FVA	PROGRAMMIERRICHTLINIE	Forschungsvereinigung Antriebstechnik e.V.

Inhaltsverzeichnis

FVA-Softwarestrategie

1 Vision

Die FVA ist international die kompetenteste Organisation zur Entwicklung und Etablierung von Berechnungsmethoden (Referenzberechnung und Auslegung) im Bereich der Antriebstechnik zum Nutzen ihrer Mitglieder. Die FVA definiert und etabliert durch ihr Netzwerk den Stand der Technik in der Antriebstechnik weltweit und ist hierdurch Technologieführer auf diesem Gebiet.

2 Mission

Die FVA unterstützt die kontinuierliche Erweiterung der Berechnungsansätze bzw. die Erarbeitung neuer Ansätze durch die Entwicklungen in den Arbeitskreisen und den Arbeitsgruppen, die dazu dienen, die Wirtschaftlichkeit der Produkte der FVA-Mitglieder zu verbessern. Die FVA-Gremien unternehmen alles, um die Weiterentwicklung der Berechnungsansätze voranzutreiben. Weiterhin stellen die Arbeitsgruppen und Arbeitskreise die Zuverlässigkeit und die Qualitätssicherung der von ihnen entwickelten Berechnungsansätze sicher. Unter Berücksichtigung der speziellen Anforderungen der kleinen und mittelständischen Unternehmen als Anwendergruppe erfolgt die Entwicklung von neuen Berechnungsmöglichkeiten.

3 Zielposition

Als die führende Berechnungsplattform etabliert die FVA die FVA-Workbench global. Die FVA-Workbench ist das zentrale Instrument, um die Ergebnisse aus den Arbeitskreisen (Arbeitsgruppen) der Branche zugänglich zu machen. Zudem unterstützt und forciert die FVA die Entwicklung der FVA-Workbench als antriebstechnische Berechnungsplattform. Ziel ist es, dass die Ergebnisse von Forschungsprojekten als anwendbarer Programmbestandteil in der FVA-Workbench verfügbar sind. Die FVA-Workbench ist bei jeder FVA-Programmentwicklung zu unterstützen und zu stärken. Sie ist die Plattform, die es ermöglicht, die Berechnungsvielfalt der FVA zu nutzen und effizient anzuwenden. An Stellen, an denen der Einsatz der FVA-Workbench nicht möglich ist, sollten kommerzielle Programme eingesetzt werden, die es erlauben, Berechnungsmethoden auf dem Stand der Technik in die FVA-Forschungsarbeit einzubinden (MKS, FEM, CFD, BEM). Entscheidend für die FVA ist eine schnelle Realisierung von Softwarelösungen, die den Stand der Technik widerspiegeln.

4 Umsetzung

Möglichst alle Berechnungsansätze der FVA sind in einer der strategischen FVA-Berechnungsplattformen zu integrieren (FVA-Workbench bzw. SIMPACK). Die FVA konzentriert sich dabei auf die (Er-)Forschung von antriebstechnischen Themen und nicht auf deren softwaretechnische Realisierung. Die Umsetzung der FVA-Berechnungsansätze (Forschungsergebnisse) in eine breit anwendbare Software wird durch die FVA GmbH realisiert. Die FVA betreibt die internationale Verbreitung und Standardisierung ihrer Berechnungsansätze über die FVA GmbH. Durch Wartung, Support, Seminare und Qualitätssicherungsprozesse stellt die FVA GmbH die Betriebssicherheit der FVA-Programme langfristig sicher und stellt die FVA frei von Ansprüchen externer Dritter.

Hinweis: Um diese gemeinsamen Ziele zu erreichen, ist es notwendig, dass sich in der Organisation FVA e.V. alle an der Softwareentwicklung beteiligten Parteien an den in diesem Dokument beschriebenen Vorgehensweisen bzw. Richtlinien orientieren.

0 Allgemeine Einführung

Bei der Entwicklung von Berechnungsprogrammen auf Basis von Forschungsarbeiten wird in der FVA eine einheitliche Vorgehensweise und Strategie angewendet. Im Wesentlichen müssen hierbei zwei Alternativen in Betracht gezogen werden. Die FVA unterstützt zwei wesentliche Softwareplattformen in ihrer Berechnungswelt. Diese sind die FVA-Workbench (Eigenentwicklung der FVA) und die Simulationssoftware SIMPACK (eine Entwicklung der SIMPACK AG). Ziel des geplanten Vorhabens sollte immer die Ergänzung und/oder Erweiterung der Berechnungsmöglichkeiten dieser beiden Plattformen sein. Sollten von der projektbegleitenden Arbeitsgruppe andere Anforderungen an Sie herangetragen werden, so sind diese in einem offiziellen FVA-Protokoll zu dokumentieren.

1. Für die Entwicklung von Programmen gilt generell, dass diese als Stand-Alone Version lauffähig sein müssen und derart gestaltet sind, dass diese als Berechnungsmodul in die FVA-Workbench integriert werden können.

2. Im Bereich der dynamischen Mehrkörpersimulation hat die FVA als Standardplattform die Software SIMPACK definiert. Ob SIMPACK nun die geeignete Plattform für den Berechnungsansatz ist, muss aber in Zusammenarbeit mit der projektbegleitenden Arbeitsgruppe definiert werden. In jedem Fall ist es anzustreben, die einheitliche Plattform zu nutzen, um den Transfer der Ergebnisse in die Industrie zu erleichtern.

3. Für alle anderen Programme gelten Ausnahme- oder Sonderregelungen, die durch die Industrie in Abstimmung mit dem FVA-SoftwareService definiert werden.

In der Regel bestehen Forschungsprojekte (mit dem Ziel das Wissen in Form von Software anwendbar zu machen) immer aus zwei Themenblöcken. Zum einen den theoretischen Forschungsanteil, zum anderen den Teil in dem die Erkenntnisse in die Software implementiert werden. Diese Phasen sollten, insbesondere bei der Zeitplanung, getrennt betrachtet werden. Die im Folgenden beschriebenen Prozesse beziehen sich im Wesentlichen auf die Teile eines Forschungsprojektes, die sich mit der tatsächlichen Umsetzung der Ergebnisse in eine Software befassen. Diese sollen helfen die Arbeiten im Detail zu planen und das Projekt erfolgsorientiert abzuschließen.

1 Ablauf der Softwareerstellung in der FVA

1.1 Rahmenbedingungen der Antragsstellung

Für die Realisierung des Programmes sollten Sie die Projektphasen wie folgt abschätzen:

- Planung und Strukturierung 30% der Projektlaufzeit
- Realisierung 30% der Projektlaufzeit
- Testen 40% der Projektlaufzeit

Diese Werte sind Richtwerte, die bei der Programmentwicklung gängige Praxis sind.

Folgende Rahmenbedingungen sind für die Beantragung, Durchführung und den Abschluss von Vorhaben, mit dem Ziel ein Forschungsergebnis in ein Berechnungsprogramm zu transformieren, zu beachten:

1.1.1 Grundregeln

1.) Generell wird die Erstellung von grafischen Oberflächen durch Forschungsstellen von der FVA nicht finanziert.

2.) Zu jedem angestrebten Programm ist eine projektbegleitende Arbeitsgruppe zu bilden.
Die Arbeitsgruppe hat folgende Aufgaben:

   a. Definition der Anforderungen an das Programm
   b. Fachlicher Ansprechpartner für den durchführenden Sachbearbeiter
   c. Kontrollgremium, das die reibungslose Abarbeitung des Projektes überwacht und sicherstellt
   d. Qualitätssicherungsgremium, das sicherstellt, dass die im Rahmen des Projektes entwickelte Software den Anforderungen der Industrie genügt
   e. Die Abnahme des Programms, welches im Rahmen des Projektes entwickelt wird

3.) Die Integration des Programmes in eine grafische Oberfläche wird von der FVA nur im Rahmen der Einbindung in die FVA-Workbench unterstützt (Diese Aufgabe wird von der FVA GmbH wahrgenommen).

4.) Forschungsanträge, die in einem Projekt Forschungs- und Versuchsanteile haben, sowie die Erstellung eines Programms beinhalten, bestehen aus folgenden Unterlagen:

   a. Die Beschreibung der Forschungsarbeit an der Hochschule inkl. (Weiter-) Entwicklung eines FVA-Programms unter Berücksichtigung der Programmierrichtlinie.
   b. Definition der Datenschnittstellen von und zur FVA-Workbench
   c. Eine Spezifikation für die Integration in die FVA-Workbench. Dies erfolgt in Zusammenarbeit mit dem FVA-SoftwareService.

5.) Bei der Definition des Programmumfanges (der Funktionalität) ist die FVA-Workbench als Referenz heranzuziehen. Das zu entwickelnde Programm soll sich nahtlos in die vorhandene Betrachtung von Getriebesystemen einfügen. Hierbei ist immer die zum Zeitpunkt des Antrages aktuelle offizielle FVA-Workbench zugrunde zu legen (Datum und Angabe der Versionsnummer).

6.) Bei der Nomenklatur Ihrer Parametrisierung ist darauf zu achten, dass Sie nur Bezeichnungen verwenden, die in der FVA-Workbench zur Anwendung kommen. Es sollte unbedingt vermieden werden, dass gleiche physikalische Größen mit unterschiedlichen Formelbezeichnungen mehrfach in die FVA-Workbench implementiert werden.

7.) Der Namensraum für die Parametrisierung ist durch die FVA-Workbench vorgegeben. Abweichungen hiervon sind mit der FVA Softwareentwicklung abzusprechen.

8.) Neue Parameter oder Parameterbezeichnungen sind nur mit Genehmigung des FVA-SoftwareService in die FVA-Workbench einführbar.

9.) Die Daten und Datenstrukturen müssen sich in die Systematik (Produktmodell) der FVA-Workbench einfügen.

Bevor ein Forschungsantrag zu einem Projekt, mit dem Ziel der Softwareerstellung, entwickelt wird, sollte eine AG aus Industrievertretern und FVA-Workbench-Entwicklern stattgefunden haben. Diese soll Ihnen helfen, den Rahmen und die Funktionstiefe Ihres Programms besser definieren zu können. Dies wird sich auf die Bewilligung des Vorhabens positiv auswirken.

1.1.2 Verantwortlichkeiten

Der Antragsteller hat die komplette Projektverantwortung. D.h. er hat dafür Sorge zu tragen, dass die Parametrisierung (durch die grafische Oberfläche) sowie der Transfer zu dem eigentlichen Berechnungskern zuverlässig und fehlerfrei funktionieren (Schnittstellen).
Dies ist bei Abgabe des Programms zu bestätigen und durch genügend Rückmeldungen aus der industriellen Anwendung abzusichern (siehe auch Anlage "Abnahme der Bestätigung zur offiziellen Freigabe von FVA-Programmen").

1.1.3 Unterlagen

Forschungsarbeit
Die der Software zugrunde liegenden Forschungsarbeiten sind im Rahmen eines Forschungsantrages zu beschreiben.

Definition der Software auf Basis der vorangegangenen Forschungsarbeiten
Die Software, die auf Basis der Forschungsarbeiten entwickelt werden soll, ist in Form einer Teilspezifikation zu beschreiben. Die Teilspezifikation gliedert sich in zwei Bereiche. Die Forschungsstelle hat eine der beiden Teilspezifikationen zu erstellen, die das Gesamtsystem Software beschreibt. Hierbei sollen insbesondere die Berechnungsfunktionalitäten beschrieben werden sowie die Schnittstellen zur grafischen Benutzeroberfläche. In Zusammenarbeit mit den von der FVA beauftragten Dienstleistern erstellen diese eine zweite Teilspezifikation, die den Leistungsteil der Oberfläche beschreibt und definiert.

1.1.3.1 Projekt und Terminplan

Für die Entwicklung der Software ist ein Termin- und Zeitplan zu erstellen, der auch mit der Entwicklung der Oberfläche abgestimmt ist. Dieser soll konkrete Zeitpunkte beinhalten, an denen ein Ergebnis der jeweiligen Partei vorliegen muss. Ebenso ist anzugeben, wann ein Review der Arbeiten erfolgen kann.

1.1.3.2 Abstimmung des Projektes

Alle Projekte, die Software zum Inhalt haben, sind mit dem FVA-SoftwareService abzustimmen. Bei der Antragserstellung sind notwendige Erweiterungen der FVA-Workbench zur Einbindung und Ansteuerung des zu erstellenden Programms detailliert zu definieren. Der Arbeitsaufwand auf Seiten der FVA-Workbench-Entwicklung ist mit dem FVA-SoftwareService abzustimmen. Die Zeitplanung muss entsprechende Zeiträume zur Implementierung, dem Testen und der Freigabe der Programms und der FVA-Workbench enthalten. Die für die FVA-Workbench-Entwicklung erforderlichen Aufwendungen sind zusätzlich anzugeben.

Das Konzept und der fertige Antrag von Forschungsvorhaben mit Softwareentwicklung sind zwei Wochen vor der begutachtenden Arbeitskreissitzung dem FVA-SoftwareService mitzuteilen und vorzulegen.

