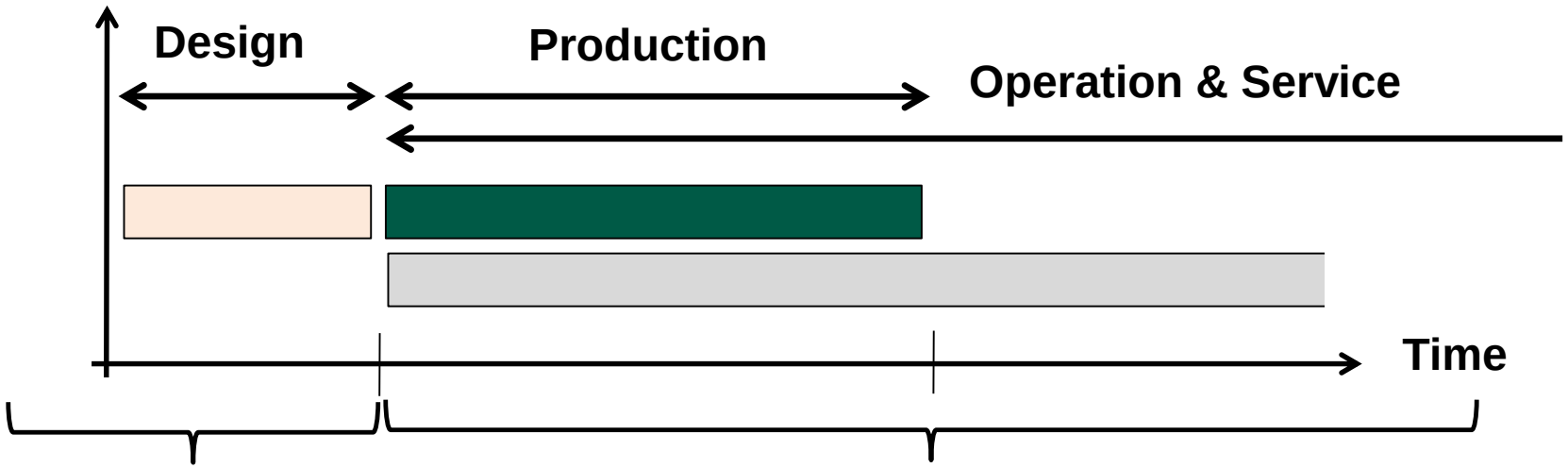
Change Management

- **Need for a systematic approach**



Configuration Management

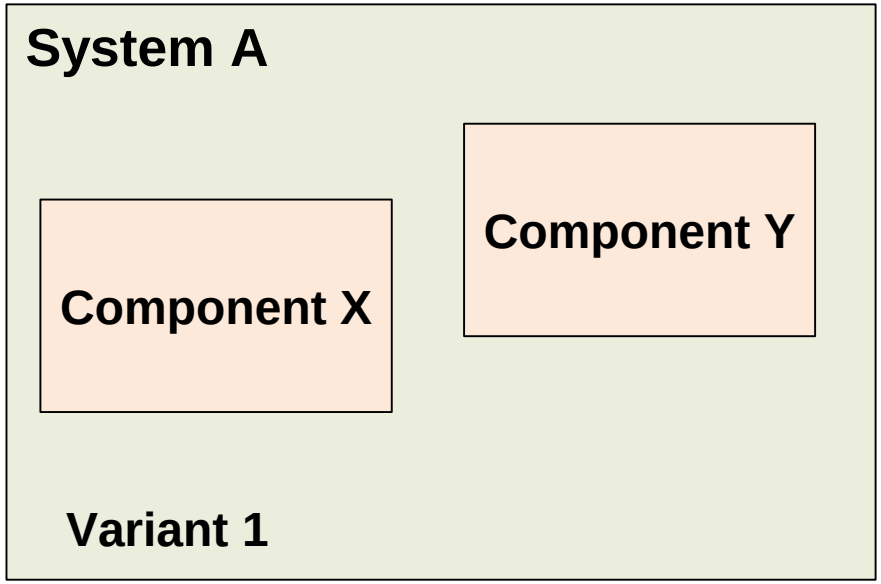
- **Different ECU requirements along the product life cycle**

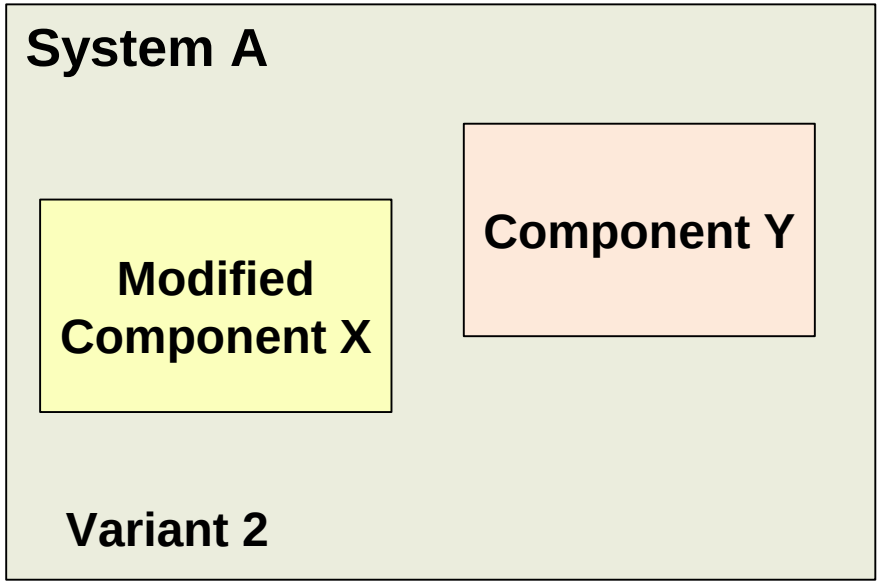


- Measurement
- Calibration
- Fast prototyping
- Download and Debugging
- Software upgrade by flashing
- Diagnostics

- Measurement
- Software parameterization
- Software configuration
- Software upgrade by flashing
- Diagnostics

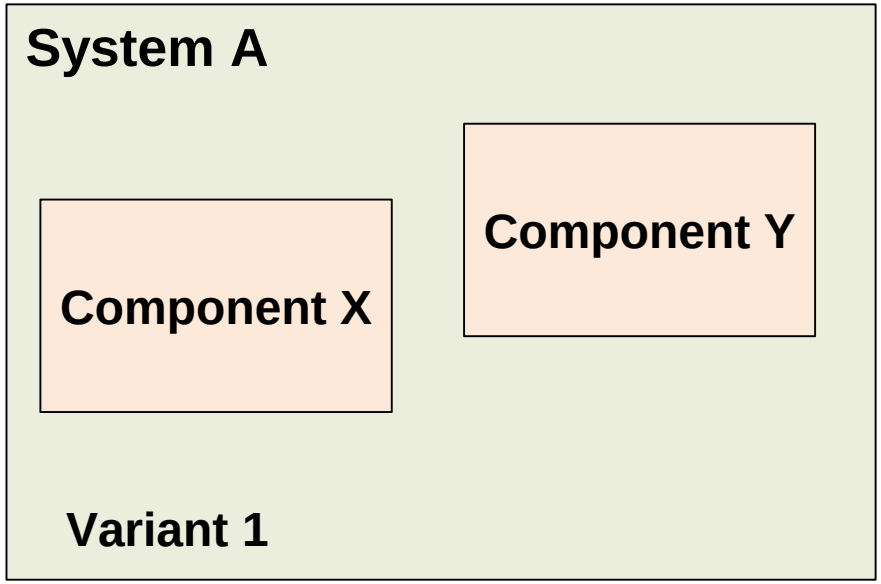
Variant by Modifying

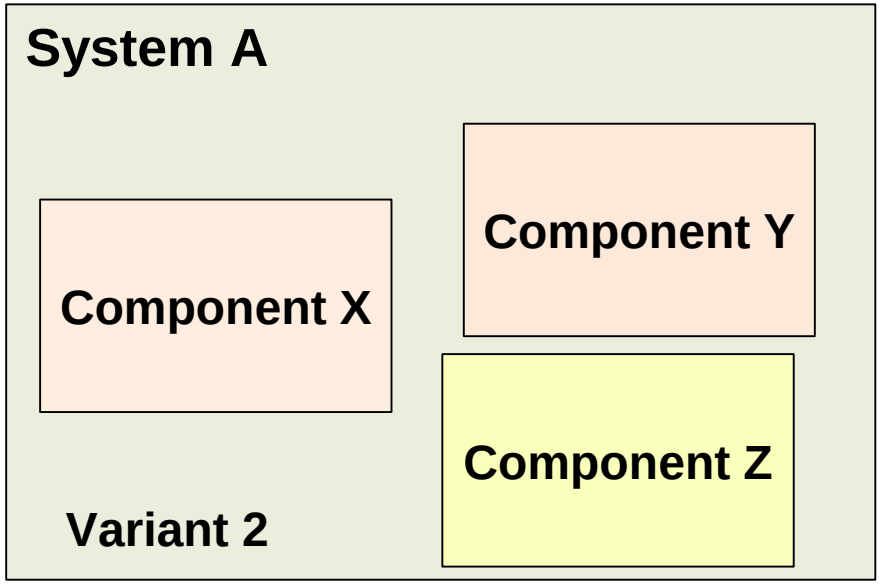




- To add new function to the system
  - Component X is modified or extended
  - Component X needs to be debugged anew

