

Software Engineering

Projekt-Management

Prof. Dr. Peter Jüttner

Inhalt

5 Methoden

5.1 Requirements Engineering

5.2 Architektur und Design

5.3 Implementierung

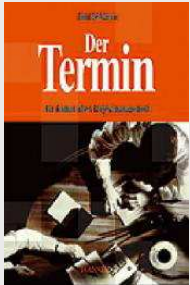
5.4 Test

5.5 Qualitätssicherung

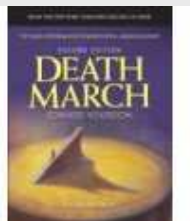
5.6 Projektmanagement

5.7 Konfigurationsmanagement

Literatur zur Vertiefung



- Tom deMarco
Der Termin: Ein Roman über Projektmanagement
Hanser Fachbuchverlag (ISBN 3-446-40165-2)



- Edward Yourdon
Death March
Prentice Hall International (ISBN 013143635X)

Inhalt

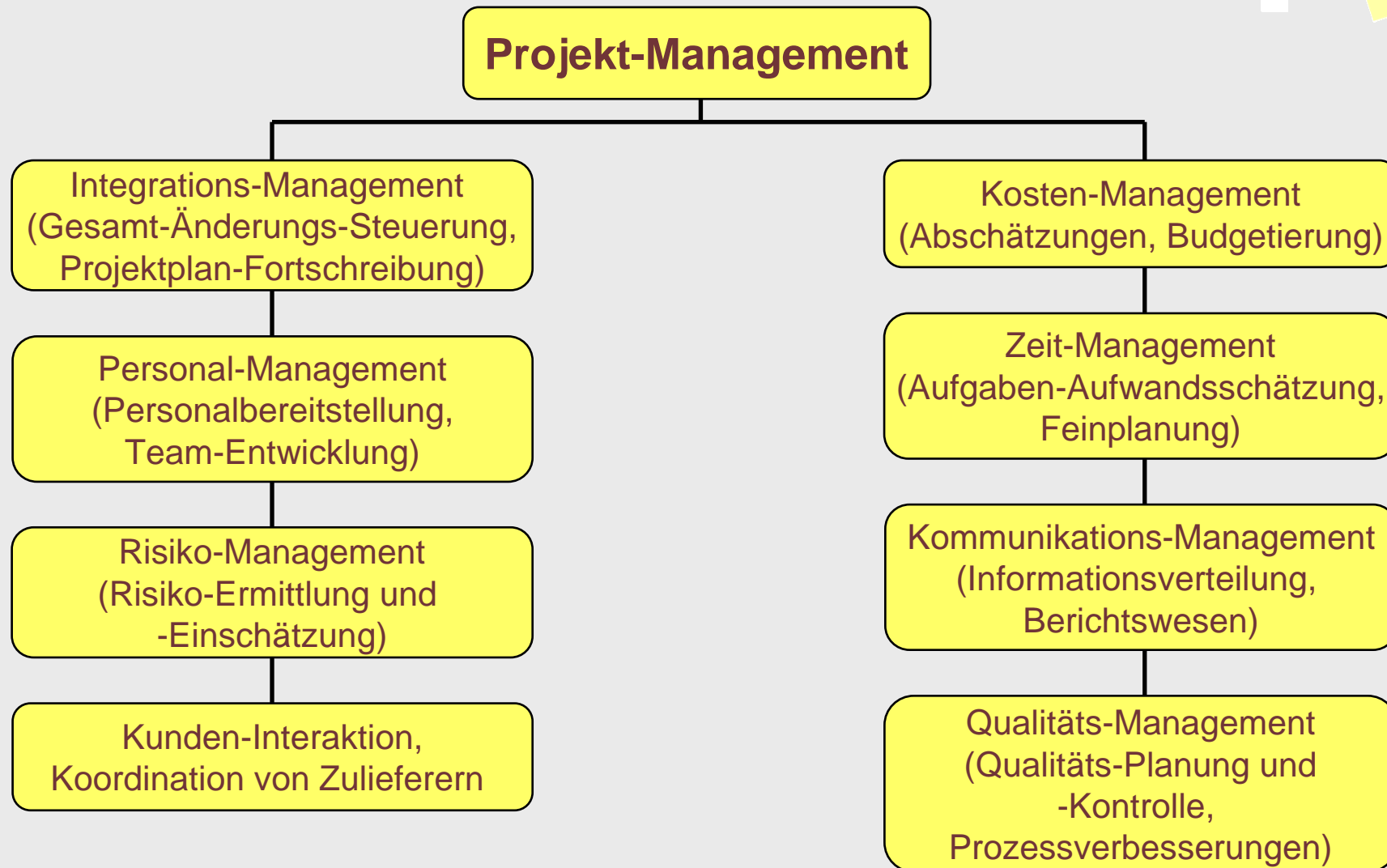
- Definition
- Aufgabenfelder
- Konflikte
- Planung und Abschätzung
- Steuerung und Kontrolle
- Risikomanagement
- Projekthandbuch
- Produkt-Abnahme
- Änderungs-Management
- 3rd Party SW

