Requirement and Architecture Design

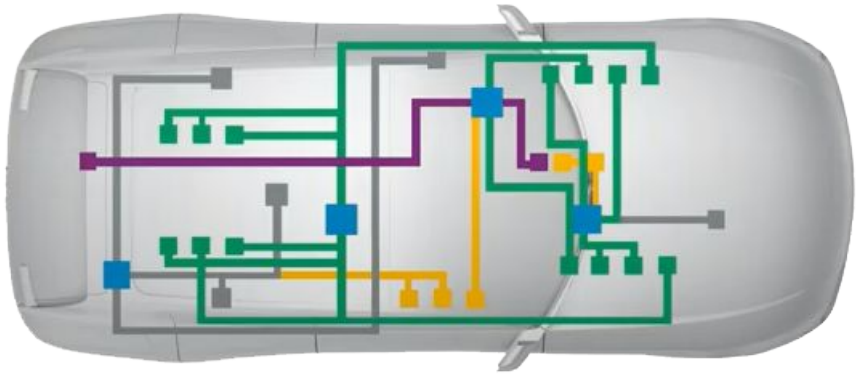
Programming for Automotive - Basics

Requirement and Architecture Design

- What is requirement?
  - Expressing of functionalities and features that a product shall have
  - Exists on different levels
- Two main types of requirement exist:
  - Non-Functional
    - No direct influence on the product.
    - E.g. cost, security, performance and etc.
  - Functional
    - Affects the functionalities of the product

Requirement and Architecture Design

- Requirements Hierarchy: requirements exist on many different levels
  - Market and customer requirements
    - Requirements from the marketing team that will result in the system requirement specification (market and customers point of view)
  - System requirement specification
    - High level technical requirements.
    - Covers all aspects of the system.
  - Subsystem requirement specification
    - Breakdown of the system requirement into domain requirement
  - Component requirement specification
    - Requirements for a specific ECU; should contain all requirements that are relevant for the ECU



Requirement and Architecture Design

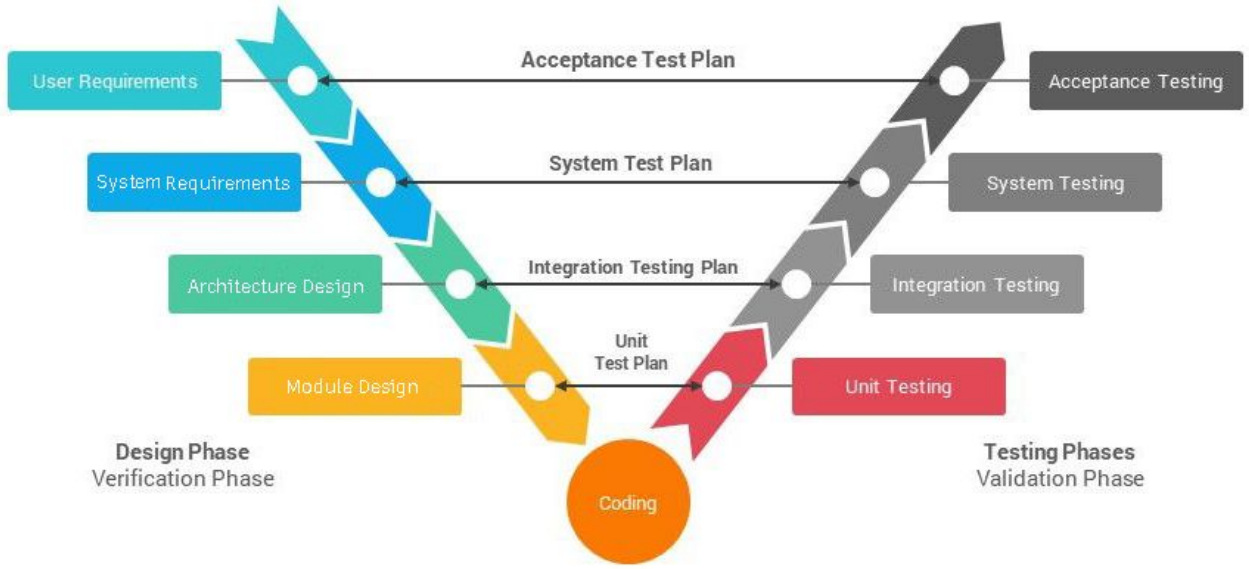
- Requirement development
  - Requirements should be developed in an iterative way
  - If you aren't an absolute expert, you can't get it right at the first time
  - Communication between developers and stakeholders is required until the requirements are good enough and fulfill the criterias
- Agile Requirement Development
  - Product owner needs to take a more active role
  - Requirements development is still an iterative process
    - 1) Product owner + Scrum master + Architects
    - 2) Team refinement
    - 3) Goto 1
  - **Result**: User stories with clear requirements