Variant by Scaling

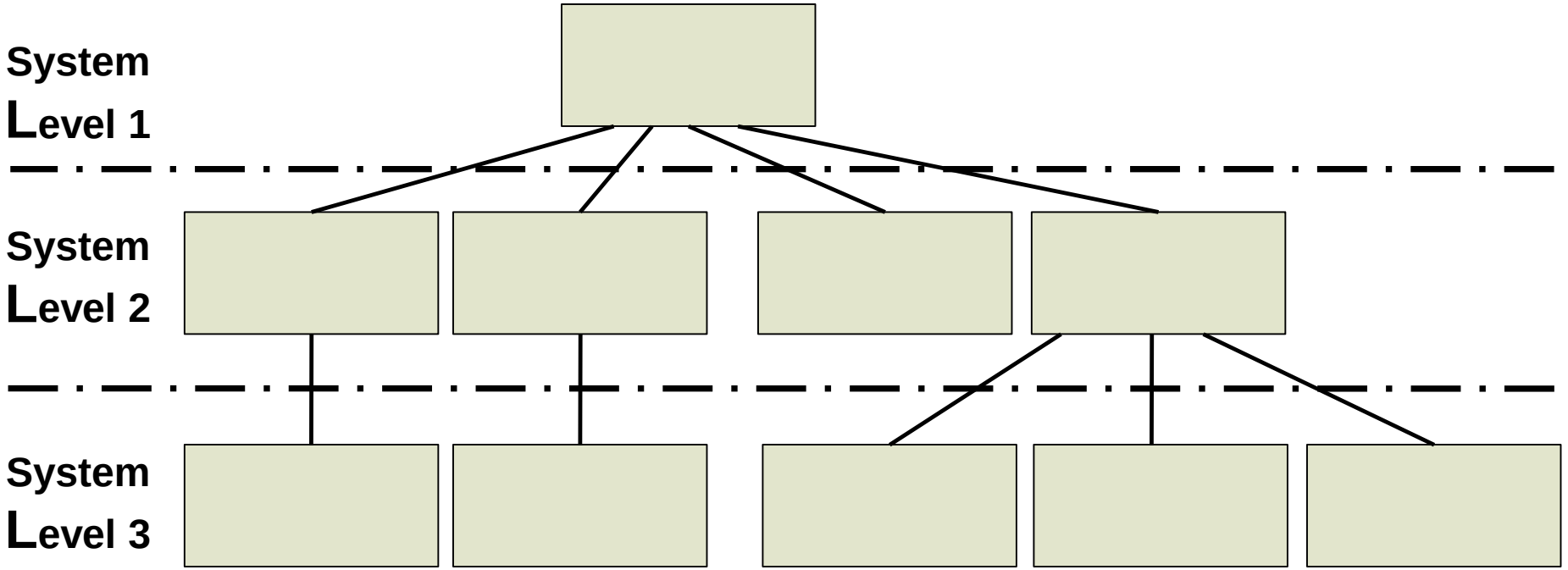




- To add new function to the system
  - Component Z is added to the system
  - Component X does not need to be debugged anew

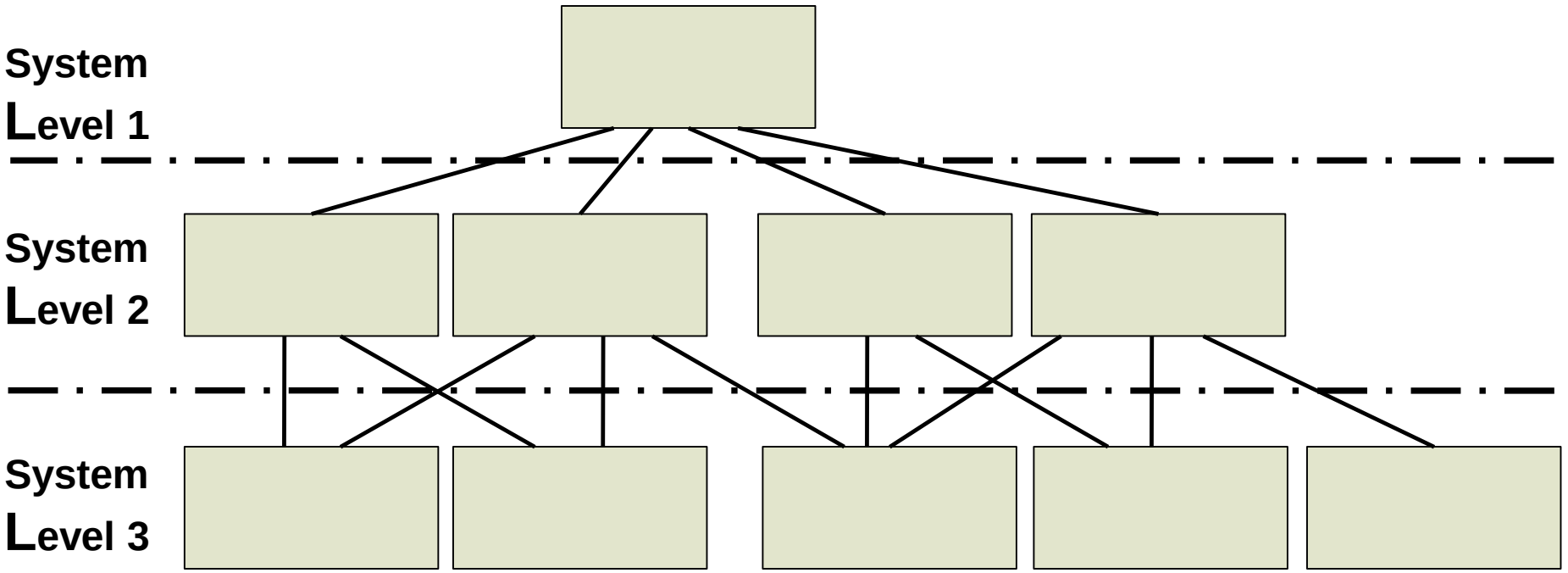
Version Management

- **A version is a well-defined state in the development of a component or system**
  - **They can be arranged in a tree-like structure**

