VL2: Softwareprojekt - Anforderungsanalyse

Inhalt

1. Struktur eines Softwareprojektes
2. Anforderungsanalyse

1. Struktur eines Softwareprojektes

Ziel dieses Semesters ist es, ein kleines Software*projekt* durchzuführen, das ein kleines Software*programm* in der Sprache C++ realisiert. Dazu sollen alle wichtigen Phasen eines Softwareprojektes beispielhaft durchlaufen werden. Im nachfolgenden Schaubild kann

<table>
<tr><th>TIME</th><th>LANGUAGE</th><th>LAYER</th><th>SWE PROCESS</th></tr>
<tr><td></td><td>NL ...</td><td>CIM</td><td>Domain Modeling</td></tr>
<tr><td rowspan="2">Since 1966</td><td rowspan="2">Diagrams (UML, ...)</td><td>PIM</td><td>Design Model</td></tr>
<tr><td>PSM</td><td rowspan="4">Implementation</td></tr>
<tr><td rowspan="2">Since 1954</td><td rowspan="2">High Level Languages (Fortran, Algo, C, ...)</td><td>Application</td></tr>
<tr><td>Middleware</td></tr>
<tr><td rowspan="2">About 1941-46</td><td rowspan="2">Assembler Binary</td><td rowspan="2">Operating System</td></tr>
<tr><td colspan="4">Hardware</td></tr>
</table>

man den typischen Aufbau eines modernen Softwareprojektes in den Grundzügen erkennen. Das Erstellen von Software mit ingenieurmässigen Methoden nennt man heute

Softwareengineering (SWE). Das hier einschlägige Knowhow ist mittlerweile sehr umfangreich und wird in eigenen Lehrveranstaltungen eingeübt. Hier beschränken wir uns nur auf die Grundelemente, die notwendig sind, um ein erstes kleines Projekt realisieren zu können.

Im Rahmen des Softwareengineerings unterscheidet man grob vier Phasen bei der Erstellung einer Software: (i) die Ermittlung der Anforderungen an eine neue Software, um ein bestimmtes Problem zu lösen (:= das Erstellen eines ersten Modells ohne Berücksichtigung, wie dies durch den Computer gelöst wird = *Computer Independet Model* = *CIM*); (ii) die Modellierung bzw. das Design einer möglichst allgemeinen Lösung, noch ohne Berücksichtigung von Details (= *Platform Independent Model* = *PIM*); (iii) die Umsetzung der Lösungsidee bzw. die Implementierung in lauffähigen Programmkode sowie (= *Platform Specific Model* = *PSM*)(iv) das *Testen* des Programms und seine Übergabe.

Unter einem *Modell* versteht man hier eine Zusammenstellung von Elementen, Beziehungen zwischen diesen Elementen und deren Interaktion in der Zeit. Unter einer *Plattform* versteht man ein konkretes System (Hardware und Betriebssystem, evtl. sogar einschliesslich einer Middleware), auf dem das zu erstellende Programm ablaufen soll.

Beispiele für spezifische Plattformen wären z.B-. PCs mit dem Betriebssystem Linux oder MS Windows. Ein Beispiel für eine Middleware ist CORBA.

In dieser Vorlesung setzen wir als *spezifische Plattform* einen PC voraus, auf dem das Betriebssystem Linux mit der Grafikoberfläche KDE3.0 sowie die Programmiersprache C++ verfügbar ist.

Als ein internationaler Standard für die Art und Weise, wie man diese verschiedenen Modellierungsschritte vornimmt, beginnt sich heute die *Model Driven Architecture (MDA)* der *Objekt Management Group (OMG*; www.omg.org) durchzusetzen. Wir berücksichtigen diesen Standard hier nur soweit, als man innerhalb der Model Driven Architecture die Diagramsprache *UML (:= Unified Modeling Language)* benutzt, um die verschiedenen Modelle zu beschreiben. Zur Zeit gültig ist UML Version 1.5; Version 2.0 steht bald vor der Verabschiedung.

