Mein Praxissemester bei **MAKA**

Markus Schnalke
MatNr: 039131

Dies ist der Bericht zu meinem ersten Praxissemester im Studiengang Wirtschaftsinformatik an der Hochschule Ulm. Ich absolvierte dieses Praktikum in der IT-Abteilung der ***MAKA - Max Mayer Maschinenbau GmbH*** *in Nersingen.*

Inhaltsverzeichnis

Teil I.
Einleitung

1. Vorwort

Als Student einer Fachhochschule habe ich die Möglichkeit in Praxissemestern Einblicke in die Praxis der freien Wirtschaft zu sammeln. Ich sehe darin eine große Bereicherung für mich, die Hochschule und das Unternehmen und bin froh diese Chancen nutzen zu können.

Für die 24-wöchige Praktikumszeit hatte ich mir die Firma **MAKA - Max Mayer Maschinenbau GmbH** im nahen Nersingen als Arbeitgeber ausgesucht. Ich habe diese Zeit auf vielfältige Weise genossen und bin mir sicher, ganz viel Erfahrung für mein weiteres Studium mitgenommen zu haben.

Ich möchte in diesem Bericht, der meine Tätigkeit bei **MAKA** beschreibt, nun als Erstes kurz das Unternehmen selbst und die IT-Abteilung, der ich zugehörig war, vorstellen. Dann allgemeine Dinge zu meiner Arbeit sagen und einige Teile meiner Projekte detaillierter beschreiben. Abschließend werde ich ein Fazit des Praxissemesters ziehen.

2. Unternehmensvorstellung

2.1. Das Unternehmen

> Die **MAKA - Max Mayer Maschinenbau GmbH** ist ein inhabergeführtes Unternehmen mit rund 170 Mitarbeitern, das seine Wurzeln im schwäbischen Maschinenbau hat. Mit über 50 Jahren Erfahrung im Maschinenbau und über 25-jähriger Erfahrung im Bau von CNC-Spezialmaschinen setzt sich **MAKA** mit an die Spitze dieser Technik im den Bereichen Holz-, Aluminium- und Kunststoffbearbeitung.
>
> Ebenso werden individuelle Lösungen für die hohen Anforderungen im Modellbau und Gussbereich angeboten. Die Hochgeschwindigkeitsbearbeitung mit 5-Achs-Technik in der Aluminium-, Holz und Kunststoffbranche wird mit **MAKA**-CNC-Spezialmaschinen erfolgreich von weltweit führenden Firmen des Automobil-, Flugzeug-, Waggon-, Yacht-, Fassaden-, Möbel-, Türen- und Treppenbaus und von Spezialisten wie Tiefzieher, Kunststoff- und Acrylglasbearbeitern umgesetzt.
>
> Unser Unternehmen hat es sich zum Ziel gemacht, die richtige technische und wirtschaftlichste Lösung auf hohem Niveau zu bieten.

So beschreibt sich das größte Unternehmen Nersingens selbst. Neben dem Hauptsitz in Nersingen[1], gehören die eigene Produktion in Neu-Ulm-Burlafingen, das Tochterunternehmen in England und 13 Vertretungen in Europa dazu.

Der Umsatz verteilt sich zur Hälfte auf Deutschland, etwa 45% im restlichen Europa (davon knapp 20% im Vereinigten Königreich) und verbleibenden fünf Prozent in den USA und den Vereinigten Arabischen Emiraten.

Abbildung 1: Das Firmenlogo

Als Etwas das viel über ein Unternehmen aussagt, möchte ich hier noch die Philosophie von **MAKA** anfügen.

> Made in Germany, Alles aus einer Hand und Kundennähe sind die tragenden Säulen der **MAKA**-Kundenphilosophie und Eckpfeiler der Erfolgs.

Das Unternehmen ist auch im Internet vertreten. Unter `http://maka.com` ist die Webpräsenz zu erreichen. Ebenfalls online zu finden ist der Unternehmensteil **MAKA - Tischlereimaschinen** mit dem 1952 das Unternehmen entstanden ist.

[1] Zehn Kilometer östlich von Ulm gelegen

2.2. Die IT-Abteilung

Die IT-Abteilung bei **MAKA** umfasst drei Personen. Das sind: Der Abteilungsleiter, der sich, neben den vielfältigen Aufgaben seiner Position, auch um das ERP-System kümmert. Ein Systemadministrator der Hardwarebelange und die Standardsoftware unter sich hat. Und schließlich mein Betreuer, einstmals Student an der FH-Ulm, jetzt Programmierer und Verantwortlicher für das *DBSRV*-Projekt bei **MAKA**.

Hardware und Software werden intern verwaltet; Berater und einzelne Spezialisten werden von extern bezogen. Ganz allgemein wird ein "möglichst viel in-house"-Konzept verfolgt.

Die IT-Abteilung ist stark mit dem restlichen Unternehmen verwoben und hat, meiner Ansicht, eher wenig Entscheidungsmacht.

2.3. MAKA und die Hochschule

MAKA setzt als ERP-System das niederländische *Baan* ein. Dieses, in einigen Punkten suboptimale, System, bietet vor allem eines nicht: Übersichtlichkeit. Viele Masken und Reports sind sehr im Detail. Sie ermöglicht zwar alle nötigen Aktionen und auch Abfragen wie "Alle Fehlteile eines Projekts" sind implementiert; wenn man aber wissen möchte, welche Bestellungen für diese Fehlteile laufen und wann diese vorraussichtlich geliefert werden, dann steigt der Aufwand für diese Informationen schnell exponentiell: Drei verschiedene Masken, nervige Nummerntipperei und eine große Zeitinvestition müssen dafür in Kauf genommen werden.

Um diese Unzulänglichkeiten des ERP-Systems auszugleichen wurde 2004 von **MAKA** über den Dozent Herrn von Schwerin eine Zusammenarbeit zwischen dem Unternehmen und der Fachhochschule Ulm angeregt: In Projektcharakter analysierten die damaligen Studenten Andreas Heinemann und Fabian Heß die Situation, informierten sich über mögliche Lösungen, entwarfen das Konzept der heutigen Lösung und programmierten schließlich die ersten Anwendungen.

Was als Projekt in ihrer Freizeit begann, führt über Praxissemester und Diplomarbeit, bei einem der Beiden sogar zur Festanstellung bei **MAKA**.

Erfolg auf ganzer Linie. Ein Musterprojekt der Hochschule!

Seit dieser Zeit wurde die Zusammenarbeit mit der Hochschule weiter gepflegt. Ich selbst bin der inzwischen vierte Wirtschaftsinformatik-Student der bei **MAKA** die Verbindung, und damit den Wissens- und Erfahrungsaustausch, zwischen Forschung und Wirtschaft herstellt.

3. Rahmenbedingungen

Bevor ich näher auf meine Tätigkeit für die **Max Mayer Maschinenbau GmbH** eingehe, möchte ich zuerst die gegebene Entwicklungssituation beschreiben.

Grundsätzlich hatte ich sehr viele Freiräume die Situation an meine Wünsche und Bedürfnisse anzupassen.

Die Entwicklung der Programme fand stets im direkten Kontakt mit den späteren Anwenden statt. Das war, vor allem zu Beginn aber auch später noch, notwenig, da oft nur sie das nötige Hintergrund- und Praxiswissen hatten.

Des Weiteren entwickelte ich direkt am Live-System. Anfangs war ich darüber eher überrascht, hätte ich mir dies in einem Unternehmen wie **MAKA** nicht vorstellen können. Doch diese scheinbare Unprofessionalität stellte sich schnell als sehr erfolgreich heraus. Änderungen und Neuentwicklungen meinerseits waren auf diese Weise sofort verfügbar und in Verbindung mit dem engen Anwenderkontakt konnte ich von ihrem direkten Feedback profitieren. Mit der Entwicklung am Live-System wird aber natürlich in Kauf genommen, dass es entwicklungsbedingt zu Ausfällen kommen kann. Da das System nicht von *entscheidender* Notwendigkeit ist, sondern eher Zusatznutzen und Komfort bietet, ist dies akzeptabel. Im Allgemeinen treten Ausfälle nur in Programmteilen auf, an denen gerade programmiert wird.

Ein weiterer Punkt um dieses Vorgehen verständlich zu machen, ist die Dynamik des Systems selbst. Während ich mir einem Programmteil beschäftigt war, habe ich nebenher auch kleine Änderungswünsche und Bugfixes für andere (ältere) Programmteile umgesetzt. Eigentlich gibt es kein *abgeschlossen* bei diesen Programmen, denn sie alle sind einer ständigen Veränderung und Anpassung unterworfen. Dieses System *lebt* ... auch weil ständig daran programmiert wird.

Teil II.
Aufgabenbeschreibung

4. Das DBSRV-Projekt

Wie in der Einführung beschrieben, wurde von den ersten Praktikanten bei **MAKA** das DBSRV-Projekt ins Leben gerufen. Der DBSRV ist eine Art Framework für allerlei Programme die den Mitarbeitern zur Verfügung stehen. Zum Teil erweitern diese Programme die Funktion des ERP-Systems, aber es gibt auch Programme die komplett neue Funktionalitäten anbieten[2]. All diese Programme sind von Praktikanten nach und nach geschrieben worden.

Ich habe hier einfach an der Stelle weitergemacht an der mein Vorgänger aufgehört hatte. Pflichtenhefte, oder vielleicht besser "Konzepte" existierten dazu bereits.

Im Folgendem werde ich den Begriff *DBSRV* verwenden wenn ich dieses DBSRV-Projekt meine.

Während meines Praktikums habe ich an drei Programmen gearbeitet:

1. Einkaufsinformationssystem (kurz EIS)
   Eine Abfragensammlung mit Reports wie Bestellobligo, Realer Wiederbeschaffungszeit und einem Fehlteilmanagement. Dieses Programm war von meinem Vorgänger schon zur Hälfte fertig gestellt.

2. Montageinformationssystem (kurz MontIS)
   Dem EIS recht ähnlich aber für die Montage. Zu implementieren war das Fehlteilmanagement und die Liefertreueanalyse.

3. Translator-Datenbank & Dictionary
   Eine Übersetzungsdatenbank für fremdsprachige Artikelbezeichungen um entsprechend lokalisierte Stücklisten drucken zu können.

