# T U M

I N S T I T U T F Ü R I N F O R M A T I K

# Prozessintegration und -anpassung

Marco Kuhrmann

TUM-I0712
April 07

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-04-I0712-0/1.-FI


Druck: Institut für Informatik der
Technischen Universität München

# Prozessintegration und -anpassung

## Eine Fallstudie zur Konzeption und Ermittlung von Prozessintegrationsverfahren für das V-Modell XT

Marco Kuhrmann
Technische Universität München
Institut für Informatik, Software & Systems Engineering
Boltzmannstr. 3
85748 Garching, Germany
*kuhrmann@in.tum.de*

## Zusammenfassung

Das V-Modell XT ist ein modulares Vorgehensmodell, das zugunsten einer möglichst breiten Anwendbarkeit auf die Implementierung konkreter Methoden verzichtet. Während der Einführungsphase des V-Modells wurden eine Reihe von Pilotprojekten durchgeführt, die das neue V-Modell auf dessen Anwendbarkeit prüfen sollten. Hierbei wurde deutlich, dass neben dem zur Verfügung gestellten Rahmen weitere Verfahrensweisen und weiter gehende Werkzeugunterstützung erforderlich sind.

In diesem Beitrag werden Integrationsverfahren für das V-Modell XT diskutiert, die einerseits die Integration einer konkreten Methode, anderseits die Intgegrierbarkeit in ein Werkzeug beispielhaft beleuchtet. Am Beispiel des Microsoft Solutions Framework und des Visual Studio Team Foundation Server werden hier zwei Integrationspartner für das V-Modell XT näher untersucht.

**Keywords**

Sofware Engineering, Process Engineering, Software Development Process,
V-Modell XT, Microsoft Solutions Framework,
Visual Studio 2005, Team Foundation Server

**CR-Classification:** D.2

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1. Einführung und Überblick

Dieses Kapitel führt in die beiden zu betrachtenden Vorgehensmodelle ein. Es beleuchtet Sie anhand ihrer zugrunde liegenden Philosophien, Basisstrukturen und fasst einige ihrer Besonderheiten zusammen. Den Abschluss dieses Kapitels bildet eine kurze Gegenüberstellung und Abgrenzung der beiden Modelle. Es werden Optionen zur Integration skizziert und weitere Vorgehensweisen für die Analyse sowie die zu erwartenden Ergebnisse ermittelt, bzw. festgelegt.

## 1.1. V-Modell XT und Microsoft Solution Framework

Dieser Abschnitt stellt die beiden zu betrachtenden Vorgehensmodelle, bzw. Prozesse gegenüber. Erläutert werden hierbei insbesondere die Basiskonzepte und das Wording. Sofern erforderlich werden Begrifflichkeiten in einen weiteren Kontext gestellt. Die grundlegenden Inhalte wurden in [KT06] bereits herausgearbeitet. Sie werden hier noch einmal detaillierter dargestellt.

### 1.1.1. V-Modell XT – Überblick

Das V-Modell XT [RH06] ist als Nachfolger des V-Modell 97 (AU-250, [DW99]) im Rahmen eines Public-Private-Partnership Projekts konzipiert worden. Nach seiner offiziellen Vorstellung 2005 ist es für die Erstellung aller neuen IT-Systeme des Bundes im Geltungsbereich Bundesministerium des Innern (BMI) und Bundesministerium der Verteidigung (BMVg) verpflichtend anzuwenden. Gemäß Beschluss des Interministeriellen Koordinierungsausschusses (IMKA) vom 4. November 2004, ist es auch für alle anderen Ministerien auf der Bundesebene zur Anwendung empfohlen. Auch auf Länderebene setzt sich das V-Modell XT zunehmend weiter durch, wie zum Beispiel die Entwicklung des V-Modell Bayern zeigt.