1.1.3.3 Bewilligungsfähige Forschungsvorhaben in Verbindung mit Software

Es werden nur Vorhaben - die Software zum Inhalt haben - bewilligt, die eine komplette Beschreibung der Forschungsarbeiten des Berechnungskerns und der Oberflächen beinhalten. Sollte ein Forschungsvorhaben, das Software beinhaltet, keine Oberfläche benötigen, so ist das frühzeitig anzuzeigen und zu erklären. Die Forschungsstelle hat sich im Vorfeld der Vorhabensdefinition aktiv darüber zu informieren, welche Möglichkeiten bisher in der aktuellen Version der FVA-Workbench vorhanden sind und wie ihr Vorhaben dort eingebunden werden kann. Hierzu kann sich die Forschungsstelle an den FVA-SoftwareService wenden, der ihr beratend zur Seite steht.

1.1.3.4 Zeitplanung

Bei der Definition des Projektzeitrahmens sind folgende Zeitabschnitte zu berücksichtigen und mit einzuplanen (s. Abbildung 1):

Konzeptphase:

Zu Beginn des Forschungsvorhabens soll der Programmaufbau bzw. die Eingliederung der neuen Programmteile in das bestehende Programm (bei Erweiterungen) entworfen und vorgestellt werden.

Entwicklungsphase:

Es soll ausreichend Zeit für die Entwicklung der Algorithmen sowie die Programmierung eingeplant werden. Zu beachten ist, vorallem der Zeitaufwand zur Dokumentation des Codes sowie zur Umsetzung einer umfassenden Fehlerbehandlung. Vielfach werden Anforderungen in den begleitenden Arbeitsgruppen- und Arbeitskreissitzzungen auch zur Laufzeit des Vorhabens noch geschärft.

Testphase:

Es ist ein ausreichend großer Zeitraum vorzusehen, in dem es der Industrie möglich ist, das Programm ausreichend zu testen. Den Testern sind Testbeispiele vorzulegen, die einen möglichst großen Anwendungsbereich der Berechnungssoftware abdecken. Ein Test durch die beteiligten Industrievertreter ist erst nach Vorliegen einer zuverlässigen und vollständigen Programmversion möglich. Dies schließt auch die Integration in die FVA-Workbench mit ein.

Abnahmephase:

Die Abnahmephase wird i.d.R. von einem BETA-Workshop eingeleitet. Das bedeutet, dass das durchführende Institut mit der projektbegleitenden Arbeitsgruppe einen Termin vereinbart, zu dem das Programm gemeinschaftlich an einem Tag intensiv getestet wird. Hierzu erstellt die Forschungsstelle Schulungsunterlagen und Testbeispiele, die geeignet sind, die im Rahmen des Vorhabens geschaffenen Funktionen vorzuführen.

Die erfolgreiche Abnahme ist dem FVA-SoftwareService in Form eines Abnahmeprotokolls mitzuteilen. Dieses muss vom Federführenden der Arbeitsgruppe gegengezeichnet sein (Vorlage siehe Anhang).

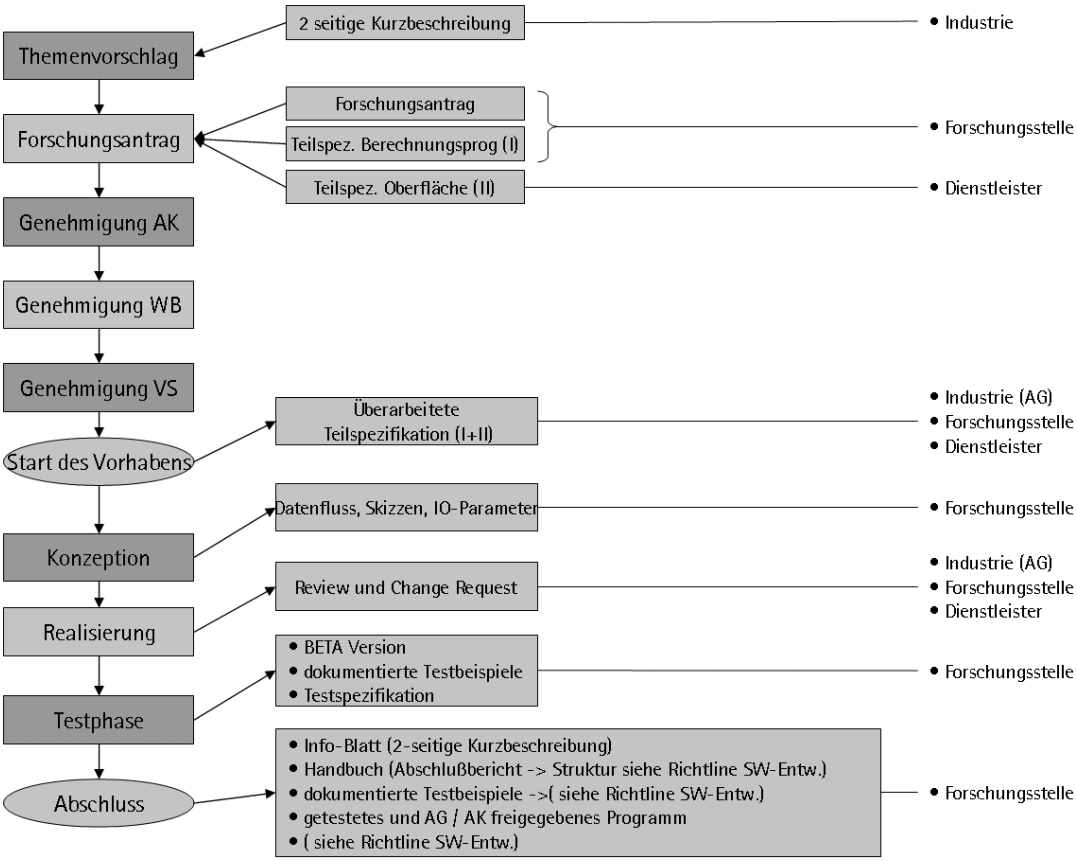

Abbildung 1: Ablauf und notwendige Dokumente für die Durchführung eines Vorhabens mit dem Ziel eine Software zu erstellen

Wesentlich für eine Softwareentwicklung ist das zielgerichtete Vorgehen. Um dies zu erreichen, ist es notwendig, die Entwicklung durch klar definierte Phasen zu strukturieren.

1.2 FVA-Entwicklungsprozess bei Forschung mit Softwareentwicklung

Hierzu hat sich der V-Prozess in der Softwareentwicklung als Standardvorgehensweise etabliert.

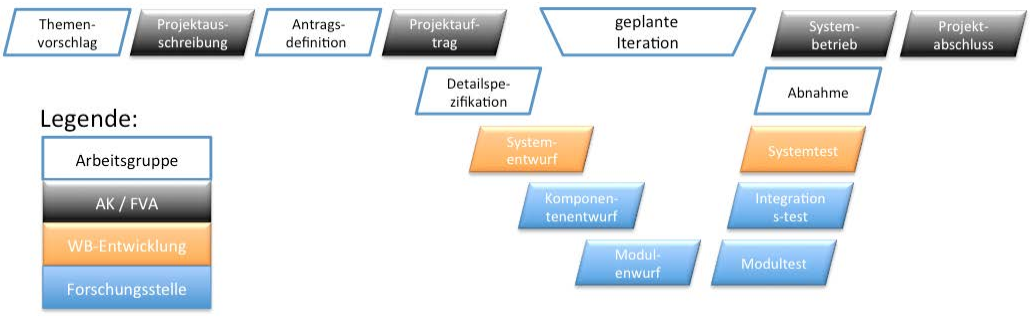

Abbildung 2: Vorgehen bei der Realisierung von FVA-Programmiervorhaben

Beschreibung der Prozessschritte finden Sie in **Abbildung 2: Vorgehen bei der Realisierung von FVA-Programmiervorhaben:** Die Schritte ***Themenvorschlag*** bis zum ***Projektauftrag*** sind für alle FVA-Vorhaben analog abzuhandeln. Hierbei gibt es keinen Unterschied zwischen theoretischen und Software Vorhaben.

Detailspezifikation:

Nachdem, von der FVA der offizielle Projektauftrag vergeben wurde, ist von dem Projektnehmer in Zusammenarbeit mit dem projektbegleitenden Ausschuss eine Detailspezifikation zu entwerfen. Diese soll nach Möglichkeit das Ergebnis der Softwareentwicklung konkret beschreiben. Anhand dieses Dokumentes ist auch die Abnahme der Ergebnisse durch den projektbegleitenden Ausschuss zu erwirken. In dieser Phase sollte nach Möglichkeit das Abnahmeprotokoll definiert werden. Somit ist im Verlauf des Projektes sichergestellt, dass die zu prüfenden Ergebnisse bzw. die Funktionalitäten den zu Beginn definierten Umfängen entsprechen.

Systementwurf:

Hier wird festgelegt wie das Modul in das System eingegliedert wird, z.B. die Definition von Schnittstellen, Parametern, System und Berechnungsgrenzen. Dies erfolgt im Rahmen von Workflowbeschreibungen sowie der Beschreibung der Schnittstellen (Sinn, Zweck, Verwendung, Formate, Datenquellen und Datensenken) der notwendigen Softwaresystemelementen. Dies sollte in Abstimmung aller am Projekt Beteiligten erfolgen. In diese Phase bestimmen Sie die Systemstruktur und beeinflussen damit wie die Komponenten und Module in der Gesamtanwendung, bzw. der Zusammenarbeit. Im übertragenen Sinne ist das die technische Zeichnung und die Stückliste. Das System ist die in der FVA-Workbench integrierte Berechnungskomponente.

Komponentenentwurf:

Nachdem festgelegt ist, wie das System strukturiert ist, müssen nun die Details ausgearbeitet werden. Das bedeutet, dass nun die Funktionalitäten einzelner Systemkomponenten festgelegt und entwickelt werden. Idealerweise werden jetzt schon Kriterien oder Testfälle definiert, gegen welche die Komponenten geprüft werden. Die Komponente ist der aus Modulen zusammengesetzter Berechnungskern, welcher über Ein- und Ausgabeschnittstellen mit dem System kommuniziert.

Modulentwurf:

In dieser Phase werden die Softwaremodule entworfen und entwickelt, sodass alle Funktionen für die Aufgaben in den Systemkomponenten nach dieser Phase zur Verfügung stehen.

Modultest:

Nachdem die Module entwickelt wurden, sind diese den Modultests zu unterziehen. D.h. jedes Modul sollte für sich getestet werden, um Fehler oder Fehlfunktionen auszuschließen. In dieser Phase ist auch die Fehlerbehandlung, welche später im Gesamtsystem benötigt wird, fertigzustellen und im Rahmen des Modultests zu validieren.

Integrationstest:

Nachdem sichergestellt ist, dass alle Module fehlerfrei arbeiten bzw. die Fehlermeldungsmechanismen anschlagen, wenn eine Bedingung verletzt wurde, ist nun zu prüfen, ob die einzelnen Module im Verbund ein kontrolliertes Verhalten zeigen. Die in der Komponentenentwurfsphase definierten Testfälle sollten jetzt geprüft werden und bei Bedarf der Komponentenentwurf angepasst werden.

Systemtest:

Nachdem die Berechnungskomponente soweit fertiggestellt ist und die jeweiligen Funktionstests durchgeführt wurden, ist diese Komponente nun in die FVA-Workbench zu integrieren. Nachdem die Integration erfolgt ist, sind nun die Tests für die Funktionsfähigkeit im Gesamtsystem zu testen. Auch hier sollte auf die in der Systementwurfsphase definierten Testfälle zurückgegriffen werden. Nach dem die Entwicklung den internen Systemtest erfolgreich bestanden hat, haben Sie mit der Software einen Alpha Status erreicht. Diese Alpha-Version ist nun Basis für den folgenden BETA-Test in der nun der Anwender mit der Software konfrontiert ist und ein Bestandteil der Testprozedur wird.

Abnahme:

Die Abnahme erfolgt in Zusammenarbeit mit dem projektbegleitenden Ausschuss, der im Rahmen eines strukturierten Workshops mit dem Programm konfrontiert werden sowie auf Basis der des zu Beginn erstellten Abnahmeprotokolls die Funktionalitäten prüfen und abnehmen sollte. Nachdem der BETA-Workshop durchgeführt wurde, hat die Software einen BETA-Status. Das bedeutet, dass der projektbegleitende Ausschuss im Nachgang die Möglichkeit hat, weitere Praxistests durchzuführen. Eventuell während der Nutzung aufgetretene Mängel können jetzt noch von der Entwicklung behoben werden, sodass bis zum nächsten anstehenden Arbeitskreis die offizielle Abnahme und der Abschluss des Vorhabens erfolgen kann.

Die oben beschriebenen Prozessschritte sind keine Einbahnstraße. Ab dem Punkt „Detailspezifikation“ bietet diese Vorgehensweise immer die Möglichkeit, Funktionen und Anforderungen iterativ anzupassen und zu detaillieren. Diese Iterationen sollten aber personell und inhaltlich abgestimmt sein, sodass die Projektlaufzeit nicht ungeplant in die Länge gezogen werden. Vielmehr sollten gewünschte (nicht spezifizierte) Funktionserweiterungen in ein Folgeprojekt verschoben werden. Damit ist garantiert, dass diese Projekte ein definiertes Ende haben und allen am Projekt Beteiligten der jeweilige Status der Entwicklung transparent ist.