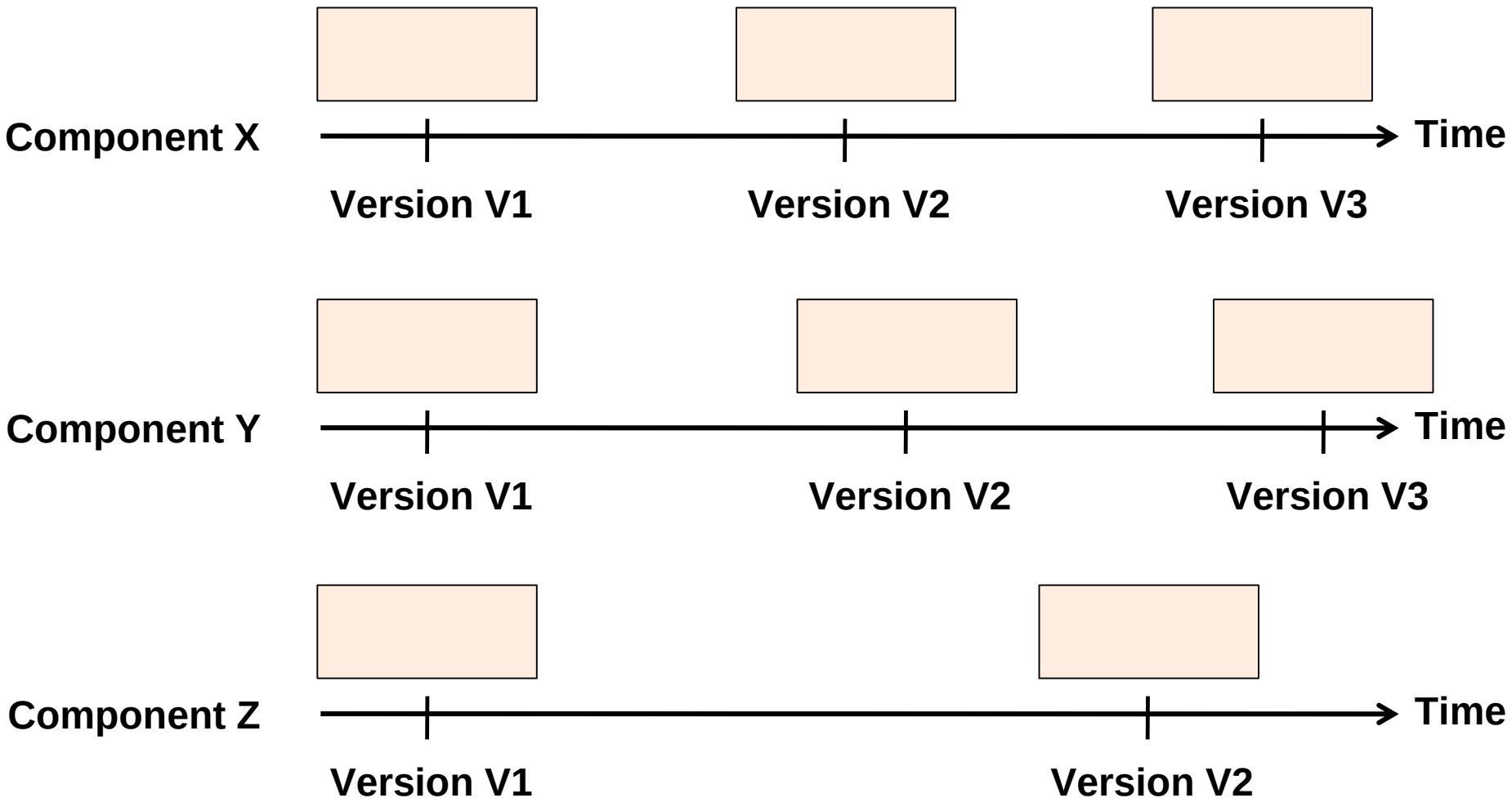


Version Management

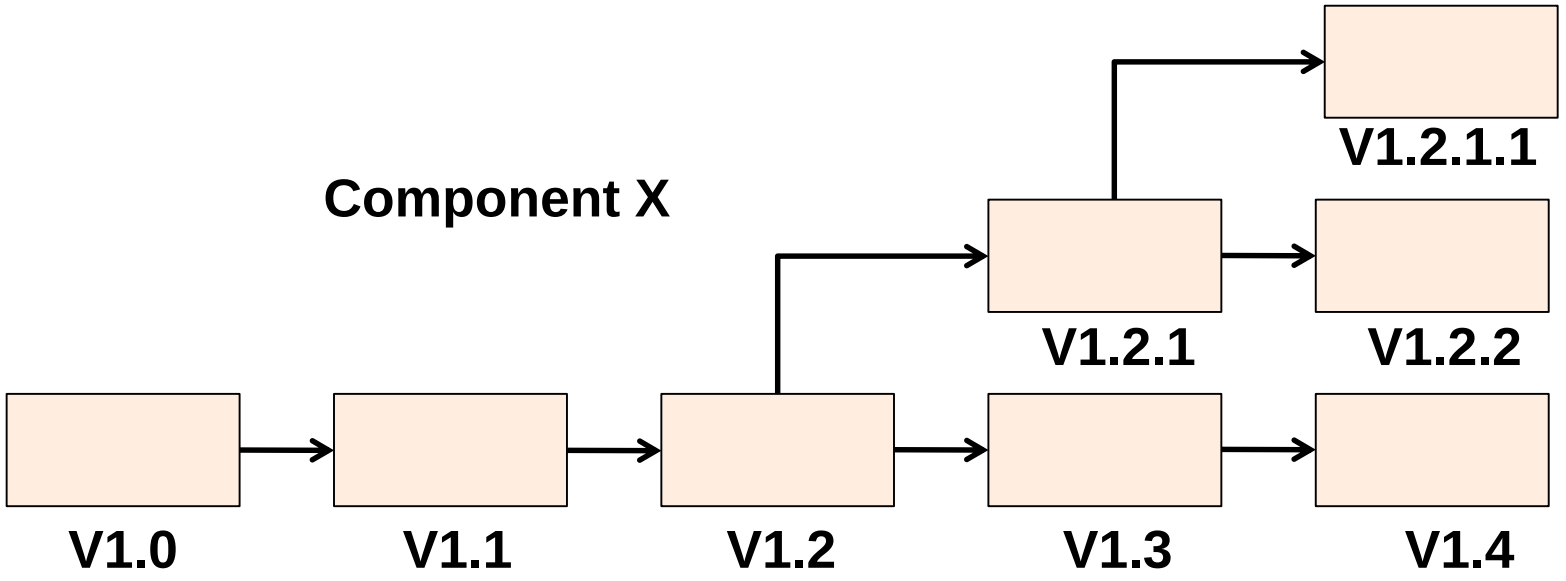
- **Versions can be also be arranged in a network-like structure**
  - **Assumed to be default in Configuration Management**



Versions of a Component

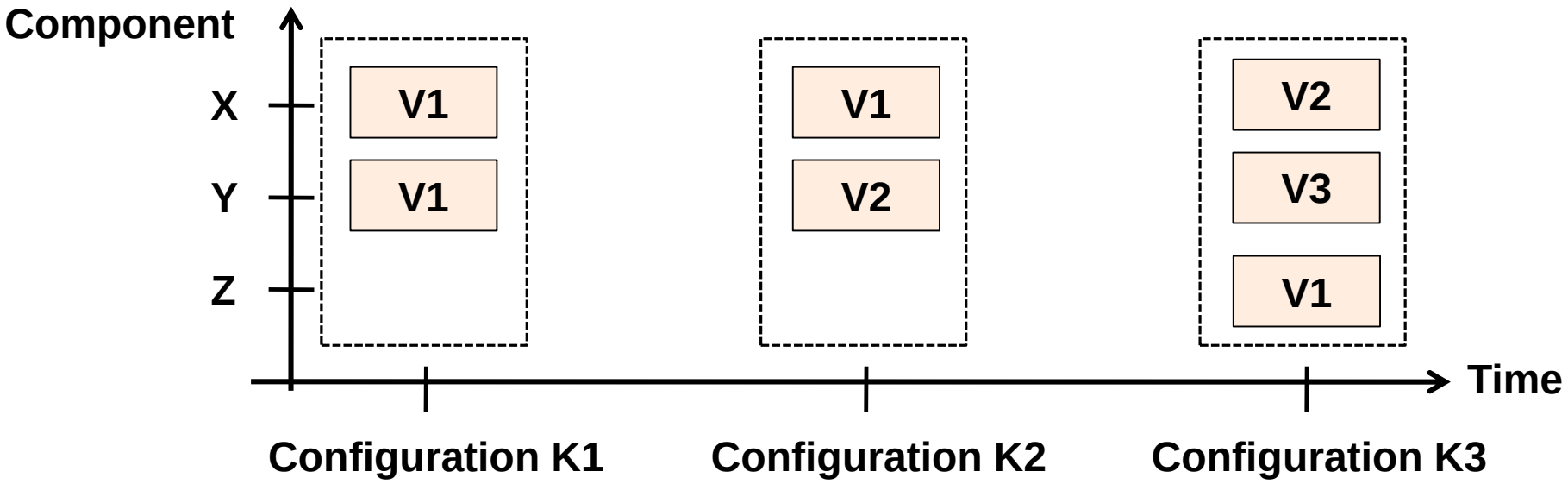


History of Versions



Configuration Management

- **A configuration is collection of versioned components**
  - **It administers the relations between versioned components**
    - **It does not administer the versions of components**
  - **It has a version itself**