Projektmanagement: Definitionen

- Definition Projekt:
 - Vorhaben mit definiertem Anfang und Abschluss
 - zeitliche Befristung
 - Neuartigkeit
 - Einmaligkeit
 - Komplexität
 - einmaliger Ablauf
 - Beschränkung der Ressourcen

- Definition Projektmanagement:
 - Anwendung von Methoden, Know-how, Techniken zur Planung, Organisation Führung und Steuerung von Projekten

Aufgabenfelder



Projektziele und mögliche Konflikte

- zweiwöchentliche Lieferung
- volle Prozesseinhaltung
- begrenztes und niedriges Budget
- zusätzliche Anforderungen
- mangelnde Ressourcen



Copyright © 2002 United Feature Syndicate, Inc.

Planung

Planung heißt,
den Zufall durch
den Irrtum zu
ersetzen!



Manchmal laufen
Dinge nicht nach
Plan, weil es nie
einen Plan gab!

Planung: Was muss geplant werden

- Arbeitspakete (Work Package)
 - Inhalt
 - Aufwand
 - Abhängigkeiten
- Termine (Anfang, Ende, Meilensteine, ...)
- Kundentermine (Zwischenstände, Produktiv-Lieferung (Produktionsstart))
- Kosten
- Personaleinsatz (wer macht was, Auslastung der Mitarbeiter)
- Material (z.B. CDs für SW-Auslieferung)
- Kommunikation (z.B. Projektbesprechungen, Kundenkontakt, Berichterstattung)
- Equipment (Tools, Testfahrzeuge)
- Anwendung von (Entwicklungs-) Prozessen
- Qualität

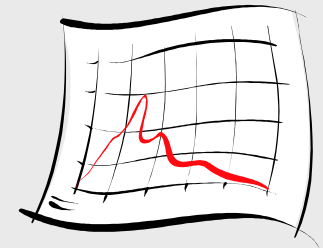
Planung: Projekt- und Systemstruktur

- Ziel: Grobüberblick über das Projekt
- Schritte:
 - Analyse der Inputs
 - relevanten Rahmenbedingungen und Voraussetzung für Projektdurchführung
 - Analyse der Outputs
 - Zielsetzung des Kunden im Hinblick auf Nutzung
 - Abgleich mit den Vorstellungen des Kunden
 - Personelle Schnittstellen abklären und dokumentieren
 - Organisation innerhalb und außerhalb des Projektteams
 - Funktionen definieren
 - Einzelfunktionen des Geräts, keine Arbeitspakete
 - Externe technischen Schnittstellen abklären
 - zur entwickelten HW, Mechanik, Fahrzeug, Produktion

Abschätzung und Planung

➤ Planung für SW-Projekte: warum?