Das Schema stellt Ihnen einen Rahmen zur Verfügung, der es Ihnen erleichtert Ihre Projektarbeiten zu strukturieren und zu definieren, u.a. an welcher Stelle Sie sich im Projekt befinden. Aus diesem Grund sollten Sie sich an diese Vorgehensweise halten. In der Kommunikation mit der Industrie ist es oft hilfreich zu erkennen, in welcher Projektphase Sie sich befinden. Es sollten insbesondere die im V-Modell beschriebenen Stufen durchlaufen werden. Diese Vorgehensweise kann aber auch allgemein bei der Entwicklung beliebiger Inhalte verwendet werden. Es bietet sich immer an, bei jeder Entwicklung eine strukturierte Arbeitsweise zu verfolgen.

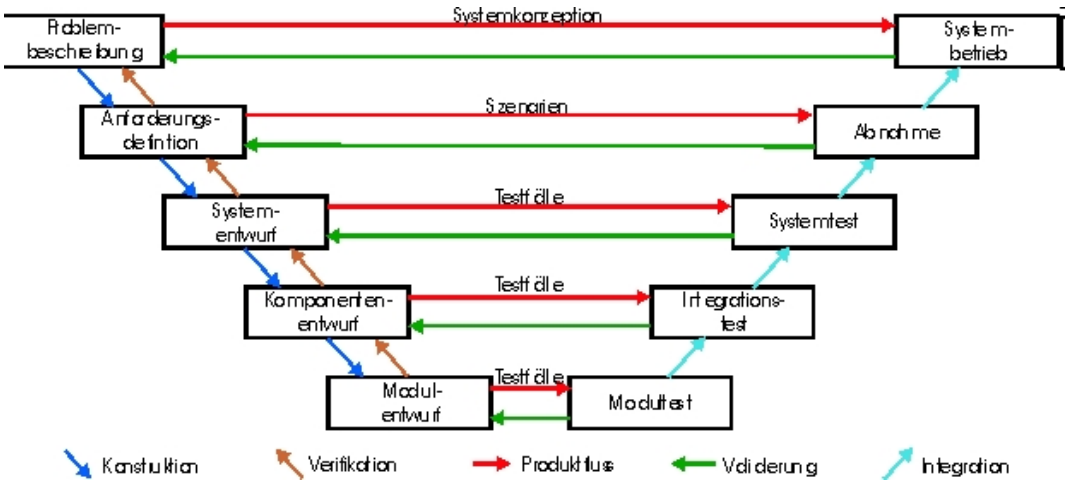

Abbildung 3: Schematische Darstellung des V-Prozessmodells wie es in der FVA für die Softwareentwicklung verwendet werden sollte.

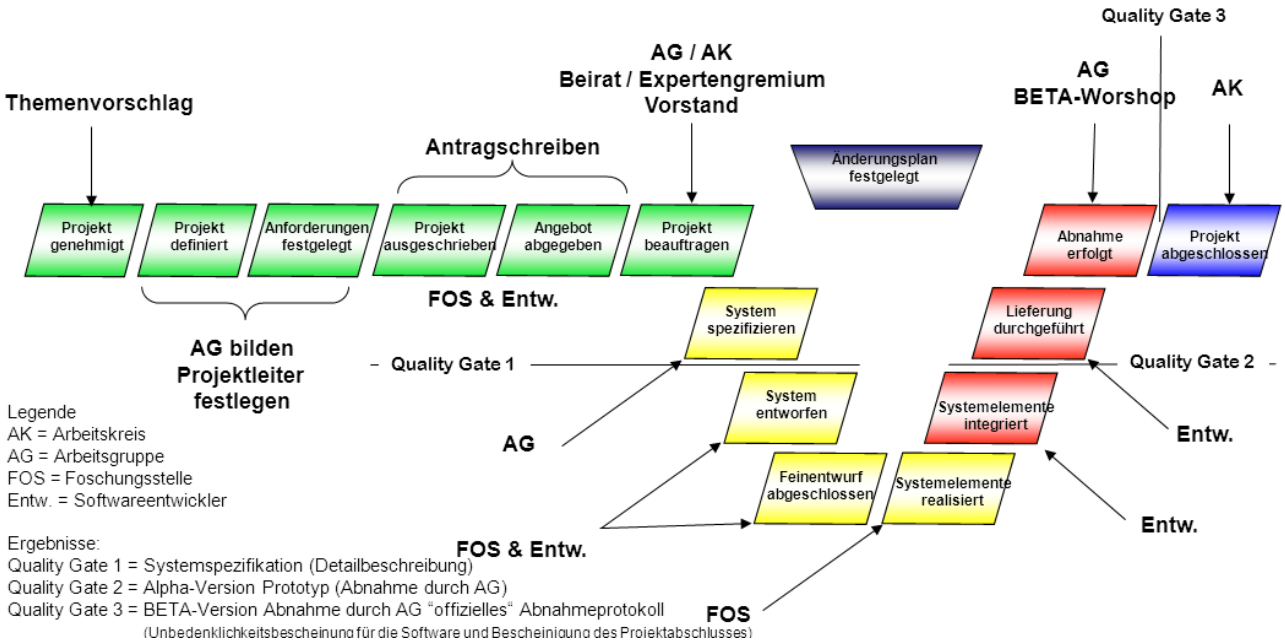

Abbildung 4: Schematische Darstellung des V-Prozessmodells als Standardprozess in der FVA

Die sogenannten Quality Gates sind zu definieren. Diese sind bestimmte Zeitpunkte im Projekt, zu denen ein Ergebnis vorliegen muss bzw. das weitere Vorgehen auf Basis der Ergebnisse abgestimmt werden kann.

Bei der Programmentwicklung sind die folgenden Ergebnisse einem Review zu unterziehen:

Quality Gate 1: Detaillierte Systemspezifikation

Quality Gate 2: Funktionierende Alpha-Version, welche von den Anwendern gegen die Spezifikation geprüft werden kann.

Quality Gate 3: Funktionierende BETA-Version, die von den Anwendern gegen die Spezifikation geprüft werden kann. Das Projekt gilt als abgeschlossen, wenn der BETA-Workshop erfolgreich von der Arbeitsgruppe dokumentiert (Abnahmeprotokoll) absolviert wurde. Erst nach erfolgtem BETA-Workshop kann eine Entlastung durch den AK beantragt werden.

1.3 Beteiligte Institutionen FVA-Softwareentwicklung und deren Rollen

Damit das Ergebnis Ihrer Forschungsarbeit, das von allen Mitgliedsfirmen der FVA finanziert ist, auch allen Mitgliedsfirmen zur Verfügung steht, hat die FVA weitere Institutionen, die Sie bei der Entwicklung der Software und bei der Nachbereitung (Einführung der Software) in den Mitgliederkreis unterstützen. In diesem Kapitel sind diese benannt und kurz beschrieben. Für weitere Informationen steht Ihnen die FVA-Geschäftsstelle zur Verfügung.

1.3.1 FVA-Kompetenzzentrum

Rolle: Beratungs-, Qualitätssicherung und Abnahmeinstanz

1) Unterstützung von Anwendern in der Industrie bei Fragen zu FVA-Programmen
- Installation und Benutzung auch von Altprogrammen
- Inhaltliche Fragen zu theoretischen Grundlagen und der Anwendung auf Praxisfragestellungen
- Fragen zur Bedienung (bisherige Benutzeroberflächen und aktuelle FVA-Workbench)
- Informationen zum Quelltext und zur Übersetzung / Portierung

2) Organisation der FVA-Softwareentwicklung
- Erstellen und Pflegen der Programmierrichtlinie
- Formale Abnahme von Programmen (Vollständigkeit, Dokumentation, Einhaltung der Programmierrichtlinie)
- Verwaltung des FVA-Softwarearchivs (Betrieb und Wartung des Quelltextverwaltungssystems für die FVA)
- Aktualisierung und Pflege des Software-Bereichs in ProMeta, Bereitstellung der FVA-Programme für die Mitgliedsfirmen

3) Beratung anderer Institute bei der Erstellung von FVA-Programmen
- Anforderungen an Software festlegen
- Unterstützung bei Problemen in der Entwicklung
- Beratung zur Spezifikation erforderlicher FVA-Workbench-Erweiterungen

4) Unterstützung der Dienstleister, die die technische Entwicklung der FVA-Workbench übernehmen sowie strategische Beratung der Entwickler zur Anwendbarkeit
- Mitarbeit in der strategisch-technischen Projektplanung der FVA-Workbench
- Unterstützung bei Programmintegrationen
- Weiterentwicklung der Bedienung der FVA-Workbench
- Erstellung von Spezifikationen zum Systemverhalten
- Unterstützung bei der Weiterentwicklung des Datenmodells der FVA-Workbench
- Anforderungen an Programmintegrationen festlegen
- Beratung und Support der FVA-Firmen beim Einsatz der FVA-Workbench
- Absicherung der Bedienung durch Tests und Verbesserungsvorschläge

1.3.2 FVA-Software Service

Rolle: Koordination übergeordnetes Projektcontrolling

1) Strukturierung und Koordination der FVA-Softwareentwicklung
- Beratung und Betreuung der FVA-Gremien
- Betreuung und Ausbau der FVA-Softwareinfrastruktur
- Kommunikation und Öffentlichkeitsarbeit zur FVA-Software
- Überwachung und Regelung der rechtlichen Rahmenbedingung für FVA-Software
- Beratung des wissenschaftlichen Beirates und des Vorstandes der FVA
- Ausrichtung der FVA-Softwareentwicklung, sodass die Strategie der FVA unterstützt wird bzw. die formulierten Ziele erreicht werden

2) Organisation der FVA-Softwareentwicklung
- Erstellen, Pflegen und Anpassen der Programmierrichtlinie
- Formale Abnahme von Programmen (Vollständigkeit, Dokumentation, Einhaltung der Programmierrichtlinie)
- Verwaltung des FVA-Softwarearchivs in ProMeta
- Strukturierung der Wartungs- und Supportinfrastruktur

3) Beratung anderer Institute bei der Erstellung von FVA-Programmen
- Anforderungen an Software festlegen
- Unterstützung bei Problemen in der Entwicklung
- Beratung zur Spezifikation erforderlicher FVA-Workbench-Erweiterungen

4) Leitung und Organisation der Dienstleister und Forschungsstellen
- Freigabe der FVA-Workbench
- Festlegung und Überwachung der Zeit- bzw. Projektpläne
- Ressourcenmanagement

1.3.3 Dienstleister

Rolle: Systemimplementierer und Systemqualitätssicherung

1) Integration der Forschungsergebnisse (Kernprogramme) in die FVA-Workbench
- Unterstützung der Forschungsstellen bei der Gestaltung ihrer Programme
- Betreuung und Ausbau der FVA-Softwareinfrastruktur
- Betrieb und zur Verfügungsstellung eines Qualitätssicherungssystem

1.3.4 Arbeitsgruppen

Rolle: Kontroll-, Projektpartner und Steuergremium

1) Definition des Forschungsbedarfs
2) Begleitung der Forschungsstelle bei Durchführung ihrer Softwareentwicklung (Forschungsvorhaben)
3) Anforderungsdefinition aus Sicht der Industrie
4) Abnahme der Ergebnisse
5) Verantwortung für die Abnahme der Leistungen der Forschungsstelle
6) Sicherstellen der Zuverlässigkeit und Relevanz der entwickelten Berechnungsansätze

1.3.5 Arbeitskreise

Rolle: Auftraggeber und Abnahmeinstanz

1) Beratungsgremium für die Arbeitsgruppe
2) Formale Institution zur Beantragung bzw. zur Abnahme der Forschungsergebnisse
3) Informationsgremium für FVA-Mitglieder

1.4 Arbeitsschritte

1.4.1 Die Realisierung

Im Laufe der Entwicklung gibt es wiederholt den Bedarf, die geplanten Strukturen und/oder Ergebnisse anzupassen. Jedoch sollten im Laufe eines Vorhabens die Funktionalitäten und Eigenschaften der Software grundlegend erweitert werden.

Als Ergebnis dieser Phase sollte den Anwendern eine ALPHA-Version (Definition siehe Qualitätsmanagementhandbuch für die FVA-Workbench-Entwicklung) zur Verfügung gestellt werden. Nachdem alle Beteiligten die Eigenschaften und Funktionen der Software abgenommen haben, sollten Sie das Programm für die BETA-Phase vorbereiten. Das bedeutet, dass Sie Ihre Software so erweitern, dass alle möglichen Fehlnutzungen abgefangen werden. Die Ausgaben sollen so gestaltet werden, dass diese in gefälliger Weise dem Anwender die Ergebnisse darstellen. Die verschiedenen Möglichkeiten der Ergebnisdarstellung werden in weiteren Kapiteln dieses Dokumentes behandelt.

Das Arbeitsergebnis dieser Phase ist eine BETA-Version Ihres Programmes. (Definition siehe Qualitätsmanagementhandbuch für die FVA-Workbench-Entwicklung)

1.4.2 Die Testphase

Idealerweise wurden die Funktionen und Eigenschaften der Software in der Alpha-Phase abgenommen. Nun sollte das Programm, welches Sie erstellt haben, alle wesentlichen Merkmale einer anwendbaren Software haben. Die BETA-Version ist nun zahlreichen Tests zu unterziehen, d.h. dass Sie genügend Praxisbeispiele haben, an denen Sie die Verlässlichkeit Ihrer Berechnungen nachweisen können. Sie können aber auch theoretische Beispiele nutzen, um nachzuweisen, dass die implementierten Algorithmen korrekt funktionieren.

