Author

Title

Sub/Work Title

University of Applied Sciences: Vorarlberg

Department of Computer Science
Professor / Supervisor

Dornbirn, 2014

Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

Listings

Abbildungsverzeichnis

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

3 Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.