#### Basiskonzepte des V-Modell XT

Das V-Modell ist ein modulares Vorgehensmodell, das auf strikte Erweiterbarkeit und Anpassbarkeit ausgelegt ist. Weiterhin ist das V-Modell XT den so genannten generischen Vorgehensmodellen zuzuordnen. Es adressiert in seiner Allgemeinheit eine Vielzahl von Anwendungsfällen (Projekttypen/-klassen), verzichtet dafür aber auf die Benennung konkreter Methoden. Ermöglicht wird dies durch die konsequente Orientierung an Projektergebnissen (Produkten). Anstelle des Weges werden also das Ziel und die an das Ziel gestellten Qualitätsanforderungen definiert.

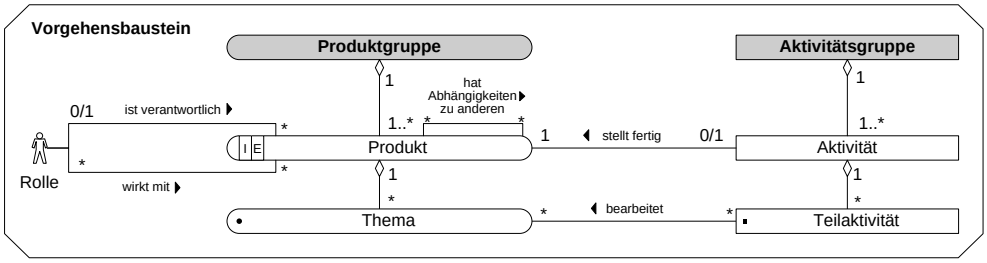



**Abbildung 1.1.:** Metamodell eines V-Modell XT Vorghensbausteins

Das V-Modell fasst die Ergebnisse und alle weiteren, den Ergebnissen angehörenden Elementen in Vorgehensbausteinen zusammen (Abbildung 1.1). Ein Vorgehensbaustein ist die modulare Einheit des V-Modells. Er fasst Produkte (Ergebnisse und Teilergebnisse), Rollen und Aktivitäten, sowie auf der fein granulareren Ebene auch Themen und Teilaktivitäten zusammen. Ein Vorgehensbaustein deckt in der Regel eine Projektdisziplin ab. Projektdisziplinen sind zum Beispiel: Projektmanagement, Anforderungsfeststellung oder aber SW-Entwicklung. Die Inhalte eines Vorgehensbausteins orientieren sich an den Disziplinen, sodass beispielsweise im Vorgehensbaustein Projektmanagement Produkte wie das Projekthandbuch, der Projektplan oder diverse Berichtsdokumenttypen definiert sind. Zu diesen gehören die entsprechenden Rollen wie Projektmanager oder Projektleiter. Ebenfalls sind die Aktivitäten, sowie wiederum die Themen und Teilaktivitäten mit Bezug zum Projektmanagement hier hinterlegt. Anlog verhält es sich zum Beispiel auch mit dem Vorgehensbaustein SW-Entwicklung (Abbildung 1.2), welcher aufbauend auf der Systemerstellung alle spezifischen Elemente zur Entwicklung von Software bereitstellt.