Das Ergebnis dieser Phase ist das von mindestens dem AG-Federführenden unterzeichnete Testprotokoll. Ohne das Testprotokoll und die Freigabe der begleitenden AG wird das Vorhaben nicht abgeschlossen. Das Testprotokoll ist bei der FVA-Geschäftsstelle mit dem Abschlussbericht einzureichen.

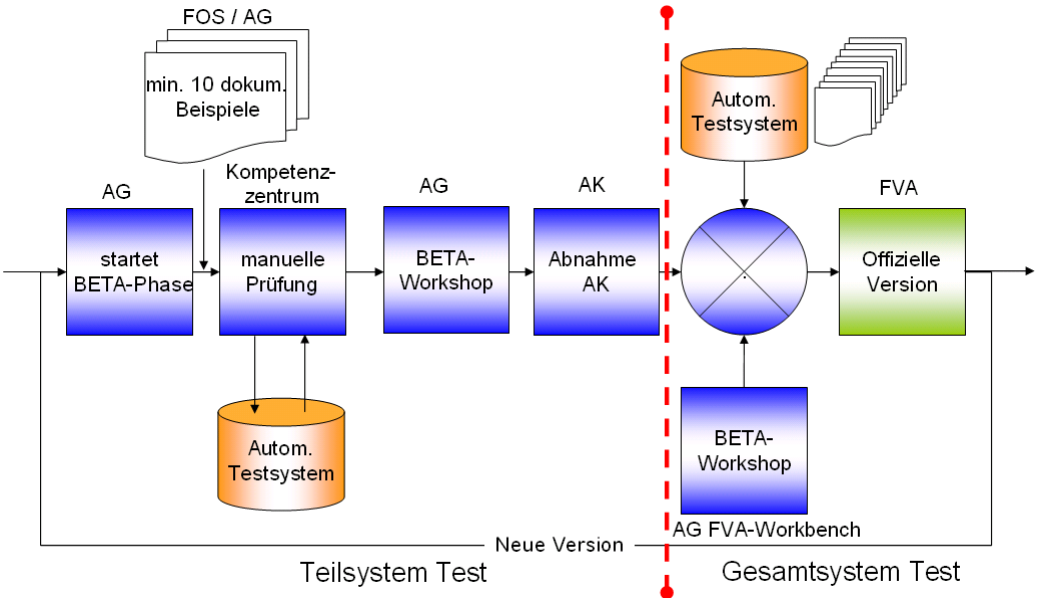

Abbildung 5: Schematische Darstellung des Abnahmeprozesses für FVA-Software

1.4.3 Die Wartungsphase für Programme aus den Forschungsstellen

Der Lebenszyklus einer Software fängt nach deren Fertigstellung erst an. Das bedeutet, dass insbesondere die Nutzung der Software in der Praxis oft Mängel bzw. Grenzen offenbart, die bei der Entwicklung nicht berücksichtigt wurden. Darüber hinaus bedingt die Nutzung einer Software auch die Beratung von Anwendern, insbesondere bei der Interpretation von Ergebnissen. Nach Veröffentlichung vergeht meist ein Zeitraum von mehreren Monaten bis zu einem Jahr, bevor die ersten Rückmeldungen aus der Praxis eintreffen. Für die projektausführende Forschungsstelle ist es Voraussetzung, dass der Mitarbeiter, der die Software entwickelt hat bzw. ein Mitarbeiter, der sich mit der Software auskennt, noch mindestens 24 Monate nach Projektabschluss an der Forschungsstelle zur Verfügung steht. Damit ist gewährleistet, dass die Software, welche im Rahmen des FVA-Projektes entwickelt wurde, weiter betreut wird (Unterstützung der Anwender und der FVA-Softwareentwicklung). Sollte die Forschungsstelle nicht in der Lage sein, diese Unterstützung am Ende eines Vorhabens zu erbringen, wird die FVA (dies liegt in der Verantwortung der Arbeitsgruppe) prüfen, ob sie das Projekt mit der Forschungsstelle durchführt. Im Zweifelsfall kann es bedeuten, dass die FVA das Projekt mit der ausgewählten Forschungsstelle nicht genehmigt. Eventuell kann die Forschungsstelle im Projektantrag ein Transferszenario entwickeln, um die langfristige Nutzbarkeit der Software zu gewährleisten.

2 Pflichten und Anforderungen an die Programmierung

2.1 Programmerstellung und –architektur

Das Standardvorgehen zur Getriebeberechnung in der FVA ist die Dateneingabe in die FVA-Workbench. Diese erstellt die spezifische ASCII-Eingabedatei für den Berechnungskern, schreibt die Konfigurationsdatei und ruft den Berechnungskern auf. Nach Durchlauf der Berechnung zeigt die FVA-Workbench die Text- und Grafikausgabedateien an. Ausgewählte Ergebnisse sollen in die Datenbasis der FVA-Workbench übernommen werden. Abweichungen von diesem Verhalten müssen vor Projektstart mit allen beteiligten Entwicklungspartnern geklärt werden.

Vor der Programmierung der Funktionalität des Berechnungskerns ist der Programmaufbau zu entwerfen und in Diagrammen als Ablaufplan bzw. als Programmstruktur darzustellen. Bei Erweiterungen vorhandener Programme ist die Einordnung der neuen Funktionalität in die bestehende Struktur zu dokumentieren.
Grundsätzlich ist die Gliederung des Programmes abhängig vom Funktionsumfang. Das Programm ist in Module zu gliedern, die größere Berechnungsaufgaben erfüllen. Jedes Modul muss über eine klare Schnittstelle verfügen, über die sämtliche Eingangs- und Ergebnisdaten übergeben werden. Diese Schnittstelle wird durch einen Subroutine-Aufruf dargestellt. Die Datenübergabe erfolgt durch die Formalparameterliste. Ein Datentransfer „im Hintergrund“ über Common-Blöcke ist nicht zulässig.
Jedes Modul ist in weitere Unterprogramme und Routinen zu ordnen. Innerhalb von Modulen ist der Datenaustausch zwischen den Routinen auch über globale Variablen zulässig, soweit die Notwendigkeit dafür begründet werden kann.

Ein Programmmodul zeichnet sich aus durch:

- eine klare explizite Schnittstelle und Formalparameterliste der Berechnungsroutine
- Ein-/Ausgabe (Daten-)Schnittstelle (Einlesen der Eingabedatei, Schreiben der Ausgabedatei)
- Recheninhalt in Form von Subroutine/Objekt/Module

Zu jedem Berechnungsmodul sind Routinen zum Einlesen der Eingabedaten aus der ASCII-Datei sowie zur Ausgabe der Berechnungsergebnisse in Textform und als Grafik zu erstellen.
In **Abbildung 6** ist der Aufbau eines Moduls am Beispiel der Berechnung von Flankenkorrekturen für Stirnräder gezeigt.
Die Routine „Korrekturmatrix berechnen“ enthält die Rechenalgorithmen und kommuniziert ausschließlich über die Formalparameterliste mit dem aufrufenden Programmteil. Das Einlesen der

Daten aus der ASCII-Eingabedatei wird von der Routine „Einlesen“ durchgeführt. Diese Routine liest die Eingabedatei und gibt die Daten danach weiter.
Die Routine „Text- und Grafikausgabe erstellen“ erhält die Ergebnisdaten. Diese werden verwendet, um die Ergebnistexte und –grafiken in Dateien abzuspeichern.

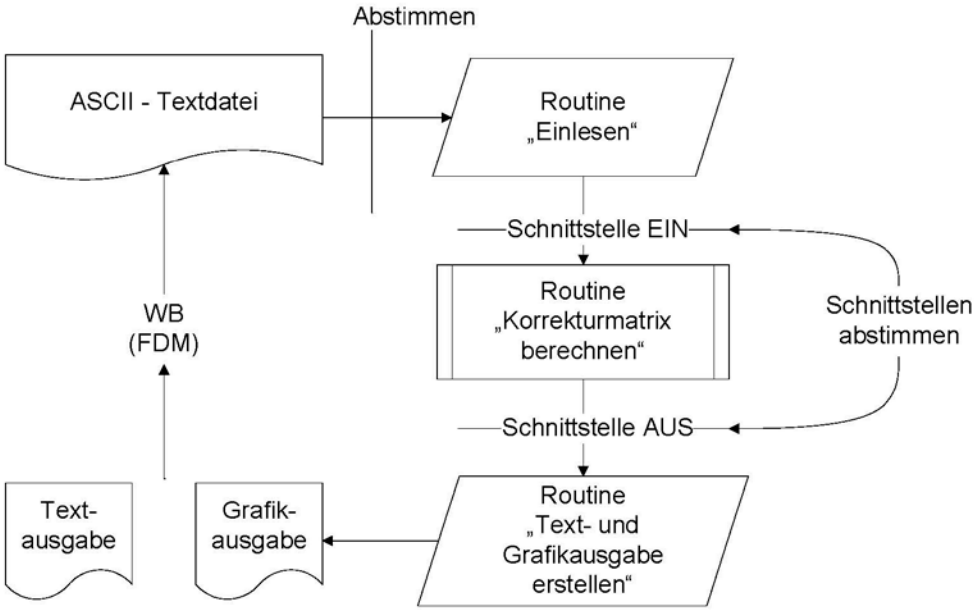

Abbildung 6: Beispiel für Teile und Schnittstellen eines Moduls (Schema des Moduls zur Berechnung von Flankenkorrekturen mit Schnittstellen)

Falls notwendig kann aus Berechnungs-, Einlese- und Ausgaberoutinen durch eine zusätzlich zu implementierende Steuerroutine ein ausführbares Programm erstellt werden.

Folgende Programmodule existieren in der FVA bereits:

- LAGER2 Wälzlagerberechnung
- LAGER2HP Hauptprogramm zu LAGER2
- WELLAG Berechnung des Welle-Lager-Systems

Es gibt zwei gängige Formen der ausführbaren Programmmodule: Die sogenannten .exe-Datei und .dll-Datei.

.exe-Datei

Die als .exe-Datei kompilierten Programmmodule laufen selbstständig. Bei der Integration in ein Gesamtsystem benutzt das Programmmodul normalerweise eine Dateischnittstelle. Wenn dieses Modul aber mit mehreren anderen Modulen Daten gleichzeitig austauschen oder von mehreren Modulen benutzt werden soll, führt eine Aufteilung eines Gesamtprogrammes in .exe-Dateien zu Problemen.

.dll-Datei

Die als .dll-Datei kompilierten Programmmodule bieten eine flexible Benutzung. Die .dll-Datei wird nur zur Laufzeit gebunden und nur eine Kopie wird in den Hauptspeicher geladen. Unterschiedliche Prozesse benutzen die einzige Kopie für eigene Funktionen. Ist die Schnittstelle der .dll-Datei gut definiert, so hat ein Austausch der .dll-Datei z.B. bei einem Update, keinen Einfluss auf das Gesamtsystem. Die Aufteilung in mehrere .dll-Dateien ist für ein Projekt sehr günstig, weil der Rechenkern in unabhängigen Funktionalitäten zerlegt werden kann.

Funktion oder Subroutine

Die Funktion/Subroutine ist die einfachste Variante der Elemente und Grundbausteine der Programmierung. Es gibt bei dieser Variante keine direkte Ansteuerung von außen. Der Datenaustausch ist schnell.

In Tabelle 1 sind die Vorteile und Nachteile den vorgestellten Varianten aufgelistet.

	.exe-Datei	.dll-Datei	Funktion/Subroutine
Geschwindigkeit der Ansteuerung (Mehrfachaufruf)	0	+	++
Speicherbelegung	-	++	++
Debug-Fähigkeit (Wenn Element von außen angesteuert wird)	--	-	+
Organisationsaufwand (bei Implementierung in ein Gesamtsystem)	+	+	-

Tabelle 1: Die Vorteile und Nachteile der Ausführung des Programmmoduls

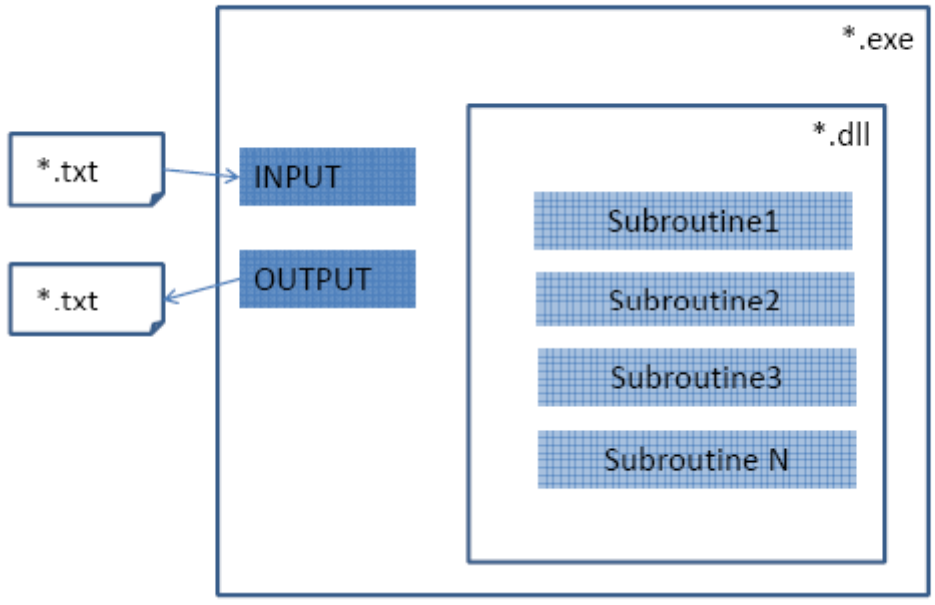

Abbildung 7: Schalen der Modulausführung

Um die Flexibilität des Programms zu erhöhen, kann eine Kombination von den Varianten herstellt werden (siehe Abbildung 7). Die Funktionalitäten der Software können als .dll Datei kompiliert werden und mit eigener Schnittstelle autonom benutzt werden. Das ganze Programm mit .dll Komponenten kann als das Ganze kompiliert werden. Als Schnittstelle des ganzen Programms können ASCII-Dateien benutzt werden.

