Todo list

1 Einführung

Productlifecyclemanagment, oder kurz PLM, ist der Versuch alle Daten, die im Lebenszyklus eines Produktes anfallen, zu verwalten. Dabei überlappen manche Funktionalitäten mit denen eines Versionskontrollsystems (VCS), welches in der reinen Softwareentwicklung verwendet wird.

PLM kann in die drei folgenden Bereiche eingeteilt werden:

- Product Development
- Product Manufacturing
- Product Ownership

Product Development und Product Manufacturing beschäftigen sich mit der Produktnummer bzw. um das einzelne Produkt, wohingegen Product Ownership eine ganze Produktserie behandelt.

Produkte werden auch noch in Produktfamilien und Produktvarianten eingeteilt. Eine Produktfamilie fasst ähnliche Produktvarianten zusammen, welche sich nur in einzelnen Attributen voneinander unterscheiden.

Produkte sind anhand ihrer Produktdefinitionsdaten nachbaubar, würde in Software also dem Quellcode entsprechen. Im Gegensatz dazu existieren noch die Produktspezifikationsdaten, welche nur das Verhalten des Produktes beschreiben, und somit den Schnittstellendefinitionen einer Software entspricht. Bei der Entwicklung dieser Daten sollten auch die bereits existierenden Normen anderer Firmen oder Organisationen beachtet werden.

1.1 Lebenszyklus

Der Lebenszyklus eines Produktes hängt ganz wesentlich von der Industrie ab. Die Industrien werden in verschiedene Typen eingeteilt.

Process Production Hier läuft ein Prozess durchgehen ab, wie zum Beispiel beim Verarbeiten von Rohöl in einer Pipeline.

Batch Production Bei diesem Typ werden sehr große Mengen, wie zum Beispiel ganze Paletten, von einem Produkt produziert.

Embedded Systems Hier ist die Verwendung einer reinen Versionskontroll etwas schwierig, da die Software Teil eines physikalischen Produktes ist. Wenn man diese Produktdaten zusammen verwalten will, muss man ein PLM-System einsetzen.

Construction Das Bauwesen stellt einen Sonderfall dar, da das Produkt, typischerweise ein Gebäude, vor Ort erstellt werden muss.

2 Spezifizierungstechniken

2.1 Objektorientierte Grundlagen

Alles (z.B. ein Fenster, eine Person) kann ein Objekt sein. Alle Objekte verfügen zudem über verschiedene Charakteristiken (z.B. Gewicht, Größe) und Verhalten (z.B. offen, geschlossen). Objekte die die selben Arten von Charakteristiken, Verhalten und Constraints haben, können in Klassen zusammengefasst werden.

2.1.1 Klassen

Klassen kann man sich im Informatikbereich als eine Art Bauplan für Objekte vorstellen. Im Prinzip sind Klassen selbst auch wieder Objekte und können reale Dinge oder computer-interne Representationen sein. Ein Objekt kann zu mehr als einer Klasse gehören (Mehrfachvererbung).

Spezialisierung und Generalisierung Die Beziehung zwischen Klassen kann so gesehen werden, dass jedes Objekt einer spezielleren Klasse auch ein Objekt einer genereleren Klasse ist.

- Multiple Klassifizierung
- Überlappende Klassifizierung
- Dynamische Klassifizierung

2.2 UML Klassendiagramm

UML ermöglicht verschiedene Sichtweisen auf ein und dasselbe Model. Je nach Programmiersprache gibt es angepasste Varianten von UML die nur das als UML zulassen was sich auch in der jeweiligen Programmiersprache wirklich umsetzen lässt.

2.2.1 Klassen

Eine Klasse ist in UML eine Repräsentation von einer Gruppe von Dingen mit denselben Charakteristiken und Verhalten. Operationen von Klassen sind die Definition eines Verhaltes einer Klasse.

- Die UML-Spezifikation sagt nichts darüber aus, welche Art von Klassifizierung verwendet werden darf / kann.
- UML sagt auch nichts darüber aus welche Art der Konfliktlösung bei Mehrfachverwendung von selben Operationsnamen oder Attributnamen verwendet wird (z.B. bei Diamantenproblem). Darum ist die Forderung nach einer expliziten Neudefinierung eines Attributs / einer Operation möglich.

2.2.2 Assoziation

Eine Assoziation ist eine Verbindung zwischen 2 oder mehreren Objekten und wird als Linie zwischen den dazugehörigen Klassen dargestellt. Verbindungen können auch auf die selbe Klasse verweisen um mehrere Instanzen einer Klasse miteinander zu verbinden. Assoziationen sind an sich auch wieder Klassen und bieten somit die Möglichkeit bei den Verbindungen selbst weitere Attribute / Operationen zu definieren.

Nie mehr als 2 Klassen miteinander Verbinden da es zu Probleme mit den Kardinalitäten bzw. deren logischer Auflösung führen kann.

Navigation In welche Richtung eine Assoziation führt wird mit Pfeilen ausgedrückt - ohne Pfeile bedeutet es, dass die Assoziation in beide Richtungen führt.

Aggregation Eine Verbindung die Darstellt welche Klasse ein Teil / im Besitz einer anderen Klasse ist(leere Raute).

Composition Eine Komposition ist auch eine Verbindungsvariante zwischen Objekten, wobei ein Objekt nur so lange exisitieren kann, solange es von einem anderen Objekt besessen wird. Außerdem kann das Objekt nicht von mehr als einem anderen Objekt besessen werden (ausgefüllte Raute).

Dependency Dient der Dokumentierung irgend einer Verbindung ohne weitere Aussage darüber, welcher Art diese Verbindung ist (strichlierte Linie).

2.3 Weitere Diagramm-Arten

Neben herkömmlichem UML gibts es auch weitere Diagrammarten:

SysML Anforderungsdiagram Diese Diagramme gibt es in graphischer und tabellarischer Form sowie als Metamodel.

UML Objekt Diagramm Das Objektdiagramm ist ein Strukturdiagramm und umfasst ausprägungen von Klassen und Assiciationen.

2.4 UML Communication Diagramm

Das Kommunikationsdiagramm ist ein Verhaltensdiagramm. Es zeigt eine bestimmte Sicht auf die dynamischen Aspekte des modellierten Systems und stellt Interaktionen grafisch dar, wobei der Austausch von Nachrichten zwischen Ausprägungen mittels Lebenslinien dargestellt wird.

2.4.1 Lifeline

Eine Lifeline ist in UML eine Art instance eines Objektes, allerdings enthält es keinen Status der Instance. Also besser gesagt ist es ein Platzhalter für ein konkretes Objekt.

2.4.2 Konkreter Nachrichten Fluss

Informationen in diesem Diagramtyp:

- Abfolge der Nachrichten
- Richtung der Nachrichten
- Typen der Nachricht (synchron, asynchron)

Eine andere Darstellungsform für dieses Diagramm ist das Sequenz Diagramm. Es enthält die selben Informationen, stellt diese aber anders dar.

2.4.3 Abstrakter Nachrichten Fluss

Informationen in diesem Diagramtyp:

- Nicht technische Darstellung
- Richtung der Nachrichten
- Typen der kommunizierten Nachrichten

Beispiel Crawler für eine Suchmaschiene

Beispiel aus den Folien

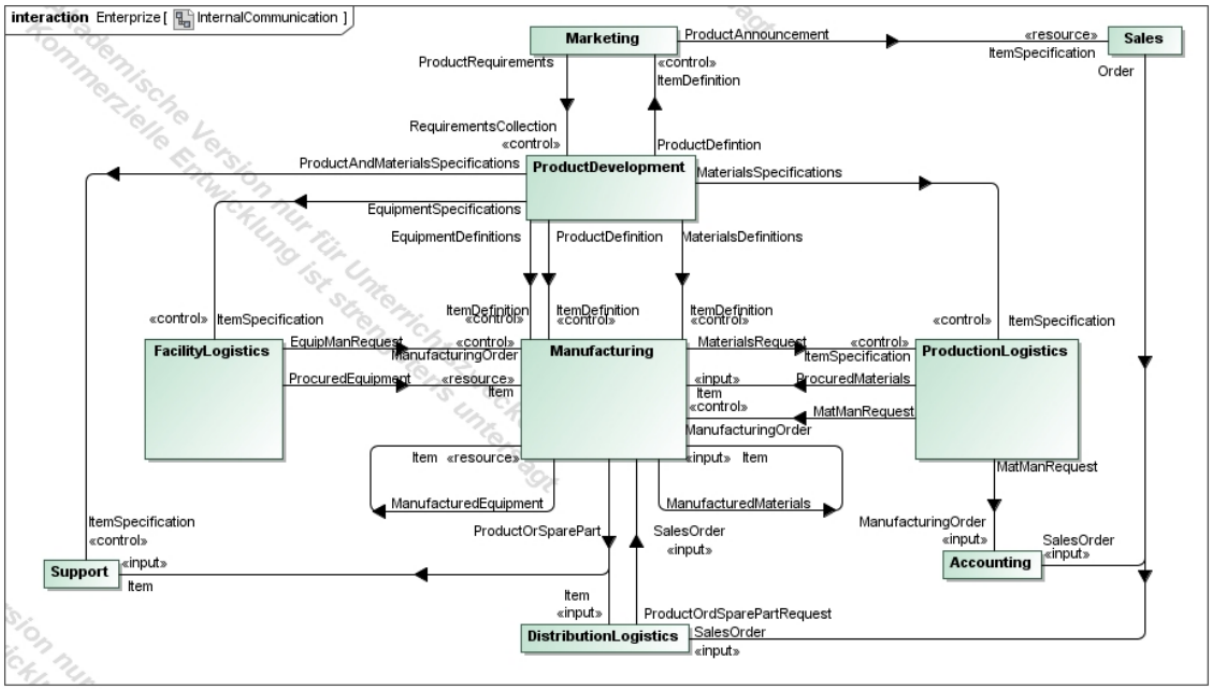

Abbildung 2.1: Abstrakter Informations Fluss

Notizen:

- Marketing schickt anforderungen an ein Produkt an Product Development (PD)
- PD entwickelt das Produkt => Ergebniss Product Definition
  - Materials = Material um das Auto zu bauen (Stückliste)

  - Equipment = Zubehör (Ablauf, Werkzeuge, Prozesse ...) um das Auto zu bauen
- Product Definition geht an Manfacturing
- PD -> MaterialSpecification wird an Products Logistics geleitet wen das Teil eingekauft wird
- Manufacturing bestellt über MaterialsRequest
- Manufactoring verbaut den Motor (input item)
- Fals gewisse Teile doch selber (oder andere Firma) Produziert werden ManufactoringOrder
- FacillityLogistics bekommt ManufactoringOrder und liefert eine Resource item (unterstützt den Prozess)
- PD liefert ProductDefinition an Marketing für neues Produkt
- Marketing liefert ProductSpecification an Sales um Order zu erstellen
- Sales liefert Order an DistributionLogistics (DL)
- DL liefert (input) an Manufactoring
- Stereotypen:
  - Input wird verbracht (nach der verarbeitung nicht mehr da)
  - Output
  - Resource ... Hilfsmittel um die Aufgabe zu erledigen
  - Control ... Steuert den Ablauf im Block

Beispiel: Kompilierung

- Input ist der Source code (weg für den Kompiler)
- Resource ist der Kompiler, BS
- Control ist z.B. Compiler direktiven

2.4.4 UML Aktivitäts Diagramm

Beschreibt das Verhalten (Fluss), nicht die statischen Strukturen.

Anwendungen

- Geschäftsprozesse
- Systemprozesse
- ...

Petri Netze

Mathematische Modelierung von Verteilten Systemen. Entwickelt 1962 con Carl Adam Petri. Grundlagen für Petri Netze sind gerichteten (bipartiten) Graphen:

- Nodes (Knoten):
  - Places: tragen Tokens in sich
  - Transition: Forked oder Joined Kanten
    - Input Place: Verbunden über Kante am Eingang
    - Output Place: Verbunden über Kante am Ausgang
  - Die Transition feuert, wenn jeder Input Place mindestens soviel Tokens wie das Gewicht der Kante besitzt.
  - Die Output Places erhalten nach feuern der Transition soviele Tokens, wie das Gewicht der verbindenden Kante

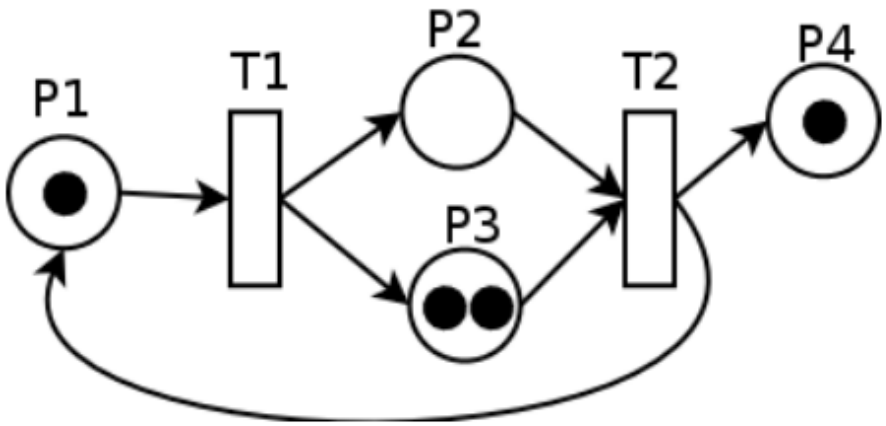

Abbildung 2.2: Activity Diagram

Activity Entspricht einem Verhalten im System und besteht aus mehereren Actions.