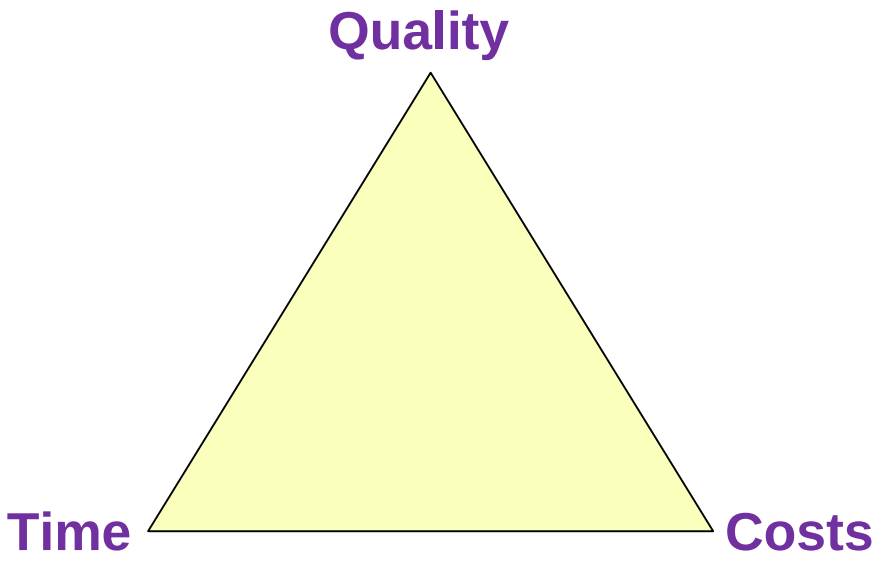


Configuration Management

- **The configuration Management**
  - **Covers all phases in the development process**
    - **Includes the exchange between different departments**
  - **Besides the versions of components it includes versions of tools that have been used**
  - **The purpose is to guarantee reproducible results, hence, the following is considered:**
    - **Requirements**
    - **Specification**
    - **Implementation (history of versions and configurations)**
    - **Description data: parameterization, diagnostics, etc.**
    - **Documentation, etc.**

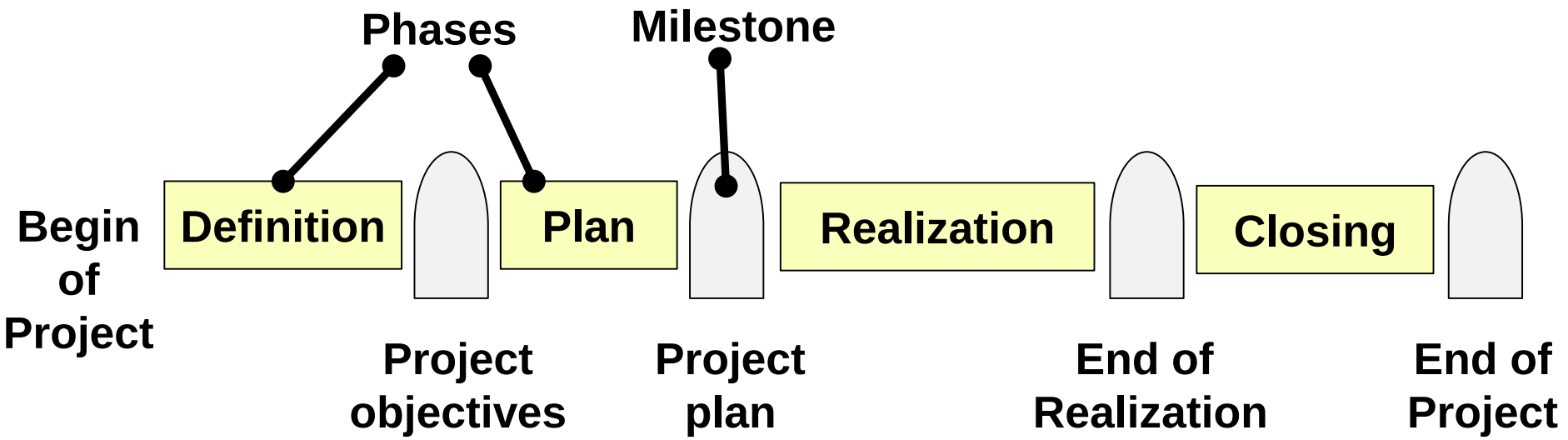
Project Management

- **A project is a task or a set of tasks that need to be performed**
  - **Tasks have an economic risk**
  - **Clear objectives**
  - **Time-bounded**
    - **With a start and an end**
  - **Limited resources**
  - **Usually with dependencies**
    - **Within one company**
    - **With other companies**



Project Phases and Milestones

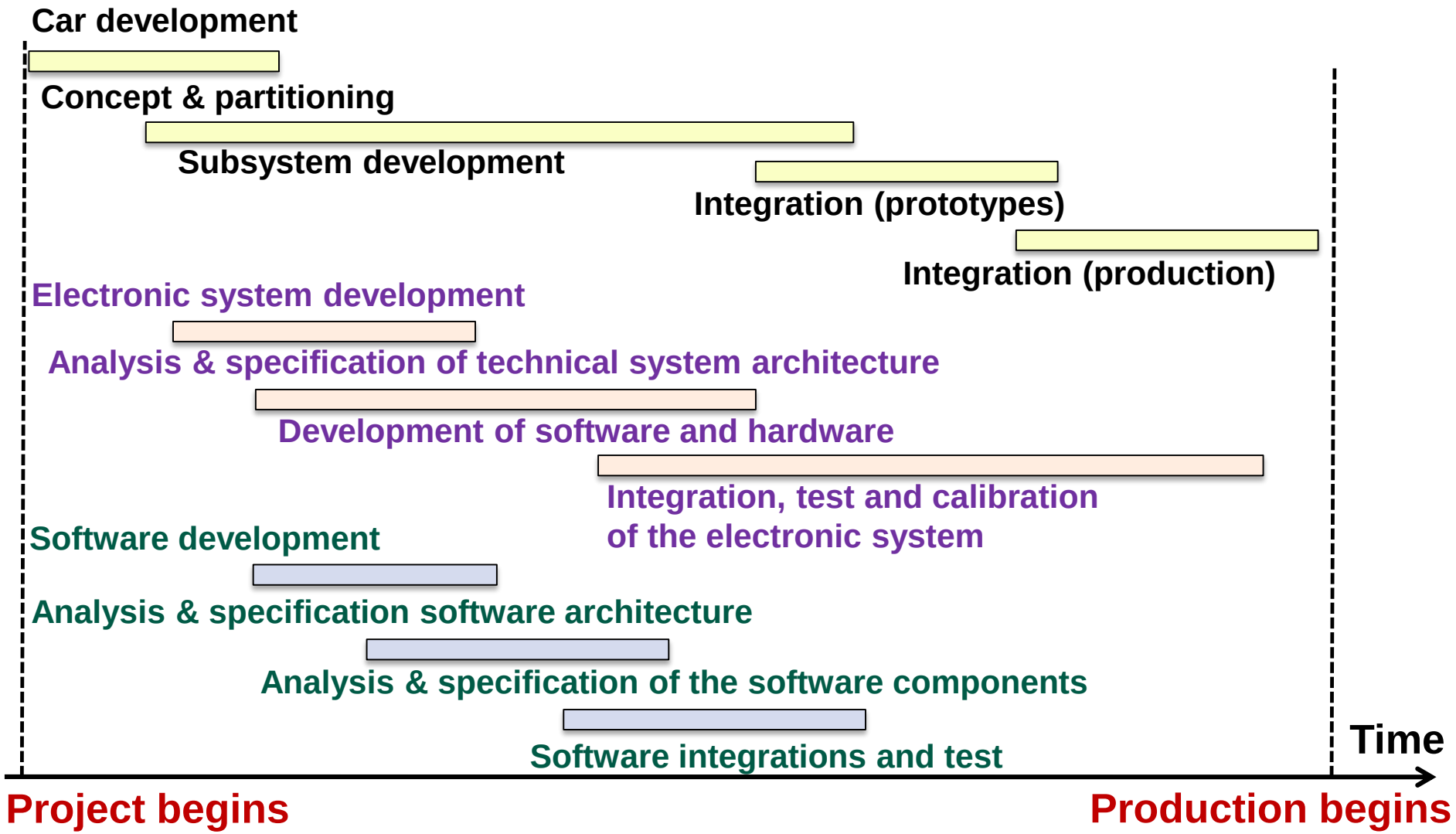
- **Project phases**
  - **Action that need to be performed**
- **Milestones**
  - **Results of the phases or actions**



