# Beschreibungsmuster für Modellelemente

Die folgenden Beschreibungsmuster finden sich in dieser Form in

Peter Hruschka, Chris Rupp: *Agile Softwareenwicklung für Embedded Real-Time Systems mit der UML, Hanser Verlag, 2002*

## Beschreibungsmuster für Systemprozesse (Kite-Level)

|  |  |
| --- | --- |
| **Name** | Aussagekräftiger Name, oft Substantiv + Verb (Bsp: Fahrstuhl anfordern) |
| **Akteur** | Initiator |
| **Auslösendes Ereignis** | Trigger des Prozesses |
| **Kurzbeschreibung** | 2-4 Sätze |

## Beschreibungsmuster für Use Cases (Wave oder Fish-Level)

|  |  |
| --- | --- |
| **Name** | Aussagekräftig (Subjekt + Verb) |
| **Akteur** | Wer initiiert die Aktion / Aktivität |
| **Auslösendes Ereignis** | Trigger-Ereignis |
| **Kurzbeschreibung** | Informell |
| **Vorbedingungen** | Anforderung an globalen Zustand, Werte von Parametern o.ä. |
| **Essentielle Schritte** | |  |  | | --- | --- | | **Intention der Systemumgebung** | **Reaktion des Systems** | | Schritt 1: Akteur + Event | Reaktion 1 | | … | …. | |  |  | |  |  | |  |  | |  |  | |
| **Ausnahmefälle** | Später einfügen, Initial nur Normalverhalten |
| **Nachbedingungen** |  |
| **Zeitverhalten** | Wenn relevant |
| **Verfügbarkeit** | So etwas wie erwartete / notwendige MTBF o.ä. |
| **Fragen/Kommentare** | Als Merkzetten, Diskussionsgrundlage |

Kürzer als 2 Seiten

Intention/Reaktion für Wave Level; Fish Level: Szenario mit technischen Details – nur wo benötigt!

## Beschreibungsmuster für Klassen

(Abstrakt; zu verfeinern wie unten beschrieben)

|  |  |
| --- | --- |
| **Name** | aussagekräftig |
| **Definition** |  |
| **Zusatzinformation** | (tagged values) |
| **Attribute** | Sofern schon bekannt |
| **Fragen/Bemerkungen** |  |

## Beschreibungsmuster für Anforderungen und Randbedingungen

|  |  |
| --- | --- |
| **Beschreibung** | Umgangssprachliche Formulierung |
| **Quelle** | Wer hat die Anforderung erstellt, auf wen geht sie zurück? |
| **Begründung** | Warum ist diese Anforderung sinnvoll |
| **Abhängigkeiten** | Verweis auf andere Anforderungen, Randbedingungen (übergeordnet oder Wechselwirkung) |

## Beschreibungsmuster für Knoten (Deployment)

|  |  |
| --- | --- |
| **Name** | aussagekräftig |
| **Verantwortung** | Rolle des Knotens im Gesamtsystem (ev. Verweis auf HW/SW Zuordnung – wenn bekannt) |
| **Funktionale Anforderung** | Prozesse, Daten, Subsysteme eventuell Verweis auf HW/SW Matrix |
| **Designbegründung** | Warum diese Knotenart? Alternativen? |
| **Designrechtfertigung** | Warum leistet der Knoten seine Aufgaben? |
| **Nicht-funktionale Anforderungen** | QA Anforderungen und Randbedingungen die diesem Knoten zugeordnet sind (über Id der Anforderung) |

Ergänzung: HW/SW Zuordnungsmatrix (s.u.)

## Beschreibungsmuster für Verbindungsspezifikationen

|  |  |
| --- | --- |
| **Name** | aussagekräftig |
| **Charakteristika** | Kenngrößen und Einschränkungen |
| **Funktionale Anforderungen** | Zuordnung zu Requirements die die Information betreffen, die über diese Schnittstelle gehen |
| **Designbegründung** | Warum diese Verbindungsart? Welche Alternativen |
| **Designrechtfertigung** | Warum leistet diese Verbindung die geforderten Aufgaben? |
| **Nicht-funktionale Anforderungen** | QA Anforderungen und Randbedingungen die dieser Verbindung zugeordnet sind (über Id der Anforderung) |

## Beschreibungsmuster für Subsystemspezifikationen

|  |  |
| --- | --- |
| **Name** | aussagekräftig |
| **Verantwortung** | Aufgabe des Subsystems in seiner Umgebung |
| **Designbegründung** | Nach welchen Kriterien wurde das Subsystem gebildet? Welche Alternativen wurden betrachtet und warum verworfen |
| **Designrechtfertigung** | Begründung warum dieser Entwurf des Subsystems die genötigten Aufgaben erfüllt |
| **Nicht-funktionale Anforderungen** | QA Anforderungen und Randbedingungen die diesem Subsystem zugeordnet sind (über Id der Anforderung) |