Requirement and Architecture Design

- Some **user stories** of a greenhouse (customer's point of view)
  - As a ***gardener*** I want to be able ***to keep the temperature inside the greenhouse in a configurable range*** using a ***temperature sensor***, a ***heater*** and a ***window***.
  - As a ***gardener*** I want to be able ***to keep the light intensity inside the greenhouse in a configurable range*** using a ***light intensity sensor*** and an ***LED lamp***.
  - As a ***gardener*** I want to be able ***to keep the soil moisture of the greenhouse in a configurable range*** using a ***soil moisture sensor***, a ***water pump***, a ***water valve***, a **fan** and a ***window***.

Requirement and Architecture Design

- What is a good requirement?
  - Atomic
  - Testable
  - Well described
  - Tagged



- Atomic requirement
  - It is detailed enough so that it cannot be broken down into smaller requirements
  - **Bad** example: When the system receives an unlock request it shall unlock the doors
  - **Good** example: When the system receives **<input_unlock_all_doors>** signal it shall activate the following outputs within 100ms in order to unlock the doors: **<output_unlock_front_left_door>**, **<output_unlock_front_right_door>**, **<output_unlock_rear_left_door>** and **<output_unlock_rear_right_door>**

Requirement and Architecture Design

- Testable requirement
  - Testability is critical. Otherwise we cannot ensure that the requirement is fulfilled
  - Untestable: The system shall react to any lock/unlock request immediately
  - Testable: The system shall fulfill any lock/unlock request within 100ms
- Well described requirement
  - The requirement contains enough information to be implemented and tested
  - For an embedded system the following information should be present:
    - Input (or stimuli): What is the input? A specific value or an event?
    - Transformation: Modification algorithm
    - Expected output: Is the output an event, multiple event or specific values?
    - Timing: When should the output be generated. For how long etc.

Requirement and Architecture Design

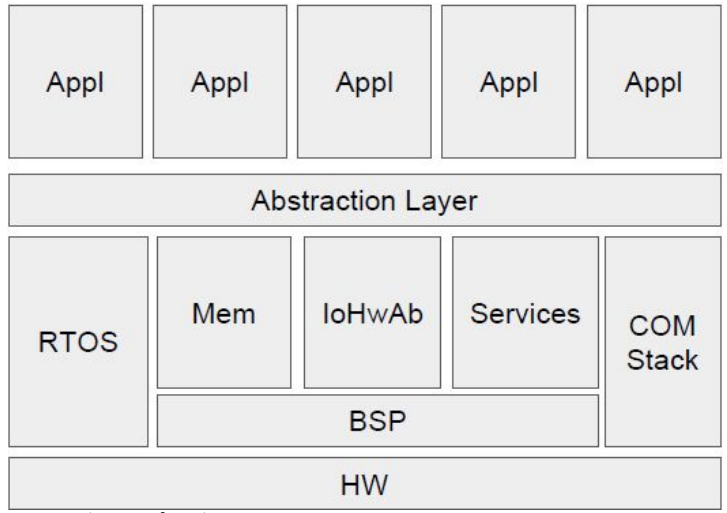
- Tagged requirement
  - A tag helps us to follow the chain from requirement to implementation and to test.
  - E.g. [ReqID:0010v01] When the <input_lock_trunk> signal is activated, the trunk shall be locked.

```
if (unlock_request_old == 0 && input_lock_trunk== 1 )  // Fulfills ReqID:0010
{
        lock_trunk();
}
```