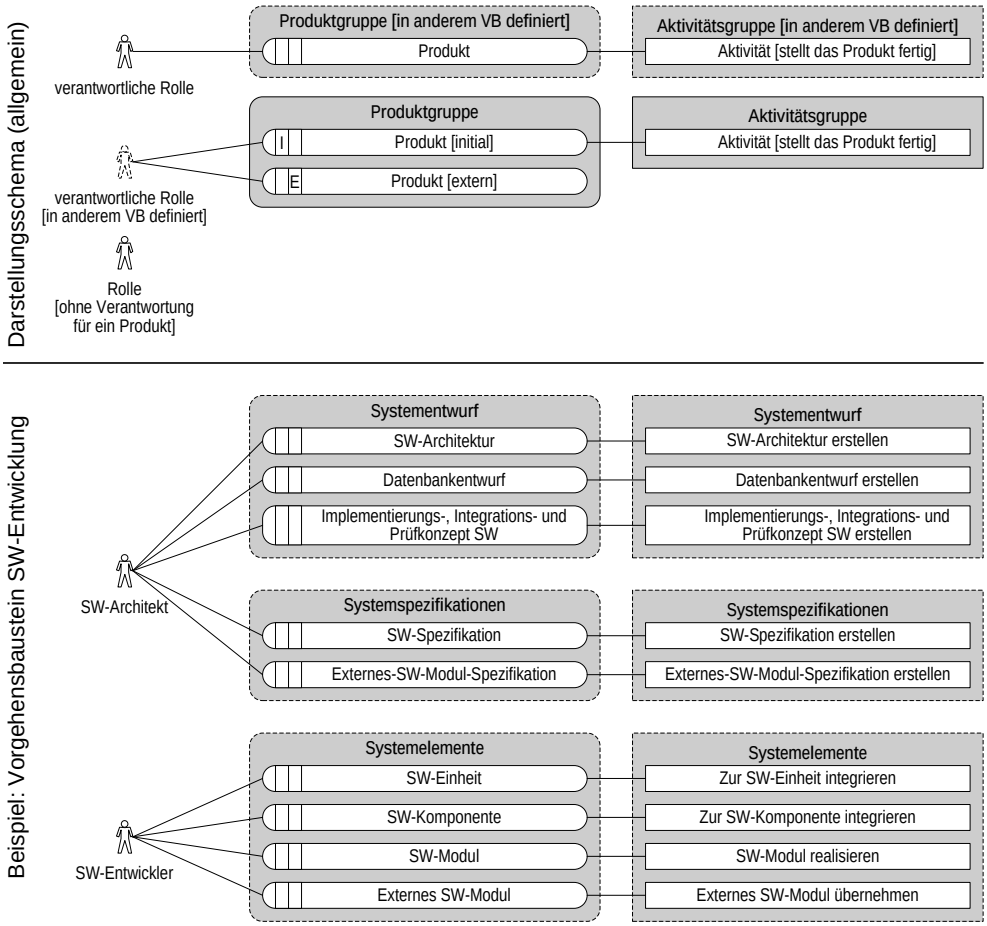



**Abbildung 1.2.:** Darstellung eines V-Modell XT Vorghensbausteins (oben) und ein konkretes Beispiel anhand des Vorgehensbausteins SW-Entwicklung

Vorgehensbausteine legen Strukturen und Ergebnisse fest. Die Ergebnisse definieren sie durch die Produkte, die Strukturen durch Produkte und Abhängigkeiten zwischen diesen. Sie machen jedoch keine Annahme über die Projektabwicklung, bzw. -durchführung als solche. Hierzu sieht das V-Modell die Projektdurchführungsstrategien (Abbildungen 1.3 und 1.4) vor. Eine Projektdurchführungsstrategie besteht aus einer Menge so genannter Entscheidungspunkte. Diese werden durch die Projektdurchführungsstrategie in eine Reihenfolge gebracht. Dazu definieren Projektdurchführungsstrategien die möglichen Pfade zwischen den Entscheidungspunkten – sie definieren somit also mögliche Abläufe eines Projekts. Jedem Entscheidungspunkt sind Produkte zugeordnet, die zu einem festgelegten Zeitpunkt in einem qualitäts-

gesichertem Zustand vorliegen müssen. Ein Entscheidungspunkt entspricht einem Meilenstein im Projektplan inklusive der Funktion eines Qualitygates. Die Kombination zwischen der Reihenfolge der Entscheidungspunkte und der Produktzuordnungen erlaubt eine detaillierte Planung eines V-Modell Projekts, da hierdurch eine Abarbeitungsfolge bestimmt wird.