Abbildung 2.3: Activity

Action Eine Aktion ist ein einzelner Schritt in einer Activity. Muss aber nicht Atomar sein (kann also aus weiteren Activities und Actions bestehen)

Abbildung 2.4: Action

Start und Endknoten Start enthält genug Token für die Eingaben, Feuert am start der Aktivität. Wen ein Token den Ende Knoten erreicht ist die Activity zu ende.

Kanten Die Kanten des Diagramm besitzt einen Namen und weitere Inforamtionen wie den Fluss (Richtung, Bedinungen, weiteres Verhalten). Zusätzlich spezifiert wird die Kante über Stereotypes.

Weitere Informationen zu Kanten:

- Gewicht {Geschwungenen Klammern}
- Guard (Filter oder Evaluationen) [Eckige Klammern]
- Zuerst Guard dann Gewicht überprüfen

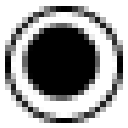

Abbildung 2.5: ActivityFinalNode

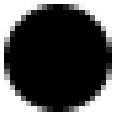

Abbildung 2.6: InitialNode

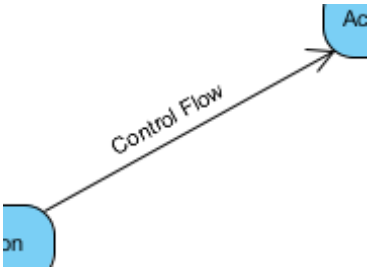

Abbildung 2.7: Knoten

Ein und Ausgänge (In/Output Pins) Entspricht Parameter in der Softwareentwicklung. Damit kann bestimmt werden welche Eingaben welche Rollen spielen. Damit können verschiedenen Eingabeparameter verschiedene Verhalten zugewiesen werden.

Objekt Knoten Repräsentiert Objekte die in Aktionen verwendet werden.

Parameter Knoten Erweiterte Version von Ein und Ausgängen

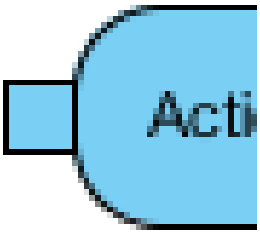

Abbildung 2.8: InputPin

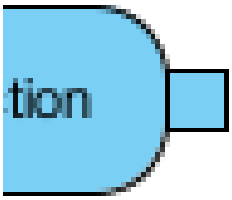

Abbildung 2.9: OutputPin

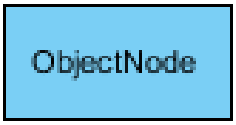

Abbildung 2.10: ObjectNode

Entscheidungsknoten Nur ein Ausgang wird gefeuert. Bedinung für jeden Ausgang an der jeweiligen Kante.

Merge Es reicht wenn ein Eingang gefeuert wurde, damit der Ausgang gefeuert wird.

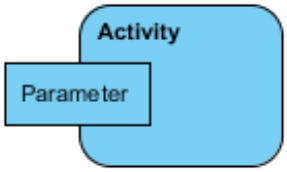

Abbildung 2.11: ParameterNode

Abbildung 2.12: DecisionNode

Abbildung 2.13: MergeNode

Fork / Join Erzeugt / Synchronisiert Parallele Flüsse.

Abbildung 2.14: Fork / Join Node

Advanced Activity Modelling

Einführung von Activity Partitions, also mehrere Swimlanes. Swimlanes können später noch weiter unterteilt werden (FHV kann im Laufe des Prozesses z.B. noch in einzelne Räume unterteilt werden).

Der external Stereotype beschreibt einen Bereich, der nicht von uns konstruiert wird, sondern von außen vorgegeben ist.

Wenn auf eine große Menge von Daten die gleiche Aktion angewendet werden soll, dann können sogenannte Expansion Regions verwendet werden. Dazu gibt es optionale Stereotypen für die Verarbeitung in einem iterativen, parallelem oder einem Streaming-Prozess. Die Verwendung dieser Stereotypen ist zum Teil allerdings an gewisse Hardware gebunden.

Darüberhinaus gibt es auch noch für die Darstellung von Loops die Standard Loops und Loop Nodes. Bei den Loop Nodes werden drei Bereiche angegeben: Setup, Test und einen ausführenden Body.

Streaming Actions können Output generieren, während dem man einen Input erhaltet.

Central Buffer stellen Puffer als Knoten dar, was z.B. bei einer Produktionskette ganz nützlich sein kann. Eine Erweiterung dazu stellt der Data Store Node dar, auf welchen auch noch Daten gespeichert werden. Diese Daten können auch später noch einmal abgerufen werden.

Statecharts

In vielen Fällen hängt das Verhalten eines Objekts von seinem Zustand ab, und die Zusammenhänge zwischen diesen Zuständen werden in einem Statechart mit Zustandsübergangen gezeigt. Wenn ein Übergang zwischen zwei States nicht definiert ist, könnte man das als nicht erlaubter Übergang interpretieren.

Status Ein Status haltet immer eine invariante Bedingung, die sich nicht ändert, solange man den Status nicht wechselt. Diese Bedingung ist im Normalfall implizit definiert.

Es gibt verschiedene Verhalten bei einem Status. Am Anfang gibt es ein Eintrittsverhalten, welches ausgeführt wird, sobald man den Zustand betritt. Während dem Verharren in einem Zustand gibt es ein Verhalten, welches gültig ist, solange der Status aktiv ist. Zuletzt gibt es noch ein Austrittsverhalten, welches beim Verlassen des Status ausgeführt wird.

Status können auch verschachtelt, und in dieser Verschachtelung auch noch in Swimlanes angeordnet werden. Man kann auch Submaschinenstatus verwenden, um die mehrfache Verwendung des gleichen Status in verschiedenen Diagrammen zu verwenden.

Zustandsübergang An einem Zustandsübergang hängt ein Trigger, welcher beschreibt bei welchem Event der Übergang eintritt. Dazu gibt es auch noch einen Guard, also ein boolscher Ausdruck, welcher wahr sein muss um den Übergang zu erlauben. Zuletzt gibt es einen Effekt, der beim Zustandsübergang ausgeführt wird.

Pseudostatus Es gibt auch sogenannte Pseudostatus, allen voran der Startzustand. Allerdings ist der Endzustand kein Pseudozustand, da an diesem wirklich etwas ankommt.

Darüber hinaus gibt es auch noch einige andere Pseudostatus, wie z.B. der Entscheidungsknoten, das Forken und Joinen, das Zusammenführen usw.

Signal Dieser Teil ist vor allem für Embedded Software wichtig, damit kann man das Empfangen und Versenden von Signalen modellieren.

3 Produktmanagment

Jedes Produkt hat verschiedene Lebenszyklen:

- product business lifecycle
- product engineering lifecycle
- manufacturing engineereing lifecycle
- product support lifecycle
- operator lifecycle

Je nachdem in welchem Zyklus sich ein Produkt befindet hat es mehr oder weniger Auswirkungen auf Entscheidungen von Produktmanagern –> z.B. wegen Rentabilität | Break Even Point.

Wichtig im Bereich des Produktlebenszyklusmanagement sind die Bereiche Herstellungs- und Produktionsprozesse. Hier kann unterschieden werden zwischen:

- Kontinuierlicher Produktion
- Stapelproduktion
- Diskrete Herstellung
    1. Einzigartiges Produkt
    2. Produkt in geringem Umfang mit und ohne Varianten

Der Produktmanager ist das Bindeglied zwischen Produktion, Marketing, Vertrieb usw. und muss vermitteln. Außerdem sollte er deb vielen verschiedenen Breichen (Technik, Wirtschaftlich, Marketing) ein wenig Ahnung und auch Verständniss dafür haben.

Neben dem Wissen über die ganzen Teilgebieten (Produktion, Marketing, Verkauf) sollte ein guter Produktmanager auch wissen wieso Kunden ein Produkt kaufen oder auch nicht. (Beispiel v. Prof.)

	Product Definition Lifecycle	Process Definition Lifecycle	Physical Product Lifecycle	Prod. Equipment Lifecycle
Continuous Process Production				
Batch Production				
Discrete Manufacturing				
Unique Product				
Low Volume Product				
no Variants				
with Variants				
Mass Product				
no Variants				
with Variants				

Abbildung 3.1: Klassifizierung von Industrien (Rot ist das Wichtigste)

Abbildung 3.2: Produktmanagerverantwortungen

3.1 Anforderungsmanagementprozess

Hier wird festgestellt ob Anforderungen gültig sowie konsisten untereinander (keine Wiedersprüche) sind. Außerdem wird überprüft ob es keine Duplikate gibt und die gesamte Sammlung der Anforderungen valide ist (nichts vergessen und nichts was entfernt werden sollte).

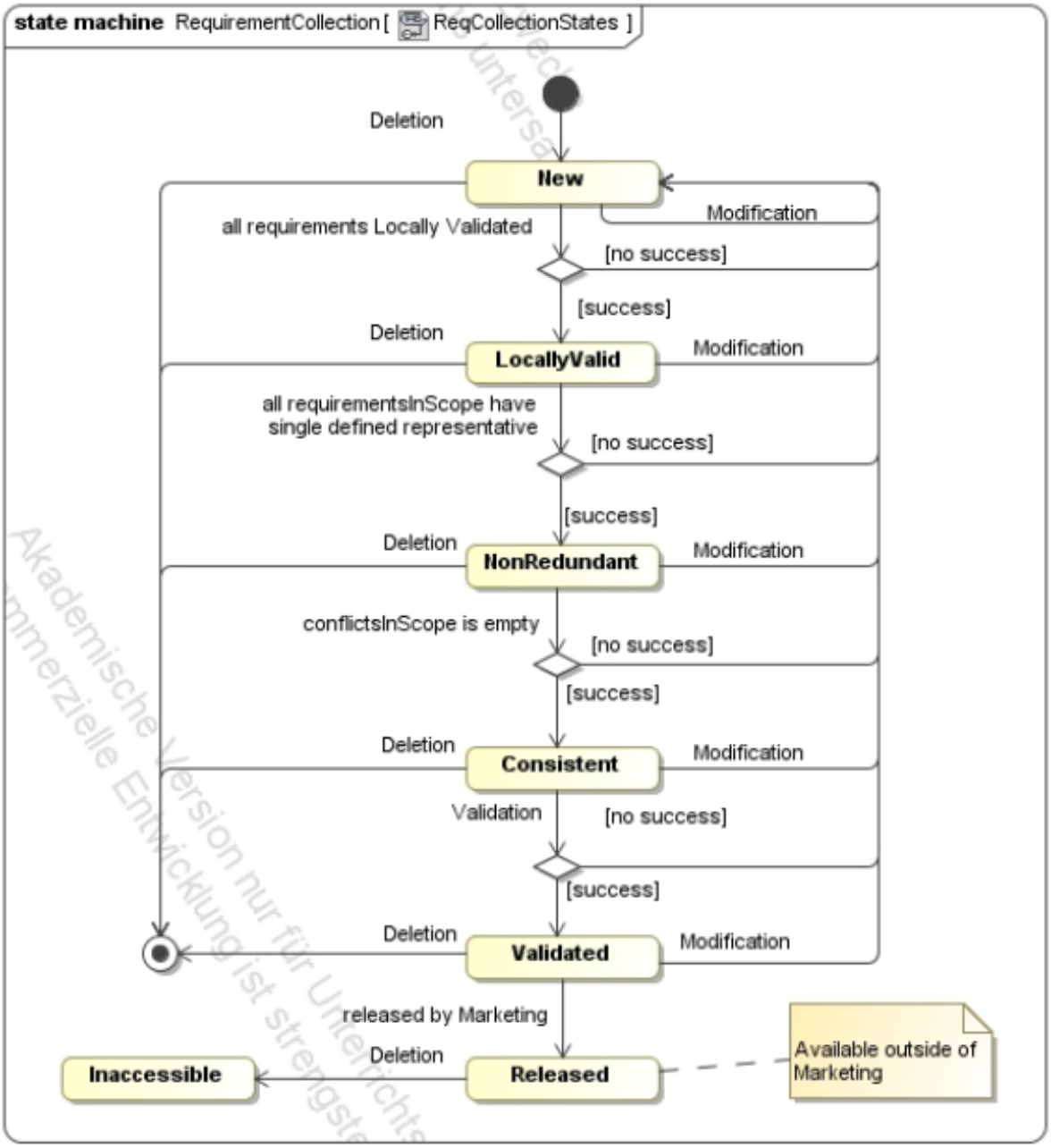

Abbildung 3.3: Anforderungsmanagement-Prozess

3.2 Item und Item-Context

Item Spezifikation Resultat einer technischen Entwicklung, unabhängig von dessen Gebrauch als Teil eines größeren Produktes und Zubehör für die Produktion.

Item Kontext Wichtige Informationen um die Spezifikation richtig zu interpretieren.