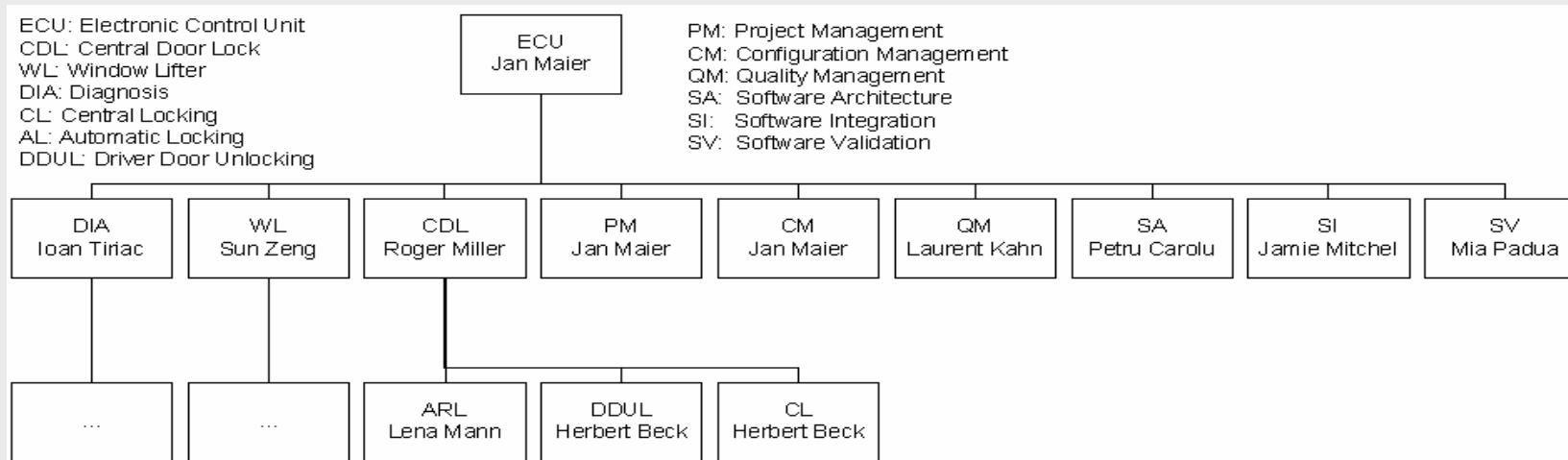
- Kostenüberlauf
- Terminverzug
- schlechte Qualität, viele Fehler/Bugs
- keine Abschätzung von Änderungen, da Annahmen und Aufwand nicht dokumentiert



- Ziel: Verbesserung zum Vorgängerprojekt anstreben
- Erste Aufwandsabschätzung erfolgt bereits in der Angebotsphase
- Grobterminplan als Bestandteil des Angebots
- Verfeinerung bei Entwicklungsstart

Work-breakdown structure

- Ziel: Aufbrechen des Gesamtprojekts in hierarchisch gegliederte Teilpakete
- Grundsätzliche Vorgehensweise.
 - Entwicklungszyklus-orientiert
 - Funktionsorientiert (Requirements)
 - Rollenbasiert
- Beispiel ECU auf oberster Ebene (funktions- und rollenbasiert)



Projektmanagement: Arbeitspaket

- kleinste Einheit in der Work BreakDown Structure (WBS)
- geschlossene überschaubare Arbeitsmenge (eigene und fremde Leistung) (50 bis max. 500 Arbeitsstunden)
- Inhalt:
 - Umfang mit nachweisbarem Ergebnis
 - Eingangsvoraussetzungen
 - die zur Überwachung nötigen Arbeiten (Reporting)
 - Kostenvorgaben
 - eindeutiger Verantwortlicher
 - für eine Projektphase
- Summe aller Arbeitspakete ergibt alle zur Fertigstellung des Projekts notwendigen Arbeiten
- Abhängigkeit der Arbeitspakete untereinander

<u><Responsible></u>	<ID>
<Task>	
<effort>	<cost >

Übung

Work Breakdown Structure

Beschreibung:

Sie planen eine große Geburtstagsfeier mit Freunden.

Aufgabe:

Zerlegen Sie die Organisation in Arbeitspakete (Work Breakdown Structure)

Begründen Sie Ihr Ergebnis!

Arbeiten Sie ggf. zusammen mit einem Partner.

Zeit:

10 Minuten



Schätzmethoden

Voraussetzung: Arbeitspakete liegen vor

1. 4-Augen-Methode

- Schätzung eines Arbeitspaket von bearbeitenden Person
- Vergleichsschätzung durch Erfahrungsträger
- Überprüfung durch SW-Projektleiter
- Dokumentation der Annahmen



2. Prozent-Methode

- basierend auf Vorgängerprojekt wird Arbeitspaket als prozentualer Anteil am Gesamtprojekt abgeschätzt
- historische Daten als Basis

3. Schätzklausur mit Expertenrunden

- mindestens 3 Experten schätzen die Arbeitspakete voneinander unabhängig
 - Diskussion von größeren Abweichungen und Einigung
- => aufwändig aber effektiv !