Grundlegende Typen von Systemen:

* Steuerung und Koordination
* Datensammlung
* Datenanalyse
* (Basis-)Dienste

Technische Kategorien:

* Mensch/Maschine-Schnittstelle
* Eingabeschnittstelle
* Ausgabeschnittstelle
* Systemdienste

# Beschreibungsmuster für SW Elemente

## Beschreibungsmuster für Klassen

|  |  |
| --- | --- |
| **Name** | Aussagekräftiger Name, eventuell Synonyme |
| **Definition** | Was stellt die Klasse dar und warum ist es für die Klasse wichtig? |
| **Zusatzinformation** | Details, etwa Info der Stakeholder, sollte im Laufe der Zeit verfeinert werden: zu Attributen, Methoden, Zuordnung zu einer Klasse etc |
| **Merkmale/ Eigenschaften** | Frei definierbar (tagged values); zB Autor, Instanziierbarkeit, Ableitbarkeit, Persistenz |
| **Regeln für Konstruktion und Destruktion** |  |
| **Sichtbarkeit** |  |
| **Interfaces** | Schnittstelle, die die Klasse implementiert |
| **Mengengerüst** | Anzahl der Objekte dieser Klasse, die gleichzeitig existieren können |
| **Fortschritt der Implementierung** | Prozent der Klasse, die in Code umgesetzt ist |
| **Fragen / Bemerkungen** | Schnittstelle, die die Klasse implementiert |

## Beschreibungsmuster für Beziehungen

|  |  |
| --- | --- |
| **Name** | Aussagekräftiger Name, eventuell Synonyme |
| **Definition** | Was stellt die Beziehung dar und warum ist es für die Klasse wichtig? |
| **Zusatzinformation** | Details, etwa Info der Stakeholder, sollte im Laufe der Zeit verfeinert werden: zu Attributen, Methoden, Zuordnung zu einer Klasse etc |
| **Merkmale/ Eigenschaften** | Frei definierbar (tagged values); zB,Persistenz |
| **Regeln für Konstruktion und Destruktion** |  |
| **Multiplizitäten** | Anzahl der in Beziehung stehenden Objekte |
| **Ableitungsinformation** | Bei abgeleiteten Beziehungen |
| **Mengengerüst** | Anzahl von Beziehungen dieser Art, die gleichzeitig existieren können |
| **Fragen / Bemerkungen** | c |

## Beschreibungsmuster für Attribute

|  |  |
| --- | --- |
| **Name** | Aussagekräftiger Name, eventuell Synonyme |
| **Definition** | Was stellt das Attribut dar und warum ist es für die Klasse wichtig? |
| **Zusatzinformation** | Details, etwa Info der Stakeholder, sollte im Laufe der Zeit verfeinert werden: zu Attributen, Methoden, Zuordnung zu einer Klasse etc |
| **Merkmale/ Eigenschaften** | Frei definierbar (tagged values); zB, Veränderbarkeit, Persistenz |
| **Datentyp** | Programmiersprachunabhängig, dh. Abstrakte Sicht |
| **Wertebereich** | Werte die angenommen werden können (Teilmenge des Datentyps) |
| **Initialwert** |  |
| **Struktur des Attributs** | Bei komplexen Attributen |
| **Ableitungsinformation** | Quelle der Attribute, eventuell Ableitungsregeln |
| **Fragen / Bemerkungen** | Notizen der Designer / Implementierer |

## Beschreibungsmuster für Operationen

|  |  |
| --- | --- |
| **Name** | Aussagekräftiger Name, eventuell Synonyme |
| **Definition** | Was macht die Operation, wer führt sie aus? |
| **Zusatzinformation** | Details, etwa Info der Stakeholder, sollte im Laufe der Zeit verfeinert werden: zu Attributen, Methoden, Zuordnung zu einer Klasse etc. |
| **Merkmale/ Eigenschaften** | Frei definierbar (tagged values); zB, Überschreibarkeit, Instanziierbarkeit |
| **Eingabe** | Input Parameter und ihr Typ |
| **Ausgabe** | Rückgabewerte und ihr Typ |
| **Exceptions** | Solche die von der Operation generiert werden können und wann |
| **Ablauf** | Algorithmus bzw allgemein Ablauf – eventuell Hinweis auf ein Modell |
| **Dauer** | Maximale Ausführungszeit (wenn sinnvoll) |
| **Vorbedingungen** | Bedingung an Eingabeparameter etc die nicht geprüft werden und so vom Aufrufer sichergestellt werden müssen |
| **Nachbedingungen** | Garantierte Bedingung nach Fertigstellen |
| **Fragen / Bemerkungen** | Notizen der Designer / Implementierer |

Verschiedene Guidelines für die Orientierung:

Aus Hruschka/Rupp (s.o.)