Implementierung des Subsystems

Bei der Implementierung der Subsysteme und der in den Subsystemen zusammengefassten Komponenten ist vor allem darauf zu achten, ob die Aufgaben, die im Entwurf für das Subsystem festgelegt wurden, in der Implementierung auch eingehalten werden.

Bei der Implementierung der Subsysteme kann von innen nach außen (d.h. zuerst werden die Programmmodule implementiert) oder von außen nach innen (d.h. zuerst werden die Schnittstellen implementiert) vorgegangen werden. Der Vorteil des zweiten Vorgehens ist, dass die Schnittstellen dadurch bereits feststehen und eine willkürliche Abänderung unwahrscheinlich wird. Dies setzt allerdings voraus, dass Schnittstellen auf Basis des vorhandenen Wissens über die Systemstruktur bereits fest definiert werden können.

Richtlinien für den Programmcode

Syntax und Struktur

- Strukturieren Sie den Code so, dass er immer leicht lesbar bleibt
- Verwenden Sie Einrückungen (Tabulatoren und Leerzeichen)
- Beschränken Sie die maximale Zeilenlänge auf 80 Zeichen. Es gibt viele Anwendungssysteme, deren Anzeigevermögen auf 80 Zeichen begrenzt ist
- Verwenden Sie nur eine Anweisung pro Zeile
- Vermeiden Sie unverständliche Abkürzungen
- Legen Sie sich ein konsistentes Benennungssystem zurecht und halten Sie es ein

Kommentare

- Sparen Sie nicht mit Kommentaren, insbesondere bei nicht offensichtlichen Annahmen
- Beschreiben Sie vor jeder Methode ihre Funktionsweise
- Sparen Sie nicht an begleitender Dokumentation

System

- Vermeiden Sie den Gebrauch globaler Variablen. Eine Methode verwendet als Daten im Normalfall nur Parameter und lokale, initialisierte Variablen
- Implementieren Sie eine effektive Fehlerbehandlung

Klassenebene

- Verwenden Sie Vererbung nur, wenn Klassen in inhaltlichem Zusammenhang stehen
- Fassen Sie in Subsystemen nur zusammen, was auch zusammengehört
- Vermeiden Sie zu allgemeine Klassen. Jede Klasse soll eine genau spezifizierte Funktionalität besitzen, für die sie Spezialist ist

Methodenebene

- Benutzen Sie aussagekräftige Variablen- und Methodenbezeichner. Verwenden Sie zur besseren Lesbarkeit Groß- und Kleinschreibung
- Halten Sie private und öffentliche Methoden strikt auseinander
- Jede Methode muss testbar (Unit-Tests) sein, d.h. es soll zu entscheiden sein, ob sie korrekt implementiert wurde

Verwendung bestehender Module unter der FVA-Workbench

Grundsätzlich sind auch Teilfunktionen bestehender Programme zukünftig modular nutzbar.
Unter der FVA-Workbench kann durch die Definition von Berechnungsketten und die automatische Übernahme von Ergebnissen bereits ein „modulähnliches“ Verhalten erreicht werden.

Beispiel: Die Berechnung der Stirnradgeometrie mit STplus bietet dem Anwender zahlreiche Alternativen zur Eingabe der erforderlichen Grundgeometriegrößen. STplus berechnet zahlreiche daraus abgeleitete Größen. Diese stehen in der FVA-Workbench nach der STplus Berechnung zur Verfügung und können z.B. für die Ansteuerung von RIKOR verwendet werden. Dadurch kann unter der FVA-Workbench STplus als Modul zur Geometrieberechnung angesteuert werden, um nicht auf die eingeschränkten Möglichkeiten der Geometrieeingabe in RIKOR begrenzt zu bleiben. Außerdem

existiert dadurch eine einheitliche Geometrieberechnung für alle Stirnradprogramme unter der FVA-Workbench.

Zeitlicher Ablauf

Bei Start des Projekts erfolgt die Abstimmung der Schnittstellen zur FVA-Workbench. Die Datenübergabe zwischen FVA-Workbench und Rechenkern erfolgt über eine ASCII-Eingabedatei mit blockweiser Datenstruktur (s. GEAS Abschlussbericht FVA 555).
Zur eindeutigen Festlegung der Schnittstelle sind die Eingabedaten beginnend von den Attributen der FVA-Workbench bis zu den Bezeichnungen in der ASCII-Eingabedatei festzulegen (Screenname, Kategorie, Einheit, ASCII-Bezeichner, Position in der ASCII-Datei).

Im Anhang ist eine Excel-Tabelle zu finden, aus der das Schema ersichtlich ist.

Grundsätzlich gilt beim Eintragen neuer Variablen in das FVA-Workbench-Produktmodell:

- Neuanlage von Attributen ist nur nach Prüfung zulässig, ob diese nicht bereits vorhanden sind
- Änderung von Attributen nur bei vollständiger Dokumentation der Abwärtskompatibilität
- Zeitplan für die Umsetzung
- Absprache mit FVA-Kompetenzzentrum und FVA-SoftwareService

2.2 Programmiersprache

Matlab, Python oder andere Skriptsprachen sind nur für den Entwurf eines Prototypen zu verwenden. Dies kann bei Neuprogrammierungen ratsam sein und ist mit der betreuenden Arbeitsgruppe zu klären.

Zum Abschluss des Forschungsvorhabens ist das fertige Programm in FORTRAN nach ISO-Standard vorzulegen. Optional kann bei Neuprogrammierungen C/C++ als Programmiersprache eingesetzt werden. Grundsätzlich besteht die Möglichkeit, beide Sprachen über Bibliotheken (libs / Dlls) zu kombinieren. Innerhalb eines Forschungsvorhabens ist EINE Sprache zu verwenden, es können aber vorliegende Module sprachübergreifend integriert werden, soweit erforderlich.
Abweichungen von dieser Vorgabe sind mit dem FVA-SoftwareService abzustimmen und müssen vom FVA-Kompetenzzentrum freigegeben werden.

Zur Verarbeitung der Ein- und Ausgabedaten sind im gesamten Programm einheitliche, standardisierte Bibliotheksfunktionen zu verwenden. Die verwendete Ausgabebibliothek muss die Textausgabe auf Basis von Maskendateien/Templates erstellen. Damit ist sicherzustellen, dass sämtliche Ausgabetexte unabhängig vom Programmcode (ohne erneutes Compilieren und Linken) geändert werden können. In der FVA steht die IMELIB für Ein- und Ausgabeoperationen zur Verfügung. Alternative Bibliotheken sind zulässig, solange mindestens die IMELIB-Funktionalität erfüllt ist.

2.3 Programmaufbau und -schnittstellen

Das Berechnungsprogramm kommuniziert über diverse Schnittstellen mit der FVA-Workbench oder firmeneigenen Programmen bzw. kann vom Anwender über die Dateischnittstellen bedient werden.

Folgende Schnittstellen sind in Dateiform für jedes Programm vorzusehen:

Konfigurationsdatei im ASCII-Format:

- ASCII-Eingabedatei
- Zuordnungsdatei
- Maskendatei
- Schnittstellendatei

Konfigurationsdatei im ASCII-Format:
Die Datei enthält die wichtigsten Informationen zum Programmaufruf des Rechenkerns. Enthalten sind Daten zum Anwender, Pfad und Name der Eingabedatei sowie der Ausgabedateien und der Templatedatei für die Ausgabeerstellung (Maskendatei). Die Zuordnungsdatei dient zur Entkoppelung von ASCII-Eingabedatei und Einlesebefehlen im Programm. In der Zuordnungsdatei wird für jeden Bezeichner in der Eingabedatei ein interner Bezeichner festgelegt, unter dessen Verwendung auf den entsprechenden Wert zugegriffen wird. Außerdem sind in der Zuordnungsdatei Defaultwerte und Wertebereich für jedes Datum angegeben. Es können weitere Einträge für die Konfigurationsdatei definiert werden, wenn erforderlich. Ein Beispiel für den Aufbau einer Konfigurationsdatei ist in **Abbildung 8** zu sehen.

```
Firmenname = FZG, TU Muenchen
Sachbearbeiter = Dr.-Ing. M. Otto
Eingabedatei = ..\..\STplus\work\eingabe.ste
Ausgabedatei =  ..\..\STplus\work\ausgabe.sta
Graphikdatei =  ..\..\STplus\work\zzzg67.stz
Graphikdatei_2 =  ..\..\STplus\work\zzzg672.stz
Zuordnungsdatei =  ..\..\STplus\bin\qelesed.stq
Werkzeugdatei_lokal =  ..\..\STplus\wkz\wkz.dat
Werkstoffdatei_lokal =  ..\..\STplus\wst\wst.dat
Schmierstoffdatei_lokal =  ..\..\STplus\oel\oel.dat
Pfad_Maskendateien =  ..\..\STplus\zubeh\
Schnittstellendatei =  ..\..\STplus\work\ausgabe.sts
STEP_Konfiguration =  ..\..\STplus\work\config.dat
```

Abbildung 8: Beispiel einer Konfigurationsdatei

ASCII-Eingabedatei
Hier werden Aufbau und Struktur des Getriebemodells definiert sowie die Eingabewerte übergeben. Die Daten sind blockweise geordnet, jeder Wert wird einem Bezeichner mittels „=“-Zeichen zugewiesen. Die Datei muss mit „$ Anfang“ beginnen, Ende der einzulesenden Daten ist mit „$ Ende“ erreicht. Ein Beispiel ist in dargestellt.

```
$ Anfang

$ Ausgabe

A_KOPFRUECKNAHME = 0
A_DECKBLATT = 0
RIKORVERSION = H 2.2

$ Welle

IW = 1
DREHRI = -1
TEXT = Beispiel 1: Antriebswelle (Welle1)
UK = 0 DA = 108 TUA = -1
UK = 222 DA = 114
UK = 334.5 DA = 114 LAGER = 1
UK = 353 DA = 143
UK = 359 DA = 138
UK = 410 DA = 114
UK = 667 DA = 138
UK = 718 DA = 143
UK = 724 DA = 114
```

```
UK = 742.5 DA = 114 LAGER = 1
UK = 789 DA = 0

$ Welle

IW = 2
U_0 = 289
DREHRI = 1
TEXT = Beispiel 1: Abtriebswelle (Welle2)
UK = 0 DA = 146
UK = 53 DA = 146 LAGER = 1
UK = 65 DA = 171.45
UK = 186 DA = 465
UK = 313 DA = 146.05
UK = 446 DA = 146.05 LAGER = 1 TUA = -1
UK = 832 DA = 0

$ Stufendaten

IW = 1 2
IR = 1 1
ACHSWIN = 180.
T1 = 5975
BERE = 1
S_BAUART = 1

$ Zahnrad

IW = 1
IR = 1
MN = 8
Z = 14
X = 0.4713
DNA = 137.97
UKA = 470
B = 137
HA0 = 1.47
ZAHNRADWERKSTOFF = 18CrNiMo7-6
BETA = -11

$ Zahnrad

IW = 2
IR = 1
Z = 59
X = 0.4161
BETA = 11
DNA = 503.63
UKA = 186
B = 127
HA0 = 1.47
MN = 8
ZAHNRADWERKSTOFF = 18CrNiMo7-6

$ Ende
```

Abbildung 9: Beispiel einer Eingabedatei (RIKOR, bsp1.rie)

Zuordnungsdatei
Im Sinne einer Übersetzungstabelle zwischen ASCII-Eingabedatei und Rechenkern inkl. der Möglichkeit Defaultwerte und Ober-/Untergrenzen für jeden Parameter angeben zu können, ist die sogenannte „Zuordnungsdatei“ aufzubauen.

Maskendatei
Die Maskendatei ist für die Ausgabekonfiguration und enthält die Templates der Ausgabetexte.