Ablauf einer Schätzklausur

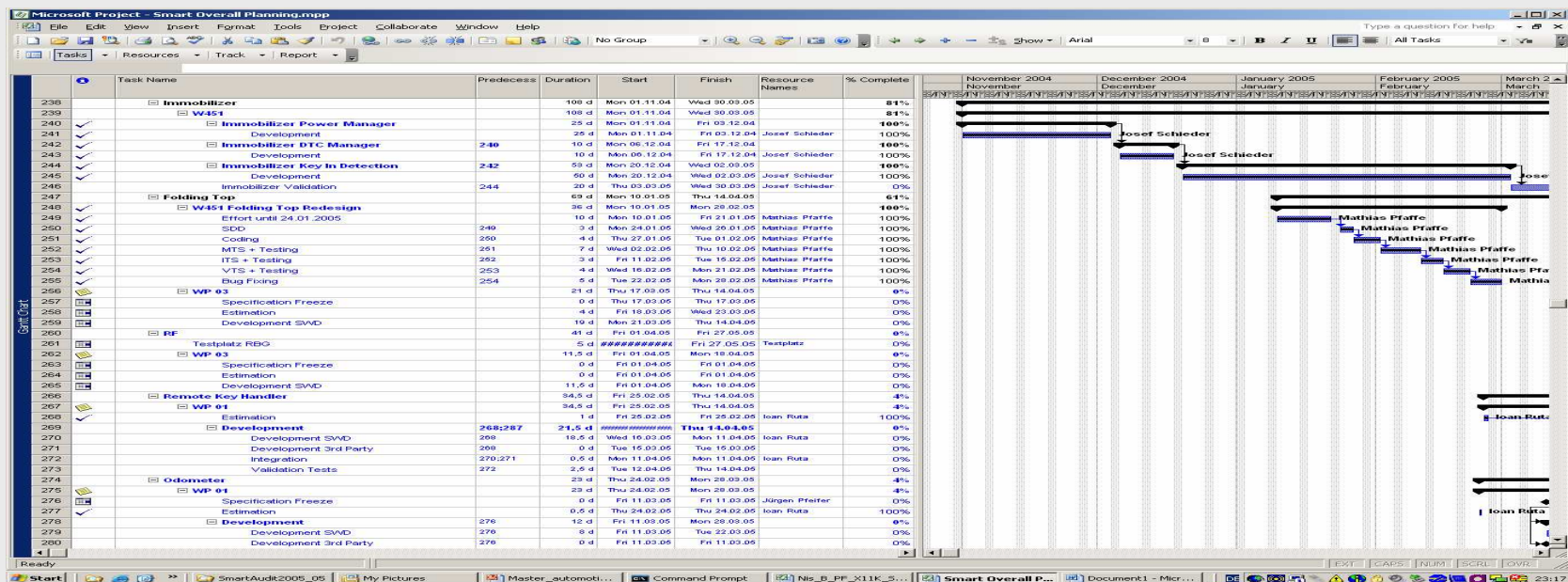
Schritte einer Schätzklausur:

1. Arbeitspakete definieren (durch Moderator oder Projektleiter)
Größe: 1 MW bis 2 MM
2. Arbeitspaket-Liste an Projektmitglieder verteilen
3. Jeder schätzt alle Arbeitspakete (vor der Schätzklausur) ab
4. In der Schätzklausur:
bei nicht-signifikanten Abweichungen Durchschnitt bilden
(Streuung bis ca. 50%, z.B. zw. 10 und 15 MT)
5. Bei signifikanten Abweichungen diskutieren und sich (möglichst)
eilvernehmlich auf eine Aufwandszahl einigen
6. Resultate dokumentieren !



Projektmanagement: Terminplan

- meist "Microsoft-Project" als Tool
- Abhängigkeiten der Arbeitspakete
 - Projekte werden in eine logische Reihenfolge gebracht
 - Arbeitspakete mit Ressourcen hinterlegt
 - Ressourcen und Terminkonflikte sind zu lösen