2. Anforderungsanalyse

Zur Anforderungsanalyse gehören die folgenden in der Tabelle blau unterlegten Felder:

SWE	
Domain	Festlegung eines Anwendungsbereiches
Requirements Analysis	Klärung der Aufgabenstellung
Data	Zusammenstellung aller wichtigen Dokumente, die die Aufgabe beschreiben; Lastenheft
CIM	Erstellen eines Usecase in UML, ergänzt um weitere Texte; Pflichtenheft
PIM	Erstellen eines abstrakten Modells mit UML
PSM	Anpassung des allgemeinen Modells an eine bestimtme Plattform
Kodierung	Übersetzen des speziellen Modells in lauffäigen Programmkode
CIM-based testing of PSM	Testen des Programmkodes anhand des Pflichtenheftes

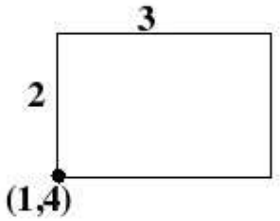
2.1 Anwendungsbereich (Domain)

Als Anwendungsbereich soll hier die Übersetzung von ASCII-Texten in Postscript-Diagramme angenommen werden. D.h. irgendwelche *Autoren* erstellen *ASCII-Texte*, die dann mittels einem zu erstellenden *ppmp2ps-Compiler* in *Postscript-Programme* übersetzt werden, die sich dann mit den entsprechenden *Postscript Readern* anzeigen lassen. Gleichzeitig kann man Postscripttexte auch über *postscriptffähige Drucker* ausdrucken. Über die Art der zu erstellenden ASCVII-Texte bzw. über die zu erstellenden Postscript-Programmen wird zunächst nichts Besonderes festgelegt.

2.2 Aufgabenstellung (Requirements Analysis)

Die eigentliche Aufgabe besteht in der Erstellung eines *ppmp2ps-Compilers*. Dies soll ein C++-Programm sein, das unter Linux läuft und bei Angabe einer ASCI-Datei (___.txt) diese Datei in eine Postscriptdatei (___.ps) übersetzt. Allgemein wird nur gefordert, dass

die ASCII-Texte solche Texte sind, die Strukturen beschreiben, die sich in Diagramme übersetzen lassen (Für ein Beispiel siehe Tabelle).

___.txt	***___.ps***	***PS-Ausgabe***
Qudrat((1,4),2,3)	...	
Ein Quadrat bei (1,4) mit Höhe 2 und Breite 3	...	

2.3 Daten/Lastenheft

An dieser Stelle wäre es jetzt wichtig, konkrete Daten zusammenzustellen, sofern man diese allgemeine Aufgabe mit konkreten Vorgaben bearbeiten wollte, z.B.:

1. Übersetzung von C-Programme in Flussdiagramme (Festlegen, welche Elemente eines C-Programmtextes in welche Diagrammelemente zu übersetzen wären)
2. Übrsetzen von boolschen Formeln in Schaltbilder mit Logikbausteinen
3. Übersetzen Betriebsanleitungen in Handlungspictogramme
4. Übersetzung von Spielzügen eines Schachspieles in Stellungsbilder
5. Übersetzung von Graphenbeschreibungen in Bilder von Graphen
6. ...

Die erste Aufgabe für jedes Team wird darin bestehen, sich eine solche konkrete Übersetzungsaufgabe herauszusuchen und die Details dieser Übersetzung genauer zu beschreiben.

2.4 CIM/Usecase + Pflichtenheft

Das Anwendungsfalldiagramm (Usecase) ist im vorliegenden Falle sehr einfach. Man hat einen Akteur, den User/Benutzer, der verschiedene aktionen vornehmen möchte. So soll der benutzer einen ASCII-Text auswählen können, diesen nach Postscript (PS) übesetzen können, als Postscript-Datei anschauen und/oder ausdrucken können, von Postscript nach PDF konvertieren und wiederum PDF-Dateien anschauen und/oder ausdrucken können. Man kann ein solches Anwendungsfalldiagramm mithilfe des

opensource Programms *umbrello* erstellen. Dieses ist Teil der Suse-Linux-Distribution.

