What is Automotive Software Engineering?

- Process models:
  - Capability Maturity Model Integration® (CMMI)
  - Software Process Improvement and Capability Determination (SPICE)
  - V-Model
- Standards:
  - MISRA-C standard
  - AUTOSAR

What is Automotive Software Engineering?

- Configuration management
  - Product and Life Cycle
    - development
    - production
    - operation and service
  - Variants and Scalability
    - components variants
    - scalable system architecture
  - Versions and Configurations
    - versions of a configuration evolve over time
    - separation of hardware characteristics from software

What is Automotive Software Engineering?

- Configuration management facilitates the administration of relations between systems and components
- Managed items:
  - Requirements
  - Specifications
  - Implementations, such as program versions and data versions
  - Description files, such as for diagnostics, software updates, and software parameterization
  - Documentation

What is Automotive Software Engineering?

- Project management, because
  - Limitation in time
  - Limitation in resources
  - Fine-tuning of organizational structure to suit objectives
  - Cross references with other projects
- Objectives:
  - Quality
  - Costs
  - Deadlines or milestones
What is Automotive Software Engineering?

Project planning
• Phases:
  • definition
  • planning
  • implementation
  • Completion
• Quality planning
• Cost planning
• Project scheduling

What is Automotive Software Engineering?

Project planning (continued)
• Development roles and Responsibilities

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function development</td>
<td>Analysis of user requirements, specification of logical system architecture</td>
</tr>
<tr>
<td>System development</td>
<td>Analysis of logical system architecture, specification of technical system architecture</td>
</tr>
<tr>
<td>Software development</td>
<td>Analysis of software requirements, plus software specification, design, implementation and testing</td>
</tr>
<tr>
<td>Hardware development</td>
<td>Analysis of hardware requirements, plus hardware specification, design, physical implementation and testing</td>
</tr>
<tr>
<td>Peripheral development</td>
<td>Analysis of peripheral specific requirements, plus their specification, design, physical implementation and testing</td>
</tr>
<tr>
<td>Integration, testing and calibration</td>
<td>Integration, testing and calibration of onboard vehicle systems and their functions</td>
</tr>
</tbody>
</table>

What is Automotive Software Engineering?

Subcontractor management
• System and component responsibility
• Interfaces for specification and integration
• Defining the cross-corporation development process

Activities Common to Software Projects

• Requirements and specification includes:
  • domain analysis
  • defining the problem
  • requirements gathering
    – obtaining input from as many sources as possible
  • requirements analysis
    – organizing the information
  • requirements specification
    – writing detailed instructions about how the software should behave
Activities Common to Software Projects

• Design
  • Deciding how the requirements should be implemented, using the available technology
  • Includes:
    − Systems engineering: Deciding what should be in hardware and what in software
    − Software architecture: Dividing the system into subsystems and deciding how the subsystems will interact
    − Detailed design of the internals of a subsystem
    − User interface design (probably less important)
    − Design of databases (probably less important)

Modeling

• Creating representations of the domain or the software
  − Use case modeling
  − Structural modeling
  − Dynamic and behavioral modeling

Programming

• Quality assurance
  • Reviews and inspections
  • Testing
  • Deployment
  • Managing the process

Requirements Engineering

• Domain analysis
  • The process by which a software engineer learns about the domain to better understand the problem:
    − The domain is the general field of business or technology in which the clients will use the software
    − A domain expert is a person who has a deep knowledge of the domain
  • Benefits of performing domain analysis:
    − Faster development
    − Better system
    − Anticipation of extensions

Starting point for software projects
Requirements Engineering

- Problem and scope:
  - A problem can be expressed as:
    - A difficulty the users or customers are facing,
    - Or as an opportunity that will result in some benefit such as improved productivity or sales.

- The solution to the problem normally will entail developing software
- A good problem statement is short and succinct
- Narrow the scope by defining a more precise problem
  - Exclude some of these things if too broad
  - Determine high-level goals if too narrow

What is a Requirement?

- It is a statement describing either
  - 1) an aspect of what the proposed system must do,
  - or 2) a constraint on the system’s development.
- In either case it must contribute in some way towards adequately solving the customer’s problem;
- the set of requirements as a whole represents a negotiated agreement among the stakeholders.

- A collection of requirements is a requirements document.

Types of Requirements

- Functional requirements
  - Describe what the system should do
- Quality requirements
  - Constraints on the design to meet specified levels of quality
- Platform requirements
  - Constraints on the environment and technology of the system
- Process requirements
  - Constraints on the project plan and development methods in relation to suppliers

Functional Requirements

- What inputs the system should accept
- What outputs the system should produce
- What data the system should store that other systems might use
- What computations the system should perform
- The timing and synchronization of the above

- Use cases very suited to describe functional requirements
Quality Requirements

• All must be verifiable
• Examples: constraints on
  • Response time
  • Throughput
  • Resource usage
  • Reliability
  • Availability
  • Recovery from failure
  • Allowances for maintainability and enhancement
  • Allowances for reusability