Schnittstellendatei
Zur programmbasierten Auswertung der Berechnungsergebnisse ist eine zweite Ausgabedatei zu erstellen. Diese Datei wird auch von der FVA-Workbench verwendet, um im weiteren Rechenverlauf benötigte Daten aus vorhergehenden Berechnungen einzulesen (s. **Abbildung 10**). In die Schnittstellendatei sind alle Eingangsdaten sowie alle Berechnungsergebnisse des Programms aufzunehmen. Minimalumfang sind alle Daten, die dem Anwender in den ASCII-Textausgabedateien zur Verfügung gestellt werden.

```
$ ANFANG

$ ALLGEMEINE_DATEN

BENUTZERTEXT1           = 1. Beispiel
BENUTZERTEXT2           = Ritzel durch zwei Werkzeuge bearbeitet

$ GEOMETRIEDATEN

NORMALEINGRIFFSWINKEL   =         20.00000
STIRNEINGRIFFSWINKEL    =         21.17283
BETRIEBSEINGRIFFSWINKEL =         22.11058
SCHRAEGUNGSWINKEL_TEILK =        -20.00000
SCHRAEGUNGSWINKEL_GRUND =        -18.74724
ZAEHNEZAHL              =              25              65
VIRTUELLE_ZAEHNEZAHL    =         25.00000         65.00000
ZAEHNEZAHLVERHAELTNIS   =          2.60000
STANDUEBERSETZUNG_OZ    =          0.00000
STANDUEBERSETZUNG_OH    =          0.00000
UEBERSETZUNG_SONNE_STEG =          0.00000
ACHSABSTAND             =        241.00000
NORMALMODUL             =          5.00000
STIRNMODUL              =          5.32089
NORMAL_DIAMETRAL_PITCH  =          5.08000
DIAMETRAL_PITCH         =          4.77364
SUMME_X                 =          0.31864
PROFILVERSCHIEBFAKTOR   =          0.25000          0.06864
PROFILVERSCHIEBUNG      =          1.25000          0.34318

# TEILUNGEN, UEBERDECKUNGEN ETC.
STIRNTEILUNG            =         16.71607
NORMALTEILUNG           =         15.70796
AXIALTEILUNG            =         45.92702
STIRNEINGRTEILUNG       =         15.58765
NORMALEINGRTEILUNG      =         14.76066
EINGRIFFSSTRECKE        =         16.23110
GEMEINS_ZAHNHOEHENF     =          1.34321
PROFILUEBERDECKUNG      =          1.04128
SPRUNGUEBERDECKUNG      =          1.52416
GESAMTUEBERDECKUNG      =          2.56544
$ ENDE
```

Abbildung 10: Beispiel einer Schnittstellendatei zur Ergebnisweitergabe

ASCII-Textausgabedatei(en)

Die Ergebnisse sind in Form von Textdateien auszugeben. Die Formatierung entsprechend **Abbildung 11** enthält Name der Größe, Formelzeichen, Wert und Einheit.

```
1       ********************************************************************
        * FVA-Stirnradprogramm STplus 6.0. FZG, TU Muenchen Blatt  1 von 6  *
        * Ein Programm der Forschungsvereinigung Antriebstechnik, Frankfurt *
        ****************************************** 13.03.11  13:16:12 *****
        1. Beispiel
        Ritzel durch zwei Werkzeuge bearbeitet


        ----------------------------------------------------------------------
        Geometrieberechnung nach DIN 3960 (Maerz 1987)
        ----------------------------------------------------------------------
                                  (Treiber=1)  Ritzel=1        Rad=2
        Normaleingriffswinkel . . . . . . alfa_n            20.00000     Grad
          Stirneingriffswinkel  . . . . . alfa_t            21.17283     Grad
          Betriebseingriffswinkel . . .  alfa_wt            22.11058     Grad
        Schraegungswinkel am Teilkreis  . . beta           -20.00000     Grad
          Schraegungswinkel am Grundkreis beta_b           -18.74724     Grad
        Zaehnezahl  . . . . . . . . . . . . . z        25          65    -
          Zaehnezahlverhaeltnis . . . . .  z2/z1             2.600       -
        Achsabstand . . . . . . . . . . . . . a           241.000       mm
        Normalmodul . . . . . . . . . . . . . m_n           5.00000     mm
          Stirnmodul  . . . . . . . . . . . . m_t           5.32089     mm
```

Abbildung 11: Beispiel einer Ausgabedatei

Grafikausgabe
Die grafische Ausgabe der Kenngrößen erfolgt in Dateiform (z.B. LRZ (alt), XML, SVG). Die Ergebnisgrafiken können danach in einem Viewer betrachte werden können.

Errordatei
Hierhin sollten optional alle Fehlermeldungen umgeleitet werden können. Zusammen mit aussagekräftigen Kommentaren kann eine derartige Datei auch der Entwicklung zur Fehlersuche dienen.

Weitere Dateischnittstellen
Nach Bedarf sind Daten-Dateien aus Messung mit 3D-Koordinatenmessmaschine, etc. umfassend zu dokumentieren.

Ausführlich zu dokumentieren sind das Format der Dateien (sofern nicht vorgegeben), der Inhalt mit zugehörigen Koordinatensystemen und Bezugsangaben, Einheiten der Daten, etc.
Es ist eine klare Versionierung vorzusehen. Die Ein- und Ausgabedateien müssen den Programmnamen und die Versionsnummer als Parameter enthalten. Berechnungsprogramme müssen beim Einlesen der Eingabedatei prüfen, ob die Programmversion kompatibel zur der in der Eingabedatei angegebenen Programmversion ist.

2.4 Programmierung

Auf eine klare Programmstruktur ist zu achten. Es ist als Maßgabe eine Subroutine pro File vorzusehen. Es sollen Module gebildet werden, die klare und dokumentierte Schnittstellen aufweisen. Es ist im Programm durchgängig eine Gliederung vorzusehen. In Abhängigkeit des Programmumfanges und des Berechnungsziels ist zu entscheiden, ob Datenstrukturen und eventuell objektorientierte Programmieransätze verwendet werden sollen. Ein Modul soll eine Sammlung von Subroutinen darstellen, die nur gemeinsam sinnvoll verwendet werden können.

Folgende Randbedingungen sind einzuhalten:

- Daten in Formalparameterleiste bzw. Schnittstelle in Ein-/Ausgabe trennen
- Innerhalb der Module globale Datenhaltung erlaubt
- Trennung: Berechnungsaufgaben am Element
- Bearbeitung komplexer Struktur (Gesamtgetriebe etc.) hat moderne Datenorganisation und Datenhaltung (Liste, Baum) zur Folge
- Errorbehandlung in Unterprogrammen sowie Rückgabe des Errorcodes (EXIT(XXX))

2.5 Entwicklungsumgebung

Als Entwicklungsumgebung ist INTEL FORTRAN Compiler für Windows in der aktuellen Version zu verwenden. Zusätzlich kann als Compiler GNU gfortran eingesetzt werden.
Der Quelltext ist in einem Versionsverwaltungssystem abzulegen, z.B. einem Subversion-System (svn).
Weitere Hilfsprogramme können nach Erfordernis verwendet werden, einige Beispiele sind in der folgenden Liste genannt:

- UNDERSTAND for FORTRAN Erstellt u.a. Abhängigkeitsgraphen der Software
- Visustin Erstellt Nassi-Shneiderman-Diagramme

2.6 Anforderungen an Quelltext

Die ausführliche Kommentierung des Quelltextes ist selbstverständlich. Es ist ein Data-Dictionary (Variablenkatalog) zu erstellen, in dem alle verwendeten Variablen dokumentiert sind. Die Ein- und Ausgabedaten sind in einem Datenflussdiagramm zu dokumentieren.
Es ist ein Standard-Header am Anfang jeder Routine zu verwenden. Es darf auf eine getrennte Variablendokumentation verwiesen werden, wenn im Quelltext keine vollständige Kommentierung enthalten ist.

2.7 FVA-Workbench-Integration

Folgende Punkte sind bei der Integration von Rechenprogrammen in die FVA-Workbench zu beachten:

- Rechtzeitige Absprache mit dem FVA-Kompetenzzentrum und der FVA-Workbench-Entwicklung über Nutzung der ProduktModellDatenBank (PMDB)
- Umsetzung der Definition von benötigten Attributen
- Umsetzung der Einordnung in Kategorien
- Darstellung, welche Attribute und Kategorien bereits vorhanden sind und welche neu anzulegen sind
- Erstellung von Filtern (Basis, Regulär, Gesamt)
- Erstellung von Logiktabellen (Vorlage siehe Anhang)
- Erstellung und Einordnung der Entwicklung in einem Berechnungsprozess (Berechnungsplan)

Die genannten Punkte müssen vom Entwickler des Rechenkerns bearbeitet werden. Die Programmintegration in die FVA-Workbench erfolgt auf Basis der Arbeitsergebnisse. Die erstellten Filter und Logiktabellen werden in die FVA-Workbench mit dem Rechenkern integriert.

Zur detaillierten Information über die Attribute der FVA-Workbench wurde die ProduktModellDatenBank (PMDB) angelegt.

URL der PMDB: http://www.plan-software.net/pmdb/

Ansprechpartnerin für einen Zugang ist die Fa. Plansoft.
Der Bugtracker zur Meldung von Fehlern in der FVA-Workbench ist die allgemeine Plattform zum Eintragen von Nachbesserungswünschen.

URL des Bugtrackers: http://www.plan-software.de/bugtracker/login_page.php

2.8 Dokumentation

Die zu implementierenden Funktionen und Gleichungen müssen vollständig definiert und dokumentiert werden.
Die Dokumentation ist in einen theoretischen Grundlagenanteil, eine Benutzeranleitung sowie eine Programmdokumentation zu gliedern. Folgende Hauptgliederungspunkte sollten enthalten sein:

Theoretische Grundlagen
- Überblick
- Entwicklungshistorie
- Theoretische Grundlagen
- Im Programm umgesetzte Formeln und Rechenmethoden

Benutzeranleitung
- Eingabemöglichkeiten
- Ergebnisausgaben
- Fehlerbehandlung
- Beispiele

Programmdokumentation
- Programmaufbau
- Struktur
- Subroutinen-Verzeichnis sowie Variablendokumentation
- Schnittstellen

Parametermanual (tabellarischer Anhang)
- Auflistung aller Parameter, die vom Programm gelesen werden
- Auflistung aller Parameter, die vom Programm ausgegeben werden
- Beschreibung des Parameters
    - Herkunft (Quelle: z.B. DIN, Konstruktionsmaß, physikalische Eigenschaft)
    - Zweck
    - Einfluss auf die Berechnung
    - Englische Übersetzung

Die Abschnitte sollen folgende Inhalte beschreiben:

Theoretische Grundlagen:
Der Überblick enthält eine Einführung in die Berechnungssoftware (Wozu dient die Berechnung? Was kann berechnet werden? Wie kann berechnet werden? Wo sind die Grenzen der Berechnung?)
Dieser Teil sollte maximal zwei Seiten betragen (ähnlich dem Info-Blatt).
Die Entwicklungshistorie soll einen knappen Überblick über den Leistungsumfang früherer Programmversionen geben und stellt die Release-Dokumentation dar.
Die theoretischen Grundlagen für die Berechnung sind zu dokumentieren. Die im fertigen Programm tatsächlich verfügbaren Rechenmethoden sind darzustellen.

Benutzeranleitung
Die Eingabephilosophie und -möglichkeiten, das typische Vorgehen bei der Definition der Berechnungsaufgabe, eine Beschreibung der Besonderheiten wie Materialdatenbanken, Zusatzdateien, etc. sind hier zu dokumentieren. Die Beschreibung der grafischen Oberfläche ist nicht Bestandteil des

Forschungsberichtes. Es muss jedoch eine Zuordnungstabelle zwischen den Eingabegrößen in der FVA-Workbench und den ASCII-Eingabevariablen zur Verfügung gestellt werden.
In der Benutzeranleitung sollen die Beispiele, die den Testern (und Endanwendern) zur Verfügung stehen, beschrieben werden. Es ist die konkrete Anwendungssituation, die Parameter (Bedeutung, Quelle, Gültigkeit) sowie die Interpretation und Diskussion der Ergebnisse und deren praktische Bedeutung für das berechnete System zu beschreiben.

Programmdokumentation
Die Entwicklungsdokumentation ist die Bedienungsanleitung für den Programmierer.
Sie beinhaltet weniger die Dokumente zur Entwicklung selbst, sondern behandelt die Nutzungsmöglichkeiten der zahlreichen, wieder verwendbaren Funktionen in den Anwendungssystemen durch die Softwareentwickler.

Anhand von Diagrammen ist der Programmablauf zu beschreiben. Die Programmstruktur kann getrennt dokumentiert werden, falls erforderlich.
Im Programm enthaltene Subroutinen sowie Variablen sind nach Name und Aufgabe zu dokumentieren. Die Ein- und Ausgabeparameter müssen eindeutig definiert sein. Es kann auf umfangreich dokumentierte Quelldateien verwiesen werden bzw. die Header der Quelltextdateien können zur Dokumentation abgedruckt werden.
Die Dokumentation der Programm-Schnittstellen bezieht sich auf alle Ein- und Ausgabedateien des Programms. Aufbau und Inhalt sind systematisch zu dokumentieren.

Unterlagen zum Workshop
Die Schulungsunterlagen müssen aus folgenden Dokumenten bestehen:

- Präsentation mit einer Einführung in die Theorie der Berechnungsgrundlage
- Eine vollständig erklärte Beispielrechnung (Praxisbeispiel)
- Schrittweise Anleitung durch das Programm und seine Möglichkeiten

Es ist ein Schulungskonzept zu entwickeln, welches die Berechnungsmöglichkeiten des Programms anhand eines Praxisbeispiels darstellt (dient auch zur Vorbereitung des BETA-Workshops).