- Use of the standard terms, **shall**, **will**, and **should** in industry
  - Requirements use **shall**. Mandatory to be implemented, and its implementation shall be verified.
  - Statements of fact use **will**. Not subject to verification.
    - E.g. The building's electrical system will power the XYZ system
  - Goals use **should**. Recommended requirements. Not formally verified.
    - E.g. The system should minimize the life cycle costs

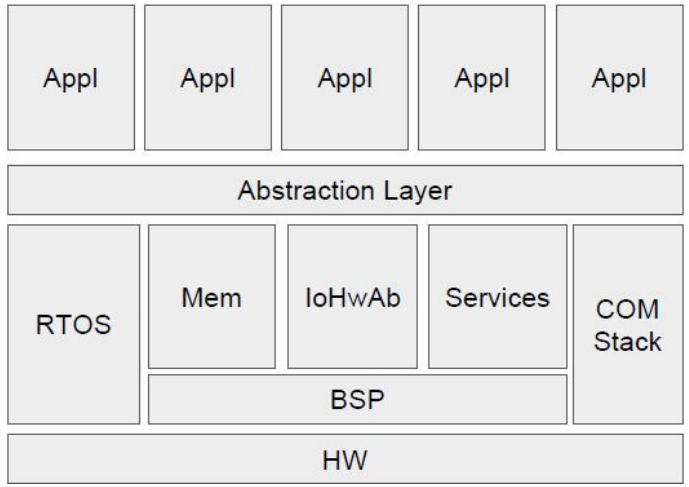
Requirement and Architecture Design

- Software Architecture
  - BSP
    - Abstraction from the hardware
  - RTOS
    - Manages the timing of the entire architecture
  - Memory Manager
    - Handles Non-volatile memory access, buffering and etc.
    - Loads calibration parameters at startup
  - Services
    - Contains generic services like startup and shutdown, mode management and etc.
  - IO Hardware Abstraction
    - Abstracts the HW from the abstraction layer. E.g. set_led_state(uint_8 state) -> digitalWrite(13, state)
  - COM Stack
    - Layered stack to abstract from communication protocols
  - Abstraction Layer
    - Provides stable and fixed interfaces for the applications
  - Applications
    - Business logic and functional blocks and execution requirements



Requirement and Architecture Design

- Tips for mapping the requirements to the software architecture
  - IO requirements (requirements that specify hardware dependencies)
    - The interfaces are mapped to the IO Hardware Abstraction (IoHwAb)
    - Signal interaction is done via the abstraction layer
  - Network interaction (communication)
    - Network signals are mapped to respective stack
    - Access to signals is achieved via the abstraction layer
      - Application does not care about the source/target network type
      - Only the characteristics matters: timing, value and etc.
  - Application Architecture
    - Use common sense
    - bundle by functionality
    - Separation of duties
    - E.g. Bundle everything related to ambient temperature into one block



Requirement and Architecture Design

- Example: some detailed requirements for the **watering system** in the greenhouse

[ReqId:01v01]: It shall be possible to have a **terminal** application to view and set numeric values via UART.

[ReqId:02v01]: The terminal shall have a function to set parameter <soil_moisture_min>.

[ReqId:03v01]: The terminal shall have a function to set parameter <soil_moisture_max>.

[ReqId:04v01]: The terminal shall restore all the parameters from the EEPROM at startup

[ReqId:05v01]: The terminal shall have two functions to display the parameters for the user.