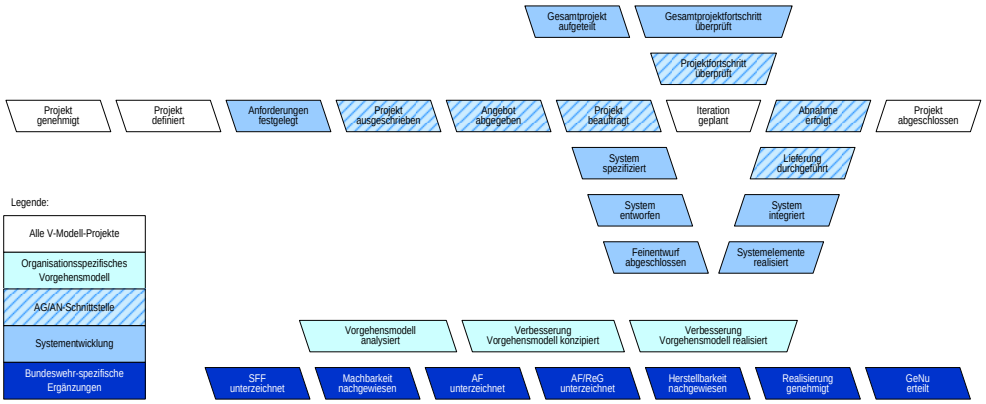



**Abbildung 1.3.:** Entscheidungspunkte des V-Modell XT. Aus diesen werden die Projektdurchführungsstrategien zusammengestellt.

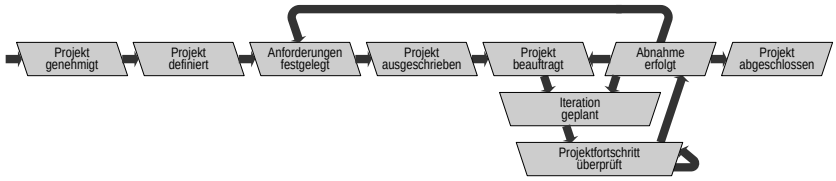



**Abbildung 1.4.:** Vergabe und Durführung eines Systementwicklungsprojekts (AG) als Beispiel für eine konkrete Projektdurchführungsstrategie

Das V-Modell definiert eine Vielzahl von Produkten, Aktivitäten und Rollen in mehr als 20 standardmäßig definierten Vorgehensbausteinen, 18 Entscheidungspunkten und mehreren Projektdurchführungsstrategien[1]. Um dieses Leistungsangebot auf konkrete Projektbedürfnisse hin anzupassen und das V-Modell somit erst anwendbar zu machen, kann der integrierte Tailoring-Mechanismus verwendet werden. Dieser kann aufgrund verschiedener Parametereinstellungen, bzw. aufgrund der Kombination von Vorgehensbausteinen zu einer erheblichen Reduktion des Umfangs des V-Modells führen [NK05]. Beispielsweise kann in Form eines so genannten Anwendungsprofils festgelegt werden, dass der Projektgegenstand eine Softwareentwicklung ist. Infolge dessen greift das Tailoring und entfernt im Rahmen der projektspezifischen Anpassung alle für die Softwareentwicklung nicht relevanten Teile. Die Philosophie des V-Modells ist: *Nur das muss gemacht werden, was notwendig ist*. Erfahrungen, die wir im Rahmen der Pilotierungsphase machen konnten [KB05, NK05] zeigen, dass hier eine Umfangsreduktion von bis zu 50% möglich ist.

### Anpassbarkeit des V-Modell XT

Erweiterbarkeit und Anpassbarkeit sind zwei wesentliche Anforderungen an das V-Modell XT. Ermöglicht wird diese Fähigkeit durch das **Metamodell** [RH06, Gna06], das dem V-Modell XT zugrunde liegt. Es definiert die wesentlichen Entitäten und die Beziehungen zwischen diesen. Dadurch, dass beispielsweise definiert ist, was ein Vorgehensbaustein ist, und was er enthält, ist es möglich, neue Vorgehensbausteine auf definierte Weise zu ergänzen. Das V-Modell sieht im Wesentlichen zwei

1 Die Anzahl ist abhängig vom V-Modell Derivat.

Ebenen der Anpassung vor (sofern man die Ausgestaltung, bzw. Implementierung im Projekt auslässt).