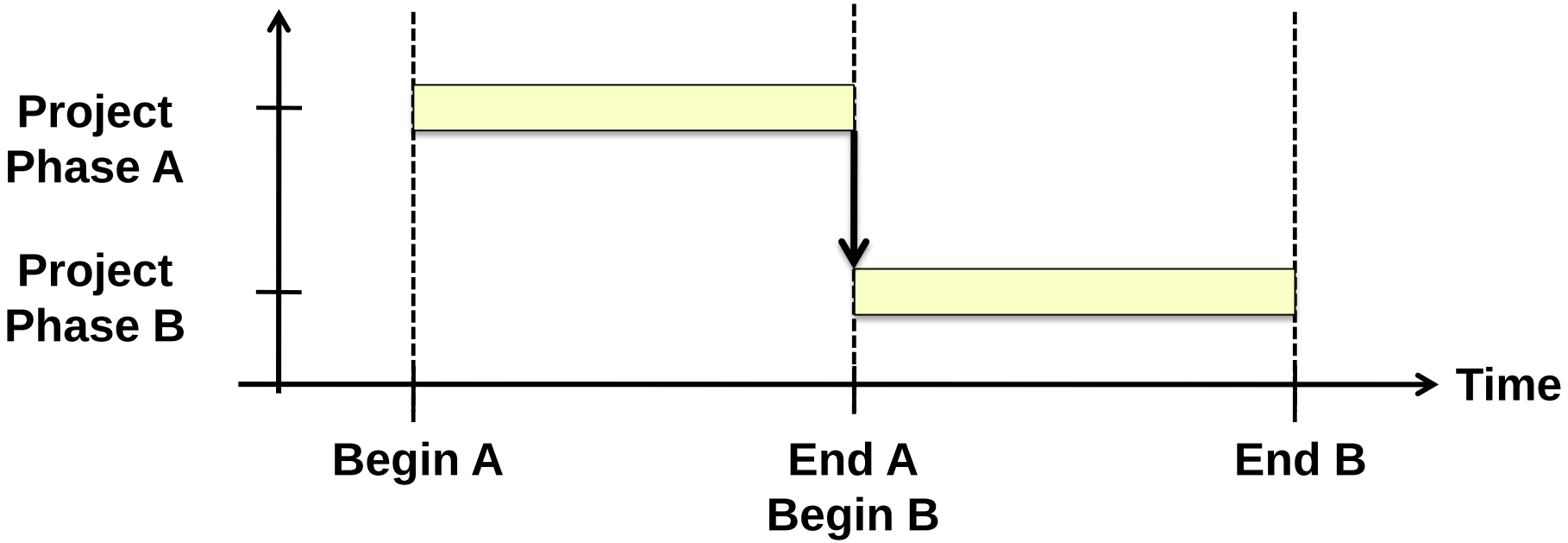
Project Planning

- **Quality Plan**
  - **Measures to provide desired quality**
  - **Measures to test quality**
- **Budget Plan**
  - **Calculate expenses**
  - **Assign resources**
- **Schedule**
  - **Fix deadlines for different phases**
  - **Consider dependencies**
  - **Consider buffer time**

Project Planning

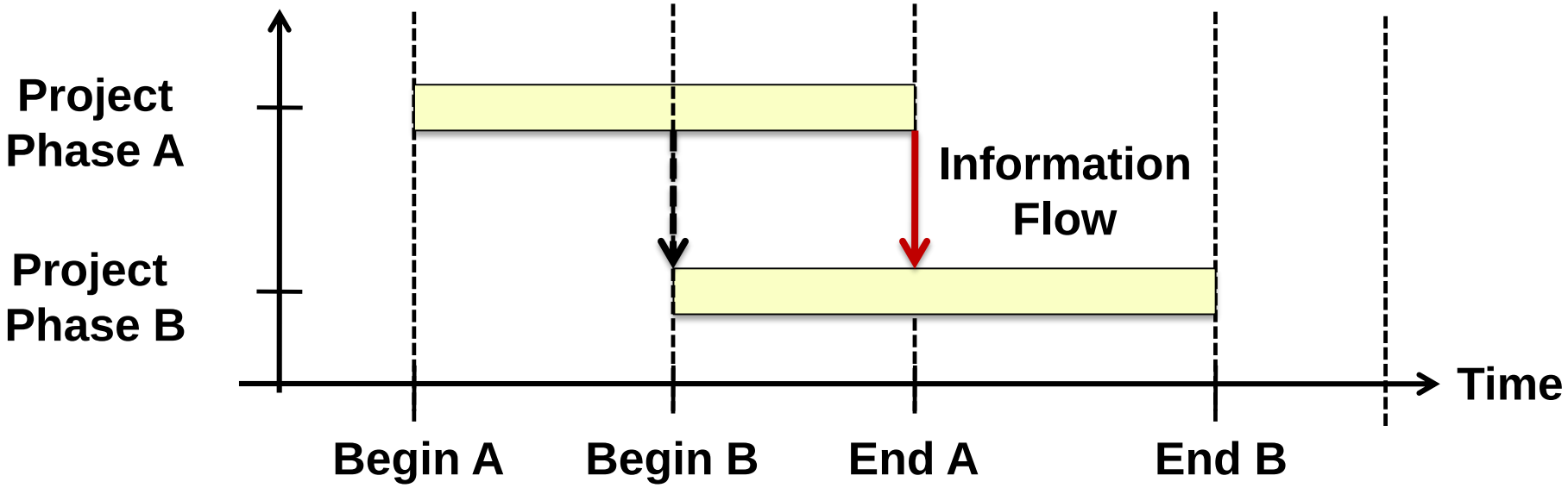


Synchronization



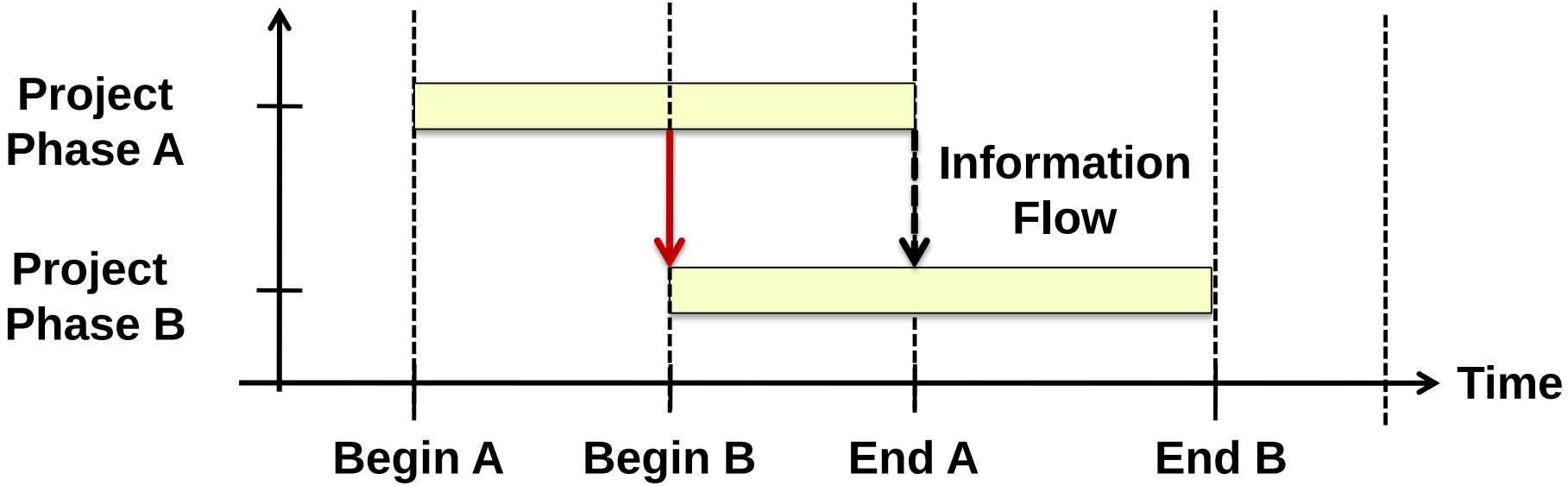
- **Sequential (once A has finished, B starts)**
  - **Advantage: sequential information flow**
  - **Disadvantage: long processing time**

Synchronization



- **Parallel with no early freezing of previous phase**
  - **Advantage: Shorter processing time**
  - **Disadvantage: Risk of iterating over in Phase B**

Synchronization



- **Parallel with early freezing of previous phase**
  - **Advantage: shorter processing time**
  - **Disadvantage: potential quality loss in Phase A**