- Detailierte Anforderungen
- Verbindungen zu anderen Teilen
- Klassifizierung
- Zweck und Status des Items
- Phase im Lebenszyklus

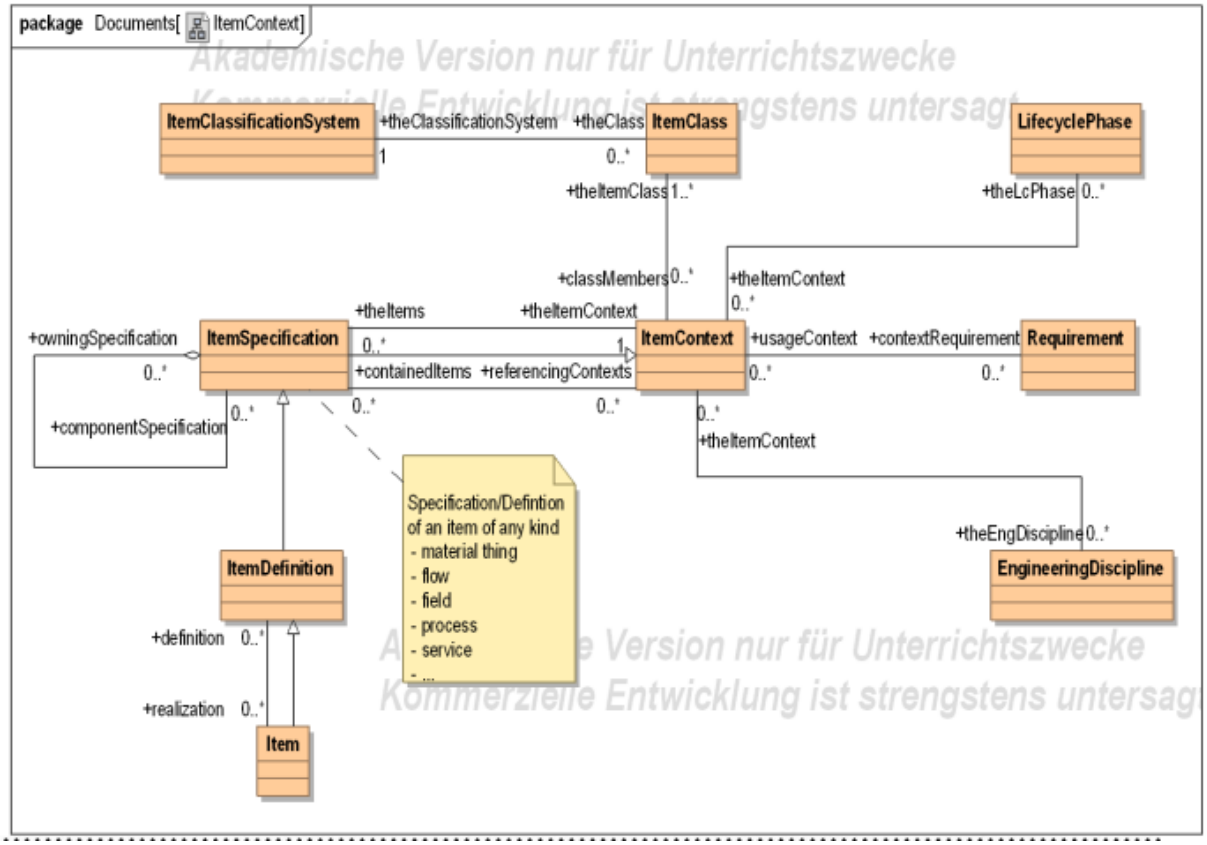

Abbildung 3.4: Item and Item Context

3.3 Item Entwicklungs Prozess

Der Prozess wird für jede Stufe durchgeführt, daher wird der Prozess rekursive ausgeführt.

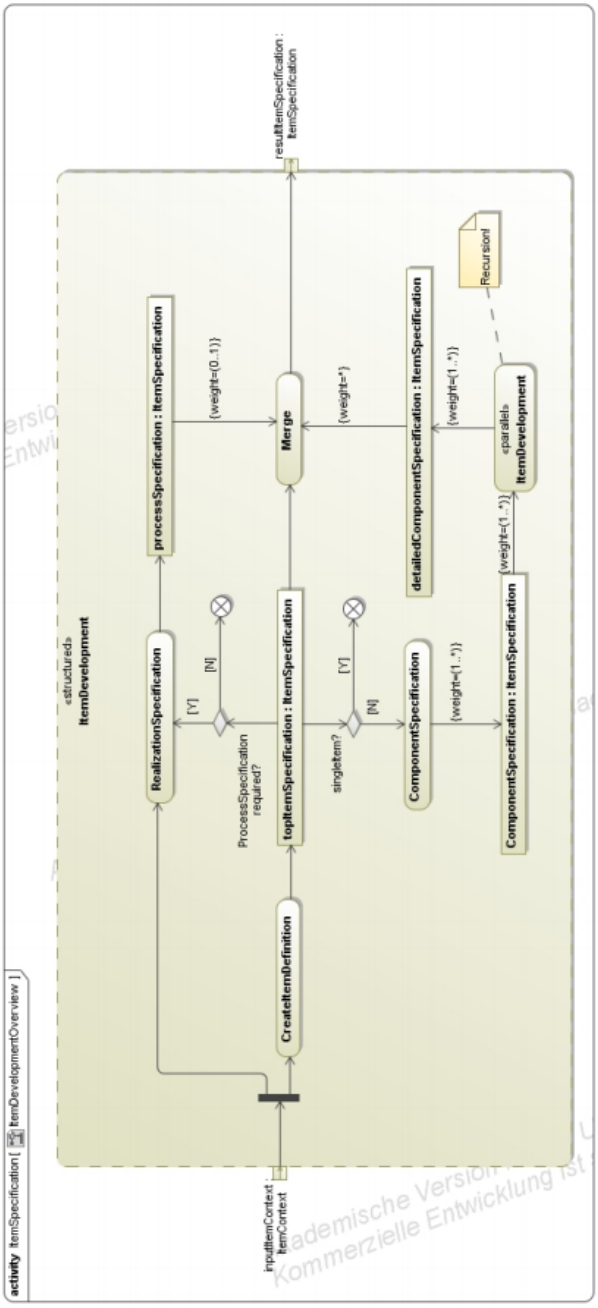

Abbildung 3.5: Item Development Process

3.4 Item Struktur

Entspricht einer einfachen Stückliste für das Item. Es ist ausreichend für die Fertigungslogistik aber nicht für die Unterstützung mit einem Informationsmanagement System.

- Idee eines Artikels
- UseCases sammeln
- Liste von Funktionen
- Funktionen auf abstrakte Komponenten zuweisen
- Suche nach bestehenden Komponenten
- Fals keines gefunden werden kann starte einen neuen Design Prozess
- Die UseCases auf dieser Ebene auf die abstrakte Komponente zugeordnet

Ideale Struktur Die M-N Beziehungen kommen daher, dass eine Funktion von meheren Komponenten und dass mehrere Komponenten eine Funktion erfüllen können.

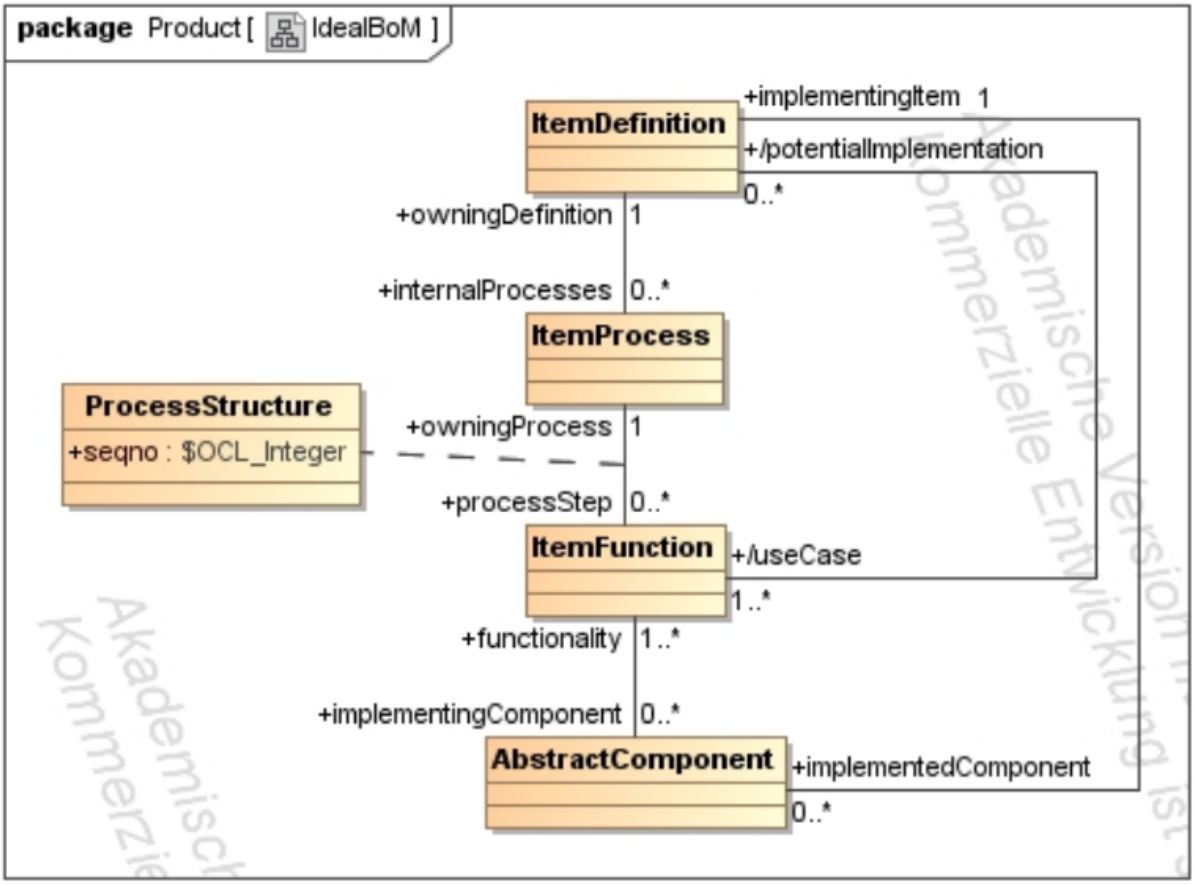

Abbildung 3.6: Ideale Item Struktur

Reale Struktur Über die Beziehungsklasse (ItemStructureRelationship) kann ein Typ definiert werden. Zusätzlich kann über das Flag physicalContainment definiert werden, ob die Beziehung eine Komposition oder Aggregation ist.

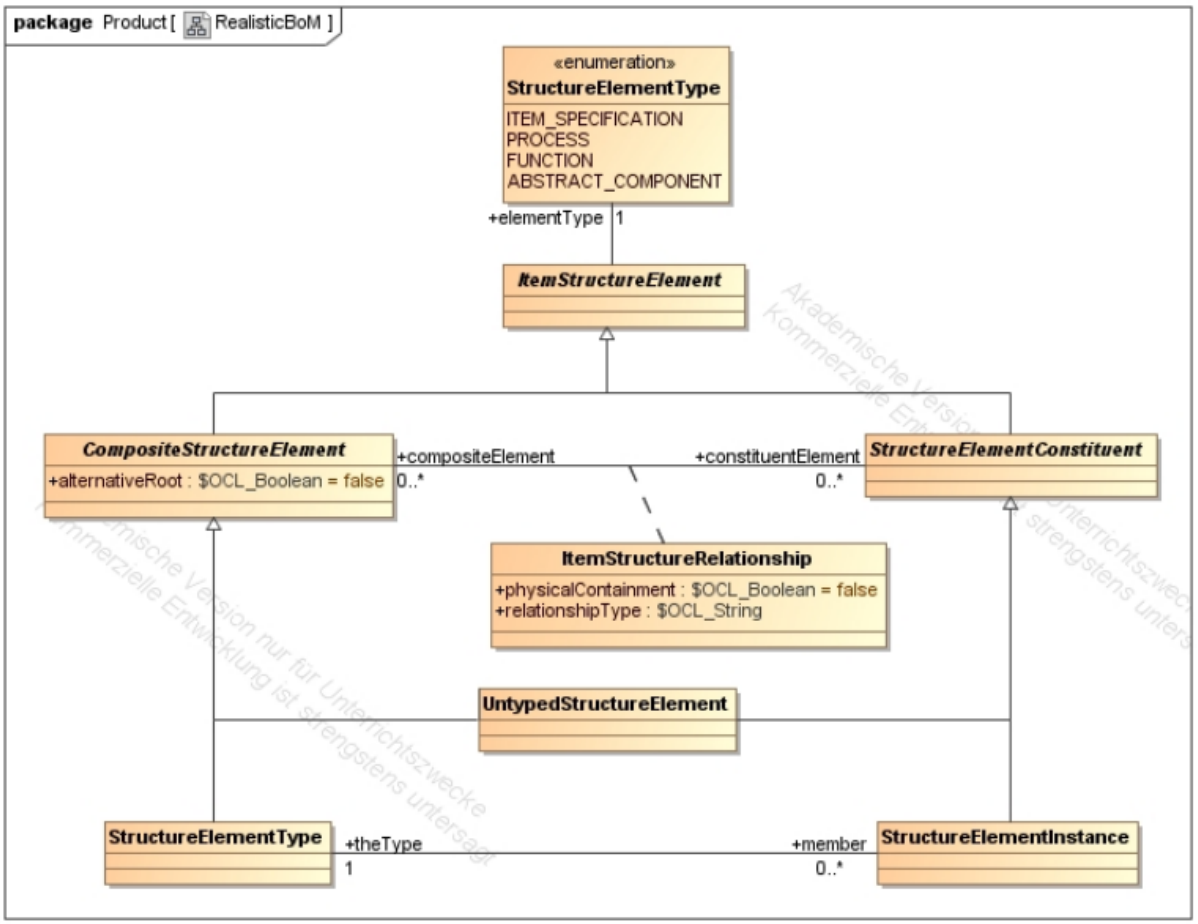

Abbildung 3.7: Reale Item Struktur

In einem gewissen Detailierungsgrad macht die Unstruktured Definitionen keinen Sinn mehr. Beispiel Heizkörper, jeder Heizkörper bekommt nun ein Instance Objekt. Dort findet sich wieder eine Stückliste mit dem Bestandteilen dieses einzelnen Items.