1. Anpassung auf der Unternehmens-, bzw. Organisationsebene (**organisationsspezifische Anpassung**)
2. Anpassung auf der Projekttypebene (**Tailoring**)

Bereits in [Kuh06] wurden die beiden Anpassungsebenen am Beispiel eins so genannten Handlungsraums gegenübergestellt und erläutert. Dies soll hier noch ein-

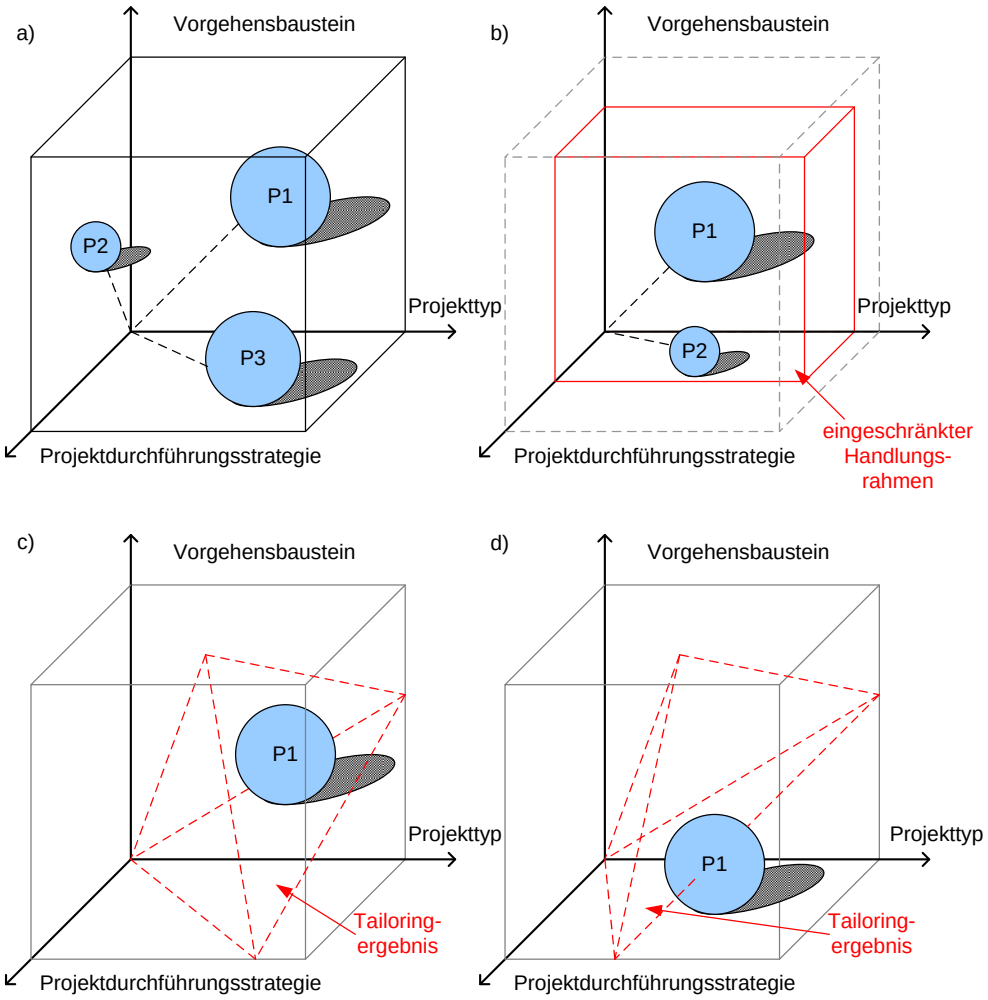



**Abbildung 1.5.:** Handlungsraum des V-Modell XT nach [Kuh06]

mal exemplarisch erfolgen. Abbildung 1.5 zeigt hierfür ein simplifiziertes Modell. Das V-Modell XT spannt durch seine vorgefertigten Bausteine (Vorgehensbausteine, Projektdurchführungsstrategien…) einen gewissen Aktions-, bzw. Handlungsraum auf. Aufgrund der hohen Generik des V-Modells ist dieser Handlungsraum sehr groß. Konkrete Projekte benötigen erfahrungsgemäß nur einen (geringen) Anteil der Funktionalität des Gesamtmodells (Abbildung 1.5a.). Es handelt sich also um einen Ausschnitt des Handlungsraums. Der Schnitt wird durch das Tailoring durchgeführt. Da das Tailoring aber entweder anhand der Projektmerkmale oder direkt auf den Vorgehensbausteinen [RH06] fest gemacht wird, welche zahlenmäßig beschränkt sind, werden auch konkrete Projekte gewissermaßen klassifiziert. So gibt es Projekte mit Softwareentwicklung, mit oder ohne Logistikkonzeption und so weiter. Tailoring ist hier eine Methode, das V-Modell auf bestimmte Problemklassen hin zu trimmen. Ein Rest Generik (Abbildung 1.5c.) bleibt auch bei der projektspezifischen Anpassung, dem Tailoring, erhalten. An dieser Stelle greift dann der Mechanismus der organisationsspezifischen Anpassung (Abbildung 1.5b.). Dieser passt den initialen Handlungsrahmen des V-Modells direkt an. Die Folgen sind:

- potenziell erweiterter oder präziser definierter Funktionsumfang
- Erweiterungen des V-Modell XT
- Reduktionen des V-Modell XT
- Spezialisierungen, z.B. neue, konkrete Methoden im V-Modell XT

Das V-Modell liefert also bereits selbst die notwendigen Mittel, um es an neue Gegebenheiten anzupassen. Sowohl auf der Werkzeugebene (V-Modell XT Editor) als auch auf der Modellebene selbst (eigener Projekttyp) ist das notwendige Handwerkszeug bereits vorhanden.

### 1.1.2. Microsoft Solution Framework - Überblick

Das Microsoft Solution Framework (MSF[2], [vB04, Tur06]) ist ein Prozessframework, das in seiner Grundintention dem V-Modell sehr ähnlich ist. Aus einem abstrakten (deskriptiven) Modell, dem MSF 4.0 Metamodell, können verschiedene konkretere Modelle abgeleitet werden. Zurzeit gibt es im Wesentlichen zwei Ausprägungen des Microsoft Solution Frameworks:

- MSF 4.0 for Agile Software Development und
- MSF 4.0 for CMMI Process Improvement

#### Historie

Der MSF ist seit etwa 1993 verfügbar. Er stellt eine Prozessbeschreibung dar, die aus der internen Entwicklung in Zusammenarbeit mit Partnern zu einen Produkt weiterentwickelt wurde. Das letzte Release 3.x [vB04] definierte bereits umfangreiche Konzepte wie ein Team- und ein Prozessmodell. Zu diesen kamen noch sog. Disziplinen als weitere Kernkomponenten hinzu. Das MSF stellt nicht nur ein Prozessmodell für die Softwareentwicklung dar, sondern verfügt weiterhin über eine Schnittstelle zum Microsoft-Standard für Softwarebetrieb, dem Microsoft Operations Framework (MOF[3]). Im Bereich der Werkzeugunterstützung war der MSF bislang jedoch eher dünn besetzt. Bis auf eine Projektvorlage in Microsoft Project 2003 (MSF Anwendungsentwicklung) gab es keine wirkliche Unterstützung.

#### Werkzeugintegration des MSF

Seit der Version 4.0 ist der MSF direkt in das Visual Studio 2005 integriert. Die Integration erfolgt einmal serverseitig im Backend, einmal clientseitig bei den Mitarbeitern im Projekt. Im Backend wird der MSF durch den Team Foundation Server (TFS[4]) realisiert. Dieser stellt Infrastruktur und Datenablage, sowie Reportingmechanismen bereit. Clientseitig spiegelt sich der MSF in Abhängigkeit der konkreten Rolle wieder. Projektmanager sehen beispielsweise Excel, Project oder das Teamportal (siehe Abbildung 1.6) als Frontend, während Entwickler vorrangig mit dem Visual Studio 2005 Team System [GP06] arbeiten werden.

#### Konzepte des MSF

Der MSF basiert ebenfalls auf einigen grundlegenden Konzepten. Viele davon sind mehr im Bereich der Richtlinien und Empfehlungen anzusiedeln und nicht klar formal erfassbar. Dennoch basiert der MSF ebenfalls auf einem Metamodell. Dieses Metamodell teilt sich im Wesentlichen in zwei Teile:

- einen strukturellen und
- einen operationellen Teil.