## Kontexte

Bei der Definition des Systemkontextes können verschiedene Modelle für verschiedene Arten von Kontexten verwendet werden. Dabei spielen auch die Schnittstellen eine Rolle, dh. Die Art wie die Kontextelemente mit dem Zielsystem verbunden sind.

* Logische Kontexte: Fokus Kommunikation mit Nachbarsysten / Nachbarkomponenten
  + Use-Case-Diagramme bei unspezifizierten Schnittstellen (meist hohe Abstraktionsebene)
  + Klassendiagramme bei Ein-/Ausgabeschnittstellen (globale Daten, Methoden)
  + Sequenzdiagramme bei Nachrichten zwischen verschiedenen Teilkomponenten
* Physikalische Kontexte: physikalische Verbindungen zu Nachbarsystemen / Nachbarkomponenten (Kanäle)

Klassifizierung von Akteuren im System als Stereotype in Modellen:

* Mensch / Strichmännchen (bei eingebetteten Systemen selten)
* <<Device>> : Ein-/Ausgabegeräte (manchmal)
* <<Sensor>>: Sensoren oder Quellen hinter Sensoren (oft)
* <<System>>: Nachbarsystem (oft)
* Zeit (oft)

## Systemprozesse finden

Einige Tricks zur Identifizierung von Systemprozessen

* Zielsystem aus Sicht des umgebenden Systems betrachten – was ist aus der Umgebung wahrnehmbares Verhalten / Funktion?
* Klassifizieren von Nachbarsystemverhalten und Ableiten von Zielsystemprozessen:
  + Aktive Nachbarsysteme (vgl Akteure oben) – lösen Systemprozesse aus
  + Kooperierende Nachbarsysteme – werden um Auskunft gebeten, gemeinsames Interesse an Zielerreichung
  + Autonome Nachbarsysteme – arbeiten eigenständig, zeitweise kooperativ
* Frühes Einbeziehen von Zustandsmodellen

## Glossary

Begriffsdefinitionen als gemeinsame Wissensbasis

* Sammlung von nicht allgemein bekannten Begriffen und ihrer Erklärung / Definition im Projektkontext.
* 20-50 Begriffe
* Regeln zur Definition
  + Definitionen im Aktiv – passiven Formulierungen fehlt häufig das Subjekt (falsch: Das XY wird erzeugt; richtig: A erzeugt das XY)
  + Verwenden einfacher Vollverben (vermeiden von machen, tun, haben, können, sein)
  + Vermeiden von Nebensätzen und redundanten Ausdrücken
* Anmerkung: Begriffsdefinitionen sind keine Anforderungen!
* Muster für Substantive über
  + Klassifizierung: Ein XY ist ein….
  + Merkmale / Eigenschaftten / Fähigkeiten: Ein XY zeichnet sich aus durch…
  + Verhalten / Aktionen
* Definition von Verben und Adjektiven:
  + Zusammenhang zu Subjekten klarmachen
  + Adjektive zusammen mit Substantiven: Ein ab XY ist ein XY dass….
  + Verben: im Zusammenhang mit Systemprozessen

## Randbedingungen sammeln

Randbedingungen und nicht-funktionale Anforderungen sollten klassifiziert werden. Die Klassifizierung hilft auch, die Herleitung und Analyse dieser Aspekte systematisch anzugehen. Kategorien von Randbedingungen:

* Qualitätsanforderungen
* Randbedingungen für den Entwurf
* Anforderungen an den Entwicklungsprozess
* Randbedingungen für das Management (Zeitraum, Budget, Verfügbarkeit von Personal, bekannte Risiken, Berichtswesen etc.)

## HW/SW-Verteilungsmatrix

Zuordnung von SW-Komponenten/SW-Subsystemen auf HW-Knoten im System:

|  |  |  |  |
| --- | --- | --- | --- |
|  | HW Knoten 1 | HW Knoten 2 | HW Knoten 3 |
| SW Komponente 1 |  |  |  |
| SW Komponente 2 |  |  |  |
| SW Komponente 3 |  |  |  |

.