Nach vollständiger Durchführung eines Beispiels sollten die Nutzer anhand eines weiteren dokumentierten Beispiels selbst das Programm erkunden. Nach erfolgreicher Durchführung der Berechnung und Interpretation der Ergebnisse sowie deren Bedeutung für die praktische Anwendung sollte der Teilnehmer versuchen eigene Fragestellungen zu lösen.

Während der Seminare sollten Sie alle Anregungen und/oder Fehlermeldungen dokumentieren und bei einer erneuten Überarbeitung des Programmes einfließen lassen.

2.9 Qualitätssicherung

Testfälle in Form von Beispieldatensätzen sind zu den wichtigsten Anwendungsfällen zu liefern: Eingabedatei für den Rechenkern (ASCII), Eingabe für die WB (wbpx), Textausgabe als Referenz.
Die Programmbeispiele sind in das Qualitätssicherungs-System der FVA-Workbench einzupflegen. Das QS-System ist online unter der URL der PMDB (siehe oben) verfügbar.

2.10 Programmabnahme und -herausgabe

Der Ablauf der Programmabnahme ist als definierter Prozess in der FVA festgeschrieben. Die Einzelheiten und Verantwortlichkeiten sind dem Qualitätsmanagementhandbuch der FVA-Workbench zu entnehmen. Die Programmabnahme wird im Rahmen einer Arbeitsgruppensitzung mit einem Standardformular vom Projektleiter aus der Industrie bestätigt. Im Folgenden wird ein Überblick über den Prozess gegeben:

2.10.1 Bereitstellen von BETA-Versionen

In der Testphase ist es wichtig, den Anwendern möglichst früh eine Testversion zur Verfügung zu stellen. Diese muss in ProMeta eingestellt werden. Der Ablageort für BETA-Versionen ist ein Ordner im Softwareverzeichnis. Dieser ist vom FVA-SoftwareService anzulegen. In diesem Ordner ist die BETA-Version des Programms abzulegen. Dies kann in Form einfacher, gepackter Dateien erfolgen. Die Dateien sind wie folgt zu kennzeichnen:

<Programmname_Erstellungsdatum>

Das Erstellungsdatum ist der Versionsstempel der BETA-Version. Diese sind im Ordner abzulegen und die Änderungen sind in der “History“ Datei zu beschreiben. Informieren Sie aktiv die Arbeitsgruppe über das Einstellen einer neuen BETA-Version sowie der Änderungen, die in der Version vorgenommen wurden.

2.10.2 Abnahme der Projektergebnisse

Um dem projektbegleitenden Ausschuss (AG) die Möglichkeit zu geben, sich über die Funktionsfähigkeit des Programmes zu informieren, sowie den potenziellen Anwendern die Gelegenheit zu bieten, sich in die Funktionalität der Software einzuarbeiten, muss zum Ende des Projektes ein BETA-Workshop veranstaltet werden. Dieser ist von der Forschungsstelle zu organisieren und durchzuführen. Dieser Workshop dient dazu, die Funktionsfähigkeit der Software zu belegen und den Mitgliedern der Arbeitsgruppe die Möglichkeit zum Testen der Software zu bieten.

2.10.3 Abnahme und Test der Programme

Prinzipiell gilt, dass die Entwicklung der Programme von Arbeitsgruppen zu begleiten ist. Diese Arbeitsgruppen sind nach Bewilligung des Vorhabens ins Leben zu rufen und sind der erste Ansprechpartner für die praktischen Anforderungen der Programme. Die Forschungsstelle hat sich aktiv um einen Termin mit der Arbeitsgruppe zu bemühen. Die AG sollte im Verlauf des Vorhabens mindestens dreimal eine Sitzung durchgeführt haben, um die Teilergebnisse des Vorhabens als zielführend zu bewerten.

Diese Arbeitsgruppe ist auch maßgeblich für die Abnahme des Programms zuständig. Die Abnahme hat in Form eines Testworkshops zu erfolgen. Diesen hat die Forschungsstelle eigenständig am Ende des Projektes einzuplanen. Der Testworkshop kann auch mit einem Übergabeseminar gekoppelt werden.

Entscheidend für den Erfolg dieses Workshops ist, dass mindestens drei Industrievertreter daran teilnehmen. Diese müssen nach Abschluss des Workshops einstimmig beschließen, dass das Programm in allen Hauptfunktionen fehlerfrei arbeitet. Die Teilnehmer des Workshops haben auf der nächsten AK-Sitzung die Aufgabe, dem AK die Freigabe zu empfehlen oder die im Workshop aufgedeckten Mängel zu berichten und das Institut aufzufordern, diese Mängel zu beheben.
Ist dies bis zur AK-Sitzung erfolgt, kann das Programm freigegeben werden und das Projekt ist abgeschlossen.

2.11 Archivierung

Das Softwareprojekt gilt erst als beendet, wenn die folgenden Vorgaben erfüllt sind.

Die Übergabe der Software erfolgt auf einem Speichermedium. Dieses Speichermedium kann je nach Art und Umfang der erstellten Software eine:

- CompactDisc (CD)
- Digital Versatile Disc (DVD)

sein.
Diese ist dem FVA-Kompetenz zu übergeben. Das FVA- Kompetenzzentrum bestätigt den Eingang der Daten und die Vollständigkeit.

Erst mit dieser Bestätigung erfolgt die Restzahlung der im Vertrag angegebenen Restsumme. Diese Bestätigung ist der FVA-Geschäftsstelle zur Prüfung vorzulegen.

Ein vollständiges Ergebnis einer Softwareentwicklung besteht aus:

- Dem vollständigen Quellcode
- Allen zur Erzeugung notwendigen Zusätzen (Libraries, Icon's, Packages, Resource Bundels, Objekt Dateien, etc.), die nicht in einer Standardentwicklungsumgebung enthalten sind.
- Der aktuellen Programmdokumentation
- Einer Protokolldatei der Entwicklungsgeschichte des Programms (Im ASCII_Format)
- Den Installationshinweise und eine Beschreibung der Methode zur Erstellung des Programms (Compiler Aufruf, Compiler Parameter, etc.).
- Beschreibung der Entwicklungsumgebung (Entwicklungsplattform mit Datum und Version, Compilerversion mit Datum, Compilereinstellungen, Versionsnummern der zusätzlichen Libraries, Angabe der Bezugsquellen von Zusatzmodulen oder Libraries)

Eine vollständige Verzeichnisstruktur muss mindestens den folgenden Aufbau haben:

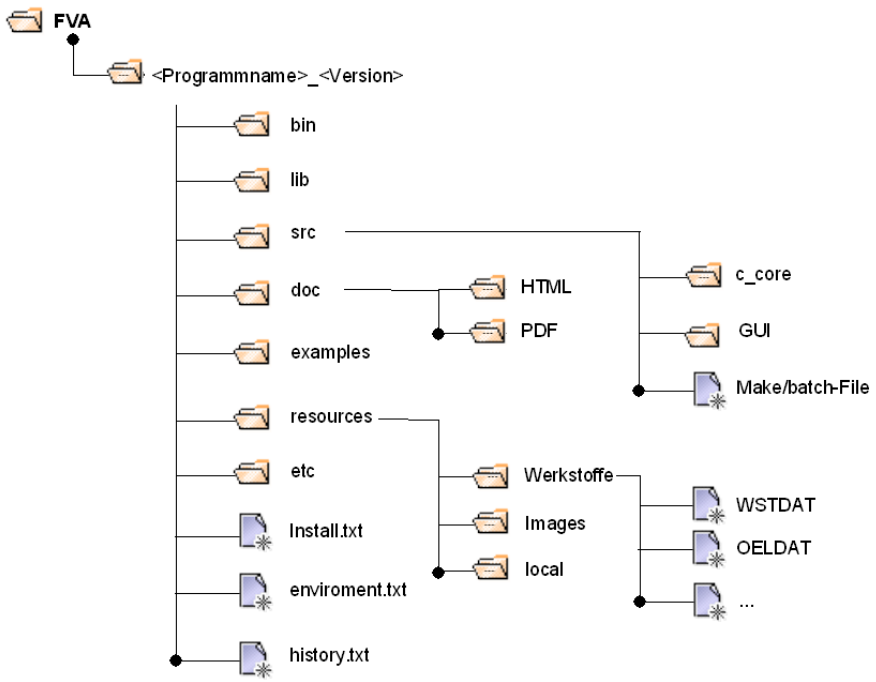

Abbildung 12: FVA-Softwareverzeichnisstruktur

Beschreibung der Dateien und Ordner:

- FVA = Der oberste Ordner auf dem Datenträger
- <Programmname>_<Version> 1= Ist der Ordner, in dem alle das Programm betreffende Daten gespeichert sind.
- bin = Hier werden alle ausführbaren Dateien abgespeichert. In diesem Verzeichnis muss sich mindestens eine unter Windows lauffähige Version befinden.
- lib = In diesem Verzeichnis sind alle Dateien (Bibliotheken) zu speichern, mit denen das Programm beim Linken verbunden wurde.
- src = In diesem Ordner werden alle Dateien, die zum Quellcode gehören, gespeichert. Zusätzlich zu den Quellen ist ein Makefile bzw. eine Workspace-Definition der verwendeten Entwicklungsumgebung zu liefern, die alle Einstellungen zur Übersetzung der Quellen und zur Erstellung des Anwenderprogramms enthält.
- doc = In diesem Ordner sind alle Dateien für die Dokumentation abzulegen. Dies gilt sowohl für die sogenannte Online-Hilfe als auch für die Programmbeschreibung im PDF- oder im HTML-Format.
- examples = In diesem Verzeichnis sind alle Beispieldateien zu speichern, die zum Veranschaulichen der Berechnungsmöglichkeiten des vorliegenden Programms dienen.
- Resources = Hier sind alle Dateien, die Zusatz Informationen enthalten, zu speichern. Dies gilt für Datenbanken, die für die Berechnungen benötigt werden, als auch für Bilddaten und Daten die für die Sprachanpassungen benötigt werden.
- Werkstoffe = In diesem Verzeichnis sind alle Werkstoffdatenbanken abzulegen.
- Images = Hier sind alle Bilder und Icons, die im Programm verwendet werden, abzulegen.
- local = Hier sind alle Dateien, die für die Internationalisierung der Programme benötigt werden, zu speichern.

Die Verzeichnisstruktur kann bei Bedarf erweitert werden:

- etc = Hier sind alle Dateien zu speichern, die zur Konfiguration und als Voreinstellungen der Programme dienen.
- Install.txt = In dieser Datei muss vom Programmersteller der Installationsvorgang beschrieben werden. Auf Besonderheiten ist hinzuweisen, vor allem auch auf Probleme, die bei einer Installation auf einer anderen Plattform auftreten können.
- Environment.txt = Hier ist die Entwicklungsplattform zu beschreiben und zu dokumentieren. (Auf welcher Plattform wurde das Programm entwickelt (Bezeichnung, Version, Datum)? Welche Werkzeuge wurden für die Entwicklung benutzt (IDE, Compiler, Libraries, Icon-Editoren etc.)
- history.txt = Hier sind alle Änderungen, die am Programm vorgenommen werden zu dokumentieren. Diese Datei wird bei der Ersterstellung des Programms angelegt und über den gesamten Lebenszyklus des Programms gepflegt. Anzugeben sind die Programmversion und die vorgenommenen Änderungen und Erweiterungen im Programm. Dies gilt für alle Programmteile.

Diese Vorgaben gelten für alle Berechnungskerne, die im Rahmen eines FVA Auftrages erarbeitet werden. Dies schließt auch Erweiterungen ein, welche für die Berechnungsplattform SIMPACK erarbeitet werden.
Die Verzeichnisbäume dürfen von der vorgegebenen Struktur und der Namenskonvention sinnvoll abweichen. Die Vollständigkeit der Informationen muss erfüllt bleiben.

2.12 Konvention zur Festlegung der Programmversion

Generell sind alle Programme mit numerischen Versionsnummern auszustatten. Die Nummerierung unterliegt folgenden Konventionen:

<Programmname>AA.BB.CC.EE_Build<ID>

AA = Die Nummer der sogenannten Major-Releases. Diese ändern sich nur, wenn signifikante Änderungen oder Erweiterungen am Programm vorgenommen werden. (Der Nutzwert der Software erhöht sich.)

Der Wertebereich für AA ist 1 – 99. Die Versionsnummer des Major-Release ist konsequent immer nur um einen Zähler zu erhöhen. Die Null ist bei einstelligen Versionsnummern wegzulassen.

Beispiel: Die Versionsnummer sollte wie folgt gestaltet werden: STPlus 4.01 -> STPlus 04.01 hingegen ist nicht zulässig.

Programme, die nur in der Majorversion verfügbar sind, sind wie folgt zu nummerieren:
Beispiel: PressFit 1.0

BB = Die Nummer der sogenannten Minor-Releases. Diese ändert sich typischerweise, wenn an dem Programm Bug-fixes oder kleinere Änderungen und/oder Erweiterungen vorgenommen werden. (Der Nutzwert der Software erhöht sich kaum.)

Der Wertebereich für BB ist 1-9. Diese muss aber nicht bis zur 9 weitergeführt werden, um einen Wechsel der Major-Release-Nummer durchzuführen.

Beispiel: BECAL 3.05 auf Becal 4.0 ist möglich.

CC = Diese Nummer dient zur Versionierung bei Behebung von Fehlern und/oder kleineren Korrekturen in Modulen oder Funktionen. Der Funktionsumfang der Software wird nicht geändert.