2 Informationen zum MSF unter: `http://msdn.microsoft.com/vstudio/teamsystem/msf`
3 Informationen zum MOF unter: `http://www.microsoft.com/technet/itsolutions/cits/mo/mof/default.mspx`
4 Informationen zum TFS unter: `http://msdn.microsoft.com/vstudio/teamsystem/team`


**Abbildung 1.6.:** Die MSF Process Guidance im Teamportal am Beispiel: MSF Agile

Im strukturellen Teil des Metamodells werden Entitäten definiert, wie zum Beispiel Rollen, Produkte oder Aktivitäten (Abbildung 1.7). Dieser Teil ist mit dem entsprechenden aus dem V-Modell XT vergleichbar. Im operationellen Teil des Metamodells werden Strukturen beschrieben, wie sie für die Instanziierung und Durchführung eines Projekts notwendig sind. Dazu zählen beispielsweise die Zusammenhänge zwischen Checkpunkten, Workstreams und Iterationen (Abbildung 1.8). Detaillierte Informationen hierzu finden Sie in Anhang A.

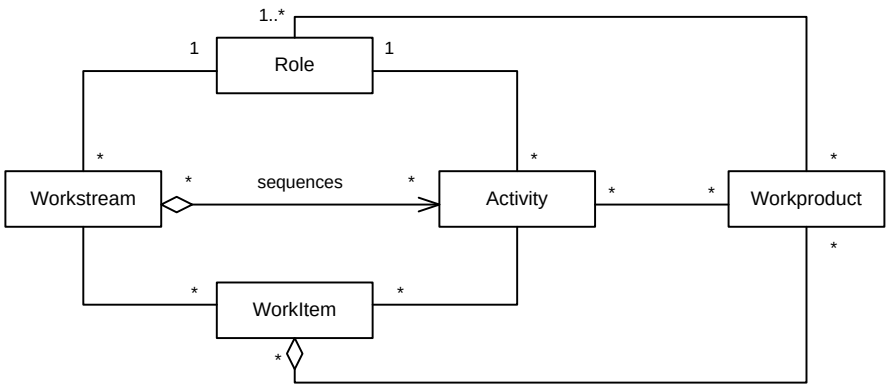



**Abbildung 1.7.:** Strukturelles Metamodell des MSF

Ein Beispiel für einen Workstream, wie er in Abbildung 1.8 zu sehen ist, ist der Workstream Guide Iteration[5]. Dieser Workstream fasst Aktivitäten des Projektmanagers zusammen, die notwendig sind, um eine Iteration eines Projekt zu planen, durchzuführen und zu bewerten.

5 Die Prozesselemente, auf die hier Bezug genommen wird, sind alle dem MSF Agile entnommen.

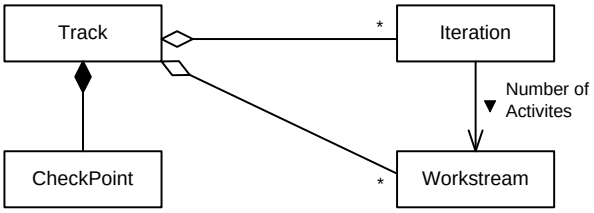



**Abbildung 1.8.:** Operationeller Metamodellanteil des MSF

#### Anpassung des MSF

Der MSF kennt kein Tailoring, so wie es das V-Modell XT versteht, kann aber trotzdem weit gehend angepasst werden. Die erste Anpassungsstufe verläuft ähnlich dem V-Modell XT. Auf der Basis des Metamodells sind verschiedene Ausprägungen ableitbar. Dies kommt dem Gedanken eines V-Modell XT Basis-, bzw. Referenzmodells sehr nahe. MSF Anwender erhalten also im Wesentlichen schon ein vorgetailortes Vorgehensmodell. Das Konzept des Handlungsrahmens (vgl. Abbildung 1.5) ist auch hier anwendbar.