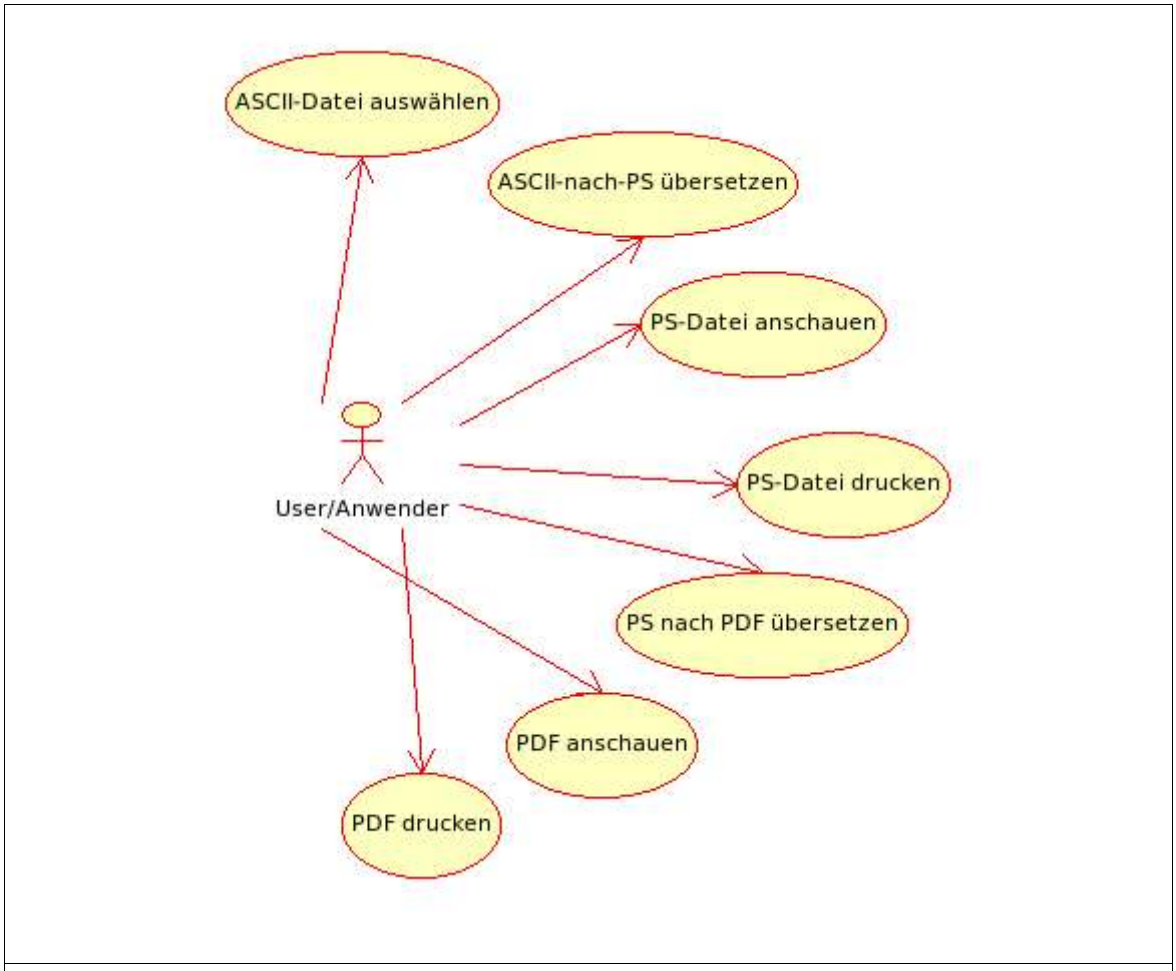


Bild 1: Anwendungsfalldiagramm für ppmp2ps

Man kann dann diejenigen Tätigkeiten auswählen, die durch ein zu erstellendes Programm realisiert werden sollen. In unserem Fall empfiehlt es sich, nur die Aufgaben "ASCII-Datei auswählen" sowie "ASCII-Datei nach Postcript konvertieren" auszuwählen, da es für die anderen Aufgaben schon fertige Programme gibt (*gv* für PS-Dateien anschauen; PS-Ausdruck können heute die meisten Betriebssysteme. PS nach PDF leistet das Programm *ps2pdf*. PDF anschauen leistet *acroread*). Diese zu integrieren überschreitet das Ziel dieser Lehrveranstaltung. Wir arbeiten also im Folgenden mit dem reduzierten anwendungsfall (siehe Bild 2).

Für die Benutzung des Programms *umbrello* hier die folgenden Hinweise: nachdem man das Programm *umbrello* gestartet hat, klickt man die Option Anwendungsfallansicht an und legt einen neuen