3.5 Item Änderungen

Im Allgemeinen eine Sourcecode Verwaltung für Item Spezifikationen. Um zu einem späteren Zeitpunkt zu einem definierten "gutenSZustand zurückzukehren.

Beinhaltet Informationen wie:

- Anforderungen
- Kontext
- Spezifikationen
- Definitionen
- Änderungsverlauf

Struktur Der Item Kontext erhält einen Vorgänger und mehrere Nachfolger.

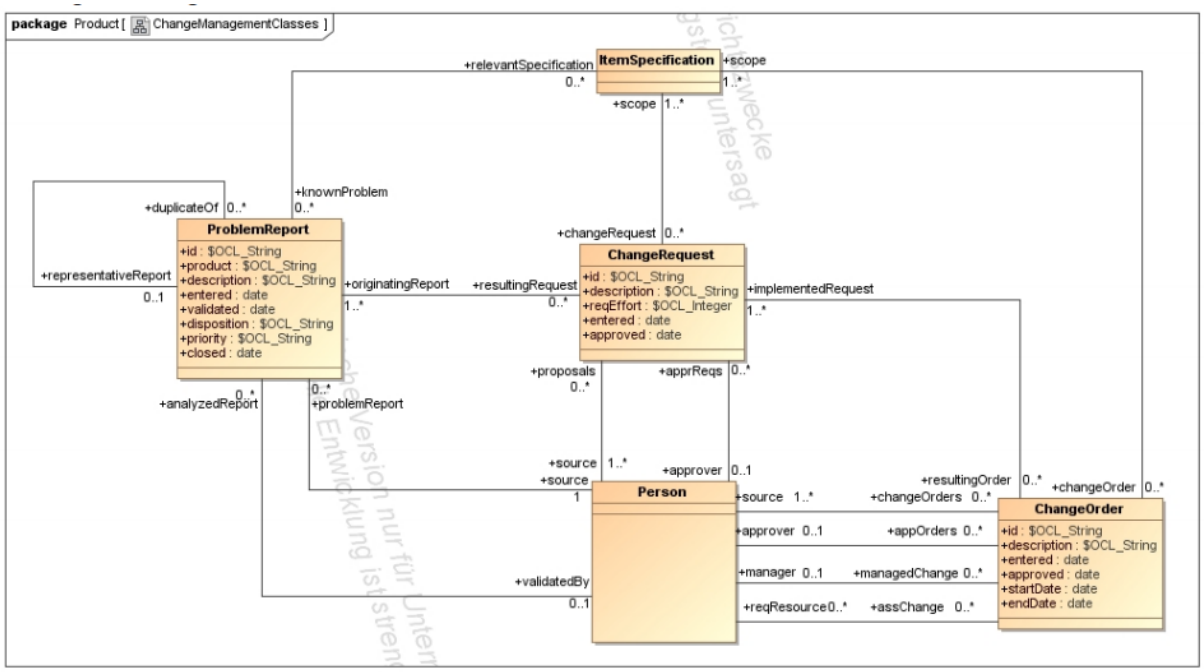

Abbildung 3.8: Change Management Classes

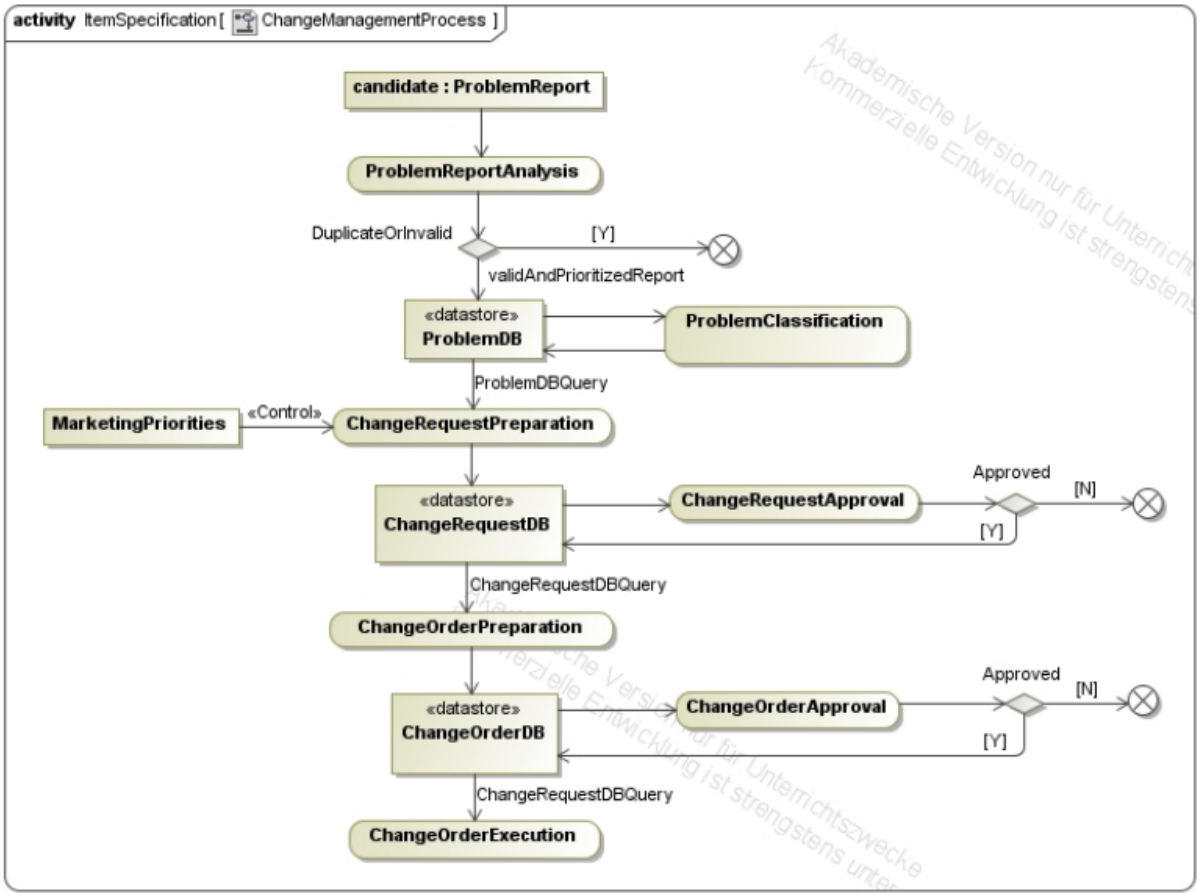

Abbildung 3.9: Change Management Process

Ablauf

- Problem Report analysieren: Gültigkeit, Duplikate, ...
- Speichern in einer Datenbank
- Kategorisieren (Verbesserungswünsche, Kritischer Fehler, ...)
- Aufgrund von Marktpriorisierung (oder normale Priorisierung) ChangeRequests erstellen
- Genemigung, Planung, Aufwandsschätzung
- Arbeitsaufträge erstellen
- Ausführung

3.6 Item Naming

Traditionelle Sicht laut ISO 10303-214:

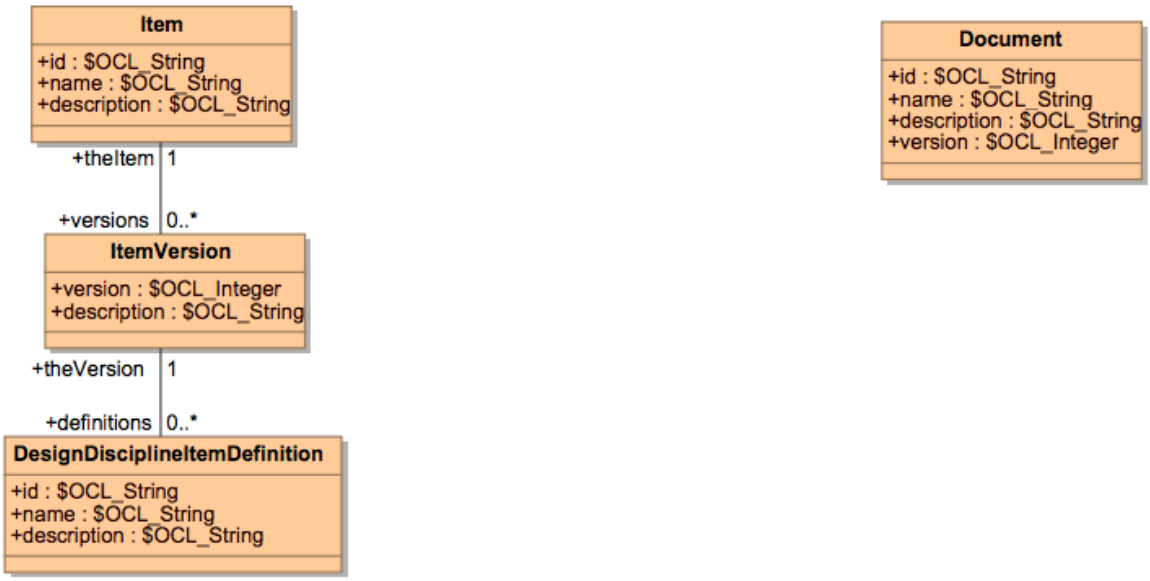

Abbildung 3.10: Traditioneles Item Naming

Zusätzlich kann jedes Teil mehrere Namen besitzen. Beispiel Personen:

- Namen
- Studentennummer
- Spitznamen
- ...

Namen können verschiedene Typen und Kontexte haben. Kontext sind zum Beispiel Familien, Firmen, ... Gutes Beispiel für Kontext sind zum Beispiel Domains ilias.fhv.at -> ilias ist nur in fhv und fhv nur in at definiert.

Beispiel für Globale Identifikation: Identifikation von Bernd Wenzel

1. Norm (UN - 0, ISO - 1, IEC - 2)
2. Land (AT - 43, DE - 49, ...)
3. Ort (Dornbirn - 6850, ...)

4. Gebäude (FHV S-Trakt - 792, ...)
5. Raum (Bernd Wenzel - 3007, ...)

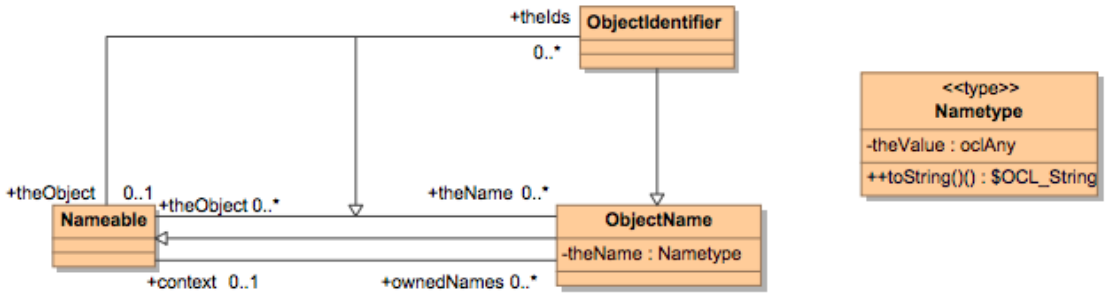

Abbildung 3.11: Endergebnis von Item Naming

3.7 Item Release Versionierung

Überlegungen

- Verschiedene Versionen einer Sache beziehen sich auf verschiedene Designs
- Zwischen zwei verschiedenen Designs können folgende Beziehungen bestehen:
  - Ein Design wurde von der anderen abgeleitet
  - Beide Entwürfe wurden von der gleichen Person erstellt
  - Beide Entwürfe wurden im Rahmen der gleichen Organisationen geschaffen
  - Ein Design wird in Zukunft Design-Arbeit die andere ersetzen
  - One-Design wird in der Produktion die andere ersetzen

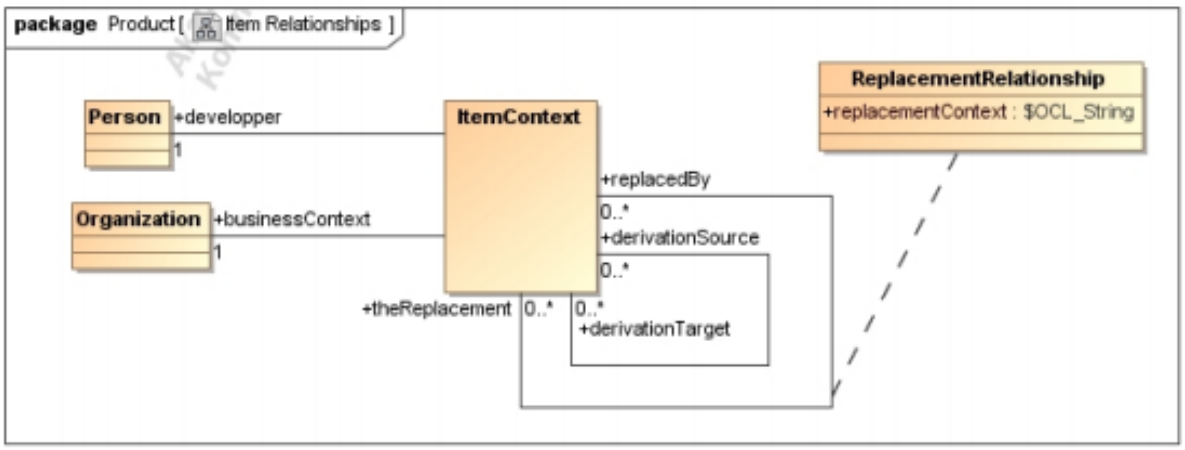

Abbildung 3.12: Versionierung

Version Veröffentlichtes Design, egal ob limitierte Veröffentlichung (limitierte erhalter) oder ganz öffentlich.