Der Wertebereich für CC ist 1-9.

EE = Versionsnummern, die für die interne Versionsnummerierung verwendet werden können. Es bietet sich an, anstatt einer rein numerischen Darstellung die Versionierung nach dem Erstellungsdatum zu klassifizieren.

Beispiel: BECAL 3.05_Build20050412. Es handelt sich hierbei um das Programm, das am 12.04.2005 erstellt wurde. In der Regel sollten diese Nummern nur zum internen Gebrauch oder zur Kenntlichmachung in der BETA-Phase verwendet werden. Es kann jedoch sinnvoll sein das man diese Nummer in Programmausgaben mit ausgibt. Somit kann identifiziert werden von welcher Version (und Binär-Datei) der Ausdruck stammt.

3 Literaturangaben:

FVA: FVA-Leitfaden, Frankfurt am Main, 2011

FVA: Qualitätsmanagementhandbuch für die FVA-Workbench-Entwicklung, Version 1.0, Frankfurt am Main, 11/2011

Intel Fortran Compiler V12: Internet-Quelle www.intel.com

Internet-Quelle: www.mantisbt.org

Internet-Quelle: Versionsverwaltungssystem Subversion, subversion.tigris.org

ISO/IEC 1539-1: 2010: Fortran, 2010

ISO/IEC 14882:2003: C++, 2003

Mulzer, F.: Getriebesystemanalyse in der FVA-Workbench, FVA-Nr. 555 V, Frankfurt am Main, 2010

Steingröver, K.: FVA-Programmierrichtlinie, FVA-Merkblatt 0/11, Frankfurt am Main 2001

Understand for FORTRAN / für C/C++

V-Modell-XT-Bund Stand 05.03.2010 Internet-Quelle http://download.4soft.de/

4 Anhang

- Formular zur offiziellen Abnahme des Programms
- Formular für die Prüfung durch das FVA-Kompetenzzentrum
- Vorlage zur Definition benötigter neuer Attribute in der FVA-Workbench
- Vorlage zur Erstellung von Logiktabellen
- Hinweis auf Filter/Konfigurationserstellung
- Verzeichnis der Ansprechpartner

4.1 Dokumentation der Änderungshistorie

Die History-Datei ist als reine ASCII-Datei auszuführen und im obersten Verzeichnispfad der Programminstallation (Plugin des Berechnungskerns) abzulegen. Der Inhalt entspricht der Entwicklungshistorie in der Dokumentation, bzw. ist detaillierter. Die History-Datei kann auf der Log-Datei aus dem eingesetzten Quelltextverwaltungssystem basieren.

Die Datei hat den Namen History.txt.

Beispiel:

History-Datei <Programmname> FVA Nr.: <xxx>

Version: x.x.x Datum: xx.xx.xxxx Bearbeiter:xx

Beschreibung von

- neuen Funktionen
- Erweiterungen
- beseitigten Fehlern

Version: x.x.x Datum: xx.xx.xxxx Bearbeiter:xx

Beschreibung von

- neuen Funktionen
- Erweiterungen
- beseitigten Fehlern

4.2 ProMeta Kurzbeschreibung

Damit das Programm nach Freigabe der AG und des AK in den offiziellen Downloadbereich der FVA aufgenommen werden kann, müssen die folgenden Angaben in Form einer wie folgt formatierten ASCII-Datei an das FVA-Kompetenzzentrum inklusive aller Quellen und kompilierten, ausführbaren Programm übergeben werden.

Folgende Angaben sind notwendig:

Name	STplus
Kategorie	Stirnräder
Zweck	Nachrechnung der Geometrie und der Tragfähigkeit von Evolventen-Radpaarungen.
Schnittstellen	ASCII, STEP, XML, etc.
Systemvoraussetzung	PC mit mindestens Windows 2000
Verantwortliches Institut	Forschungsstelle : Prof. Stahl, Prof. Höhn, FZG Garching
Beschreibung	STplus 6 ist ein Programm zur Geometrie- und Tragfähigkeits-berechnung von Evolventen-Stirnradpaarungen und der Paarung Stirnradwerkzeug, zur Geometrie-berechnung - auch bei Bearbeitung mit zwei unterschiedlichen Werkzeugen – sowie Sonderver-zahnungen, und zur Tragfähigkeits-berechnung nach unterschiedlichsten Rechenverfahren (DIN, ISO, AGMA, DNV, LRS, BV u.a.).
Ansprechpartner	Andreas Fröh (Prof. Stahl, Prof. Höhn, FZG Garching
Versions Kurzbeschreibung	Neue Version – mit Berechnung nach ISO 6336-2006
Aktuelle Version	6.0.3
Release-Datum	01.04.2011

Tabelle 2: Beispiel für eine ProMeta Kurzbeschreibung des Rechenkerns

Testbericht
zur offiziellen Freigabe für FVA-Programme

Forschungsvereinigung Antriebstechnik e.V.

Allgemeine Angaben

Berichtsdatum _____

Arbeitsgruppe _____

Name (des Berichtenden) _____

Firma (des Berichtenden) _____

Programm _____

Programm-Version _____

FVA-Workbench® Version _____

Angaben zum Test

(Diese Angaben sind von dem/der Federführer/in nach Abschluss des BETA-Tests auszufüllen.)

	JA	NEIN
Wurden Mängel im Programm festgestellt?	☐	☐
Wenn JA, bitte S. 2f. des Testberichts „Mängelbeschreibung“ ausfüllen →		
Sind die Ein- und Ausgaben des Programms praxisgerecht?	☐	☐
Sind genügend Testbeispiele vorhanden?	☐	☐
Sind die Testbeispiele nachvollziehbar?	☐	☐
Ist das Benutzerhandbuch ausreichend detailliert und verständlich?	☐	☐
Sind die Filter (Eingabe-/Ausgabewerte) auf den Berechnungskern anwendbar?	☐	☐
Programmstatus	☐	
Prototyp	☐	
Produktiv-Version	☐	
Zur FVA übergreifenden Anwendung empfohlen	☐	☐

Abnahmestatus

	JA	NEIN
Programm abgenommen	☐	☐
Programm abgenommen, sofern genannte Mängel beseitigt werden	☐	

→ Termin der Programmabnahme: _____

Notwendige Maßnahmen

_____

Datum / Unterschrift des Mitglieds der Arbeitsgruppe

Mängelbeschreibung

Lfd. Nr.	Beschreibung	Zuständigkeit Maßnahme	Termin	Klassifizierung		Verhindert Freigabe	
				Mangel	Erwei-terung	JA	NEIN
1						☐	☐
2						☐	☐
3						☐	☐
4						☐	☐
5						☐	☐
6						☐	☐

Programmname _____ **Datum** _____

Berichtender (Name, Vorname) _____

Firma _____

Mängelbeschreibung

Lfd. Nr.	Beschreibung	Zuständigkeit Maßnahme	Termin	Klassifizierung		Verhindert Freigabe	
				Mangel	Erwei-terung	JA	NEIN
7						☐	☐
8						☐	☐
9						☐	☐
10						☐	☐
11						☐	☐
12						☐	☐

Programmname _____ **Datum** _____

Berichtender (Name, Vorname) _____

Firma _____

FVA-Kompetenzzentrum
Rechenkern-/Oberflächeabnahme

Verantwortliches Institut: _____

Sachbearbeiter: _____

FVA-Vorhabensnummer: _____

Programmname: _____

Version: _____

Datum: _____

Bearbeiter FVA-Kompetenzzentrum: _____

Datum der Abnahme: _____

Durchführung der Tests:

Schritt	**OK**
Installation der FVA-Workbench mit zu testendem Berechnungskern	☐
Anlegen eines geeigneten Modells	☐
Durchführung der vom Institut zur Verfügung gestellten Berechnungsbeispiele auf dem Modell	☐
Laden eines älteren Modells (evtl.)	☐
Durchführung der vom Institut zur Verfügung gestellten Berechnungsbeispiele auf dem Modell	☐

	JA	**NEIN**
Konnte die Berechnung auf den Modellen erfolgreich abgeschlossen werden?	☐	☐

Wenn NEIN, auf welchen Modellen ist die Berechnung fehlgeschlagen?

_____

Stimmen die Berechnungsergebnisse mit den Ergebnissen aus der Berechnung ohne Oberfläche überein? ☐ ☐

Wenn NEIN, welche Ergebnisse weichen ab?

_____

Gibt es einen fachlichen Grund die Abnahme zu verweigern? ☐ ☐

Wenn JA, welchen?

_____

Programm wurde zur Beseitigung folgender Mängel zurückgewiesen an:

Verantwortliches Institut ☐

Oberflächenentwicklung / FDM ☐

Oberflächenentwicklung / Plan Software ☐

_____

_____

_____

_____

_____

_____

Programm Freigabe erteilt ☐ Programm Freigabe **nicht** erteilt ☐

_____

Datum / Unterschrift Bearbeiter des FVA-Kompetenzzentrum

FVA-Workbench®

Vorlage Attribut-Definition

Forschungsvereinigung Antriebstechnik e.V.

Freigaben					
Ersteller:			Datei:	Vorlage Attribut-Definition.doc	
Firma:	**Plan Software**	Unterschrift:	Pfad:		
Name:	**M. Demeisi**		Datum:	**04.10.2012**	
Tel.:	**0681-3792710**		Status:	**Revision 1.0**	
Prüfer I:			**Prüfer II:**		
Firma:		Unterschrift:	Firma:		Unterschrift:
Name:			Name:		
Tel.:			Tel.:		

Dokument-Historie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
1.0	Entwurf	04.10.2012	M. Demeisi	Erstellung der Vorlage

Inhaltsverzeichnis

1 Definition eines neuen Attributes

Bei der Definition von Attributen müssen folgende Angaben für das Attribut angegeben werden

Attribute dienen dazu, eine Komponente zu beschreiben und mit Standard-Werten vorzubelegen. Ein Attribut besitzt verschiedene Metainformationen (Merkmale), um das Attribut und dessen Kontext genauer zu beschreiben:

Nr.	Merkmal	Wert
1.	Komponente	
2.	Text-ID des Attributes (wird evtl. von der FVA angepasst)	
3.	Name des Attributes in deutscher Sprache	
4.	Name des Attributes in englischer Sprache	
5.	Formelzeichen	Definiert das Symbol, mit dem ein Attribut innerhalb von Editoren kenntlich gemacht wird.
6.	Maßeinheit (si Standard)	Falls es sich um ein Attribut mit Maßeinheit handelt, wird diese Angabe verwendet, um die Maßeinheit festzulegen.
7.	Attribute-Kategorie (falls bekannt)	Definiert die erste Gliederungskategorie für die Anzeige eines Attributs innerhalb von Editoren.
8.	Voreinstellungswert	Vorbelegung des Attributes
9.	Hinweise	Enthält Hinweise und Bemerkungen zum Attribut.
10.	Beschreibung	Detaillierte Beschreibung und Anwendung des Attributes.
11.	Rechenkern	Falls es sich ein rechenkern-spezifisches Attribut handelt, soll dieser Rechenkern angegeben werden.
12.	Berechnungsnorm	Falls das Attribut eine bestimmte Berechnungsnorm betrifft, ist diese anzugeben.
13.	Datentyp und Dimension (Bei Arrays und Matrizen die Dimension angeben)	Mögliche Daten-Typen : - Boolean - Integer - Double - String - Double mit Maßeinheit - Array mit Maßeinheit - Matrix mit Maßeinheit (zwei dimensionales Arrays)
14.	Constraints	Die Constraints definieren die Einschrän-

		kungen, die auf den Wert des Attributs angewendet werden müssen. Z.B.: Wert >= 0 0 <= Wert <= 1

1.1 Beispiel

Nr.	Merkmal	Wert
1.	Komponente	*planetary_gearing*
2.	Text-ID des Attributes (wird evtl. von der FVA angepasst)	*width of carrier*
3.	Name des Attributes in deutscher Sprache	Breite des Stegs
4.	Name des Attributes in englischer Sprache	width of planet carrier
5.	Formelzeichen	--
6.	Maßeinheit (si Standard)	si__milli_metre
7.	Attribute-Kategorie (falls bekannt)	Geometrieangaben
8.	Voreinstellungswert	
9.	Hinweise	
10.	Beschreibung	Gibt die Breite des Planetenträgers an. Gilt für einwangige und zweiwangige Planetenträger.
11.	Rechenkern	--
12.	Berechnungsnorm	--
13.	Typ und Dimension	*Double*
14.	Constraints	*0 <= Wert*

2 Vorlage zur Definition eines neuen Attributes

Bei der Definition von mehreren Attributen soll die Tabelle für jedes Attribut kopiert werden.

Nr.	Merkmal	Wert
1.	Komponente	
2.	Text-ID des Attributes (wird evtl. von der FVA angepasst)	
3.	Name des Attributes in deutscher Sprache	
4.	Name des Attributes in englischer Sprache	
5.	Formelzeichen	
6.	Maßeinheit (si Standard)	
7.	Attribute-Kategorie (falls bekannt)	
8.	Voreinstellungswert	
9.	Hinweise	
10.	Beschreibung	
11.	Rechenkern	
12.	Berechnungsnorm	
13.	Datentyp und Dimension (Bei Arrays und Matrizen die Dimension angeben)	
14.	Constraints	