[ReqId:06v01]: It shall be possible to have a **watering** system application that controls the soil moisture of the greenhouse

[ReqId:07v01]: It shall be possible to read soil moisture level every second

[ReqId:08v01]: When the soil moisture falls below the configurable <soil_moisture_min> parameter, <**watering**> shall

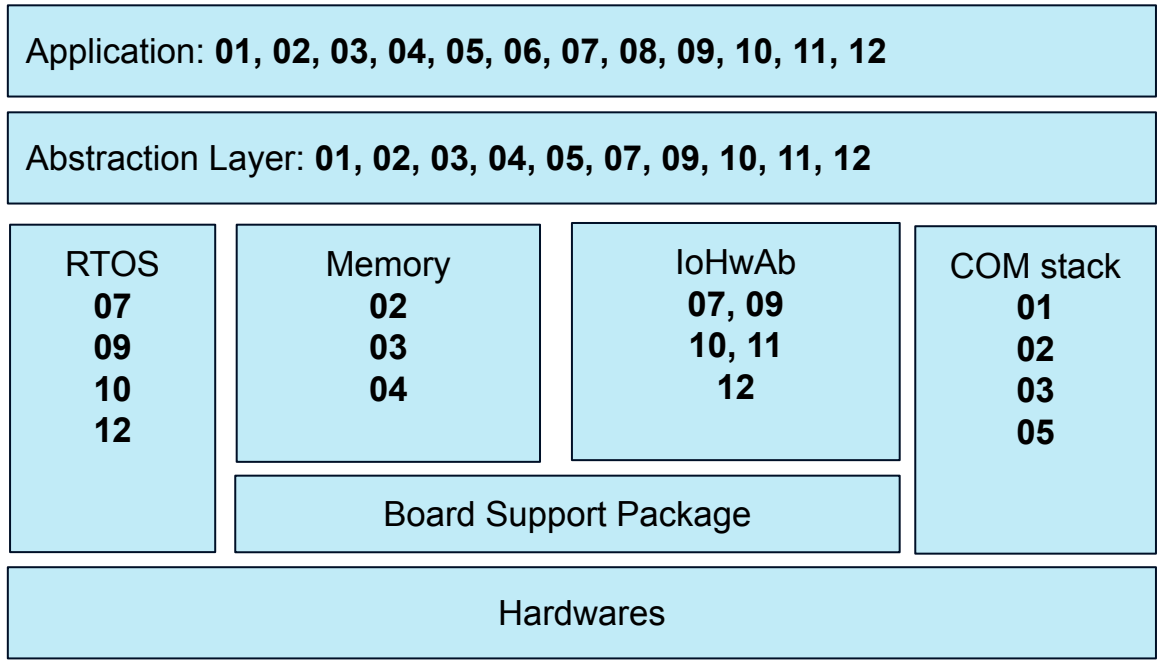
[ReqId:09v01]: activate the water pump and wait for 2 seconds to allow pressure to stabilize

[ReqId:10v01]: then open the water valve fully for 30 seconds and then turn the pump off and close the valve.

[ReqId:11v01]: If the soil moisture level goes above <soil_moisture_max>, the <**watering**> application shall open the window

[ReqId:12v01]: and activate the fan until the soil moisture level falls below <soil_moisture_max>.

Requirement and Architecture Design



- Exercise
  - Write the detailed requirements to control the ambient light and temperature
  - Map the requirements to the software architecture

Requirement and Architecture Design

- Some useful links
  - Writing Good Requirements
  - Using the correct terms – Shall, Will, Should
  - Documenting Functional Requirements
  - Documenting Non-Functional Requirements
  - Writing Requirements: Write Functional Requirements
  - Tips for Writing Clear Requirements - 1
  - Tips for Writing Clear Requirements - 2
  - What is a Functional Requirement?
  - Functional and Nonfunctional Requirements
  - Best Practices for Writing Requirements
  - Stories vs. Requirements
  - 3 Ways to Write Clearer Requirements