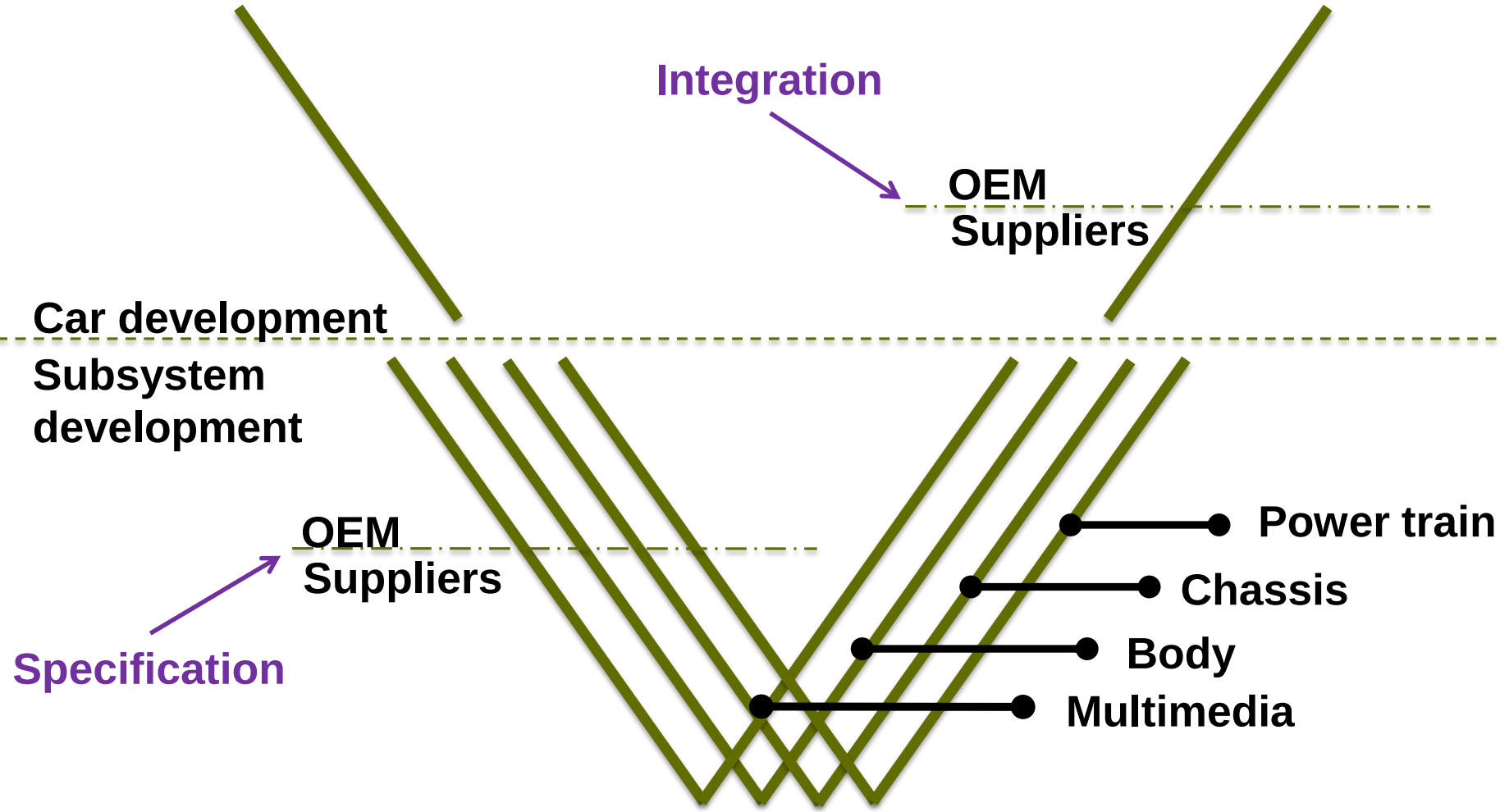
Roles in the Development Process

- Function development (logical system architecture)
- System development (technical system architecture)
- Software development
  - Software requirements and implementation
- Hardware development
  - Hardware requirements and realization
- Sensor, actuator, and setpoint encoder development
- Integration, test, and calibration

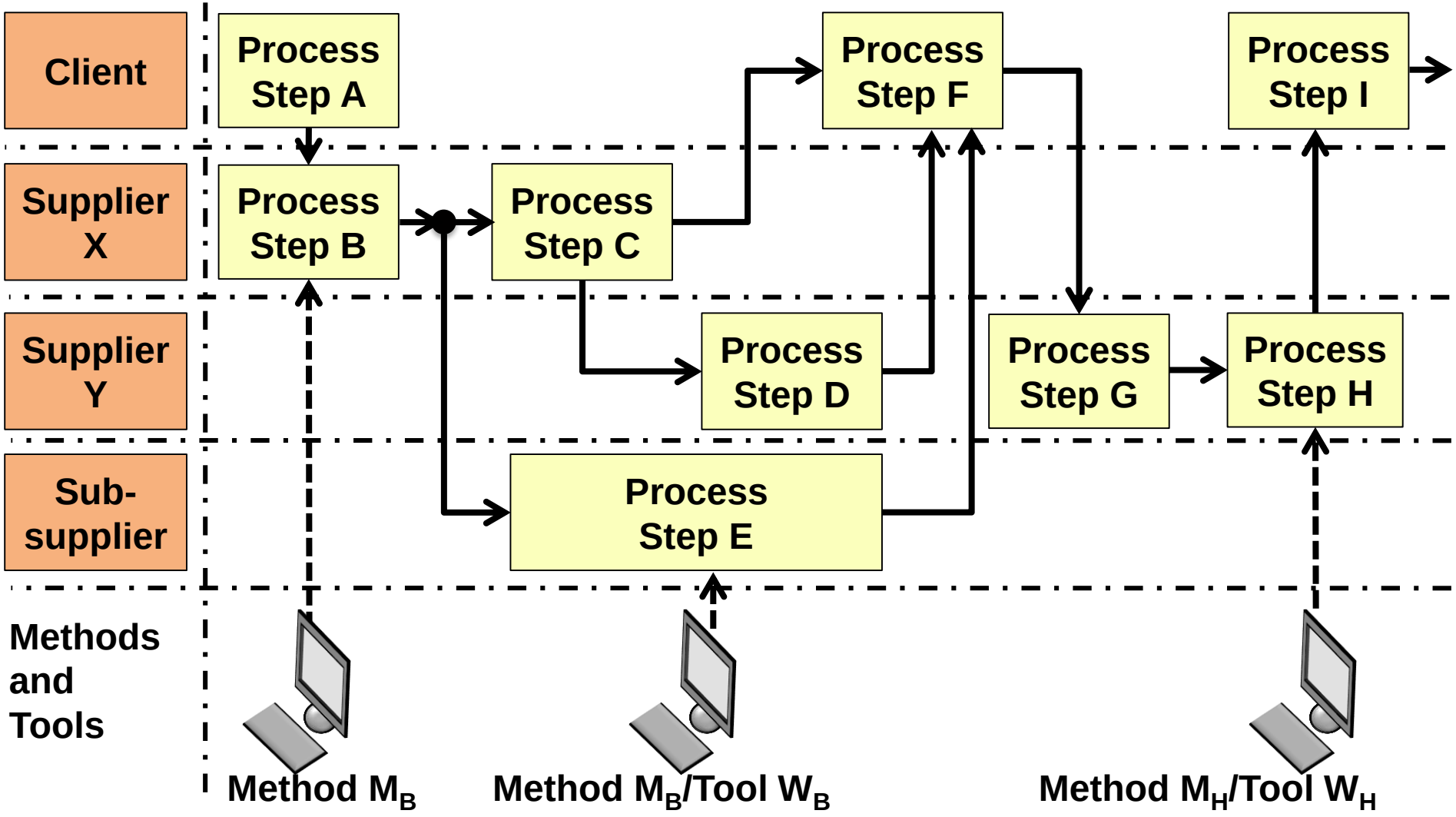
Providers Management

- Car manufacturers need to interact with providers
  - Car manufacturer=Original Equipment Manufacturer (OEM)
    - For example, VW, BMW, Toyota, etc.
  - Providers of car systems (Tier 1)
    - For example, Bosch, Continental, etc.
  - Semiconductor provider (Tier 2)
    - For example, Infineon, Freescale, etc.
- Tier 1 providers develop many systems
- OEMs validate and integrate systems, need to define clear interfaces