[2] Ein Beispiel hierfür wäre die Bilderdatenbank

5. Einarbeiten

Die erste Woche verbrachte ich damit die bestehende Codebasis des EIS und die Datenbank des ERP-Systems kennen und verstehen zu lernen. Dabei ist es natürlich eine Schwierigkeit sich die Denkmuster des vorherigen Programmierers zu verinnerlichen. Eine dabei sehr gute Entscheidung war den Code durch Refactoring kennenzulernen und gleichzeitig zu verbessern. Code zu lesen führt meist nicht zum gewünschten Erfolg (jedenfalls nicht in dem Maße). Indem man den Programmcode allerdings 'umräumt', und die dabei zwangsläufig entstehenden Fehler[3] studiert, lernt man die Funktion des Codes sehr schnell und intensiv kennen.

Nachdem ich die Programmiergewohnheiten meines Vorgängers analysiert hatte und, durch die Arbeit mit dem Code, einen gewissen Einblick in die Logik der Datenherkunft erhalten hatte, machte ich mich daran selbst meinen ersten Report zu schreiben. Das größte Problem am Anfang war dabei herauszufinden aus welchen Datenbanktabellen ich die benötigten Daten erhalte. Bei über 70 Datenbanktabellen, mit zudem recht kryptischen Namen, war das auch verständlich. Mit der Zeit erkennt man dann aber ein Schema hinter den Bezeichnungen und kann sie sich so herleiten. Als Beispiel möchte ich die Tabellenspalte `ttdpur041100.T_ORNO` anführen. Das erste `t` ist bei allen Tabellen so, dann folgt ein `td` für das Baan-Modul "distribution" (engl. für "Vertrieb" oder "Absatz"), `pur` steht für "purchase" (engl. für "Beschaffung"), die `041` ist die Tabelle "Bestellkopf" und die `100` steht für die Echtfirma. Um solche Schemen nachvollziehen zu können war auch ein Wissen über den Aufbau eines ERP-Systems notwendig, das ich mir während des Praktikums angeeignet habe. Mit der Zeit habe ich dann auch ein Gefühl dafür bekommen wo ich welche Daten finde.

[3] Die Erfahrung (und Murphy) zeigt, dass durch Refactoring immer Fehler entstehen werden. Oder wenn man es von der anderen Seite betrachtet, kommen so (Design-)Fehler zum Vorschein.

6. Das Einkaufsinformationssystem

Dieses Programm umfasst Reports aus den Bereichen

- Lieferantenanalysen
- Artikelanalysen
- Tagesgeschäft
- Fehlteilmanagement

Die ersten zwei Bereiche wurden von meinem Vorgänger schon fertig gestellt. Das Tagesgeschäft war meine erste Aufgabe. Es war gut an einem bereits begonnen Programm weiterzumachen, da das im Normalfall einfach als selbst von Grund auf Entwerfen ist. Gerade wenn man sich mit dem System noch nicht besonders gut auskennt.

6.1. Tagesgeschäft

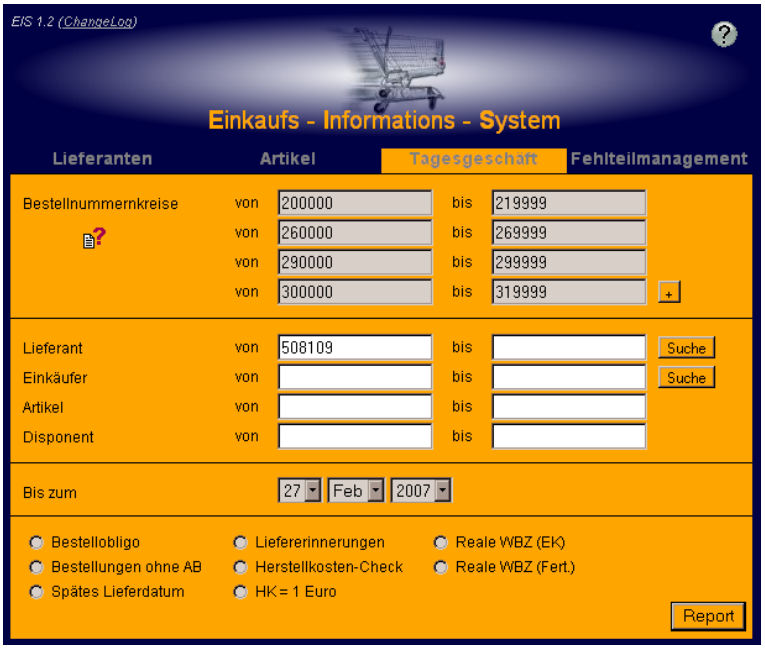


Abbildung 2: Die Eingabemaske des Tagesgeschäft-Programmteils

Der Reiter *Tagesgeschäft* enthält im oberen Bereich Formularfelder mit denen die Daten eingeschränkt werden können. Auf diese Weise kann zum Beispiel nur ein bestimmter Lieferant, oder nur eine Gruppe von Artikeln betrachtet werden. Einzelne

Einschränkungsfelder sind je nach gewähltem Report eventuell auch deaktiviert. So macht zum Beispiel beim *Herstellkosten-Check*[4] eine Einschränkung nach Lieferant oder Einkäufer keinen Sinn, da hier die Artikel ohne sonstigen Zusammenhang betrachtet werden.

Bei den Bestellnummernkreisen können beliebig weitere hinzugefügt werden. Bei Lieferant und Einkäufer kann über eine Suchfunktion die Lieferanten-/Einkäufer-Nummer anhand des Namens ermittelt werden. Und bei manchen Reports können die Datensätze zeitlich begrenzt werden.

Ich möchte nun zwei Reports näher vorstellen.

6.1.1. Bestellobligoanalyse

Dieser Report stellt eine übersichtliche Darstellung der auf das Unternehmen zukommenden Kosten im Bereich Einkauf bereit. Die Analyse bezieht sich erst einmal auf Kalenderwochen, aber Monatssummen sind vorhanden. Zudem sind für alle Zeiträume Drilldowns auf die konkreten Bestellungen vorhanden. Es wird also eine Bestellobligoübersicht für alle Detailstufen angeboten.

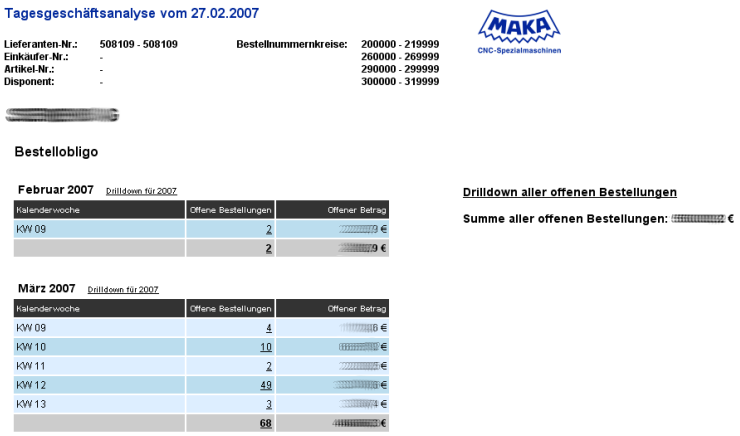


Abbildung 3: Die Bestellobligoanalyse

Im Drilldown zeigt ein Klick auf einen Lieferantennamen seine Kontaktdaten an.

6.1.2. Reale Wiederbeschaffungszeit

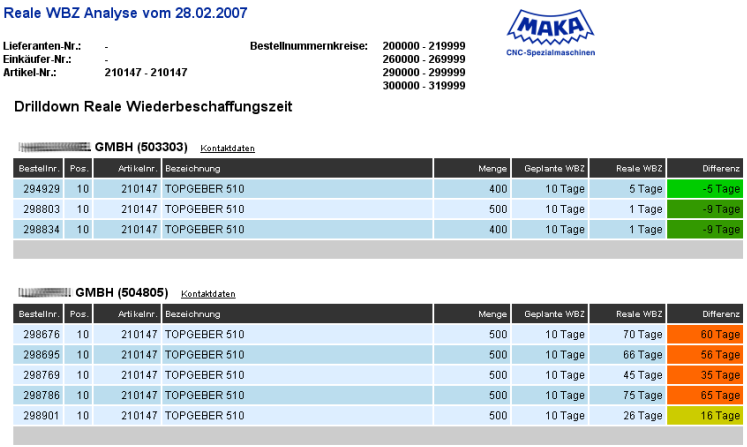
In diesem Report kann die tatsächliche (reale) Wiederbeschaffungszeit mit der, im Artikelstamm abgelegten, geplanten Wiederbeschaffungszeit verglichen werden.

[4] Eine Auflistung von Artikeln deren Soll-Herstellkosten oder letzter Einkaufspreis mehr als 5% von den Standard-Herstellkosten abweichen

Bei größeren Abweichungen, die farblich hervorgehoben sind, können dann die Artikelstammdaten korrigiert werden. Falls dies nicht gemacht würde, würde es im, vom ERP-System generierten, Produktionsplan zu Wartezeiten und Verschiebungen kommen. Um die Zeitplanung im Produktionsplan möglichst optimal zu halten (Stichwort "Just-in-time") müssen die benötigten Daten natürlich auch möglichst exakt sein.

Aber auch in der Beziehung zum Lieferant sind die reale Wiederbeschaffungszeit und ihre Abweichung zur Geplanten wichtige Daten. Mit einer anschaulichen Übersicht kann hier beim Lieferant ein gewisser Druck ausgeübt werden, falls er seine 'versprochenen' Lieferzeiten nicht einhält. Oft ist bei Gesprächen mit dem Lieferant die reine Präsenz dieser Daten und ihre sofortige und detaillierte Abrufbarkeit, der entscheidende 'Trumpf'.

Wenn Artikel von verschiedenen Lieferanten bezogen werden, dann lassen sich diese sehr schön gegenüberstellen. Und gegen die Fakten aus Abbildung 4 muss der untere Lieferant erst einmal etwas sagen.



Reale WBZ Analyse vom 28.02.2007

MAKA CNC-Spezialmaschinen

Lieferanten-Nr.: -
Einkäufer-Nr.: -
Artikel-Nr.: 210147 - 210147

Bestellnummernkreise: 200000 - 219999, 260000 - 269999, 290000 - 299999, 300000 - 319999

Drilldown Reale Wiederbeschaffungszeit

[redacted] GMBH (503303) Kontaktdaten

Bestellnr.	Pos.	Artikelnr.	Bezeichnung	Menge	Geplante WBZ	Reale WBZ	Differenz
294929	10	210147	TOPGEBER 510	400	10 Tage	5 Tage	-5 Tage
298803	10	210147	TOPGEBER 510	500	10 Tage	1 Tage	-9 Tage
298834	10	210147	TOPGEBER 510	400	10 Tage	1 Tage	-9 Tage

[redacted] GMBH (504805) Kontaktdaten

Bestellnr.	Pos.	Artikelnr.	Bezeichnung	Menge	Geplante WBZ	Reale WBZ	Differenz
298676	10	210147	TOPGEBER 510	500	10 Tage	70 Tage	60 Tage
298695	10	210147	TOPGEBER 510	500	10 Tage	66 Tage	56 Tage
298769	10	210147	TOPGEBER 510	500	10 Tage	45 Tage	35 Tage
298786	10	210147	TOPGEBER 510	500	10 Tage	75 Tage	65 Tage
298901	10	210147	TOPGEBER 510	500	10 Tage	26 Tage	16 Tage

Abbildung 4: Wiederbeschaffungszeiten eines Artikels von zwei Lieferanten

6.2. Fehlteilmanagement

Dieser Programmteil zeigt die momentanen Fehlteile und alle dafür laufenden Bestellungen auf.

Als erstes wir eine Liste mit allen Projekten in denen noch Teile fehlen generiert. Von dieser kann je ein Drilldown mit einer Gruppierung nach Lieferanten oder nach Baugruppen aufgerufen werden. Die beiden Drilldowns bieten dabei leicht unterschiedliche Funktionen.

Da das Fehlteilmanagement mich recht lange beschäftigt hat und auch im Montageinformationssystem integriert ist, möchte ich hier etwas genauer darauf eingehen.

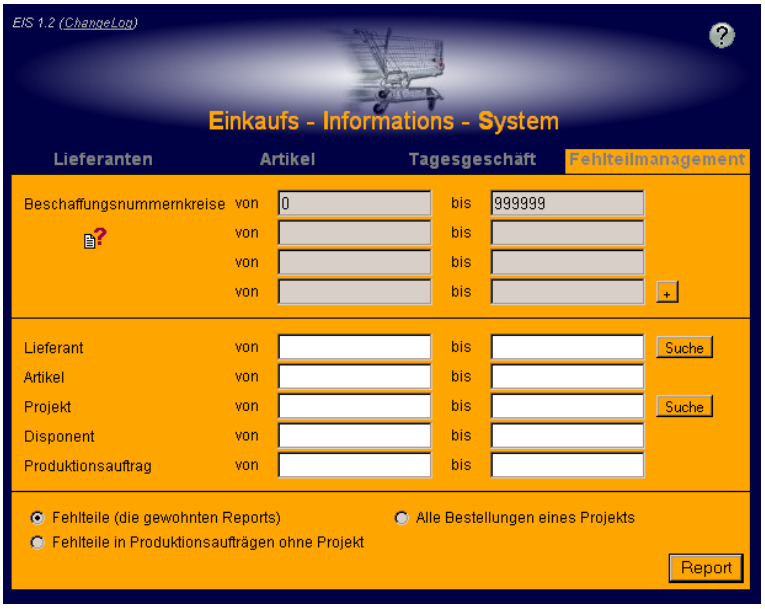


Abbildung 5: Startmaske des Fehlteilmanagements

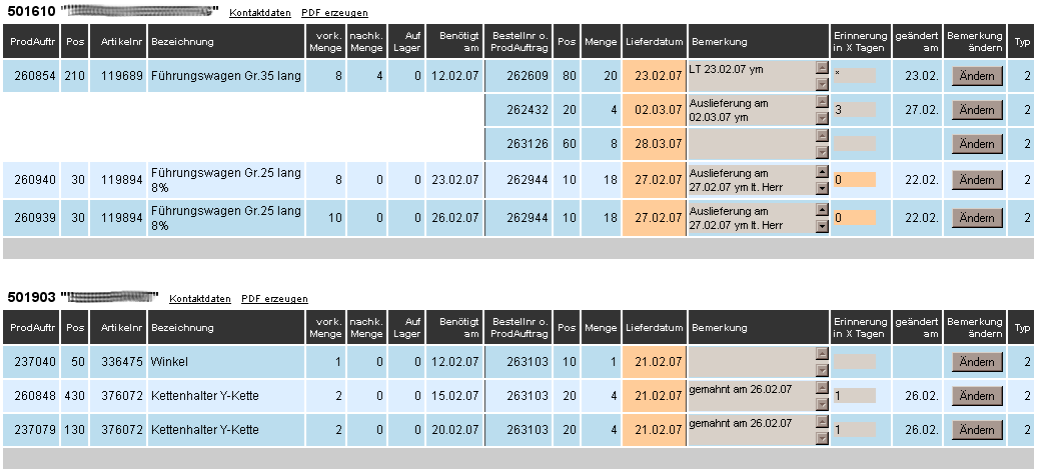
Anzumerken wäre hier vielleicht noch, dass das Fehlteilmanagement auch die komplizierteste Datenbeschaffung hatte — Joins über zwölf Tabellen, ebensoviele Where-Bedingungen und eine getrennte Behandlung von kundenspezifischen und Standardartikeln. Hier hatte ich dann auch Performanceprobleme die nun zwar nicht ganz verschwunden, aber inzwischen in einem erträglichen Rahmen sind. Im Nachhinein gesehen, wäre es vielleicht besser gewesen, einfacheres SQL und dafür komplexeres PHP zu verwenden. Inwiefern die Geschwindigkeit dabei gestiegen (und die Lesbarkeit des Codes eventuell gesunken) wäre, darüber kann ich nur spekulieren.

6.2.1. Drilldown nach Lieferanten

Die Tabelle ist logisch dreigeteilt:

- Die linken acht Spalten (bis zur senkrechten schwarzen Linie) enthalten Daten zum Artikel selbst.
- Die nächsten vier Spalten stellen die Bestellung dar. Laufen mehrere Bestellungen die auf diesen Artikel passen, so werden sie allen angezeigt. Das Lieferdatum ist rötlich hinterlegt, falls die Bestellung erst nach dem benötigten Datum eintreffen würde.

- Die letzten Spalten ermöglichen es den Einkäufern zusätzliche Informationen zu den Bestellungen einzutragen und sich selbst an Bestellungen erinnern zu lassen.



501610 [redacted] Kontaktdaten PDF erzeugen

ProdAuftr	Pos	Artikelnr	Bezeichnung	vork. Menge	nachk. Menge	Auf Lager	Benötigt am	Bestellnr o. ProdAuftrag	Pos	Menge	Lieferdatum	Bemerkung	Erinnerung in X Tagen	geändert am	Bemerkung ändern	Typ
260854	210	119689	Führungswagen Gr.35 lang	8	4	0	12.02.07	262609	80	20	23.02.07	LT 23.02.07 ym	x	23.02.	Ändern	2
								262432	20	4	02.03.07	Auslieferung am 02.03.07 ym	3	27.02.	Ändern	2
								263126	60	8	28.03.07				Ändern	2
260940	30	119894	Führungswagen Gr.25 lang 8%	8	0	0	23.02.07	262944	10	18	27.02.07	Auslieferung am 27.02.07 ym lt. Herr	0	22.02.	Ändern	2
260939	30	119894	Führungswagen Gr.25 lang 8%	10	0	0	26.02.07	262944	10	18	27.02.07	Auslieferung am 27.02.07 ym lt. Herr	0	22.02.	Ändern	2

501903 [redacted] Kontaktdaten PDF erzeugen

ProdAuftr	Pos	Artikelnr	Bezeichnung	vork. Menge	nachk. Menge	Auf Lager	Benötigt am	Bestellnr o. ProdAuftrag	Pos	Menge	Lieferdatum	Bemerkung	Erinnerung in X Tagen	geändert am	Bemerkung ändern	Typ
237040	50	336475	Winkel	1	0	0	12.02.07	263103	10	1	21.02.07				Ändern	2
260848	430	376072	Kettenhalter Y-Kette	2	0	0	15.02.07	263103	20	4	21.02.07	gemahnt am 26.02.07	1	26.02.	Ändern	2
237079	130	376072	Kettenhalter Y-Kette	2	0	0	20.02.07	263103	20	4	21.02.07	gemahnt am 26.02.07	1	26.02.	Ändern	2

Abbildung 6: Drilldown nach Lieferanten im Fehlteilmanagement

Dieser dritte Tabellenteil verbindet dann auch das Einkaufs- mit dem Montageinformationsystem. Denn die eingetragenen Notizen können über das Montageinformationssystem abgerufen werden. Eingetragene verschobene Liefertermine oder bereits vollzogenen Mahnungen ersparen so manchen Anruf.

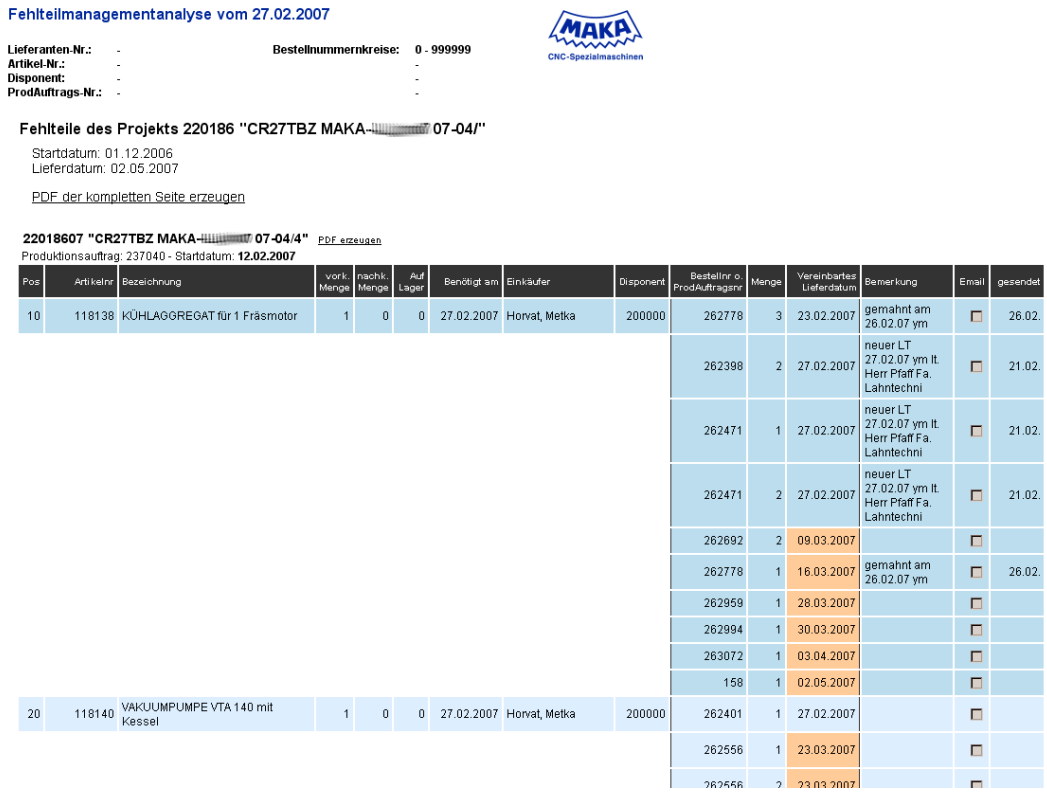
Mit der *Erinnerung in X Tagen* kann man sich in einer selbst gewählten Anzahl von Tagen eine Erinnerung zukommen lassen. Diese wird von einem Job automatisch per Email verschickt. Die Email-Adressen wurden dabei über eine Abfrage des LDAP-Verzeichnises gewonnen.

Am Anfang der Seite und über jeder Baugruppe befindet sich ein Link mit dem ein PDF-Dokument der ganzen Seite oder nur der einzelnen Baugruppe erzeugt werden kann. Dieses ist besser zum Ausdrucken geeignet und kann problemlos per Email verschickt werden. PDFs habe ich mit der FPDF-Bibliothek[5] erstellt. Die einzige Schwierigkeit dabei war das Überführen eines (eher) fließenden Layouts in HTML in ein Layout mit absoluten Maßen im PDF. Größere Probleme gab es dabei allerdings nicht.

6.2.2. Drilldown nach Baugruppen

Die Aufteilung der Tabelle entspricht der im Lieferantendrilldown.

[5] *FPDF* ist Freeware und auf http://fpdf.org zu finden.



Fehlteilmanagementanalyse vom 27.02.2007

Lieferanten-Nr.: - Bestellnummernkreise: 0 - 999999
Artikel-Nr.: - -
Disponent: - -
ProdAuftrags-Nr.: - -

Fehlteile des Projekts 220186 "CR27TBZ MAKA-[redacted] 07-04/"

Startdatum: 01.12.2006
Lieferdatum: 02.05.2007

PDF der kompletten Seite erzeugen

22018607 "CR27TBZ MAKA-[redacted] 07-04/4" PDF erzeugen
Produktionsauftrag: 237040 - Startdatum: 12.02.2007

Pos	Artikelnr	Bezeichnung	vork. Menge	nachk. Menge	Auf Lager	Benötigt am	Einkäufer	Disponent	BestellNr o. ProdAuftragsnr	Menge	Vereinbartes Lieferdatum	Bemerkung	Email	gesendet
10	118138	KÜHLAGGREGAT für 1 Fräsmotor	1	0	0	27.02.2007	Horvat, Metka	200000	262778	3	23.02.2007	gemahnt am 26.02.07 ym	☐	26.02.
									262398	2	27.02.2007	neuer LT 27.02.07 ym lt. Herr Pfaff Fa. Lahntechni	☐	21.02.
									262471	1	27.02.2007	neuer LT 27.02.07 ym lt. Herr Pfaff Fa. Lahntechni	☐	21.02.
									262471	2	27.02.2007	neuer LT 27.02.07 ym lt. Herr Pfaff Fa. Lahntechni	☐	21.02.
									262692	2	09.03.2007		☐	
									262778	1	16.03.2007	gemahnt am 26.02.07 ym	☐	26.02.
									262959	1	28.03.2007		☐	
									262994	1	30.03.2007		☐	
									263072	1	03.04.2007		☐	
									158	1	02.05.2007		☐	
20	118140	VAKUUMPUMPE VTA 140 mit Kessel	1	0	0	27.02.2007	Horvat, Metka	200000	262401	1	27.02.2007		☐	
									262556	1	23.03.2007		☐	
									262556	2	23.03.2007		☐	

Abbildung 7: Drilldown nach Baugruppen im Fehlteilmanagement

Anders sind hingegen die letzten Spalten. Der Drilldown nach Baugruppen stellt hier so etwas wie das Gegenstück zum Drilldown nach Lieferanten dar. Die Bemerkungen können hier nur gelesen werden, dafür können Aufforderungen zurück an die Einkäufer geschickt werden, dass sie ihre Bestellungen überprüfen sollten.

Funktionen wie diese gehen über das ursprüngliche Programmkonzept hinaus. Sie sind erst im Laufe der Programmentwicklung aufgekommen und herangereift. Es war häufig so, dass das Programm erst mit dem Programmieren gewachsen ist und seine endgültige Gestalt bekommen hat. Dies war vor allem durch den engen Kontakt mit den Anwendern möglich. Sie haben jede Programmversion direkt getestet und mir Verbesserungen, Anregungen und Wünsche mitgeteilt. Mir hat es gut gefallen, auf diese Weise Programme zu “erarbeiten”.

An dieser Stelle möchte ich nun zum Montageinformationssystem (kurz *MontIS*) überleiten.

7. Das Montageinformationssystem

Das Programm ist nach Konzept deutlich umfangreicher, es wurde allerdings beschlossen, dass zum jetzigen Zeitpunkt nur ein Teil dessen umgesetzt werden sollte. Dies sind die beiden Reiter:

- Fehlteilmanagement
- Liefertreue

Der Aufwand dafür wurde dadurch verringert, dass beide Programmteile auch schon im Einkaufsinformationssystem implementiert waren. Die Liefertreueanalyse von meinem Vorgänger das Fehlteilmanagement von mir. Somit war das reine Programmieren der Reports nicht das große Problem — sie mussten zum großen Teil nur angepasst und mit anderen Daten versorgt werden.

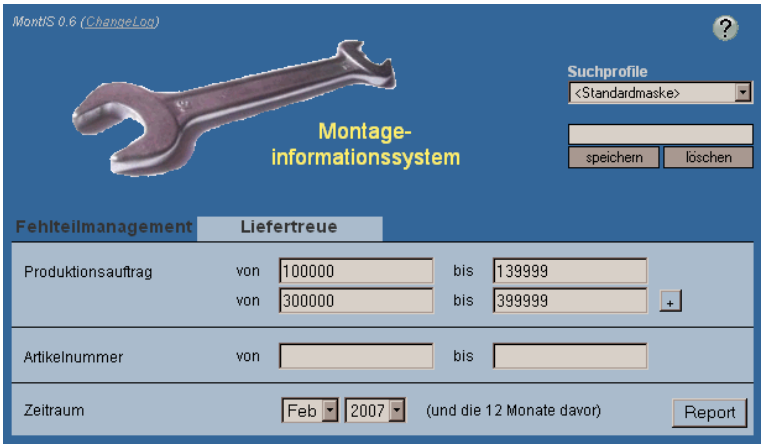


Abbildung 8: Maske der Liefertreueanalyse im MontIS

Inzwischen kannte ich mich schon ganz gut mit der Datenbank des ERP-Systems und dem DBSRV-Framework aus und so war es gut, dass ich nun mit einem neuen Programm auch selbst neue Strukturen aufbauen konnte. Inzwischen hatte ich die Probleme und Umständlichkeiten des Einkaufsinformationssystems kennengelernt und wollte an diesen Stellen das MontIS besser machen.

Ich nahm mir also Zeit um mir ein Grundgerüst zu bauen, das allgemein und skalierbar genug war um die einzelnen Reports dann ohne viel Anpassung der Programmbasis einbauen zu können. Auch habe ich zu diesem Zeitpunkt angefangen mir Klassen für wiederkehrende Anforderungen zu schreiben. Eine Datenbankklasse und eine zur Handhabung und Ausgabe von Ergebnistabellen seien hier erwähnt.

An denen Punkten meiner neuen Struktur, an denen ich keine überzeugende Lösung finden konnte, habe ich auf die alten Strukturen aus dem EIS zurückgegriffen.

So konnte ich kreativ neue Ansätze einarbeiten, hatte jedoch jederzeit auch eine funktionierende (wenn auch nicht immer gute) Implementierung als Fallback zur Verfügung. Mein Montageinformationssystem wurde dann zu eben jener Mischung aus Neuem und Altem und entspricht der Entwicklung meiner Fähigkeiten.

Auch bot das MontIS eine bessere Übersicht und Handhabbarkeit die das, meiner Meinung nach, etwas überladene Einkaufsinformationssystem missen ließ.

Das Fehlteilmanagement entspricht dabei in den meisten Punkten dem des EIS. Die Liefertreueanalyse dagegen möchte ich näher vorstellen.

7.1. Liefertreueanalyse

Die Optik und Struktur des Reports wurden sein Gegenstück aus dem Einkaufsinformationssystem angelehnt. Die Anpassungen waren eher Restrukturierungen des Source-Codes, und die andere Datenbasis natürlich.

Hier hatte ich nun auch Gelegenheit mich mit den Diagrammklassen des *jpgraph*-Pakets zu beschäftigen. Zwei damit erstellt Diagramme stellen auf der ersten Seite des Reports die grundsätzlichen Verteilungen auf einfach erfassbare Weise dar.

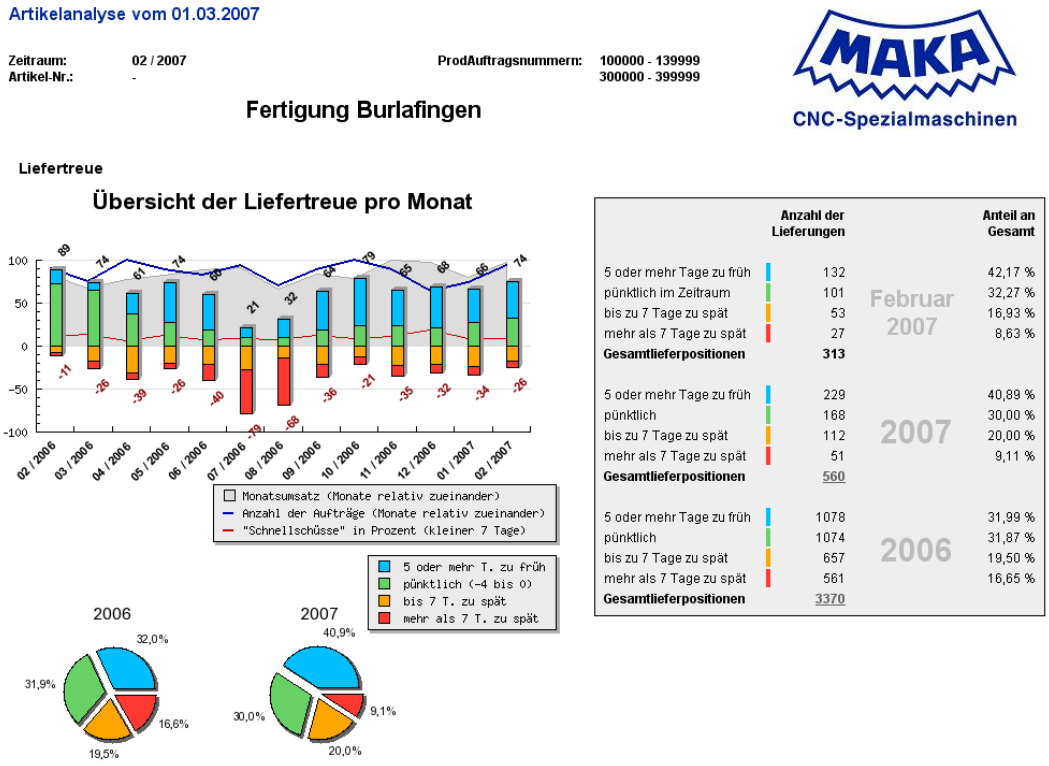


Abbildung 9: Liefertreueanalyse der Montage

Im linken oberen Bereich zeigt ein Balkendiagramm die Lieferdatumsabweichungen der Fertigung über die letzten 13 Monate an. Um eventuell Beeinflussungen durch die Auftragsmenge erkennen zu können wurde die Anzahl der Aufträge und deren Umsatz als Linien im Hintergrund ebenfalls dargestellt. Die flache rote Linie zeigt dabei die Anzahl der kurzfristig angelegten Aufträge die somit kaum geplant werden können.

Darunter befinden sich zwei Kuchendiagramme die Gesamtverteilungen für das laufende und vorherige Jahr anzeigen.

Auf der rechten Seite sind diese Werte noch einmal als Absolut- und Prozentzahlen verfügbar.

Die Zeiträume in denen Lieferungen als “zu früh”, “pünktlich”, “zu spät” und “viel zu spät” gewertet werden, sind in einer Includedatei global für das ganze Programm festgelegt und werden in den Reports dann dynamisch von dort bezogen. Ich habe stets darauf geachtet *Magic Numbers*[6] zu vermeiden und solche Werte immer dynamisch und zentral zu beziehen.

Für das laufende und vorige Jahr ist je ein Drilldown verfügbar der zu einer Monatssummenübersicht führt. In dieser lassen sind nun ganz konkret alle einzelnen Lieferpünktlichkeiten und Monate aufschlüsseln - jede Kombination und die jeweiligen Summen.

Liefertreueanalyse vom 01.03.2007

Zeitraum: 2006
Artikel-Nr.: -

ProdAuftragsnummern: 100000 - 139999, 300000 - 399999

Fertigung Burlafingen

MAKA CNC-Spezialmaschinen

Monat	5 oder mehr Tage zu früh		pünktlich geliefert		bis 7 Tage zu spät		mehr als 7 Tage zu spät		Summe
Januar	48	(22,12%)	129	(59,45%)	29	(13,36%)	11	(5,07%)	217
Februar	48	(16,00%)	218	(72,67%)	24	(8,00%)	10	(3,33%)	300
März	22	(8,80%)	162	(64,80%)	43	(17,20%)	23	(9,20%)	250
April	80	(24,02%)	123	(36,94%)	106	(31,83%)	24	(7,21%)	333
Mai	139	(46,49%)	81	(27,09%)	60	(20,07%)	19	(6,35%)	299
Juni	114	(41,16%)	53	(19,13%)	58	(20,94%)	52	(18,77%)	277
Juli	37	(11,78%)	30	(9,55%)	85	(27,07%)	162	(51,59%)	314
August	51	(21,61%)	24	(10,17%)	33	(13,98%)	128	(54,24%)	236
September	133	(44,78%)	56	(18,86%)	63	(21,21%)	45	(15,15%)	297
Oktober	184	(54,93%)	81	(24,18%)	43	(12,84%)	27	(8,06%)	335
November	122	(40,94%)	71	(23,83%)	68	(22,82%)	37	(12,42%)	298
Dezember	100	(46,73%)	46	(21,50%)	45	(21,03%)	23	(10,75%)	214
	1078	(31,99%)	1074	(31,87%)	657	(19,50%)	561	(16,65%)	3370

Abbildung 10: Monatssummendrilldown der Liefertreueanalyse

Jede unterstrichene Zahl führt weiter auf einen Drilldown mit genau diesen Aufträgen.

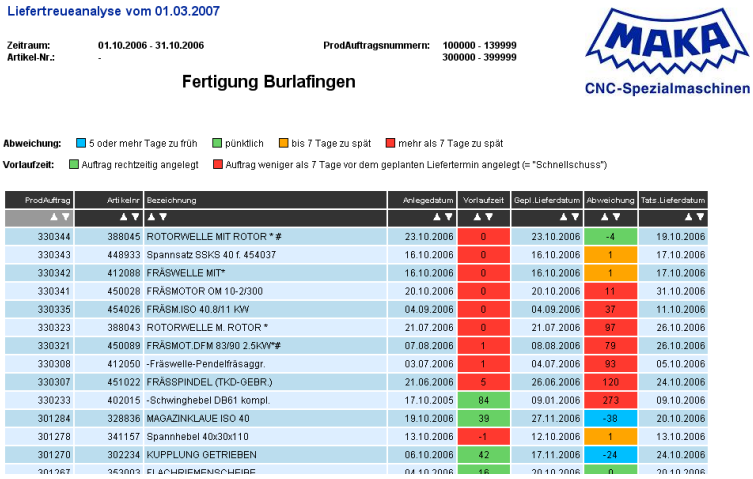
Man sieht die Vorlaufzeit[7] welche rot hinterlegt ist, falls sie kürzer als sieben Tage ist. In diesem Fall würde man sie als “Schnellschüsse” bezeichnen. Diese Schnell-

[6] *Magic Numbers* sind hier feste Zahlen im Code die nicht 0 oder 1 sind

[7] Hier die Zeit zwischen Anlagedatum und geplantem Lieferdatum. Normalerweise würde man aber nur die Zeit von Anlage bis zum geplanten Montagebeginn als Vorlaufzeit bezeichnen

schüsse entsprechen der, im Balkendiagramm auf der ersten Reportseite enthaltenen, roten Linie.

Der zweite entscheidende Wert ist die Abweichung zwischen geplantem und tatsächlichem Lieferdatum, die mit den bekannten Farben in die vier Zeiträume aufgeteilt ist.



Liefertreueanalyse vom 01.03.2007

Zeitraum: 01.10.2006 - 31.10.2006
Artikel-Nr.: -
ProdAuftragsnummern: 100000 - 139999, 300000 - 399999

Fertigung Burlafingen

MAKA CNC-Spezialmaschinen

Abweichung: 5 oder mehr Tage zu früh | pünktlich | bis 7 Tage zu spät | mehr als 7 Tage zu spät
Vorlaufzeit: Auftrag rechtzeitig angelegt | Auftrag weniger als 7 Tage vor dem geplanten Liefertermin angelegt (= "Schnellschuss")

ProdAuftrag	Artikelnr	Bezeichnung	Anlegedatum	Vorlaufzeit	Gepl. Lieferdatum	Abweichung	Tats. Lieferdatum
330344	388045	ROTORWELLE MIT ROTOR * #	23.10.2006	0	23.10.2006	-4	19.10.2006
330343	448933	Spannsatz SSKS 40 f. 454037	16.10.2006	0	16.10.2006	1	17.10.2006
330342	412088	FRÄSWELLE MIT*	16.10.2006	0	16.10.2006	1	17.10.2006
330341	450028	FRÄSMOTOR ÖM 10-2/300	20.10.2006	0	20.10.2006	11	31.10.2006
330335	454026	FRÄSM.ISO 40.8/11 kW	04.09.2006	0	04.09.2006	37	11.10.2006
330323	388043	ROTORWELLE M. ROTOR *	21.07.2006	0	21.07.2006	97	26.10.2006
330321	450089	FRÄSMOT.DFM 83/90 2.5kW*#	07.08.2006	1	08.08.2006	79	26.10.2006
330308	412050	-Fräswelle-Pendelfräsaggr.	03.07.2006	1	04.07.2006	93	05.10.2006
330307	451022	FRÄSSPINDEL (TKD-GEBR.)	21.06.2006	5	26.06.2006	120	24.10.2006
330233	402015	-Schwinghebel DB61 kompl.	17.10.2005	84	09.01.2006	273	09.10.2006
301284	328836	MAGAZINKLAUE ISO 40	19.10.2006	39	27.11.2006	-38	20.10.2006
301278	341157	Spannhebel 40x30x110	13.10.2006	-1	12.10.2006	1	13.10.2006
301270	302234	KUPPLUNG GETRIEBEN	06.10.2006	42	17.11.2006	-24	24.10.2006
301267	353003	FLACHRIEMENSCHEIBE	04.10.2006	16	20.10.2006	0	20.10.2006

Abbildung 11: Detaillierter Drilldown auf einzelne Aufträge

Wie alle größeren Tabellen in meinen Programmen wurde diese mithilfe meiner selbst geschriebenen `Ergebnisklasse.class.php` gefüllt, sortiert und ausgegeben. Wie man an den Pfeilen in der zweiten Titelzeile sieht, kann diese Tabelle nach jeder beliebigen Spalte sortiert werden. Diese Funktionalität war dank der Klasse mit minimalem Mehraufwand möglich und wurde deshalb auch konsequent verwendet.

7.2. Suchprofile

Wie bereits in einem anderen Programm auf dem DBSRV implementiert, sollte auch das Montageinformationssystem *Suchprofile* zur Verfügung stellen. Dabei handelt es sich um die Möglichkeit eine ausgefüllte Maske abzuspeichern und jederzeit wieder aufzurufen. So können bestimmte Reports ohne erneutes Ausfüllen der Maske nochmal aufgerufen werden. Ein Beispiel dafür wäre eine Abfrage die sich nur auf eine bestimmte Untermenge von Produktionsaufträgen und Artikelnummern bezieht.