Zweck der Matrix

* Überblick über HW/SW Komplexität (Management, Kunde); auch: Basis von Iterationsplanung und Clusteranalyse
* Planung und Zuordnung von Aktivitäten von Systemprozessen; Identifizieren der Aufgaben der Knoten (Designer), Implementierer
* Basis für QA und CM

## Kategorien von Klassen:

* Entity-Klassen
  + Halten und kapseln Daten
  + Aus Aktivitäts-, Sequenz- und Zustandsdiagrammen: welche Daten werden in den Klassen/Objekten benötigt?
    - Gibt es Aktivitäten oder Aktionen für die die benötigten Daten noch zu identifizieren sind?
    - Welche Informationen werden benötigt, damit Aktivitäten oder Aktionen arbeiten können
    - Werden von Aktivitäten oder Aktionen Daten erzeugt?
    - Aktivitäten, die Starten müssen wenn bestimmte Daten vorliegen (dh. Unabhängig vom Steuerfluss)?
* Sichten-Klassen
  + Kaum eigene Attribute, arbeiten mit Attributen anderer Klassen (Entity) oder abgeleiteten Attributen
  + Typische Aufgaben: Zusammenstellen und Verteilen von Daten, Verdichten von Daten – meist über mehrere Objekte hinweg für Steuerung, Kommunikation etc.
  + Nur zu bestimmten Zeitpunkten im Systemverlauf notwendig (dh. Im Sequenzdiagramm erzeugt und vernichtet
  + Meist reine Zustandsautomaten
  + Auch: Prüfen von Protokollen
  + Ermittlen zB. Ausgehend von Entitiy-Klassen
* Steuerungsklassen
  + Koordination von Abläufen, auch durch Delegation
  + Wenige eigene Attribute, zB., Status eines Prozesses, Anfangs- / Endzeitpunkte
  + Existieren nur während des Ablaufs der Steuerungsprozesse – oft nur kurzlebig, bei Gesamtsystem das Zustandsautomat ist dann auch immer
  + Oft Zustandsautomaten mit globalen Abläufen, senden Nachrichten an andere Objekte, reagieren auf Ereignisse, die von anderen Objekten erzeugt werden
  + Zu finden in parallelen Aktivitäten (Modell) oder Zustandsdiagrammen: Daumenregel: wenn mehr als 3 Objekte in Aktionen / Aktivitäten beteiligt sind lohnt sich eine Steuerungsklasse
  + Auch ablesbar aus Sequenzdiagrammen: Objekte die sehr viele andere Objekte ansprechen; entkoppeln durch Steuerungsklasse (Mediator Pattern)
* Service-Klassen
  + Kapseln Dienstleistungen, die von verschiedenen Objekten benötigt werden
  + Zu finden in
    - Use Case Diagrammen: includes
    - In SW-Prozess-Beschreibungen: Schritte, die in mehreren Teilprozessen auftauchen?
    - In State-Charts:gleiche Aktivitäten / Aktionen an verschiedener Stelle? Eventuell in mehreren State-Charts?
    - Sequenzdiagramme: mehrere Stellen an denen gleichartige Operationen auftreten bei verschiedenen Objekten?
    - Entity-Klassen: Anteile, die allgemeingültig sind und an verschiedener Stelle wieder auftauchen (könnten)?

## Kriterien zur Taskbildung

Tasks als Bestandteil von aktiven und passiven Komponenten; entsprechen in sich abgeschlossenen funktionalen Abläufen zB als Methoden oder Methodenfamilien zu implementieren.

Taskarten:

* Durch Systemumgebung motiviert:
  + Interrupt Task
    - Für aktive Eingabegeräte oder Nachbarsysteme
    - Quelle: Kontext, Gerät, Subsystem als Trigger eines Systemprozesses
  + Polling Task
    - Für passive Eingabegeräte oder Nachbarsysteme, die abgefragt werden müssen
    - Quelle: : Kontext, Gerät, Subsystem als Trigger eines Systemprozesses
  + Input Coordination Task
    - Für mehrere gleichartige aktive Eingabegeräte oder Nachbarsysteme
    - Quelle: logischer Kontext
  + User Input Task
    - Für Eingabeschnittstelle zum Bediener
    - Quelle: Kontext, Mensch als Trigger eines Systemprozesses
  + Output Decoupling Task
    - Zur entkopplung der Ausgabe (bei Bedarf)
    - Quelle: Systemprozessbeschreibung
  + Output Coordination Task
    - Für mehrere Prozesse, die das gleiche Ausgabegerät oder Nachbarsystem ansprechen
    - Logischer Kontext
  + Time Triggered Task
    - Für Prozesse mit Zeitauslösern
    - Quelle: Zeit als Trigger eines Systemprozesses
* Intern motiviert:
  + Steuerungs Task
    - Zur Koordination mehrerer Prozesse
    - Quelle: Zustandsmodelle, Aktivitätsdiagramme etc
  + Periodische Tasks
    - Zur regelmäßigen Durchführung interner Prozesse
    - Quelle: intern auftretende zyklische Überwachung oder Berechnung
  + Service Task
    - Zur Entkopplung wieder verwendbarer Prozesse
    - Quelle: interne Ereignisse
  + Rechenintensive Task
    - Zur Entkopplung zeitintensiver Prozesse
    - Quelle: Aktivitätsbeschreibungen