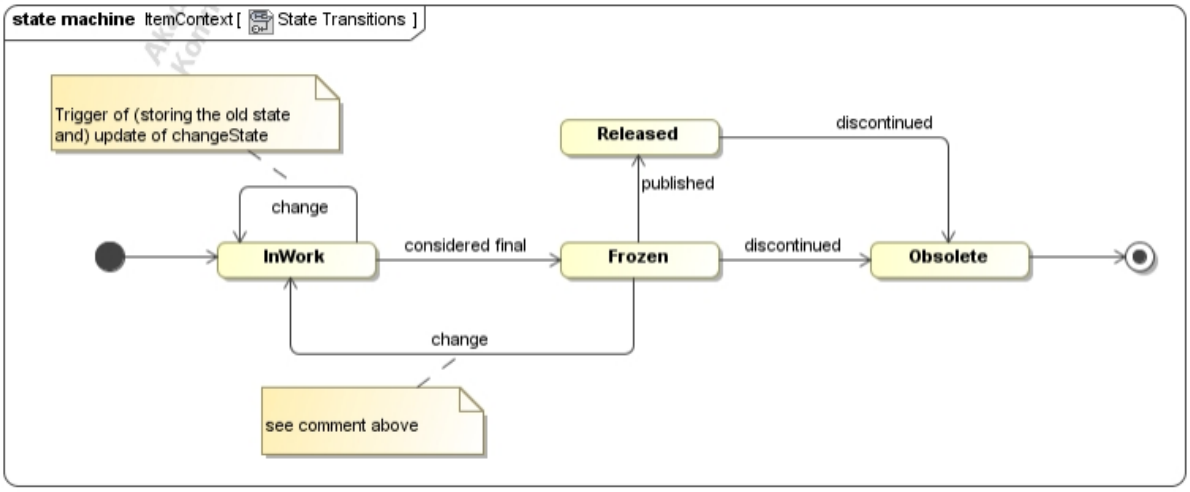

Abbildung 3.13: Versionierungs Statemachine für eien Item Kontext

Versionsnummern Versionsnummern sind Identifier innerhalb des Item Kontext. Sie müssen Global einheitlich sein für Item-Identifier und Versionsnummer.

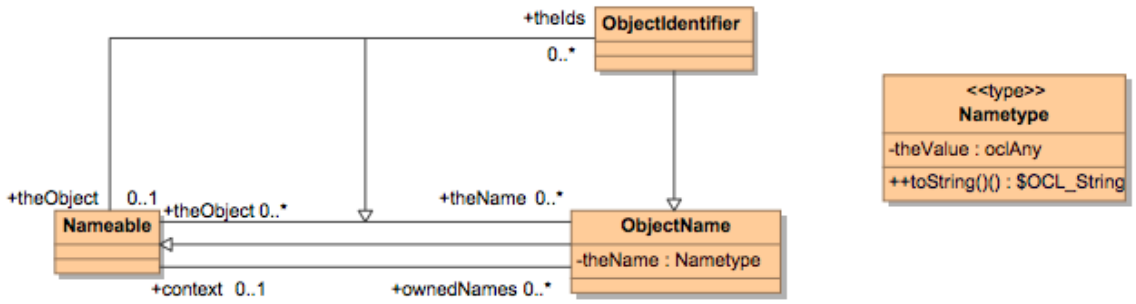

Abbildung 3.14: Versionsnummern sind Identifier

Beispiel: Magic Draw 17.0.1

- Item Identifier: Magic Draw existiert global im Unternehmen
- Version 17 Identifier: 17 existiert in Magic Draw
- Version 0 Identifier: 0 exisiert in Magic Draw 17
- ...

3.8 Übung Teilenummer

How can we represent the following information in an object model according to our naming und identification model.

The following identifiers refer to the same part

- Bosch: 4711-17
- BMW: 0814-12
- Mercedes: 9876-14

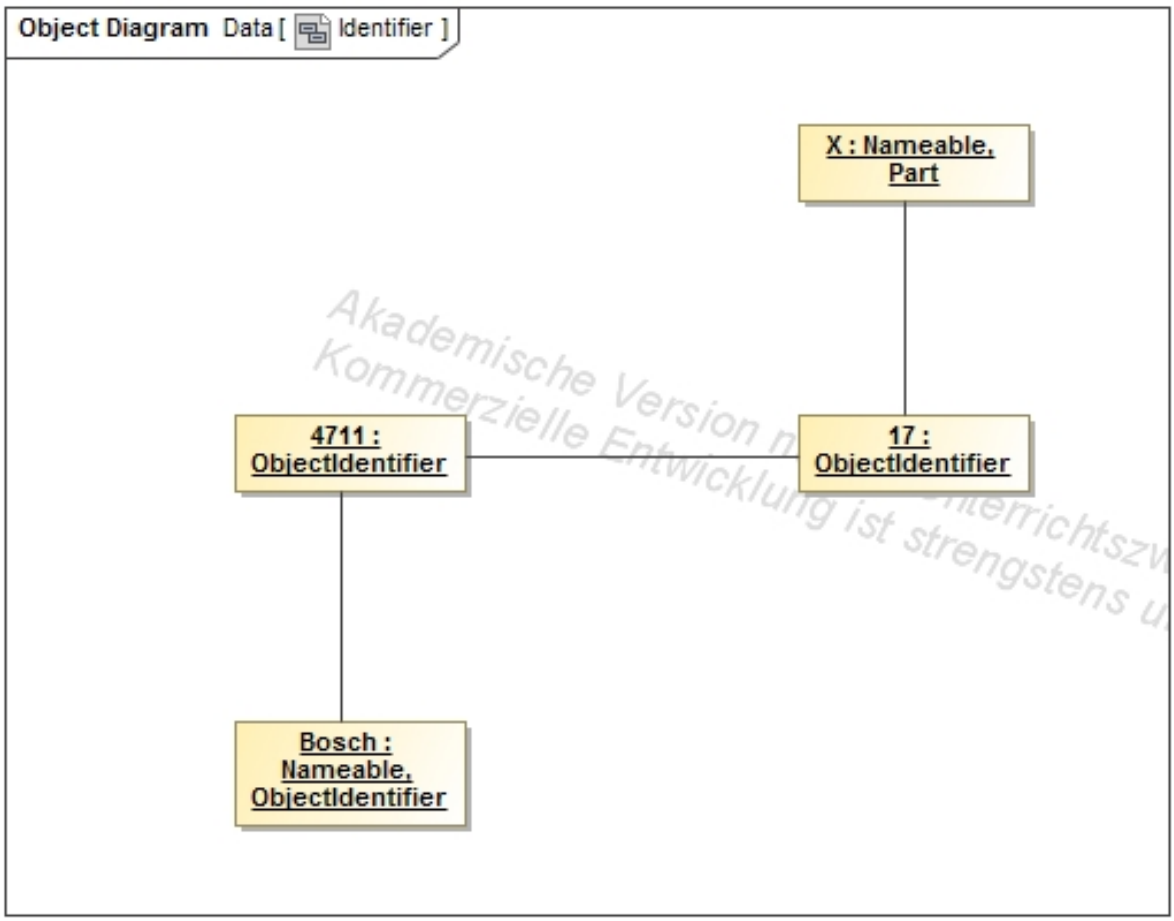

Abbildung 3.15: Lösung zur Aufgabe

4 Produktentwicklung

4.1 Items als Produkte oder Teile (Parts)

Die Begriffe "Produkt", "Partünd Ärtikelßind mehrdeutig verwendet

Bedeutung 1 Eine ItemSpezifikation eines Items werden die Erträge als eine Komponente des ganzen gesehen.

Bedeutung 2 Verwendung als Instanz wie zum Beispiel ItemSpezifikation.

Hinweis Sehr oft beziehen sich "Produktünd Ärtikeläuf den gesamten, während "Teil"bezieht sich auf einen Artikel auf beliebiger Komponentenebene.

Überlegungen zu Abbildung 4.1

- Im Frozen Zustand:
  - Die Ergebnisse werden Kollegen zur Verfügung gestellt.
  - So Stabil, dass der Prototyp kann erzeugt werden kann.
  - So Stabil, dass eine Testphase gestartet werden kann.
- External Visibility: Kann das Design begutachtet werden? Ist das Design richtig um gesetzt?
- Safety relevant: Ist das Teil Sicherheitsrelevant und Sicherheitstechnisch OK?

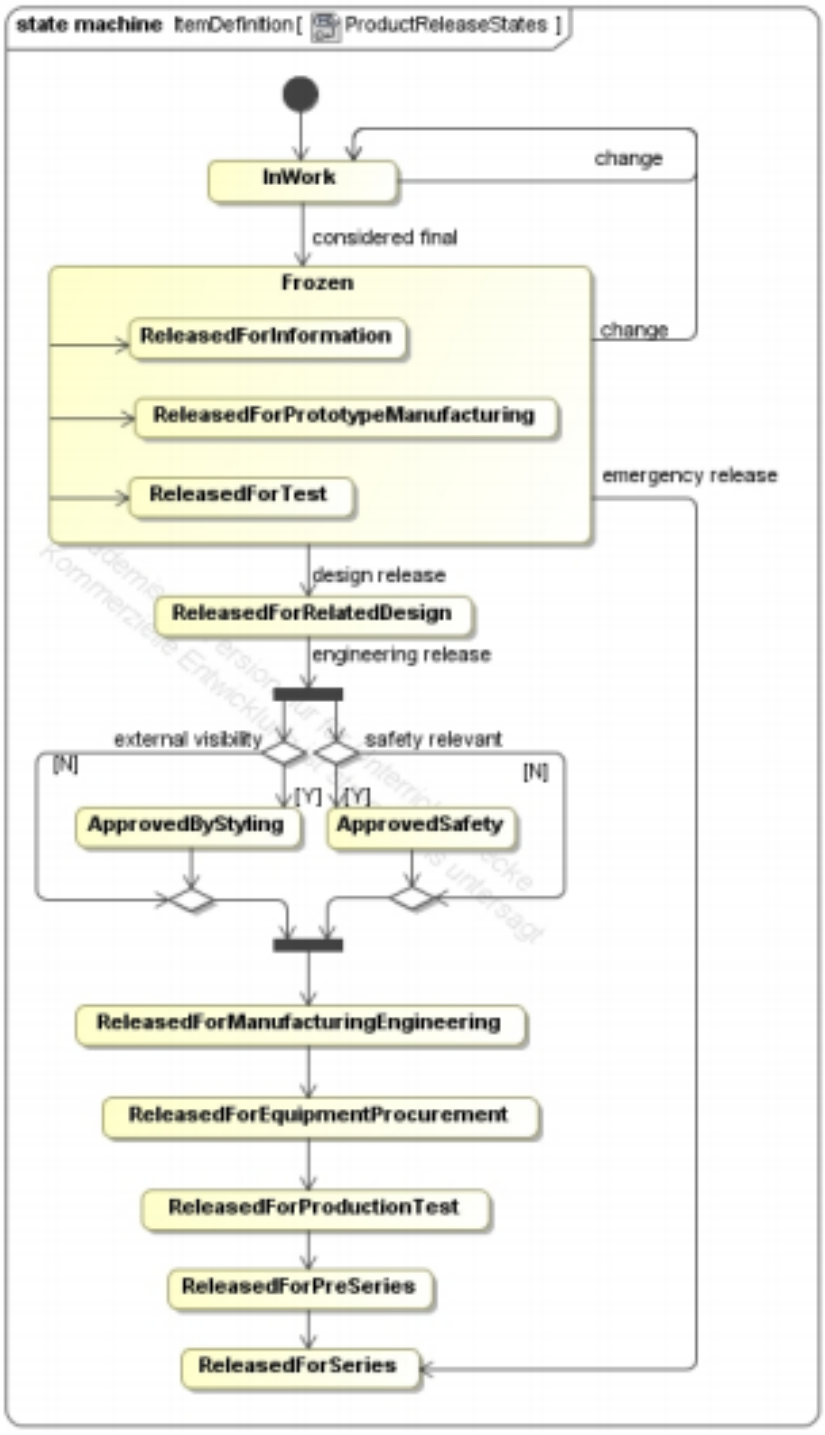

Abbildung 4.1: Beispiel eines Release Prozesses (Automobil Industrie)

4.2 Item als Zubehör (facilities), Ausrüstung (equipment) und Werkzeuge (tools)

Die Begriffe "Facility", Äusrüstungünd "Toolßind mehrdeutig verwendet.

Bedeutung 1 Als Teil des Ganzen.

Bedeutung 2 Als eigenständige Instanz.