Darüber hinaus kann auch der MSF noch weiter an die eigenen Bedürfnisse angepasst werden. Das Visual Studio Software Development Kit (SDK[6]) sieht extra hierfür entsprechende Guidelines vor und stellt auch die wesentlichen XML Schemata zur Verfügung. Der Team Foundation Server stellt ebenfalls eine Möglichkeit der Anpassung bereit. Ein bereits installiertes Process Template kann in eine Verzeichnisstruktur exportiert, dort bearbeitet und angepasst und im Anschluss wieder reimportiert werden. Auf diese Art und Weise ist es vergleichsweise einfach, ein konkretes Vorgehen für ein Projekt auf der Basis des MSF anzupassen.

### 1.1.3. Positionierung und Einordnung

Für die Entwicklung von Integrationskonzepten für die beiden Prozessmodelle ist es notwendig, die Ähnlichkeiten, bzw. Überdeckungen/Überschneidungen zu erfassen. In [KT06] haben wir hierzu schon erste Ansätze gezeigt, die wir hier wieder aufgreifen und weiter vertiefen wollen. Hierzu greifen wir auf einige allgemeine Kriterien zurück, die wir im Wesentlichen schon alle angerissen, jedoch noch nicht vertieft haben.

#### Kriterien zur Einordnung und Positionierung

Allgemeine Kriterien zur Einordnung und Positionierung der beiden Vorgehensmodelle werden aus dem Kontext der Umfänge gewählt. Die Auswahl erfolgt dabei anhand der wichtigsten Modellelemente (Rollen, Artefakte etc.) sowie beispielsweise anhand der Anwender-, bzw. Werkzeugunterstützung.

**Rollenkonzepte** Dieses Kriterium zielt auf die Art, wie Projektbeteiligte im Projekt gesehen werden. Dies betrifft die Art der Definition von Rollen, Aufbau der Beschreibungen oder Umfangsdefinition.

**Aktivitäten** Dieses Kriterium beleuchtet die Art der Projektdurchführung im Kontext der Projektbeteiligten. Wie werden Aufgaben zugeordnet und beschrieben. Wie ist es um die Detailtiefe der Aufgabenbeschreibung bestellt?

**Ergebnisse** Hier werden die Ergebnistypen betrachtet, die ein Projekt hervorbringt.

**Anwenderunterstützung** Hier wird betrachtet, wie die Anwender des Prozesses Unterstützung erfahren. Dabei ist zunächst zu klären, wer überhaupt adressiert

6 Siehe auch:

`http://zfs.in.tum.de/blogging/zfs_blogging.aspx?ItemID=e2a0aa43-25c6-47ce-a9a0-2267ec45d785`

wird und wie derjenige dann Hilfe bekommen kann, z.B. durch Instruktionen verschiedenster Granularität.

**Dokumentation** Obwohl auch die Dokumentation eine Form der Anwenderunterstützung ist, wird sie gesondert betrachtet. Hier wird beleuchtet, welche Dokumentation zur Verfügung steht und wie sie Anwendern oder Interessierten angeboten wird.

**Werkzeugunterstützung** Hier wird die verfügbare Werkzeugunterstützung des Anwenders näher betrachtet.

Alle Kriterien, die wir hier aufgezählt haben, dienen explizit nicht der Bewertung eines Vorgehensmodells, sondern dienen auf grob granularer Ebene der Identifikation möglicher Verbindungspunkte. Die Verbindungspunkte müssen bereits im Vorfeld möglichst genau identifiziert werden, um zuverlässige Aussagen über die Potenziale der Integration treffen zu können. Dabei ist es zunächst unerheblich, in welche Richtung die Integration erfolgen soll.

### Positionierung von V-Modell XT und MSF

Es lassen sich auch ohne einen detaillierten Abgleich der beiden Modelle bereits im Vorfeld Aussagen machen, wie die beiden Modelle grob zueinander zu positionieren sind.

| Kriterium | V-Modell XT | MSF |
|---|---|---|
| Rollenkonzept | Rollenmodell für Verantwortlichkeiten auf Produktebene | Rollenmodell auf Aktivitätsebene |
| Aktivitäten | abstrakte Beschreibung grober Abläufe | Workstreams, Akitivtäten und WorkItems mit klar definierten, präzisen Anweisungen, z.T. werkzeugunterstützt |
| Ergebnisse | Produkte (Dokumente, