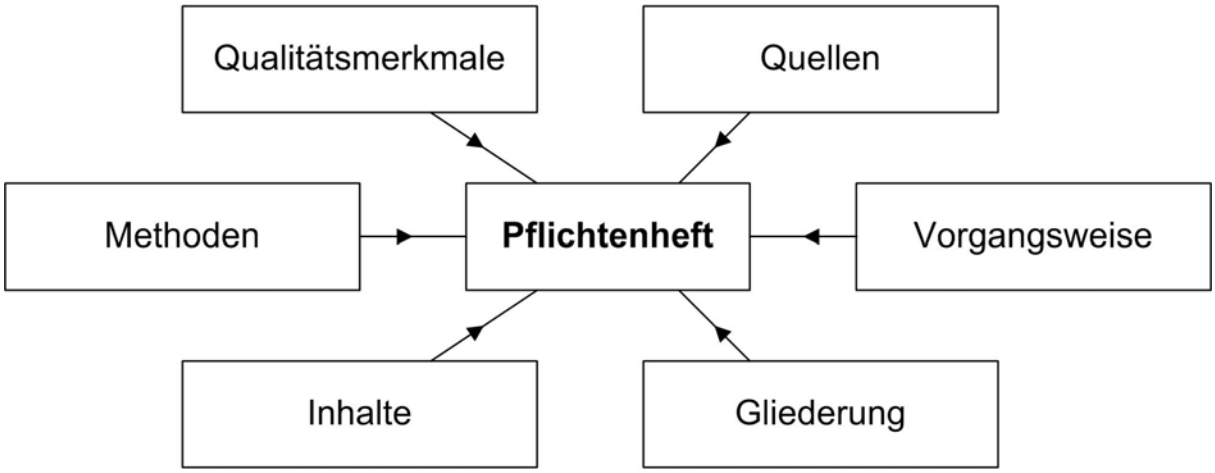
Anleitung zum Pflichtenheft

Das Pflichtenheft ist die zentrale Übereinkunft zwischen dem Auftraggeber und den Personen, die das jeweilige Softwareprodukt realisieren. Es dient dazu, die Grenzen abzustecken (was muss die Software können, was ist nicht erwünscht, usw.) und allen Beteiligten eine möglichst einheitliche Vorstellung vom Endresultat zu geben. Darüber hinaus kommt dem Pflichtenheft auch eine rechtliche Bedeutung zu, ist es doch in vielen Fällen die Grundlage von Vereinbarungen und oft sogar Vertragsbestandteil.

Als Vereinbarungskern zwischen Auftraggeber und Entwicklungsteam soll es Missverständnisse (und die daraus resultierenden Probleme und Verzögerungen) so weit wie möglich von vorneherein eliminieren. Diese Aufgabe kann ein Pflichtenheft aber nur dann erfüllen, wenn es in professioneller Weise erstellt wurde. Ein paar Zettel, auf denen die gewünschten Funktionalitäten in allgemeiner Weise beschrieben sind, *genügen in der Regel nicht*. Zu einem soliden Ergebnis wird man nur dann kommen, wenn man einen bestimmten Weg einhält. Die wesentlichen Eckpfeiler dieses Weges sind in dem vorliegenden Dokument beschrieben:



In weiterer Folge beschäftigen wir uns mit den einzelnen Eckpfeilern im Detail und geben praktische Hinweise für die Anwendung. Im letzten Abschnitt des Dokuments finden Sie einen Gliederungsvorschlag, den Sie für die Erstellung Ihres Pflichtenheftes verwenden können.

1. Aufgaben eines Pflichtenheftes

Am Anfang jedes Softwareentwicklungs-Projekts steht der Auftraggeber mit einer bestimmten Vorstellung – der Produktidee. Inhalt dieser Vorstellung sind Funktionsweisen, Arbeitsabläufe und Userinterfaces der gewünschten Software. Zwischen diesem Zeitpunkt und dem Start des technischen Designs und der eigentlichen Implementierung liegt ein steiniger und langer Weg, von dem letztendlich der gesamte Qualitätsumfang des gewünschten Produktes abhängt: Die Definitionsphase (*software requirements definition*).

Falls in dieser Phase des Softwareentwicklungs-Prozesses Funktionen oder bestimmte Qualitätsmerkmale entweder gar nicht oder nur schlecht definiert werden, führt dies zwangsläufig zu einem Produkt, mit dem der Kunde letztendlich unzufrieden ist. Um diese Gefahr weitgehend auszuschließen bedient man sich in der Definitionsphase verschiedenster formaler Methoden, von denen einige in Folge beschrieben werden.

Am Ende der Definitionsphase steht das Pflichtenheft als formale Grundlage des gesamten Projektes. Das Pflichtenheft fasst alle fachlichen Anforderungen des Auftraggebers an ein zu realisierendes Softwareprodukt zusammen. Dabei sollte stets das „Was“ und nicht das „Wie“ im Vordergrund stehen.

Das Pflichtenheft bietet in weiterer Folge die Ausgangsposition für:

- Angebotslegung
- Auftragserteilung
- Technisches Design
- Abnahme des Endproduktes

2. Quellen in der Definitionsphase

Je nach Situation ist zu unterscheiden, welche Gesprächspartner in der Definitionsphase in Frage kommen. Im Grunde sind zwei Fälle zu unterscheiden: Die Entwicklung von Individualsoftware bzw. von Standardsoftware.

Bei der Entwicklung von **Individualsoftware** werden die Vorgaben für das Softwareprojekt vom jeweiligen Auftraggeber kommen. Falls in der Definitionsphase bereits absehbar ist, welche Mitarbeiter nach der Fertigstellung das Produkt benutzen werden, sollten diese späteren User mit in den Definitionsprozess einbezogen werden. Die Beteiligung von Benutzern hat sich in der Praxis sehr bewährt, weil dadurch die Akzeptanz im Betrieb erhöht wird.
Eine in der Praxis häufig anzutreffende Situation besteht darin, dass in einem Unternehmen eine interne Softwareabteilung Software für die Fachabteilungen des Unternehmens erstellt. In diesem Fall werden die Anforderungen von der Fachabteilung gestellt, deren Mitarbeiter auch die Benutzer und Anwender sind.

Bei der Entwicklung von **Standardsoftware** für den „anonymen Markt“ werden die Anforderungen von Marketing oder Produktmanagement vorgegeben. Eine Benutzerbeteiligung lässt sich nur realisieren, falls sich Pilotkunden für das Projekt gewinnen lassen.

3. Art der Informationsgewinnung

In der Regel wird sich der zuständige Systemanalytiker aktiv um die Ermittlung der Anforderungen kümmern. Die meist fruchtbarste Art der Informationsgewinnung ist das Abhalten von Intensiv-Workshops gemeinsam mit den Beteiligten. Ebenso können in dieser Phase bereits fertige (niedergeschriebene) Anforderungsspezifikationen diskutiert und übergeben werden.

Man darf nie aus dem Auge verlieren, dass der Informationsgewinnungs-Prozess von *iterativer Natur* ist. Nach der Aufnahme von Anforderungen in den Workshops werden diese niedergeschrieben und einer Revision unterzogen, bei welcher überprüft wird, ob die Anforderungen bestimmten Qualitätszielen entsprechen.

4. Qualitätsanforderungen

Für eine Anforderungsspezifikation wie ein Pflichtenheft bestehen folgende anstrebenswerte Qualitätsmerkmale:

- Vollständigkeit
- Konsistenz
- Eindeutigkeit
- Durchführbarkeit

Der Punkt der **Vollständigkeit** beschreibt den Grad, in dem das Produkt dem Benutzer alle gewünschten Funktionen und Daten zur Verfügung stellt, um die geforderten Produktziele zu erreichen. Wird bei einer Revision der Anforderungen festgestellt, dass bestimmte Funktionen noch fehlen, so müssen sie im oben beschriebenen iterativen Prozess neu erarbeitet werden.

Konsistenz beschreibt den Grad, in dem die definierten Anforderungen untereinander widerspruchsfrei sind. Ein Beispiel hierfür wäre eine Datenbankanwendung, für die global festgelegt wird, dass keine Datensätze gelöscht werden dürfen. Eine Inkonsistenz würde hier vorliegen, wenn in einem Submodul die Funktion „Löschen von Datensätzen“ spezifiziert wird.

Eine Anforderung kann man als **eindeutig** bezeichnen, wenn sie keinen Spielraum für alternative Interpretationen zulässt. Den Satz „Das Löschen von Datensätzen sollte möglicherweise unterbunden werden.“ kann man nicht als eindeutig bezeichnen. In der Praxis zeigt sich oft, dass verbal formulierte Anforderungen oft nicht eindeutig interpretiert werden, was zur Entwicklung formalisierter Beschreibungsmethoden geführt hat (siehe nächster Abschnitt).

Eine wesentliche Frage in der Anforderungsspezifikation ist die **Durchführbarkeit** der gewünschten Produktziele. Ein Beispiel hierfür wäre eine gewünschte nicht zu überschreitende Ladezeit einer Webpage. Kann die Durchführbarkeit nicht spontan überprüft werden, so kann man zum Mittel des Prototypings greifen, um zum Beispiel die Ladedauer eines Webpage-Prototyps zu testen. Bei der Prototypenerstellung ist darauf zu achten, dass diese so einfach wie nur möglich sind und dass die Funktionalität des Prototyps auf die zu testende beschränkt ist.

5. Inhalte und Methoden

Bei der Frage, was denn nun eigentlich definiert werden soll, muss man zunächst die verschiedenen Sichten auf ein System betrachten. Helmut Balzert beschreibt in seinem Buch „Lehrbuch der Software-Technik“ vier verschiedene Sichtweisen auf ein System:

- Daten (Datadictionary, ER-Diagramm)
- Funktionen (Klassendiagramm, Datenflussdiagramm, eventuell Pseudocode)
- Dynamik (Interaktionsdiagramm)
- Benutzeroberfläche

Die Bedeutung der einzelnen Sichten ist nicht bei jeder Anwendung gleich groß – es gibt Anwendungen, bei denen zum Beispiel eine oder zwei Sichten dominieren.

Die **Datensicht** beschäftigt sich mit der Anzahl und Komplexität der Datenstrukturen der zu erstellenden Anwendung. Es wird erhoben, welche Daten langfristig gehalten werden müssen und in welcher Form die Datenhaltung erfolgen soll. Beziehungen zwischen einzelnen Klassen von Daten werden mit graphischen Methoden dargestellt. Die gängigsten Methoden hierfür sind das Data Dictionary zur Darstellung der Datenstrukturen und die Entity Relationship-Methode zur Veranschaulichung der Entitäten mit ihren Beziehungen zueinander. Ian Sommerville beschäftigt sich in seinem Standardwerk „Software Engineering“ im Kapitel „Data Modeling“ eingehend mit diesen und mit anderen Methoden.

Für die **funktionale Sicht** kann man sich neben einer ausführlichen verbalen Beschreibung mit Datenflussdiagrammen, Funktionsbäumen bis hin zum Pseudocode bei besonders anspruchsvollen Algorithmen helfen. Die Verwendung der letztgenannten Methoden wird stark von der Komplexität der abzubildenden Funktionen abhängen.

Die Sicht auf die **Dynamik** eines Softwaresystems beschäftigt sich mit den zeitlichen Zusammenhängen und Übergängen zwischen den Zuständen des gewünschten Systems sowie mit der Abfolge und Steuerung von Prozessen und den definierten Zeitbedingungen in denen sie ablaufen sollen. Falls das zeitliche Verhalten des zu entwickelnden Softwareproduktes einen bestimmten Komplexitätsgrad übersteigt und eine verbale Beschreibung nicht mehr ausreicht, hilft man sich in der Praxis mit Zustandsdiagrammen, Petri-Netzen und Interaktionsdiagrammen. Diese und weitere Werkzeuge und Methoden werden im Buch „Lehrbuch der Software-Technik“ von Helmut Balzert ausführlich beschrieben.

Falls die Ansprüche an die **Benutzeroberfläche** eines Systems hoch sind, so wird man diese Sichtweise in die Feinspezifikation des Produktes aufnehmen. Zur Veranschaulichung der verbalen Beschreibung der Benutzeroberfläche kann man entweder mit Graphikeditoren oder mit modernen Entwicklungsumgebungen einfache Prototypen der Benutzeroberfläche erstellen.

6. Gliederung des Pflichtenheftes

Wurden die Anforderungen mit den oben beschriebenen Methoden erhoben, so folgt schließlich die Erstellung des Pflichtenheftes. Im Pflichtenheft werden alle fachlichen Anforderungen zusammengefasst, die das zu entwickelnde Softwareprodukt aus der Sicht des Auftraggebers erfüllen muss.

Das Pflichtenheft ist prinzipiell an den Auftragnehmer gerichtet, welcher durch den Projektleiter, den oder die Systemanalytiker und diejenigen, die das technische Design durchführen. Weitere Adressaten sind Personen aus dem Umfeld des Auftraggebers – z. B. potentielle Benutzer oder die interne Qualitätskontrolle.

Die zu spezifizierenden Anforderungen müssen in ausreichender Detaillierung und ausreichendem Präzisionsgrad beschrieben werden. Im Pflichtenheft soll festgelegt sein, *was* das Produkt – bezogen auf Funktionen und Leistungen – erfüllen soll, *nicht wie* es sie erfüllen soll.

In der Literatur finden sich viele Vorschläge zur Gliederung eines Pflichtenheftes, die sich im Großen und Ganzen nur wenig voneinander unterscheiden. Ferner gibt es das Gliederungsschema nach ANSI/IEEE Std 830-1984. In der Praxis hat es sich bewährt, prinzipiell von einem vorgegebenen Schema auszugehen und dieses je nach Art des zu entwickelnden Softwareproduktes und je nach Kundenwunsch zu modifizieren. Zum Beispiel gibt es Applikationen, bei denen das Thema „Systemsicherheit“ ein derart wichtiges ist, dass man ihm im Pflichtenheft ein eigenes Kapitel widmet, obwohl das im ANSI/IEEE Standard nicht vorgesehen ist.

Der am meisten praxisrelevante **Gliederungsvorschlag** eines Pflichtenheftes stammt von Helmut Balzert aus seinem Buch „Lehrbuch der Software-Technik“. Die Gliederung besteht aus elf Abschnitten, die wir hier im Einzelnen kommentieren:

1. Zielbestimmung
1.1 Musskriterien
1.2 Wunschkriterien
1.3 Abgrenzungskriterien

Zweck des Abschnitts „Zielbestimmung“ ist eine Art Einleitung, indem beschrieben wird, welche Ziele durch den Einsatz des Produktes erreicht werden sollen.

Unter den Musskriterien wird aufgeführt, welche Leistungen für das Produkt unabdingbar sind, damit es für den vorgesehenen Einsatzbereich verwendet werden kann. Diese Kriterien müssen auf jeden Fall erfüllt werden.

Wunschkriterien sind nicht unabdingbar aber so gut wie möglich anzustreben. Abgrenzungskriterien sollen deutlich machen, welche Ziele mit dem Produkt bewusst nicht erreicht werden sollen.

2. Produkteinsatz

2.1 Anwendungsbereiche
2.2 Zielgruppen
2.3 Betriebsbedingungen

Dieser Abschnitt soll sich mit der Frage beschäftigen, wo und für wen das Produkt eingesetzt werden soll.

Unter „Anwendungsbereiche“ beschreibt man in wenigen Worten den groben Zweck des zu entwickelnden Systems, während unter „Zielgruppen“ die Personen beschrieben werden, für die das zu entwickelnde System bestimmt ist. Hier mag es opportun sein, explizit anzugeben, für welche Anwendungsbereiche und Zielgruppen die Software *nicht* vorgesehen ist.

Unter den Betriebsbedingungen versteht man die physikalische Umgebung des Systems sowie Parameter wie zum Beispiel die tägliche Betriebszeit.

3. Produktumgebung

3.1 Software
3.2 Hardware
3.3 Produktschnittstellen

Gegenstand dieses Abschnitts ist die Beschreibung der Produktumgebung, wobei man unter dem Punkt Software angibt, welche Softwaresysteme (Betriebssysteme, Datenbanken,…) auf dem Zielsystem vorhanden sind.

Unter Hardware wird aufgeführt, welche Hardware-Komponenten (z. B. Clientrechner, Serverlandschaft,…) in welcher Konfiguration für den Produkteinsatz vorgesehen sind.

Falls das Produkt in eine bestehende oder geplante Produktfamilie eingegliedert werden soll, so müssen die dafür notwendigen Schnittstellen im Abschnitt Produktschnittstellen beschrieben werden.

4. Produktfunktionen

4.1 Funktion 1
4.2 Funktion 2 usw.

Hier erfolgt eine Auflistung der Funktionen aus Benutzersicht, die das gewünschte Produkt erfüllen soll. Die Gliederung sollt so viele Unterteilungen aufweisen, wie das Produkt Funktionen oder Funktionsbereiche aufweist.

Die Funktionen sollten unabhängig von einem bestimmten Bildschirmlayout beschreiben werden, da diese Festlegung erst in Abschnitt 7 getroffen wird.

5. Produktdaten

Dieser Abschnitt dient zur Beschreibung der langfristig zu haltenden Daten in Verbindung mit den oben erwähnten Methoden.

6. Produktleistungen

Unter Produktleistungen werden alle Anforderungen aufgeführt, die sich auf die Zeit oder auf den Umfang beziehen wie z.B. maximale Antwortzeiten, maximaler Datenumfang usw.

7. Benutzeroberfläche

Hier werden die grundsätzlichen Anforderungen an die Benutzeroberfläche festgelegt. Abhängig vom zu entwickelnden Produkt sollte Folgendes berücksichtigt werden:

- Bildschirmlayout
- Maskenlayout
- Drucklayout
- Tastaturbelegung
- Dialogstruktur

Sämtliche Festlegungen zum Thema Benutzeroberfläche sollten sich ausschließlich auf produktspezifische Ausprägungen beschränken. Fragen der Benutzerfreundlichkeit werden im folgenden Abschnitt behandelt.

8. Qualitäts-Zielbestimmungen

In diesem Abschnitt beschreibt man, welche Qualitätsmerkmale in welcher Qualitätsstufe anzustreben sind. Ein Beispiel hierfür ist der im vorangegangenen Abschnitt erwähnte Grad an Benutzerfreundlichkeit.

9. Globale Testszenarien und Testfälle

9.1 Testfall 1
9.2 Testfall 2 usw.

Hier werden die Testfälle pro Funktion aus den Funktionsanforderungen abgeleitet und aufgelistet. Diese Testfälle sind für den Abnahmetest zu verwenden.

10. Entwicklungsumgebung
10.1 Software
10.2 Hardware
10.3 Entwicklungsschnittstellen

In den meisten Fällen wird ein System nicht auf dem Zielcomputer entwickelt, sondern auf Entwicklungscomputern. In diesem Abschnitt wird die Software- und Hardwarekonfiguration dieser Entwicklungscomputer beschrieben. Mit Software sind hauptsächlich Entwicklungswerkzeuge und Compiler gemeint. Unter dem Punkt Entwicklungsschnittstellen ist anzuführen, über welche einzuhaltenden Hardware- und Softwareschnittstellen der Entwicklungs- und Zielrechner gekoppelt sind.

11. Ergänzungen

Dieser Abschnitt dient als Auffangnetz für alle speziellen Anforderungen, die in den Abschnitten eins bis zehn keinen Platz gefunden haben. Beispielsweise könnten hier besondere Installations- und Integrationsrichtlinien festgehalten sein, die unbedingt einzuhalten sind. Ebenso finden hier rechtliche Aspekte z.B. über Lizenzen oder Patente einen Platz.

Ergänzend zu diesem Text steht Ihnen auf der InfraSoft Website ein Glossar zur Verfügung, in dem Sie die meisten der hier verwendeten Begriffe finden. Über das aktuelle Angebot an weiteren, kostenlosen Fachbeiträgen zur Softwareentwicklung informieren Sie sich bitte unter www.infrasoft.at/service.

Mag. Andreas Fandl
Wien, im Oktober 2004

Der Autor ist Mitarbeiter der InfraSoft, einem Unternehmen, das auf komplexe Softwareentwicklungen spezialisiert ist. Die Experten der InfraSoft haben langjährige Erfahrungen in der Entwicklung und verfügen über fundierte Kenntnisse in Design, Analyse, Realisierung, Test und Projektmanagement.
Für **individuelle Beratungen** zur Entwicklung von Softwarelösungen und die Bereitstellung von **Realisierungsteams** wenden Sie sich bitte an info@infrasoft.at.