Interfaces btw. OEM and Tier 1



Line-Of-Visibility (LOV) Diagram

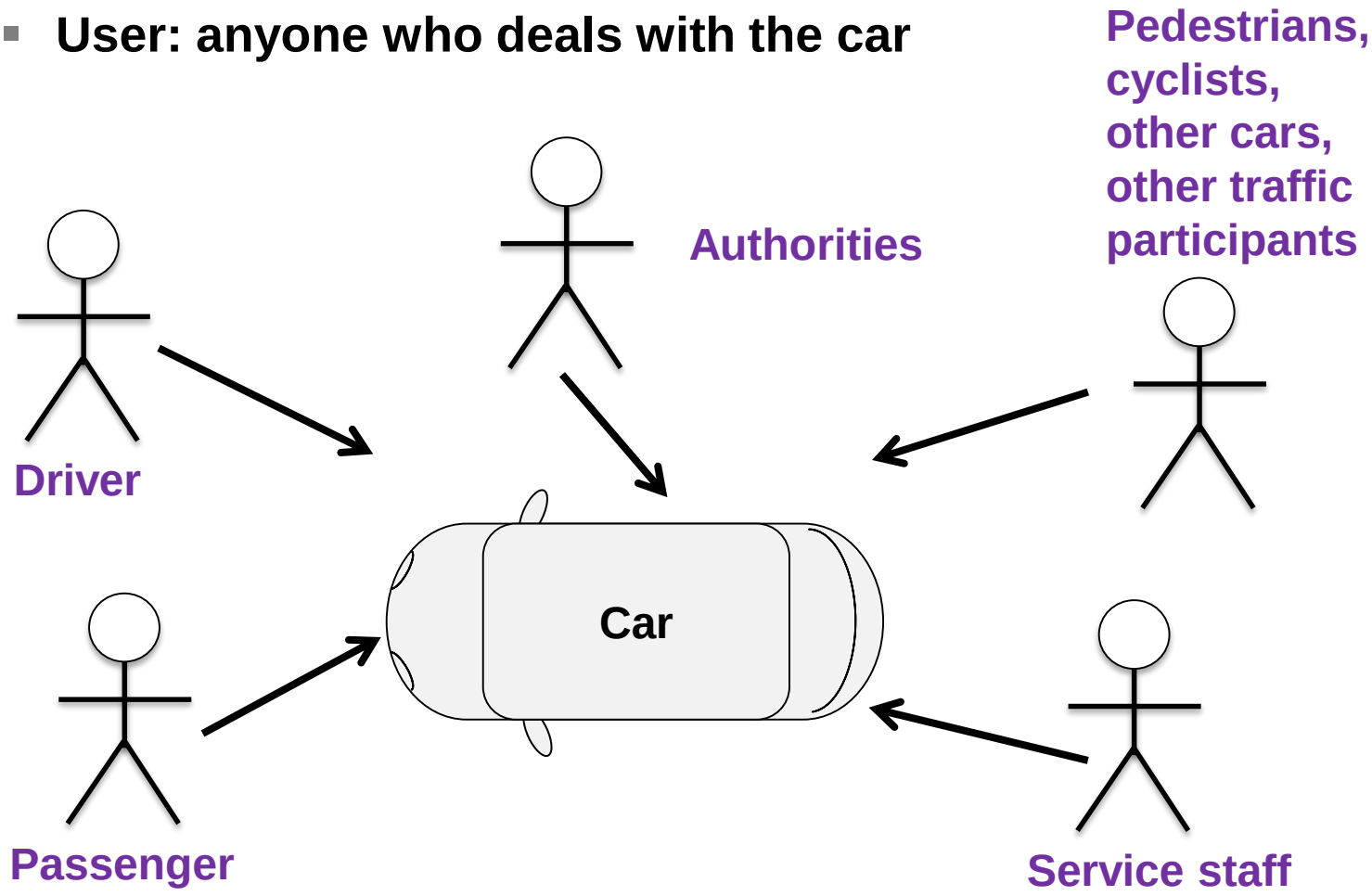


Requirements Management

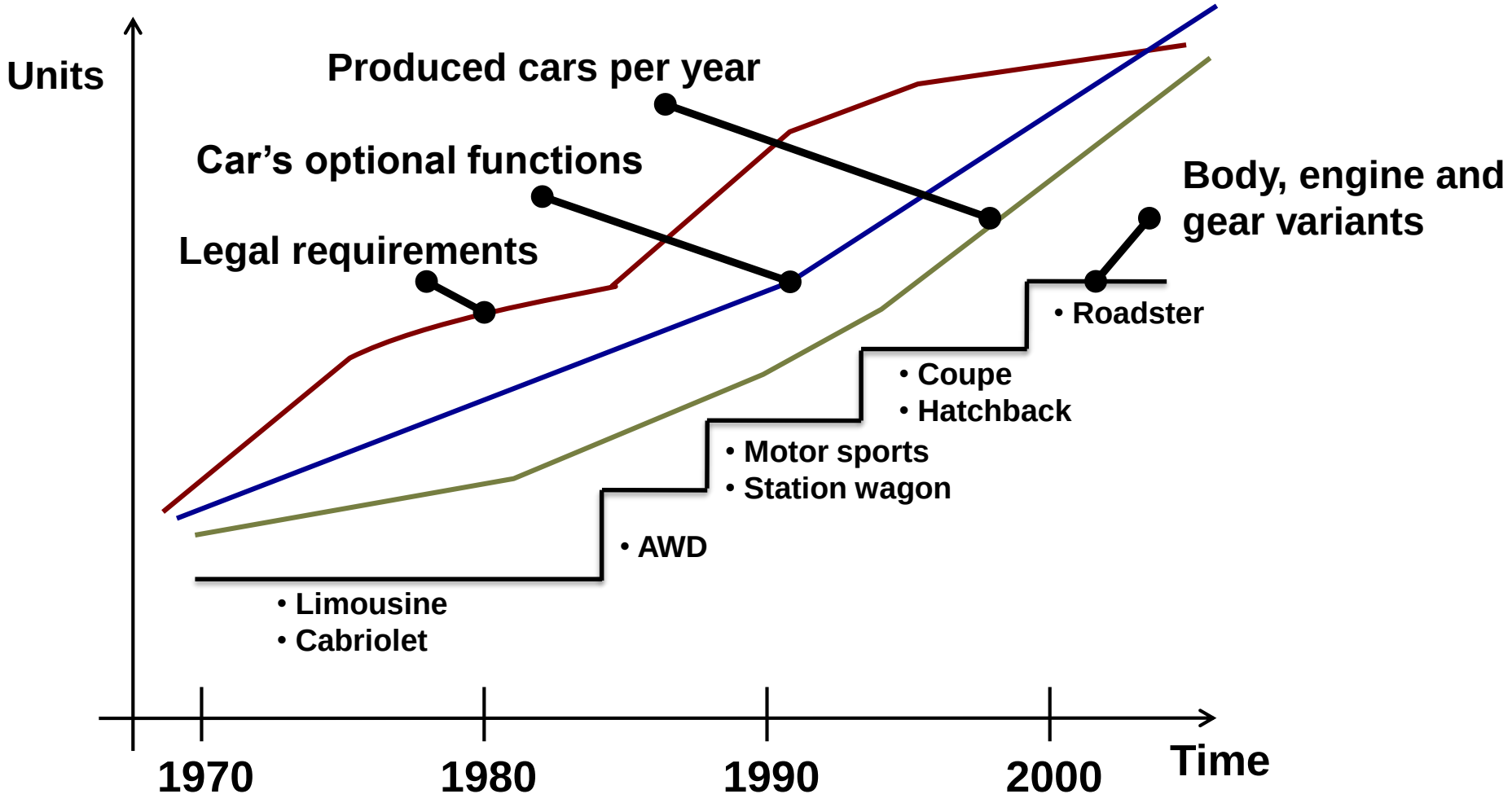
- Implies that requirements are collected and traced
- It needs to support:
  - Enterprise-wide requirements specification
    - Different departments
    - Different localities
    - Different counties
  - Long product life cycles of cars
  - Connection to Configuration Management
    - Requirements have a version
  - Integration with the whole development process

User Requirements

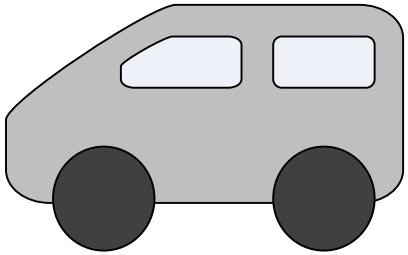
- **User requirements are paramount**
  - **User: anyone who deals with the car**



User Requirements



Automotive Requirement Classes



User interface

Functionality

Costs, complexity, & time to market

Gear & control

Quality

Real-time

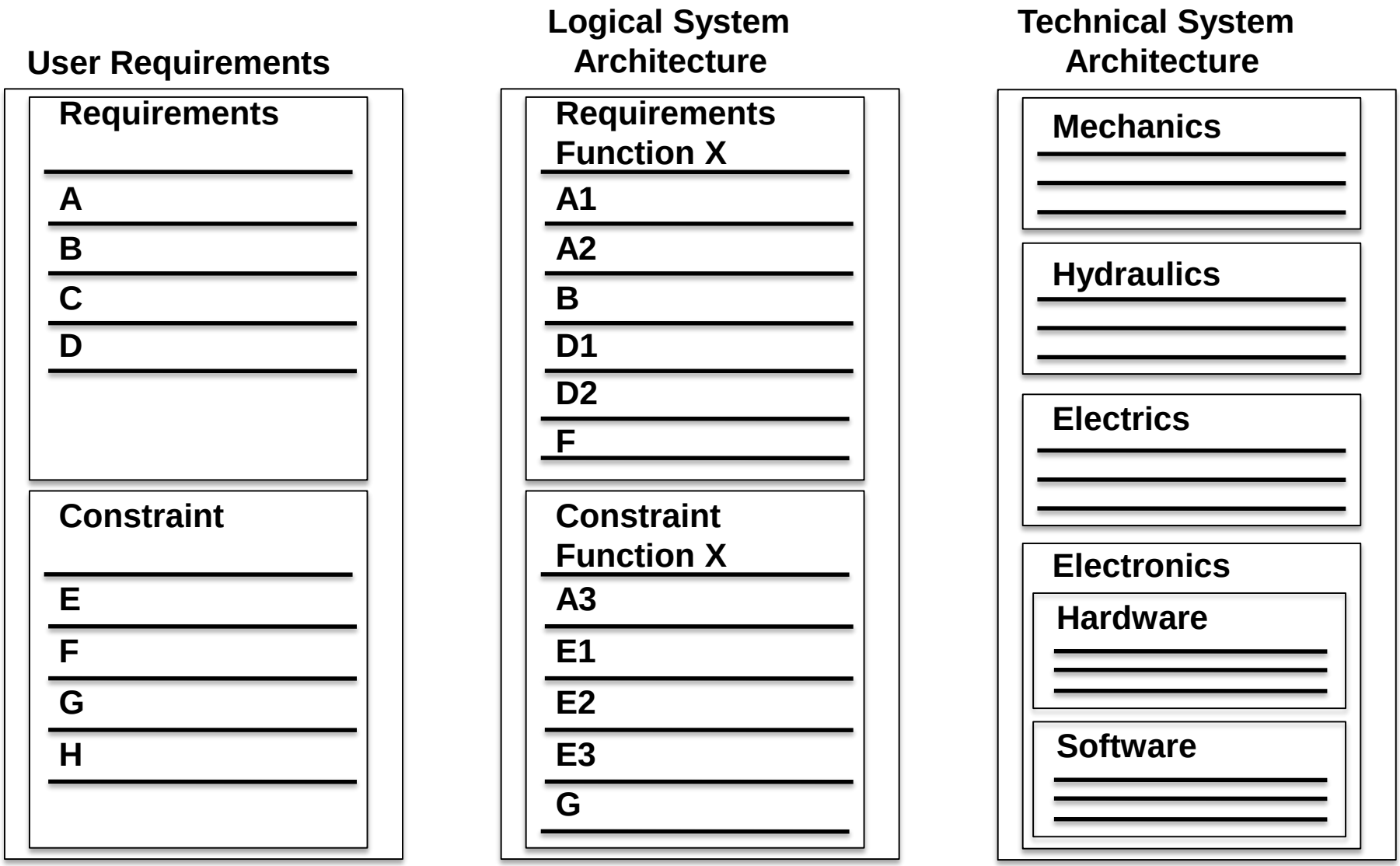
Scalability & variations

Reliability

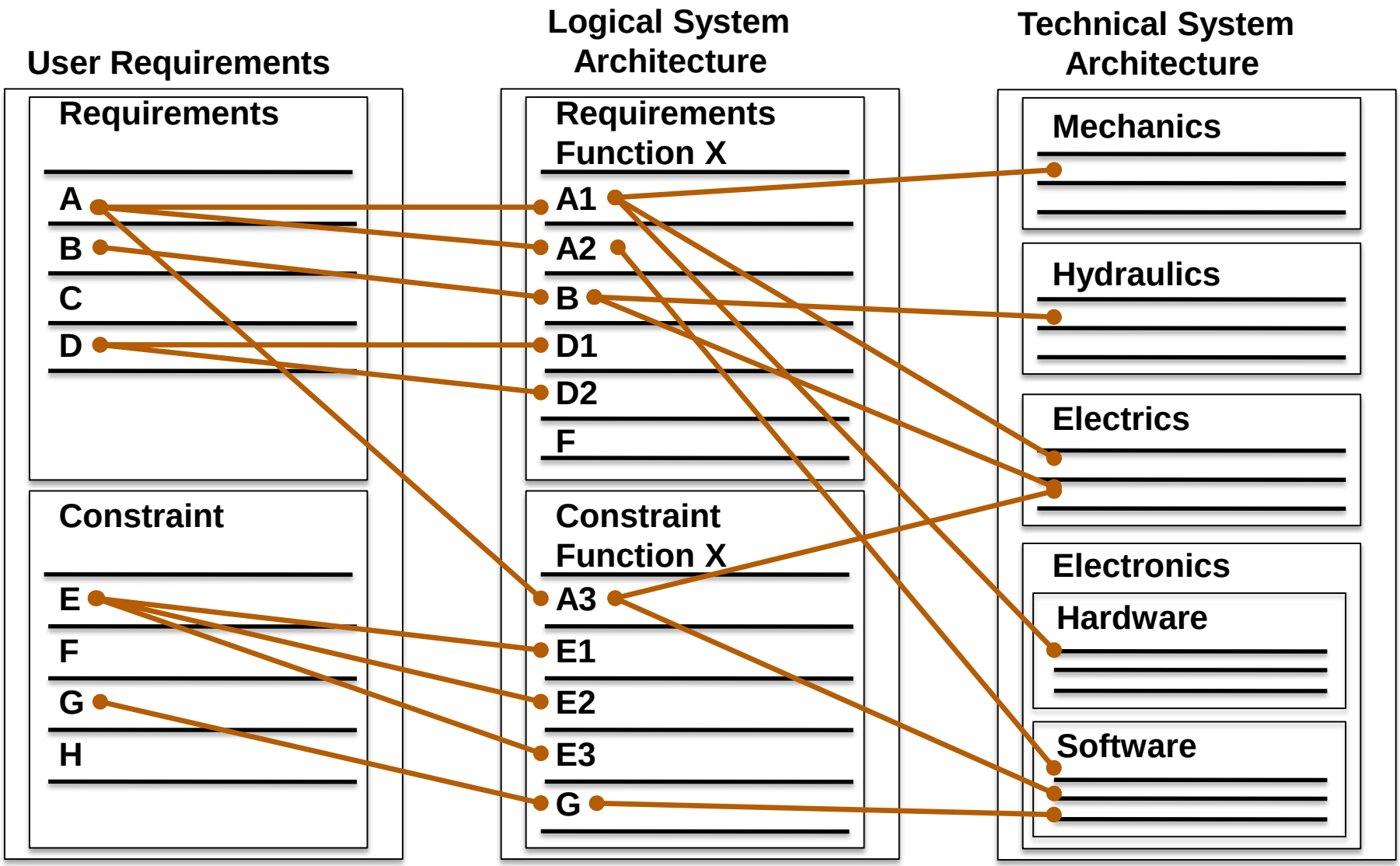
Available space, weight, & power

Safety

User Requirements/Requirements



Tracing Requirements



Quality Management

- **Policies to guarantee quality**
  - **Involve qualified staff & aims at automating manual procedures**
  - **Use of a proper development process**
  - **Use of supportive policies, measures and standards**
  - **Use of suitable tools to support the development process**
- **Measures to control quality**
  - **Specification errors**
    - **Validation**
  - **Implementation errors**
    - **Verification**

Integration and Test

- **The V-Model differentiates between the following:**
  - **Component test: checks a component against specification**
  - **Integration test: checks the system against the specification of the technical system architecture**
  - **System test: checks the system against the specification of the logical system architecture**
  - **Acceptance test: checks the system against the user specifications**
- **Component test and integration test**
  - **Count to the verification measures**
- **Acceptance test**
  - **Counts to the validation measures**

Overview Quality Management

Verification

Static Techniques

Review
Walk-through, inspection, code inspection, peer review

Analysis
Statistical analysis, formal test, control and data flow

Dynamic test, Components & integration test

Black box test
Functional capacity, stress, limit values, error rate

White box test
Structure, trail, branch, conditions, cover,…

Validation

Animation

Formal specification
Model
Simulation
Fast Prototyping

System & Acceptance Test

Functional capability
Stress and limit values test
Error rate test, Cause effect graphic, Equivalence class test

Summary

- **Supportive development processes**
  - **Requirements management**
    - **Collects and trace requirements**
  - **Configuration management**
    - **Version management**
  - **Project management**
    - **Quality, costs and time**
  - **Providers management**
  - **Quality of Control**
    - **Verification and validation**