Als der Wunsch aufkam dieses Feature ins MontIS einzubauen, war es für mich ein tolles Gefühl festzustellen, dass sich meine Grundstruktur nun auszahlte. In ein paar Stunden war es mir möglich die Funktionalität komplett einzubauen ohne größere Änderungen am Programm vornehmen zu müssen. Gleichzeitig konnte

ich die Realisierung flexibel und skalierbar halten. So ist es nun möglich beliebig neue Enschränkungsfelder zur Maske hinzuzufügen oder wegzunehmen — an den Suchprofilen muss dazu nichts verändert werden. Ebenso funktionieren auch alte gespeicherte Profile mit neuen Masken.

Würde man die gleiche Funktionalität im Einkaufsinformationssystem einbauen möchten, so würde es wohl mehrere Tage in Anspruch nehmen und wäre doch deutlich statischer.

Abbildung 12: Menü für Suchprofile

Nach Abschluss des Einkaufs- und des Montageinformationssystems folgte ein komplett anderes Projekt: Die Reimplementierung der Translator-Datenbank in PHP und MySQL.

8. Die Translator-Datenbank

MAKA ist ein Unternehmen, das seine Maschinen europaweit (und einzeln auch in alle Welt) verkauft. Per Gesetz muss die bei der Maschine mitgelieferte Dokumentation in einer landestypischen Sprache verfasst sein. Für die Stückliste, als Teil der Dokumentation gilt das somit auch. Nun sind Stücklisten bei zwei Maschinen quasi nie identisch, aber auch selten komplett verschieden. Komplette Neuübersetzungen wären also Zeit- und Geldverschwendung, Wiederverwendbarkeit von Übersetzungen ist aber auch nur auf Artikelebene gegeben.

Aus diesen Gründen wurde schon vor einigen Jahren ein Access-Programm geschrieben um sprachlich angepasste Stücklisten generieren zu können. Es konnten Übersetzungen eingetragen und dann fertige Stücklisten gedruckt werden. Dieses Programm hatte gerade bei der Performance und durch die nötige Sonderbehandlung von speziellen Baugruppen so seine Probleme. Nun war es meine Aufgabe dieses Programm in PHP und MySQL neu zu schreiben und in das DBSRV-Projekt einzugliedern.

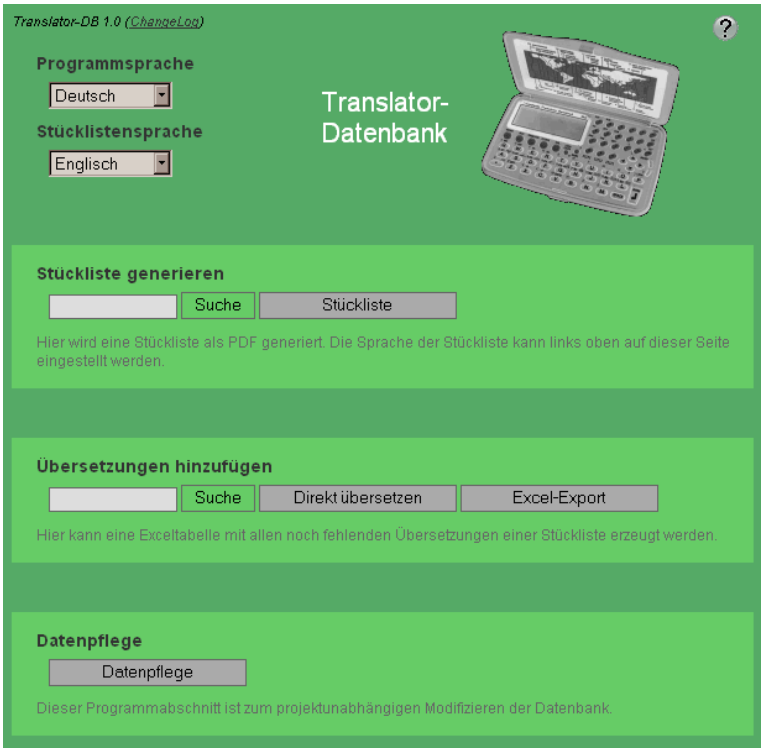


Abbildung 13: Startmaske der Translator-Datenbank

Nachdem ich im Montageinformationssystem schon einige grundsätzliche Überlegungen zur Grundstruktur des Programms umsetzten konnte, aber dennoch noch stark an die vom Einkaufsinformationssystem vorgegebene Struktur angelehnt war,

hatte ich nun mit der Translator-Datenbank die Möglichkeit ein Programm komplett auf ein selbst designtes Strukturkonzept aufzubauen. Diese Möglichkeit war mir sehr wichtig, da ich mir über Programmdesignfragen viele Gedanken mache und meine Überlegungen dazu natürlich auch in der Praxis erproben möchte. Zudem ist es ein unbefriedigendes Gefühl mit konzeptionell schlechten Programmstrukturen zu arbeiten. Oft finden sich die Unzulänglichkeiten von Konzepten halt erst bei ihrem Einsatz. Sie dann umzustellen erfordert (auch bei Extreme Programming) immer großen Refactoring-Aufwand. So läuft es meist darauf hinaus, dass man an der gegebenen, nicht optimalen, aber dennoch brauchbaren Programmbasis festhält. Wichtig ist dann halt, dass man beim nächsten Programmdesign die Probleme des Vorhergehenden zu eliminieren versucht.

Ich hatte nun die Gelegenheit die Erfahrungen aus EIS und MontIS mit meinem sonstigen neu erworbenen Wissen zu verschmelzen und daraus das bestmögliche Grundgerüst für die Translator-Datenbank zu kreieren. Schön war dabei, dass ich mich gleichzeitig recht wenig um die eigentliche Funktion des Programms kümmern musste, da es ein Vorgängerprogramm als Muster gab und die Programmlogik nicht besonders komplex war. Eine perfekte Ausgangssituation!

Die verschiedenen Funktionen der Translator-Datenbank wurden in drei Hauptteile gegliedert.

- Stückliste generieren
  Hier kann eine Stückliste in der ausgewählten Sprache als PDF erzeugt werden.
- Stückliste übersetzen
  Erzeugt ein HTML-Formular mit allen noch fehlenden Übersetzungen eines Projekts, oder generiert eine Excel-Tabelle für externe Übersetzungsbüros.
- Datenpflege
  Ermöglicht die direkte Bearbeitung der Datenbank. Dies ist nur ausgewählten Personen gestattet.

Im Folgenden möchte ich nun einzelne Funktionen der Translator-Datenbank genauer erklären.

8.1. Mehrsprachigkeit

Eine zentrale Anforderung der Translator-Datenbank war ihre Mehrsprachigkeit. Da das Programm auch von den der Tochterfirma **MAKA-UK** und später eventuell auch von den anderen Auslandsvertretungen genutzt werden sollte, musste die Benutzeroberfläche in verschiedenen Sprachen anwählbar sein. Die Menge dieser Sprachen sollte dynamisch erweiterbar sein. Ebenso war eine beliebige Anzahl von Sprachen für Stücklistenübersetzungen gefordert.

Die Benutzeroberfläche wird mit Texten aus Sprachdateien versorgt, die mittels einer PHP-Klasse `Gettext.class.php` ausgelesen werden. Im Programmquellcode selbst stehen lediglich Textidentifikator die dem Objekt der Gettext-Klasse übergeben werden. Zurückgeliefert wird dann der betreffende Text in der Sprache mit der das Objekt initialisiert wurde. Das folgende Codebeispiel demonstriert die Verwendung:

```
<?php

  include 'Classes/Gettext.class.php';

  // the language for the translation  is taken from the session
  $gt = new Gettext($_SESSION['lang_gui']);

  echo $gt->_('hello_world');

?>
```

Ausgegeben wird dabei je nach eingestellter Sprache "Hallo Welt!", "Hello world!", oder was auch immer in der jeweiligen Sprachdatei für den Identifikator `hello_world` hinterlegt ist.

Fehlende Texte werden in der Form `##identifier##` zurückgegeben. Auf diese Weise bleibt das Programm jederzeit benutzbar, auch wenn Sprachdateien nicht komplett sind. Ebenso ist es möglich auch nicht vollständig übersetzte Stücklisten zu generieren; die fehlenden Bezeichnungen bleiben leer und können gegebenenfalls auch von Hand nachgetragen werden. Eine derartige Vorgehensweise ist keineswegs angestrebt, aber trotzdem ermöglicht um die reine Funktion des Programms zu erhalten.

Die Sprachen für die Programmoberfläche und für die Stücklisten kann im oberen linken Bereich der Startmaske gewählt werden. Ein Wechsel der Oberflächesprache wird sofort wirksam. Das heißt, die komplette Oberfläche ist sofort in der gewählten Sprache nutzbar. Die Sprachauswahl für die Oberfläche zeigt dabei die Sprachnamen in der jeweiligen Sprache selbst an, die Dropdownbox mit den Sprachen für die Stücklisten ist in der gewählten Oberflächensprache. Das Programmdesign sollte an jeder Stelle die erwartete Implementierung bieten — auch *Principle of least surprise* genannt.

8.2. Standardstücklisten

Um alte Maschinen auch nachträglich mit Stücklisten versorgen zu können, wurde diese inzwischen überholte Form der Stücklisten ebenfalls implementiert. Die Unterscheidung welcher Stücklistentyp erzeugt werden muss, geschieht automatisch

anhand der Projektnummer. Die Algorithmen um die Stücklisten aufzustellen unterscheiden sich grundlegend. Wo für neue Projektstücklisten flache Datenbankabfragen genügen müssen die Standardstücklisten rekursiv als Baum aufgebaut werden. Wobei keine Baugruppe mehrfach in der Stückliste vorkommen soll, statt dessen müssen nur die Mengen angepasst werden.

Bei der Vorgängerversion der Translator-Datenbank in Access war es noch Aufgabe des Anwenders, sich um all diese Sonderfälle zu kümmern, jetzt bekommt der Benutzer von all dem gar nichts mehr mit.

8.3. Datenpflege

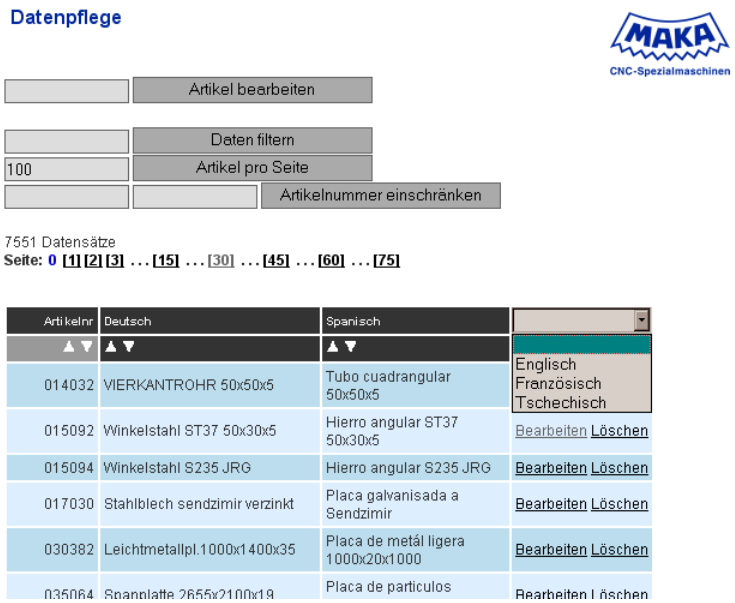


Abbildung 14: Datenpflege mit ausgeblendeten Spalten

Im Datenpflege-Fenster werden alle Datensätze der Datenbank aufgelistet. Um die vielen Datensätze handhabbar zu halten, wurde eine seitenweise Darstellung gewählt. Die Navigation um durch die Seiten zu blättern, zeigt nur eine sinnvolle Anzahl von Seitenzahlen an. Die Tabellenspalten zeigen die in der Datenbank hinterlegten Spalten an. Um die Übersichtlichkeit zu fördern können einzelne Spalten aus und wieder eingeblendet werden, wie in Abbildung 14 zu sehen ist. Die Spalten werden dynamisch verwaltet, das heißt, es werden einfach alle Spalten deren Spaltenname einem bestimmten Schema entspricht als Sprachspalten angesehen und eingefügt.

Die Datensätze können nach Artikelnummer eingeschränkt oder nach einem Begriff gefiltert werden. Zudem kann die Tabelle natürlich nach allen Spalten sortiert werden.

Die einzelnen Datensätze können hier bearbeitet, gelöscht und neue Datensätze können angelegt werden.

8.3.1. Datensatz bearbeiten

Beim Speichern von Übersetzungen habe ich erneut stark darauf geachtet, dass das Programm automatisch die erwartete Funktion ausführt. So werden Datensätze nur überschrieben wenn lediglich Übersetzungen angepasst wurden. Besteht allerdings die Gefahr, dass unbeteiligte Daten überschrieben werden, was bei einer falsch getippten Artikelnummer der Fall wäre, dann werden beide Datensatzversionen gegenübergestellt und dem Benutzer die Entscheidung überlassen, welche Daten gespeichert werden sollen. Dies ist in Abbildung 15 zu sehen.

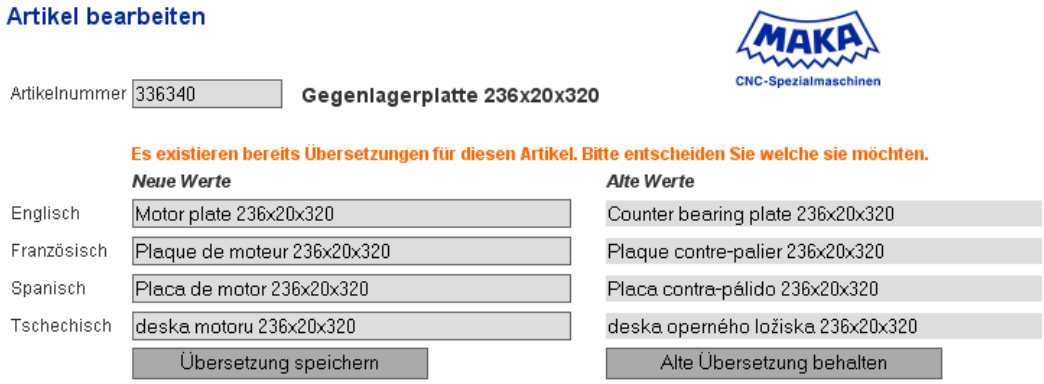
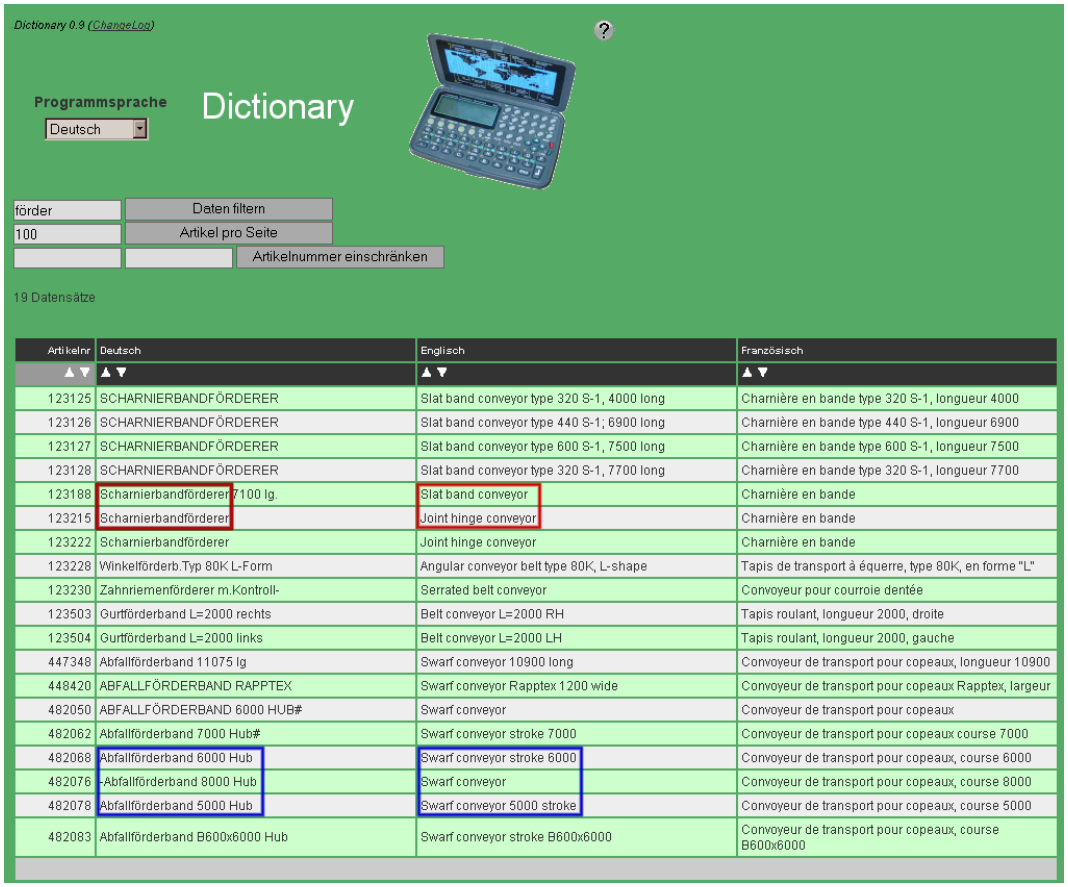


Abbildung 15: Zwei Übersetzungen kollidieren und der Benutzer muss entscheiden

8.4. Das Dictionary

Ein für den User eigenständiges Programm, aber intern doch direkt mit der Translator-Datenbank verbunden und auch aus ihr hervorgegangen, ist das *Dictionary*. Dieses ist eine Art Nachschlagewerk für übersetzte Artikelbezeichnungen und für alle Mitarbeiter zugänglich. Von einer simplen Startmaske kommt man zu einer Ansicht die der Translator-Datenbank-Datenpflege nachempfunden ist. Hier wird auf den Datenbestand allerdings nur lesend zugegriffen. Sinn des Dictionary ist es, eine Unterstützung im internationalen Kontakt zu sein, um auf einheitliche Bezeichnungen zurückgreifen zu können.

Wie auch in der Datenpflege der Translator-Datenbank können die Datensätze beliebig sortiert, gefiltert und nach Artikelnummer eingeschränkt werden. Auch lassen sich einzelne Sprachspalten ein- und ausblenden um die Übersicht zu fördern.



Artikelnr	Deutsch	Englisch	Französisch
123125	SCHARNIERBANDFÖRDERER	Slat band conveyor type 320 S-1, 4000 long	Charnière en bande type 320 S-1, longueur 4000
123126	SCHARNIERBANDFÖRDERER	Slat band conveyor type 440 S-1; 6900 long	Charnière en bande type 440 S-1, longueur 6900
123127	SCHARNIERBANDFÖRDERER	Slat band conveyor type 600 S-1, 7500 long	Charnière en bande type 600 S-1, longueur 7500
123128	SCHARNIERBANDFÖRDERER	Slat band conveyor type 320 S-1, 7700 long	Charnière en bande type 320 S-1, longueur 7700
123188	Scharnierbandförderer 7100 lg.	Slat band conveyor	Charnière en bande
123215	Scharnierbandförderer	Joint hinge conveyor	Charnière en bande
123222	Scharnierbandförderer	Joint hinge conveyor	Charnière en bande
123228	Winkelförderb.Typ 80K L-Form	Angular conveyor belt type 80K, L-shape	Tapis de transport à équerre, type 80K, en forme "L"
123230	Zahnriemenförderer m.Kontroll-	Serrated belt conveyor	Convoyeur pour courroie dentée
123503	Gurtförderband L=2000 rechts	Belt conveyor L=2000 RH	Tapis roulant, longueur 2000, droite
123504	Gurtförderband L=2000 links	Belt conveyor L=2000 LH	Tapis roulant, longueur 2000, gauche
447348	Abfallförderband 11075 lg	Swarf conveyor 10900 long	Convoyeur de transport pour copeaux, longueur 10900
448420	ABFALLFÖRDERBAND RAPPTEX	Swarf conveyor Rapptex 1200 wide	Convoyeur de transport pour copeaux Rapptex, largeur
482050	ABFALLFÖRDERBAND 6000 HUB#	Swarf conveyor	Convoyeur de transport pour copeaux
482062	Abfallförderband 7000 Hub#	Swarf conveyor stroke 7000	Convoyeur de transport pour copeaux course 7000
482068	Abfallförderband 6000 Hub	Swarf conveyor stroke 6000	Convoyeur de transport pour copeaux, course 6000
482076	-Abfallförderband 8000 Hub	Swarf conveyor	Convoyeur de transport pour copeaux, course 8000
482078	Abfallförderband 5000 Hub	Swarf conveyor 5000 stroke	Convoyeur de transport pour copeaux, course 5000
482083	Abfallförderband B600x6000 Hub	Swarf conveyor stroke B600x6000	Convoyeur de transport pour copeaux, course B600x6000

Abbildung 16: Typische Ansicht des Dictionary mit markierten Unstimmigkeiten

Da es in Abbildung 16 sehr schön sichtbar ist, möchte ich an dieser Stelle eine weitere Möglichkeit der Datenpflege-Ansicht beschreiben: Indem man nach Sprachspalten sortiert, stehen gleiche Bezeichnungen untereinander. So fallen sehr schnell Unstimmigkeiten bei den Übersetzungen auf, die in Abbildung 16 mir farbigen Rahmen hervorgehoben wurden. Diese Abweichungen können dann nach und nach ausgemerzt werden.

9. Dokumentation

Die Dokumentation von Programmen gehört ebenso wie das Programmieren selbst zur Softwareentwicklung dazu. Ich habe meine Anwendungen natürlich dokumentiert. Dies war für mich zuerst einmal eine eher neue Sache, da ich mich zwar mit der Notwendigkeit von Dokumentationen schon beschäftigt, aber noch keine selbst geschrieben hatte. Bei der technischen Umsetzung hielt ich mich an die Form die meine Vorgänger eingeführt hatten: Microsoft Word-Dokumente. Diese wandelte ich mittels OpenOffice.org in ein PDF um.

Mir war es wichtig, den Zusatzaufwand für die Dokumentation möglichst gering zu halten. Dieses Bestreben gründet zum Einen auf die allgemeine Erfahrung, dass Anwender keine Dokumentation lesen, und zum Anderen auf meine Überzeugung, dass nur der Quellcode selbst vollständig, aktuell und detailliert genug ist und deshalb vor allem er aussagekräftig und übersichtlich gemacht werden muss. Die Dokumentation sollte meiner Meinung nach nur als Ergänzung zum Quellcode gesehen werden. Sie ist jedoch trotzdem wichtig und notwendig um dem Entwickler schnell und direkt die grundlegende Logik und die nicht offensichtlichen Implementierungen des Programms offenzulegen. Des weiteren sollte sie dem Anwender Zusatzinformationen bieten, falls dieser ein Bedürfnis nach ihnen hat. Die Dokumentation für die beiden Personengruppen zu zweiteilen halte ich nicht unbedingt für sinnvoll, da auf diese Weise der Aufwand und die Redundanz steigt und bei Dokumentationen im Umfang von etwa 15 Seiten sollten die gesuchten Informationen noch recht einfach herauszufiltern sein.

Meine Dokumentation enthält somit zum größten Teil die hier aufgezählten Informationen:

- Grundsätzliche Informationen zur technischen Struktur des Programms
- Programmdesign-Prinzipien
- Kurzbeschreibung der einzelnen Programmteile und genauere Erläuterungen zu einzelnen Funktionen
- Tipps, Hinweise und Anmerkungen aller Art
- Konkrete Anleitungen falls nötig (Bei der Translator-Datenbank ist das zum Beispiel: “So füge ich eine Sprache hinzu”)

Was für mich nicht, oder nur in geringem Maße, in der Dokumentation vertreten sein sollte, sind:

- Zu detaillierte Informationen
- Variablen-, Klassen-, Dateinamen und ähnliches

Diese Informationen sorgen doch nur für einen allzu hohen Aufwand zum Aktuellhalten der Dokumente. Um auf derartige Zusatzinformationen aber nicht verzichten zu müssen, sollte man hier automatische Tools einsetzten. Ich habe an dieser Stelle das Programm *Doxygen*[8] eingesetzt. Die Ergebnisse für Klassen waren dabei sehr gut, der größte Teil der sonstigen Informationen eher weniger brauchbar. Der Aufwand für Doxygen-Kommentare im Quellcode ist gering und dient der Übersichtlichkeit des Codes selbst. Das Generieren der Doxygen-Dokumentation erfordert dann nur einen Programmaufruf, der Rest erfolgt automatisch.

Wie man im Text sicher schon heraushören konnte, habe ich keine spezielle Anwenderdokumentation in Form von einem Handbuch oder ähnlichem geschrieben. Wir erinnern uns: Der Benutzer ließt dieselbige ja doch nicht.[9] Deshalb sollte man dem Anwender die zusätzlichen Informationen zum Programm auf eine Weise anbieten, sodass er möglichst wenig davon mitbekommt und schon gar keinen zusätzlichen Aufwand hat.

Diesen Überlegungen bin ich gefolgt und habe als erstes versucht meine Programmoberflächen möglichst natürlich zu gestalten. Der Benutzer sollte alles dort finden wo er es erwartet. Konzepte wie "von links nach rechts" bzw. "von oben nach unten", "Gleiches zusammen" und "wiederkehrende Elemente" machen hierbei den größten Teil aus. Alle meine Reports sollten ein *Common Look'n'Feel*[10] haben.

Als wichtige Komponente, beim Anbieten von Zusatzinformationen ohne separates Handbuch, stellte sich dabei das Universalattribut `title` von HTML-Tags heraus. Mit ihm ist es möglich beliebigen Elementen einen Tooltip-Text zuzuweisen. Fährt man dann im HTML-Dokument mit dem Mauszeiger über das Element und verweilt kurz darauf, so öffnet sich ein kleines Informationsfenster mit dem hinterlegten Text. Diese Möglichkeit habe ich konsequent genutzt. So können zum Beispiel bei allen Tabellen durch überfahren der Spaltenüberschriften genauere Informationen zu den enthaltenen Werten gewonnen werden.

Für diejenigen Benutzer die trotz meiner Bemühungen um ein selbsterklärendes Oberflächendesign gerne die Dokumentation lesen möchten, habe ich diese auf den Startmasken der einzelnen Programme jeweils verlinkt. Ein Fragezeichen in der rechten oberen Ecke führt einheitlich zur Dokumentation in PDF-Form.

[8] *Doxygen* ist freie Software und kann von der Website http://doxygen.org heruntergeladen werden.
[9] Diese Grundannahme sollte mit ein wenig Humor verstanden werden.
[10] Eine einheitliche Optik und Bedienung

Teil III.
Abschließend

10. Fazit

Mein primärer Grund mich bei **MAKA** zu bewerben, war das Aufgabenprofil der ausgeschriebenen Stelle:

> Intranetprogrammierung mit PHP und MySQL

Da ich privat diese Technologie schon seit einigen Jahren verwende, war es mir jetzt wichtig, durch viel praktische Anwendung meine Kenntnisse zu festigen und auszubauen. Wo mein PHP-Können schon recht gut war, gingen meine (My)SQL-Fähigkeiten nur wenig über Grundkenntnisse hinaus.

Diese Erwartung an das Praxissemester wurde voll erfüllt. Meine PHP-Kenntnisse haben sich stark gefestigt, der Umgang mit der Scriptsprache ist nun solide und sicher. Aber was mir noch wichtiger ist: Die vorhandenen Kenntnisse haben sich nicht nur eingefahren, sondern ich habe sie auch konsequent ausbauen können. Das geht vom Erlernen der objektorientierten Bestandteile von PHP, über das Entwerfen von Standardklassen für die tägliche Arbeit (z.B. eine Datenbankklasse), bis hin zum *Verstehen* der Sprache an sich und ihrer Grenzen.

Ich habe das Gefühl, dass ich PHP wirklich kennen gelernt habe und dass sich mein Kenntnisniveau deutlich gehoben hat.

Als zweiten großen Part war die Datenbanksprache SQL, in ihrer Unterart MySQL, vertreten. Meine zu Beginn eher grundsätzlichen Kenntnisse erfuhren einen richtigen Sprung nach vorne. Bald schon waren umfangreiche Joins kein Problem mehr und wenig später formulierte ich sie mit sicherem Gefühl. Der Sprachumfang von SQL ist bekanntermaßen nicht die Schwierigkeit, welche dagegen im Planen des besten Weges zu den richtigen Daten zu finden ist. Gerade an dieser Stelle habe ich durch die viele Übung einen Erfahrungsschatz ansammeln können, der durch Themen wie Indizierung und Queryoptimierung sinnvoll ergänzt wird.

Abseits der Programmiersprachen habe ich die Gelegenheit genutzt, um auch meinen Workflow zu optimieren. Was diesen Bereich angeht, so ließ mir **MAKA** hier viele Freiheiten, um die ich sehr froh war. Denn nur dadurch konnte ich *meinen* Weg finden und einen Arbeitsablauf entwerfen, der mir entspricht.

Leider stand mir kein UNIX-System zur Verfügung, aber das wäre auch nur die absolute Krönung gewesen. Dennoch habe ich die Möglichkeit genutzt, den *Vim*[11]

[11] Der *Vim* ist ein verbesserter *vi* welcher unter UNIX-Systemen zum Standard gehört. Der *Vim* ist freie Software und kann unter http://vim.org heruntergeladen werden.

als Editor zu verwenden. Vor diesen sechs Monaten hatte ich ihn nur hin und wieder eingesetzt, jetzt kann ich ohne ihn nicht mehr leben! ;-)[12]

Nicht zu vergessen ist jedoch das Wirtschaftswissen, das ich mir während der meiner Zeit bei **MAKA** aufgenommen habe. Bereitwillig und von sich aus wurden mir die Vorgänge im Unternehmen erklärt. Immer wieder wurde wurden mir Gelegenheiten angeboten, die internen Abläufe zu verstehen. An vielen Stellen konnte ich dabei Verbindungen zu den Vorlesungen der ersten Semester herstellen.

Für mich war dieses Praxissemester eine große Bereicherung und die Erfahrungen die ich in diesen sechs Monaten gemacht habe, werden mir in meinem Studium mit Sicherheit von großem Nutzen sein.

Ich möchte mich dafür herzlich bei der Firma *MAKA - Max Mayer Maschinenbau GmbH* bedanken — es war eine tolle Zeit!

Markus Schnalke
Breitingen, den 11. März 2007

[12] hier soll mir ein Smiley erlaubt sein *grins*

Abbildungsverzeichnis

Milestones

Datum	**Event**
04.09.2006	Praktikumsbeginn
06.09.2006	Besprechung EIS-Tagesgeschäft
10.10.2006	Präsentations EIS-Tagesgeschäft
11.10.2006	Besprechung EIS-Fehlteilmanagement
09.11.2006	Präsentation EIS-Fehlteilmanagement
15.01.2007	Besprechung Translator-Datenbank
09.02.2007	Diskussion Translator-Datenbank
09.02.2007	Präsentation MontIS
21.02.2007	Präsentation Translator-Datenbank
02.03.2007	Praktikumsende