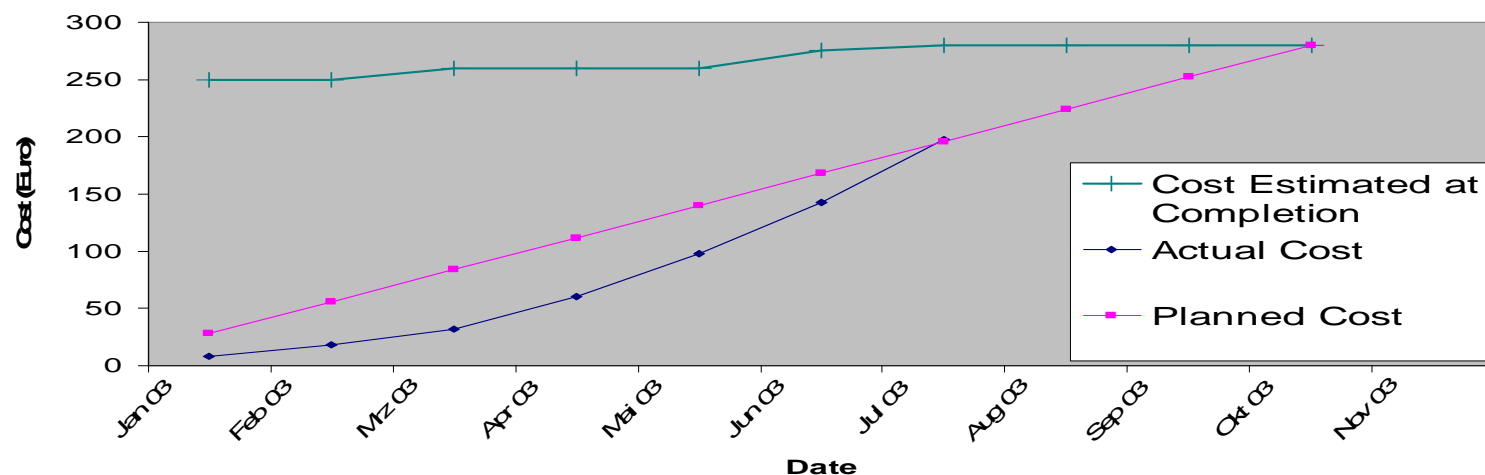
Projektmanagement: Steuerung und Kontrolle

- Definition "Controlling": Leiten, Steuern und Verfolgen
- Planung dokumentieren
- Plan in die Tat umsetzen
- Planverfolgung ("Tracking")
 - Alles was geplant wird, muss auch verfolgt werden
 - Abweichungen analysieren
 - Planung überarbeiten
 - Eskalieren (Managemententscheidung herbeiführen)
 - Tracking dokumentieren
- ggf. Plan verändern
- Berichten (Reporting)
 - Kunden
 - Management



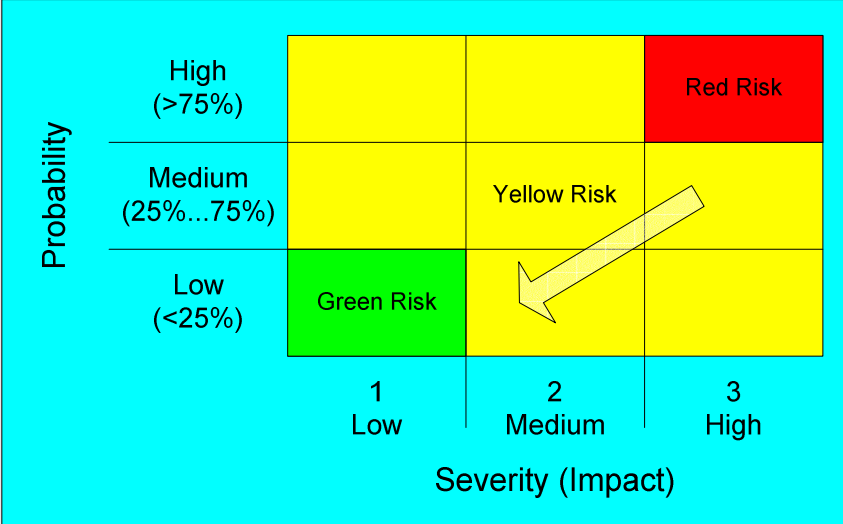
Projektmanagement: Kostenkontrolle

- identifiziert die Abweichung zwischen geplanten und tatsächlichen Kosten
- Vorsicht bei der Interpretation
 - verzögerter Kostenaufbau durch Verspätung im Projekt
 - Möglichkeit von Über- oder Unterschätzung der einzelnen Arbeitspakete



Projektmanagement: Risikomanagement

- Risiken identifizieren
- Risiken bewerten
 - Eintrittswahrscheinlichkeit
 - Auswirkungen und Konsequenzen
- Maßnahmen definieren
 - Eintrittswahrscheinlichkeit verringern durch geeignete Maßnahmen
 - bei Eintritt des Risikos liegt ein zusätzlicher Maßnahmenplan vor



Probability	High (>75%)			Red Risk
	Medium (25%...75%)		Yellow Risk	
	Low (<25%)	Green Risk		
		1 Low	2 Medium	3 High
		Severity (Impact)		