Hinweis

- Sehr oft bezieht sich "Facilityäuf der obersten Ebene in einer Produktionsressourcenstruktur.
- Sehr oft bezieht sich Ëquipmentäuf unabhängig voneinander arbeit Ebenen in einer Produktionsressource Struktur
- Sehr oft bezieht sich "Tooläuf Kunden austauschbare Funktionseinheiten in einer Produktionsressourcenstruktur

4.3 Varianten

4.3.1 Automobil Industry

In der Anfangszeit waren Autos Luxusgüter für den Adel. Gekauft wurde damals nur der Motor und der Antriebsstrang, die Karroserie wurde von sogenannten Wagenbauer verkauft.

Abbildung 4.2: Früherer Rolls Royce

Henry Ford führte die Massenproduktion ein. Das war das Ende der individuellen Wagen.

Zitat: "You may have it any color you like, provided the color is black"

Sonderwünsche Die Kunden hatten aber weiterhin spezielle Wünsche. Dazu boten die Hersteller Spezialprodukte an. Dadurch hatten die Händler einige Probleme mit der Identität und der Qualität. Die Reaktion der Hersteller war dann, dass sie sogenannte PlusMinus Stücklisten anboten (ein bzw. ausbauen).

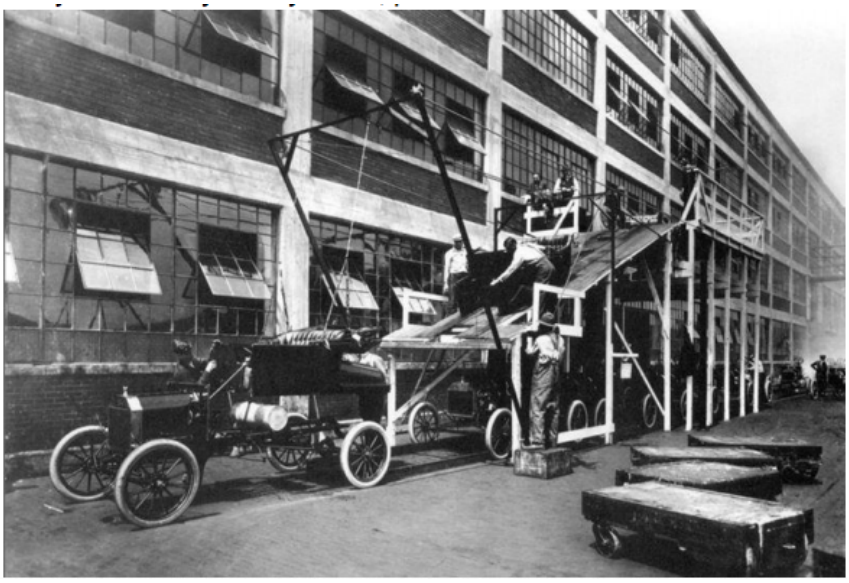

Abbildung 4.3: Ford Werk Verbindung Antriebsstrang und Karosserie

4.3.2 Varianten-Management

Eingeführt ca. 1965. Dadurch bestellt der Kunde keine Artikelnummer mehr sondern ein Model mit einer bestimmten Anzahl von Optionen.

- Typische Anzahl von Optionen: 150
- Nur geschlossene Kategorien
- Bei Luxuriösen Autos / Trucks: 1200 mit offenen Kategorien

Für jede Variante kann nicht ein Identifier erzeugt werden. Weil mindestens 150 Optionen bei jeweils mindestens 2 Möglichketen $2^{150} = 10^{45}$.

Als Vergleich das Alter des Universums in Sekunden (ca): $2 * 10^{10} * 4 * 10^2 * 10^5 > 10^{18}$

Verbreitung Flächendeckend in Deutschland und Schweden. Viele andere Europäische Hersteller limitieren das größtenteils auf Packete. Andere Länder ziehen kräftig nach und der Markt wächst stark.

4.3.3 Stückliste

Traditionelle Stücklisten können Variantenmanagement kaum abbilden. Aber auch eine Erweiterung wie in Abbildung 4.4

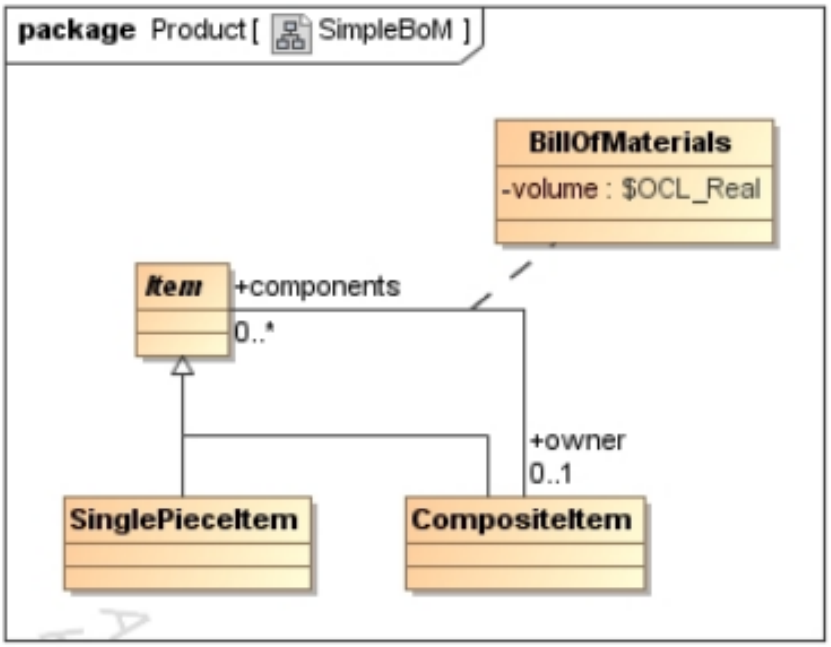

Abbildung 4.4: Klassische Stückliste

Probleme dieser Struktur:

- Dokumentation der Entwicklungsprozess
- Dokumentation von Anforderungen
- Alternative Lösungen
- Alternative Fertigungsprozesse
- Option gesteuert Alternativen
  - für Produkte
  - für Prozesse

Moderne Stücklisten sind Feingranular in Abbildung 4.5. Sie ermöglichen das Anhängen von vielen Dokumenten für die Dokumentation, Alternative Lösungen, Prozesse und die Optionsgesteuerten Alternativen für Produkte bzw. Prozesse.

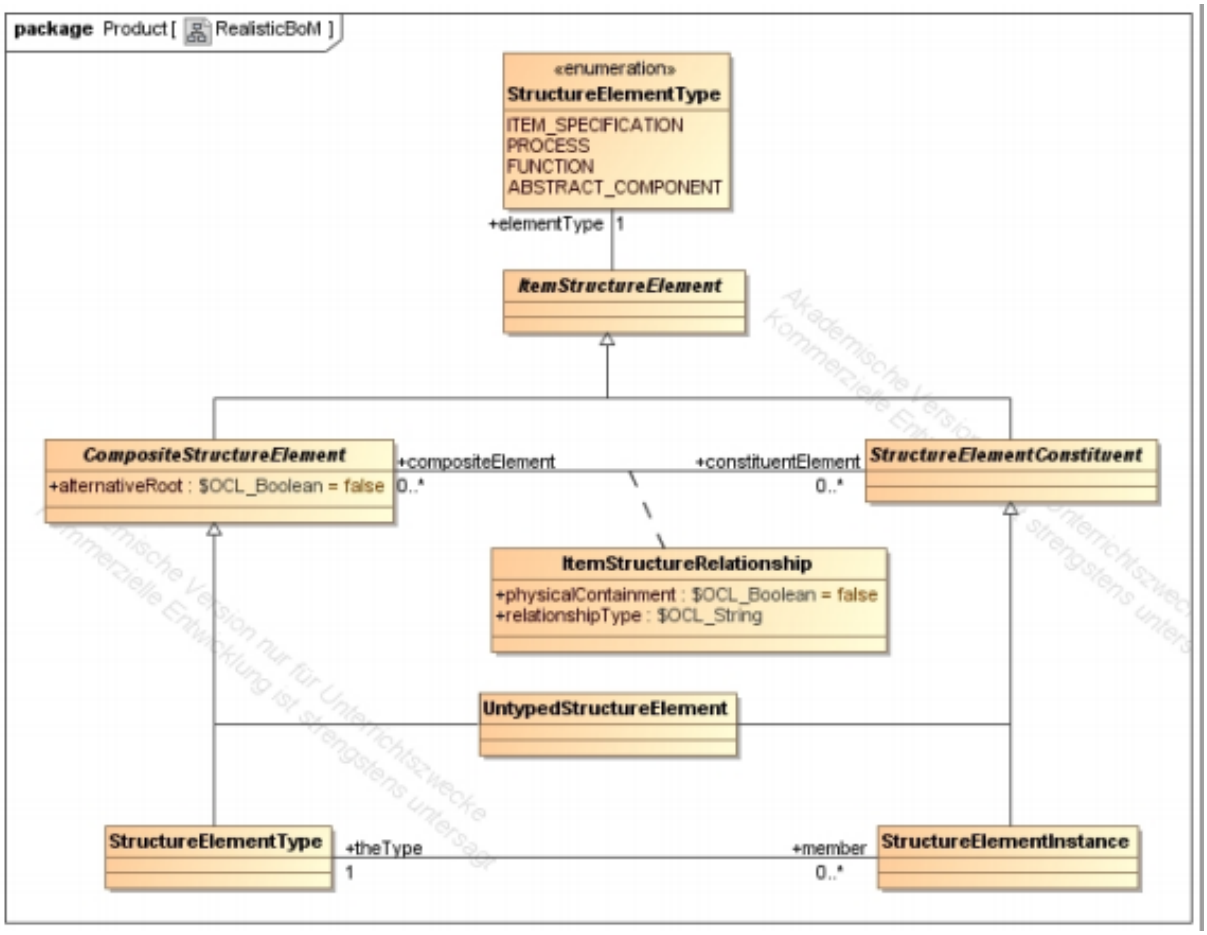

Abbildung 4.5: Moderne Stückliste

Für das Varianten Management verwenden viele Automobil Hersteller heutzutage folgende Stückliste aus Abbildung 4.6. Diese Stückliste bietet die Möglichkeit Spezifikations Expressions, um Einschränkungen abbilden zu können. Diese Expressions sind aufgebaut mit einem Syntaxtree um auch komplexere Einschränkungen abbilden zu können.

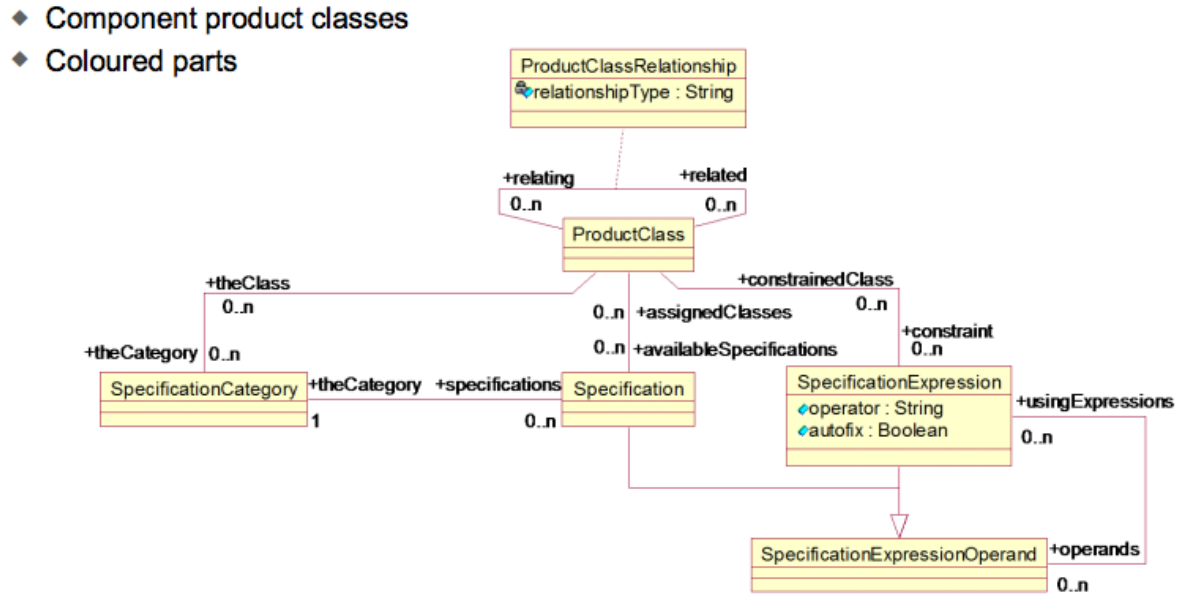

Abbildung 4.6: Automobil Industry Stückliste

Eine Flexiblere und mächtigere Struktur in Abbildung **??**. In dieser Struktur lassen sich Optionsklassen durch andere Optionen freischalten. Zum Beispiel wird die Produktklasse Vergasser nur dann angeboten wen ein alter Motor gewählt wurde oder bestimmte Optionen können nur an bestimmten Produktionsstätten gebaut werden.

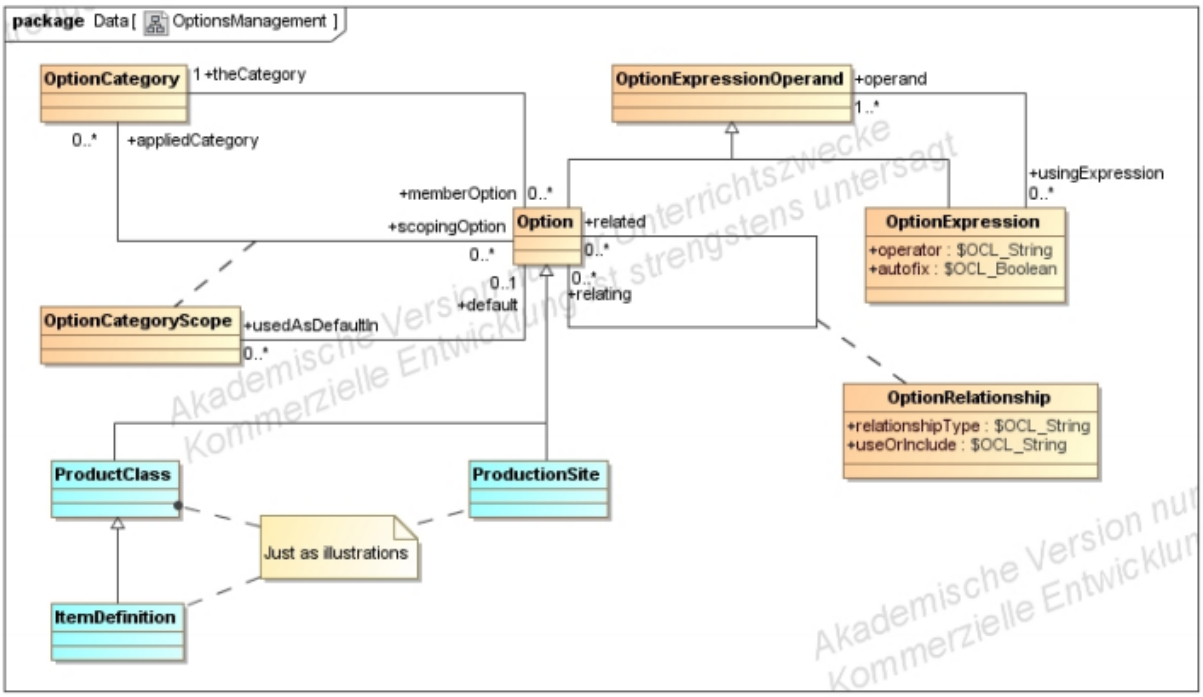

Abbildung 4.7: More powerful and flexible structure

Quelle von Optionen und Regeln

- Marketing
  - Kunden: Farben, Radio, Schiebedach, ...
  - Marktkontext: Kulturele Präferenzen in bezug auf Farbkombinationen, ...
- Produktentwicklung
  - Technische Optionen: Auf Teile-Ebene, ...
  - Technische Regeln: Kein Radio ohne Antenne, ...
- Produktherstellung
  - Herstellungs Optionen: Auf Teile-Ebene, ...
  - Herstellungs Regeln: Auswahl von Produktionsstandorten in Abhängigkeit von anderen Optionen, ...

Optionen stellen einen Kontext für jedes Element bereit. Dies ist in Abbildung 4.8 zu entnehmen.

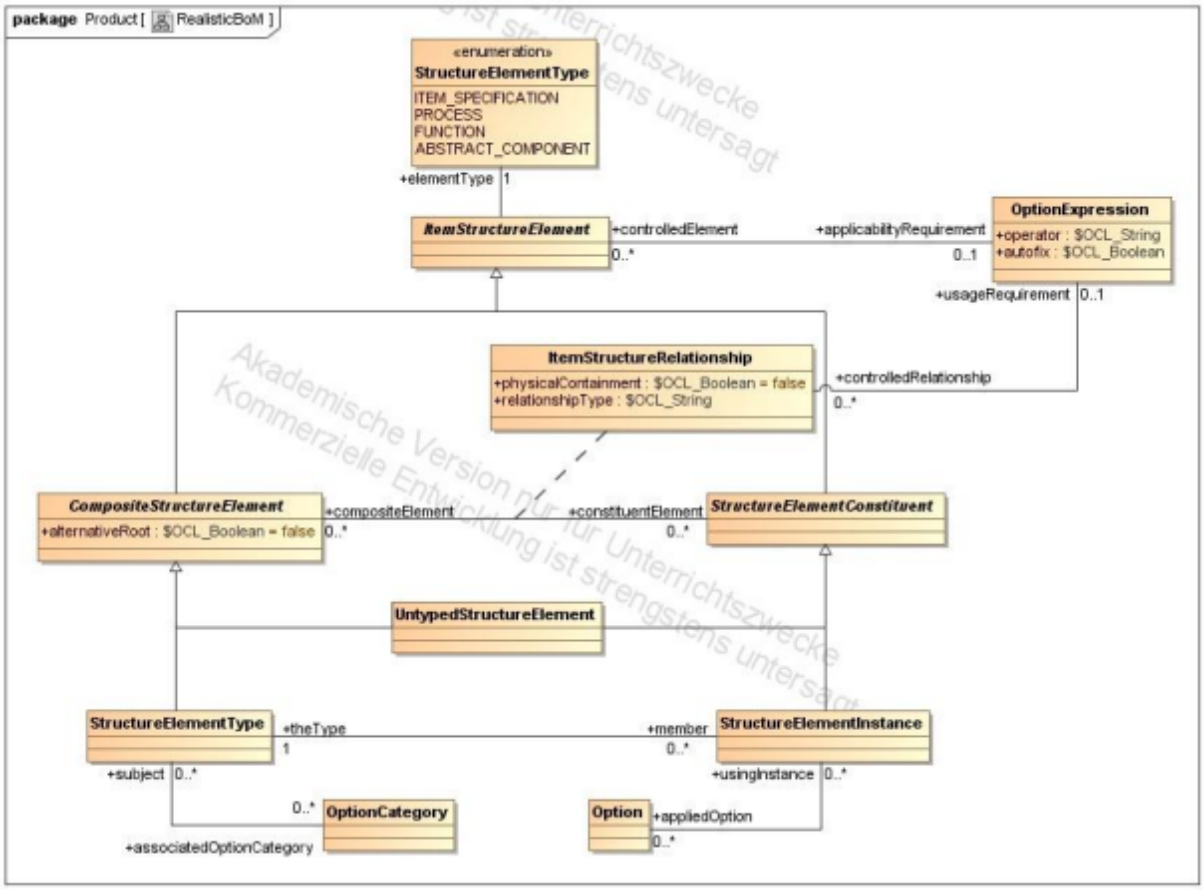

Abbildung 4.8: Stückliste + Produkt optionen

4.3.4 Produkt Optionen und Herstellungsprozesse bzw. Produkt kosten

Die Struktur in Abbildung 4.8 stellt beide Möglichkeiten schon zur verfügung.

4.3.5 Zusammenfassung

Das Variantenmanagement ermöglicht gigantische Möglichkeiten, aber es hat gewisse Limitierungen:

- Variable Abmessungen (wie in der Möbelindustry oder Sattelposition in der Automobilindustrie)
- Verbesserung: Variation Management (knowledgement Engeneering)

- Unterstützung für alle Datentypen in unserer Ausdrücke
- Generische Regelsprache ersetzt die begrenzte Option / Option Ausdruckssprache
- **Achtung:** Komplexität einer vollständigen Programmiersprache

5 Produktion

Die Produktion ist abhängig von den Fertigungswerken, der Ausrüstung und den Werkzeugen bzw. Hilfsmittel. Bevor die Produktion starten kann, müssen diese Dinge existieren/vorhanden sein. Weiters hängt auch die Effiktivität des Fertigungsprozesses von dem Grad der Instandhaltung der Fertigungswerke, der Ausrüstung und der Werkzeuge bzw. Hilfsmittel ab.

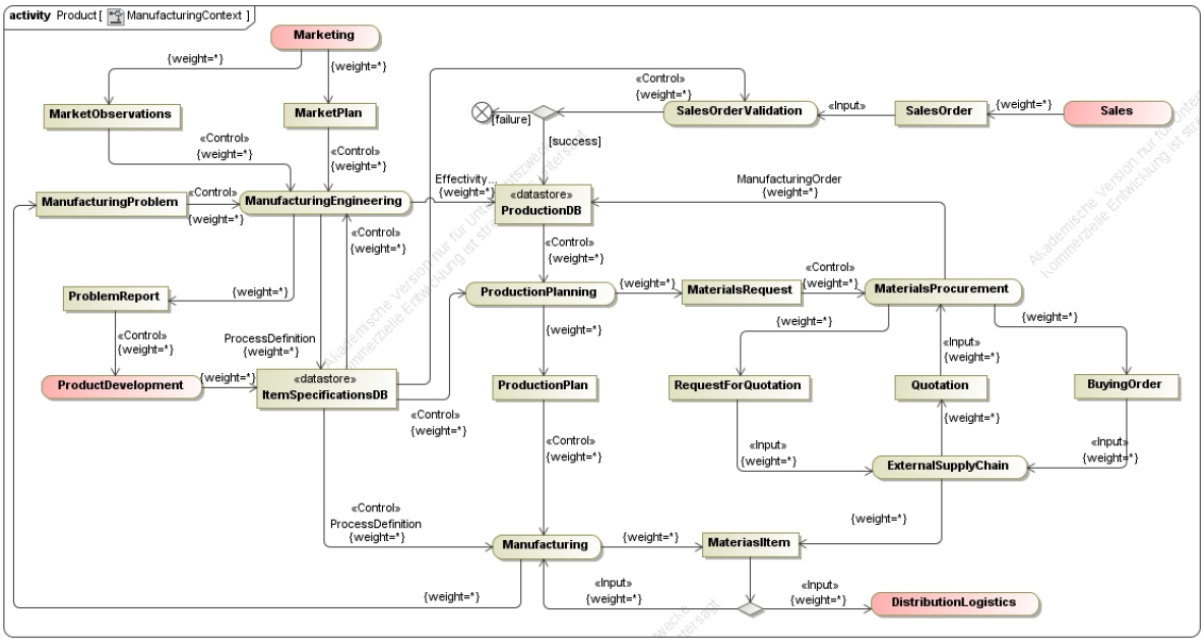

Abbildung 5.1: Produktionsprozessüberblick

5.1 Effiktivität

Effiktivität ist die Entscheidung einen Entwicklungsstand zu verwenden. Das "release level"hingegen ist die Entscheidung des Entwicklers, dass etwas fertig ist - wann es allerdings in den Release integriert wird, wird dabei nicht beeinflusst.

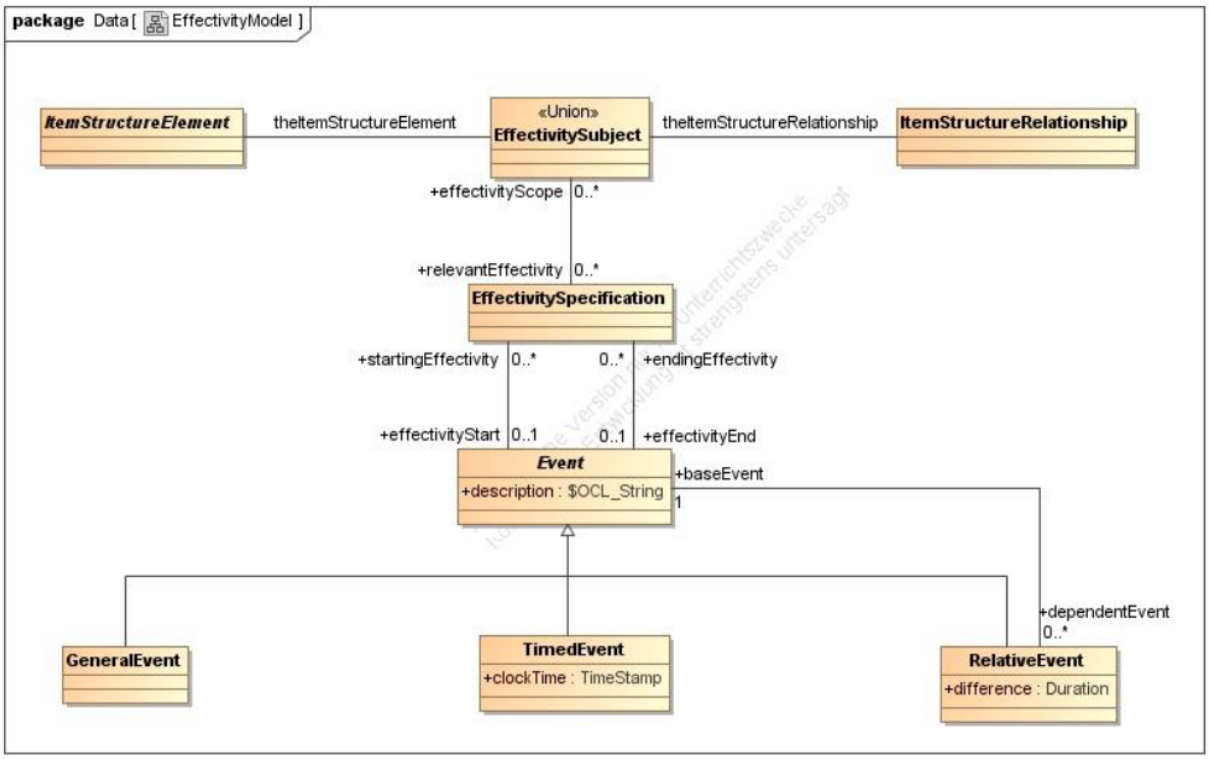

Abbildung 5.2: Effektivität

5.2 Prozesse

Manufacturing Engineering Auf Basis von Verfügbarkeit (Produktionseinrichtungen, Ausrüstungen, Geräte, ...) wird die Instandhaltung und Anpassung des Produktionsprozesse ausgeführt. Entscheidungen über die Effektivität von den Spezifikationen der Materialien werden auf Basis von Verfügbarkeit, Pläne des Marketings, Beobachtungen des Markts und der Probleme beim Produzieren getroffen. Aufgrund von Problemen beim Produktionsprozess werden Problemberichte erstellt.

Sales Order Validation Bestellungen werden vom Verkauf auf Grund von Produktspezifikationen/-definitionen und Informationen von PDM/PLM-Systemen gefiltert.

Production Planning Vorbereiten von detailierten und synchronisierten Abläufen für Produktionsstellen und Transporteinrichtungen auf Basis von Bestellungen sowie Informationen aus PDM/PLM-Systemen.

Materials Procurement Finale Entscheidung bezüglich:

- Fertigung oder Kauf
- Bestellung der Materialien
- Verhandlungen mit externen Partnern über den Kauf von Materialien

auf Basis von Materialanfragen von der Produktionsplanung. Intern wird dann entschieden ob gekauft oder gefertigt wird.

Manufacturing Die Qualitätssicherung der erhalten und produzierten Materialien sowie deren Lieferung wird auf Grund der Produktspezifikationen/-definitionen getroffen. Zeitnahe Produktion von Materialien auf Basis des Produktionsplans sowie der Produktspezifikationen/-definitionen.

External Supply Chain Platzhalter für alle Materiallieferaktivitäten die extern ablaufen.

6 Support

6.1 Produktkonfiguration

Produkt-Support-Ingeneure brauchen Informationen über den konkreten aktuellen Status bzw. die konkrete aktuelle Konfigurationen über das Produkt da dieser Status / diese Konfiguration von der Prduktdefintion des Produzenten abweichen kann. Kann gesetzliche Konsequenzen nach sich ziehen.

Als Teil der Instandhaltung sollten Produkt-Support-Ingeneuere den aktuellen Status / die aktuelle Konfiguration dokumentieren (zumindestens auf dem Level der Teilenummer). Produktbesitzer sollten Seriennummern dokumentieren.

Zu Support zählt unter anderem die initiale Installation sowie Deinstallation und Reinstallation.

Sammeln initialer Konfigurationen Das Problem bzw. die Schwierigkeiten beim Sammeln initialer Konfigurationsdaten sind die entstehenden Kosten. Außerdem sollte dieses Sammeln sehr konsequent und von beginn an gemacht werden.

6.2 Produktinstallation

- Vorbereiten der physikalischen Umgebung
- Installation des physikalischen Produkts (Abhängig von Umgebung und Transport), entfernen der transportspezfischen Teile und dokumentieren
- Installation der Supportdatenbank
- Testen
- Training der Benutzer

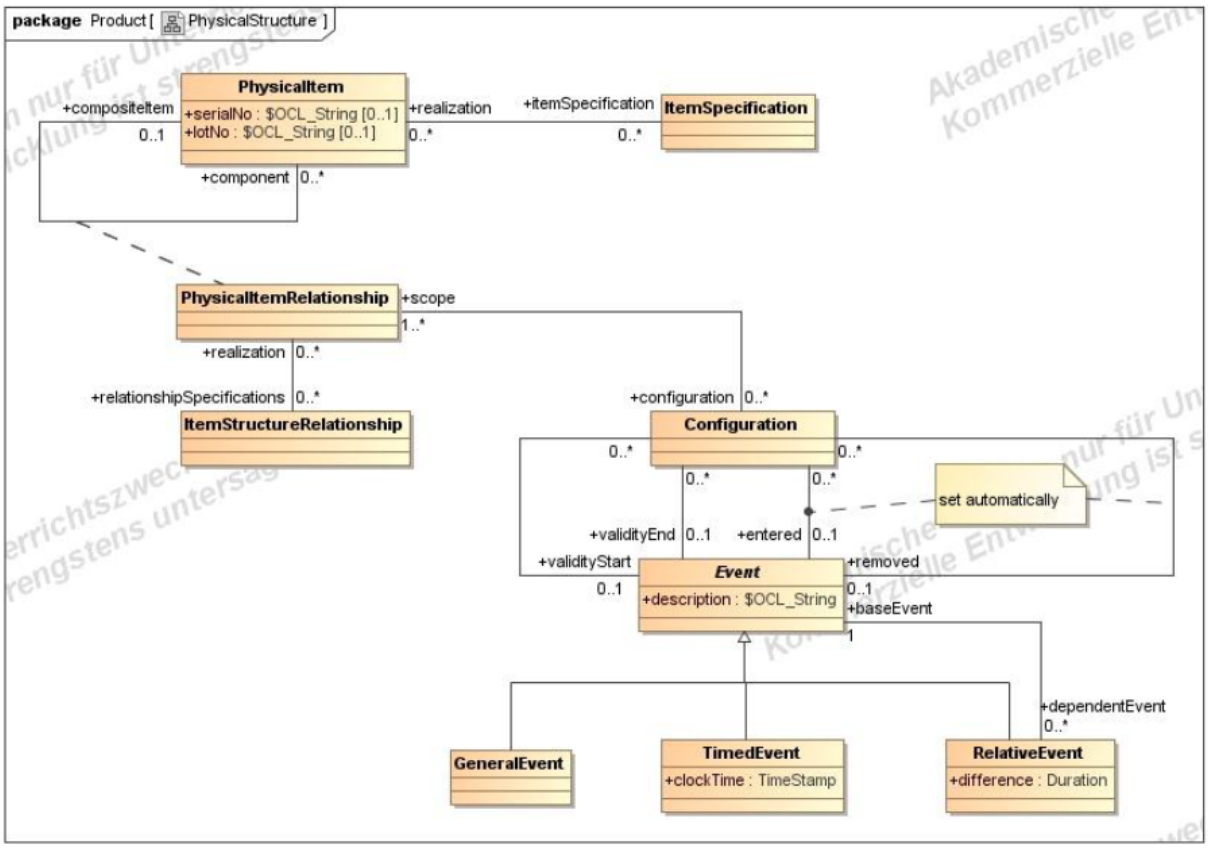

Abbildung 6.1: Produktionkonfiguration

6.3 Produktinstandhaltung und -überholung

Identifikation der Komponenten die angepasst und ersetzt werden müssen und Dokumentation der Änderungen. Gefolgt von Testen und erneutem Training der Benutzer.

6.4 Produkteinstellung und - außerbetriebnahme

Identifikation der Komponenten die entfernt/ausgetauscht oder modifiziert/geschützt werden müssen. Auch hierbei müssen die Änderungen dokumentiert und die Tests erneut durchgeführt werden.

6.5 Produktdeinstallation und -reinstallation

Nach der Außerbetriebnahme folgt die Deinstallation, Entfernen und zerlegen von physikalischen Komponenten um diese transportierbar zu machen. Nach dem Transport wird die zur Reinstallation übergegangen und gegebenenfalls werden Anpassungen (z.B. auf Grund von neuen Entwicklungen) vorgenommen. Nach dem Abschluss des Wiederaufbaus sollte wieder getestet werden und natürlich sollte jeder Schritt auch dokumentiert werden.

6.6 Produktzerstörung

Hierbei werden physikalische Komponenten entfernt und zerlegt, allerdings kann hierbei die Integrität der Komponenten ignoriert/vernachlässigt werden. Auch dieser Schritt sollte dokumentiert werden um gegebenfalls belegen zu können dass alles korrekt abgelaufen ist.

7 Produktbesitzer und Anwender

Der Anwender hängt an der Distributionslogistik. Den Anwender interessiert nicht das Design. Weiters hat er kein Interesse an der Produktnummer, sondern nur an seiner Seriennummer.

Ein gutes Beispiel für die Trennung ist die Automobilindustrie, bei der z.B. mit Chiptuning durch reine Software die Leistung eines einzelnen Autos (Seriennummer) geändert werden kann.

7.1 Kundensicht

Die Firma, die als Kunde fungiert, stellt die Spezifikation für das jeweilige Produkt. Der Auftragnehmer kann dabei Hilfestellung geben, indem gewisse Informationen an den Auftraggeber weitergegeben werden. Aus diesem Austausch definiert der Auftragnehmer eine genauere Spezifikation, welche den Produktanbietern übergeben werden. Der Anbieter muss nicht zwingend auch der Hersteller sein, es kann sich dabei z.B. auch um einen Großhändler handeln.

Die detaillierten Information sind für den Auftraggeber nicht, oder nur von marginalem, Interesse, hauptsächlich müssen die detaillierten Spezifikationen erfüllt werden, und das Produkt funktionieren.

Mit dem Kunde sollte auch definiert werden welche Normen verwendet werden, da verschiedene Länder/Organisationen unterschiedliche Normen veröffentlichen. Für die verschiedenen Zwecke sollten eindeutige Normen bestimmt werden.

Problematisch sind dabei auch die Lücke zwischen den Datenmodelliereren und der Terminologie die von den Experten in der jeweiligen Domänen verwendet wird.

Folgende weitere Punkte sind für die Zusammenarbeit auch wichtig:

- Einigung auf einen gewissen Bereich der Arbeit (Use Cases)
- Definition eines standardisierten Glossar von Klassen für das Projekt
- Einigung auf standardisierte Attribute für die Begriffe im Glossar
- Einigung auf die Maßeinheiten für die standardisierten Attribute

- Überprüfung auf die Notwendigkeit von weiteren Attributen anhand von Use Cases
- Abstimmung von zusätzlichen Attributen auf Grund von Anforderungen der Stakeholder
- Einigung auf standardisierten/normierten Austauschformat für Informationen
- Bei Möglichkeit **international anerkannte Normen** verwenden

7.2 Anwenderprozessmodell

Am Anfang muss man sich Gedanken über den Prozess machen, z.B. nach welchen Vorgehensweisen wird das Produkt hergestellt, welche Rohstoffe werden benötigt, was sind die Ergebnisse/Abfallprodukte des Prozesses.

Bei diesem Modell gibt es eine klare Trennung zwischen Prozess und Herstellung.

(Bild: Owner / Operator Process Model (2))

Dieses Bild gibt nur eine grobe Übersicht, hinter jedem Punkt hängt ein weiteres Diagramm, welches ungefähr die selbe Komplexität besitzt.

7.3 Anwenderdatenmodell

Hierbei werden die Änderungen über die Zeit am Produkt fest gehalten. Als Beispiel könnte man beim Auto das Wechseln von Einzelteilen oder sogar das Nachtanken nennen.

Dem Ganzen liegen zwei Prinzipien zu Grunde:

- Diese Änderungen können in einem flachgedrückten 3D-Raum über die Zeit dargestellt werden, welche umgangsprachlich auch Schlauchdiagramme genannt werden. Man kann sich das vorstellen, als ob die Objekte bei einer Bewegung durch den Raum kopiert werden.
- Bei temporären Teilen wird ein Bauteil oder eine Funktion über einen gewissen Zeitraum betrachtet. Bei diesen Diagrammen werden die Prozesse definiert, und die Verwendung der einzelnen Bauteilen auf einem zweidimensionalen Graph augetragen werden. Weiters könnte man auch den verwendeten Platz eines Produkts über eine gewisse Zeit abgebildet werden. (Bilder Owner /Operator Data Model(2))

(Bild Owner / Operator Data Model (1))

Bei diesem Modell wird zuerst ein Thing definiert, was eine Abstraktion für alles ist. Es werden auch Klassen definiert, welche natürlich auch ein Thing darstellen.

Es gibt auch Relationships, welche selber Things darstellen, und zwei Things in eine Beziehung stellen. Relationships haben zwei Subklassen: Specialization und Classification. Eine Specialization setzt zwei Klassen in eine Beziehung, un deine Classification ein Thing zu einer Klasse.

PossibleIndivuals gibt verschiedene Möglichkeiten für Relationships dar. Eine Unterklasse davon stellt das PhysicalObject dar, welches wie folgt weiter unterteilt werden kann.

- MaterializePhysicalObject ist ein physisches Objekt, welches im Großen und Ganzen an seiner Materie definiert werden
- FunctionalPhysicalObject wird durch seine Funktion definiert
- SpacialLocation ist ein Punkt oder eine Menge von Punkten im Raum
- Stream ist ein Fluss von Energie, Materie oder sonst irgendetwas

Die restlichen Möglichkeiten teilen sich wie folgt weiter:

- Ein anderes PossibleIndivual ist ein ActualIndiviual, wenn es in der Realität wirklich existiert.
- Ein ArrangedIndivual kann seine Funktion während seiner Lebenszeit verändern
- Ein WholeLifeIndividual hat über die gesamte Lebenszeit die gleiche Funktion
- Eine Activity ist eine Tätigkeit, wie z.B. das Reden eines Dozenten, oder das Zusammenbauen eines Teiles

Things die Activities beeinflussen sind in folgender Liste beschrieben:

- CauseOfEvent stellt zwei Things in eine Relation, die ein Event verursacht
- InvolvementByReference stellt die Verwendung Dingen in eine Beziehung
- Recognition beschreibt die Verwendung ohne Veränderung

Mit diesem Modell kann die Veränderung in Objekten der Welt einschließlich der Welt selbst über die Zeit dokumentiert werden. Es ist das einzig bekannte Modell, welches solche Veränderungen über einen so breiten Rahmen dokumentieren kann.