Ordner --z.B. ppmp2ps-- an. Dann kann man auf der Bildschirmleiste die entsprechenden Symbole auf die Zeichenfläche laden und dabei beschriften. Schliesslich kann man die entsprehenden Pfeilsymbole anklicken und die Symbole durch klicken verbinden. Abschliessend kann man das ganze Diagramm als png-Bild exportieren udnd dann in ein Textprogramm importieren.

Sofern das Anwendungsfalldiagramm die Aufgabe nicht genau genug beschreibt muss

man dann in einem zusätzlichen Text die fehlenden Detailinformationen nachliefern, z.B. genauere Angaben über die Art der ASCII-Texte und über die Art der zu erzeugenden Symbole relativ zum Text.

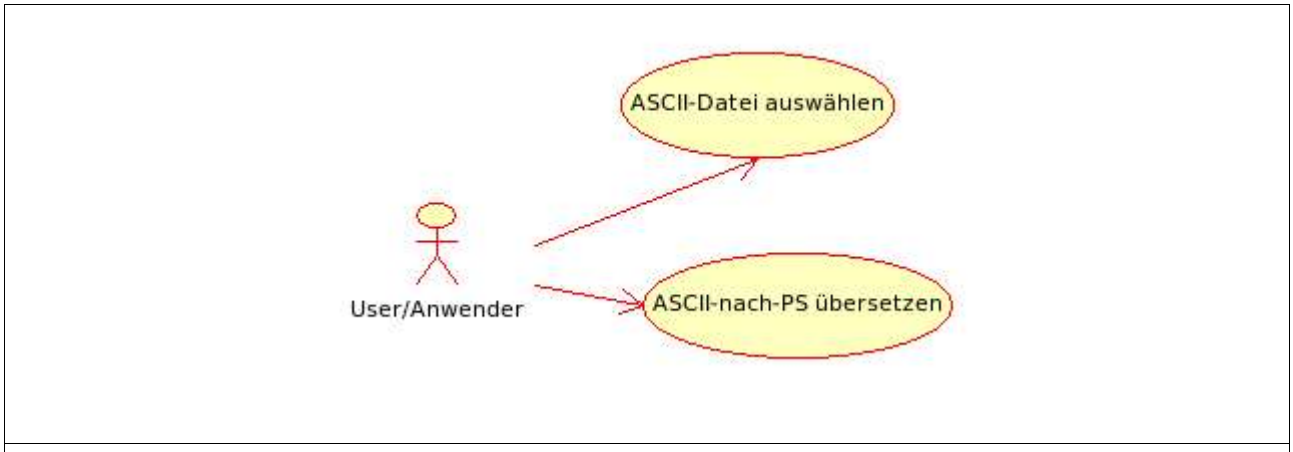


Bild 2: Reduziertes Anwendungsfalldiagramm für ppmp2ps

Am Ende sollte ein einziger Text --das *Pflichtenheft*-- im PDF-Format existieren, in dem die Aufgabenstellung inklusive Anwendungsfalldiagramm vollständig beschrieben wird.