Übung

Risikomanagement

Sie starten als Software Projektleiter ein neues Projekt zur Entwicklung eines Batteriesensors für einen völlig neuen Autobatterietyp. Ihr Team besteht zum einen Teil aus erfahrenen Mitarbeitern, zum anderen Teil aus Kollegen, die noch nie in diesem Bereich gearbeitet haben. Ihr System soll auf einem Controller laufen, der zwar von einem renommierten Hersteller entwickelt wurde, aber gerade auf den Markt gekommen ist. Auch die Entwicklungsumgebung (Compiler, Linker), die Sie verwenden sollen, ist neu. Es besteht allerdings die Möglichkeit, auf einen älteren Compiler umzusteigen. Sie wissen außerdem, dass der Fertigstellungstermin sehr früh ist. Ihr vorheriges Projekt haben Sie erfolgreich abgeschlossen, es besteht aber die Möglichkeit, dass der Kunde doch noch auf einer Änderung des Vorgängerprojekts besteht.

Aufgabe:

Identifizieren Sie Risiken in Ihrem Batteriesensorprojekt, definieren Sie Maßnahmen zur Reduzierung der Eintreffenswahrscheinlichkeit und überlegen Sie Maßnahmen falls ein Risiko doch zum Problem werden sollte.

Begründen Sie Ihr Ergebnis!

Arbeiten Sie ggf. zusammen mit einem Partner.

Zeit:

10 Minuten



SW –Development Plan / Projekthandbuch (1)

➤ Ziel:

- wichtigste Eckdaten des Projekts in ca. 10 Seiten
- internes "Vertragswerk" zwischen Organisation, SW-Projektleiter und SW-Entwicklern
- jeder Projektbeteiligte sollte es lesen

SW –Development Plan/ Projekthandbuch (2)

- Inhalt:
 - Systembeschreibung
 - Projektumfang: Was soll gemacht werden?
 - Festlegen der Verantwortlichkeiten (nach außen und Innerhalb des Teams)
 - Hauptmeilensteine
 - Projektziele
 - Arbeitsmittel (z.B. Testplätze, PCs)
 - Zuliefernsoftware
 - Prozess "Deviations" : Tailoring
 - Projektorganisation in Bezug auf
 - Meetings
 - Dokumentablage (Konfigurationsmanagement)
 - Eskalation

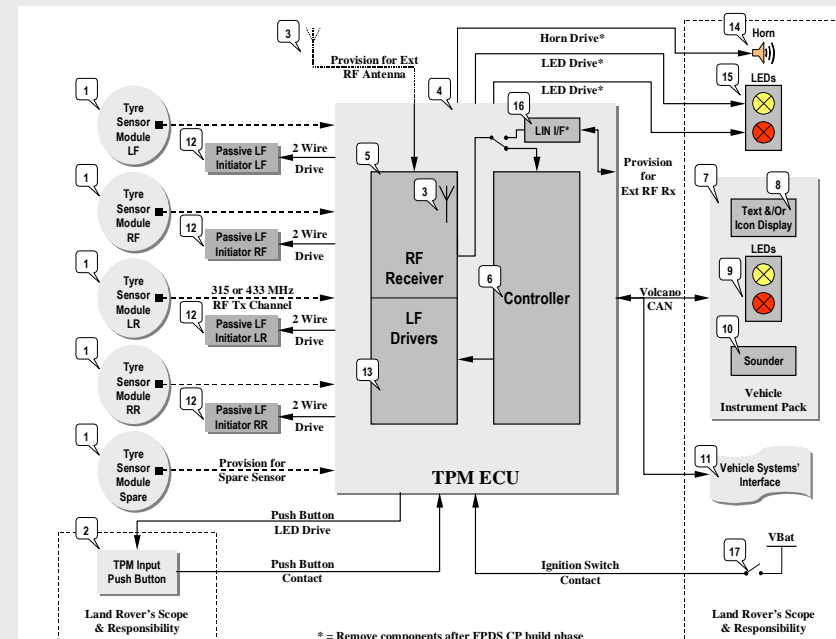
Definition des funktionellen Inhalts

- Requirements Engineering:
 - Kunde (z.B. Automobilhersteller) erstellt Lastenheft
 - Zulieferer erstellt Pflichtenheft
 - Pflichtenheft ist Implementierungsgrundlage (sollte vom Kunden abgenommen werden)
- Idealprozess:
 - Analyse des Lastenhefts
 - Klärung der offenen Punkte
 - Erstellung des Pflichtenhefts
 - Review zusammen mit dem Kunden
 - anschließende Freigabe
- Realität:
 - massive Abweichungen von den Prozessen, Plan, Budget
 - Requirements werden nie richtig festgeschrieben
 - offene Punkte werden zu spät geklärt

Beispiel: Produkt-Beschreibung

Beispiel:

- The Tireguard is a tire pressure monitor, which shall inform the driver whether the tires (spare tire included) have high or low pressure. A rapid pressure loss warning shall be implemented too. There is no warning if there is a sudden pressure loss (for example: a tire bursts). The warnings shall be visualized either by leds only or on an instrument cluster by a textual message and leds. All the communication is done via a mid-speed CAN-bus. The ECU doesn't have any safety critical functionality. The ECU shall be configurable by EOL-programming. This is needed for example to program variable pressure thresholds. In order to achieve, that the variants with a RF of 433 MHz and 315MHz (MFLBs 5WK4 7593 and 5WK4 7594) can be handled by the same software the variant coding for these variants will be programmed at the supplier production line.



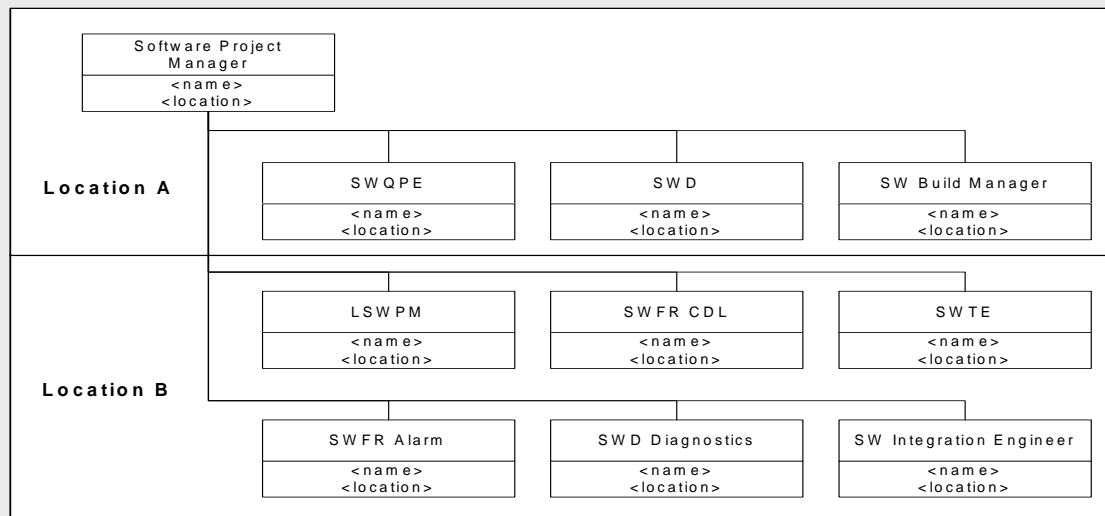
Rollen im SW-Projekt

Rolle	Verantwortlichkeit
SWPM	Software Projekt Manager <ul style="list-style-type: none">• Verantwortlich für das allgemeine Management des SW- Projekts (Planen, Kontrollieren)• Verteilt und kontrolliert Arbeitspakete• Vertritt Projekt nach innen und nach außen zu Kunde und Zulieferern
SWA	SW-Architekt <ul style="list-style-type: none">• Legt die Architektur(Schnittstelle zwischen den Funktionalitäten) für das Projekt fest• Kümmt sich um die übergreifende Design Themen wie Performance, RAM-, ROM – Verbrauch
SWQPE	Software Quality Planning Engineer / Software Qualitätsingenieur <ul style="list-style-type: none">• Durchführung der Projektphasenreviews• Reporting des Qualitätsstatus ans Management.• Teilnahme an Arbeitsproduktreviews (Code, Dokumente ...).
SWTE	Software Testing Engineer <ul style="list-style-type: none">• Entwickelt Testspezifikationen• Führt Tests durch (kann Modul-, Validierungs-, System-Tests sein)
LSWPM	Local Software Project Manager (bei Standort-übergreifender Entwicklung) <ul style="list-style-type: none">• Lokaler Vertreter des SW-Projektleiter an einem Standort falls signifikante Teamgröße• Ähnlich zu einem Teilprojektleiter
SWI	SW Integrationsingenieur <ul style="list-style-type: none">• Integration des Betriebssystems auf die HW• Integration und Zusammenbau der einzelnen SW-Funktionen
SWD	SW Developer/ Entwickler <ul style="list-style-type: none">• Erstellt den kompletten V-Zyklus für eine Funktionalität (Requirements, Design, Code, Modul- und Integrationstest)

Zusammenarbeit über verschiedene Standorte

- Schnittstellen über verschiedenen Standorte hinweg
 - funktionelle Schwerpunkte (technische Spezialisten) nur an einzelnen Standorten verfügbar
 - Kommunikation größtenteils nur über E-mail, Telefon und Netmeeting
 - Kick-Off-Meeting extrem hilfreich, regelmäßige Treffen wären wünschenswert aber sind in der Regel nicht möglich

=> Festlegung der Arbeitspakete extrem wichtig



Abnahme der Produktentwicklung

- Systemtest hauptsächlich zur funktionellen Abnahme des Produkts durch Kunden
- Verschiedene Testebenen, z.B.
 - HW mit Simulationen von Kommunikationsbussen und Lasten
 - Fahrzeug-Simulation im Labor mit realen Steuergeräten
 - komplettes Fahrzeug
- Abgrenzung zwischen Komponentenverantwortung und Systemverantwortung wird oft vermischt
- Systematische Integration der Komponenten wird oft abgekürzt durch direktes Einbauen ins Gesamtsystem (z.B. Fahrzeug) mit nicht-kompatiblen Schnittstellen

=> Zeitaufwand muss mit eingeplant werden

=> Abgrenzung der Verantwortung wichtig

Änderungsmanagement

- Anforderungen ändern sich über die Projektlaufzeit
- Teil der Anforderungen ist zu Projektstart noch nicht definiert
- Detailliertes Angebot ermöglicht Ausweisung des Zusatzaufwandes und dessen Einforderung beim Kunden
- Kontrolliertes Einspeisen der Anforderungen zum Beginn eines V-Zyklus notwendig
- Auswirkung auf Requirements Engineering und Validierung
- Anpassung von Termin-, Kosten- und Ressourcen-Planung notwendig

3rd Party SW

- Ursachen für Notwendigkeit zum Einbau von 3rd Party SW, z.B.:
 - Automobilhersteller fordert einheitliche Module über gesamtes Fahrzeug (z.B. CAN, Diagnose, OSEK, Bootloader für Reflash)
 - Entwicklung einzelner Funktionalitäten beim Hersteller oder anderem Know-how-Träger (z.B. neue Funktionalitäten wie Reifendruck, Kryptologie)

- Formen der Zulieferung
 - Source Code
 - Object Code (+ Header File(s))
 - Modelle in Modellierungssprache zur direkten Codegenerierung (z.B. Statemate, Matlab, ASCET-SD, Rapsody)

3rd Party SW: Probleme

- Architekturinkompatibilitäten
 - statisch: z.B. Namenskonflikte, verschiedene Maßeinheit von Variablen
 - dynamisch: z.B. unterschiedliche Modulinteraktion wie Event versus Polling
- unterschiedliche funktionelle Aufteilung
- unterschiedliche Design-Konventionen
 - z.B. Präemptives und nicht präemptives Scheduling
- Entwicklungsumgebung
 - z.B. Compilerabhängigkeit
- Qualitätsprobleme: Verantwortung im Fehlerfall, beschränkte Gewährleistung

3rd Party SW: Lösungsansatz

- Integrationsdokument
 - Detaillierte SW Interface Beschreibung der 3rd-Party SW (functions, global variables, interrupts, data types, timings, access mechanisms)
 - detaillierte Beschreibung der benötigten, d. h. von einem selbst zur Verfügung zu stellenden Interfaces und deren Benutzung
 - Ressourcenverbrauch (RAM incl. Stack, ROM, EEPROM, μ C) (wichtig für Embedded Systeme)
 - Umweltbedingungen und Seiteneffekte (task, interrupt, μ C operation modes, μ C registers, interrupts en-/disabled)
 - Konfiguration
 - Entwicklungsumgebung (z. B. Compiler/Linker settings)
 - Beschreibung des Integrationstests
- gemeinsame Integrationstests mit Integrations-Review
- rechtlich bindende Vereinbarung von Gewährleistung und Haftung

Zum Schluss dieses Abschnitts ...

Noch Fragen ??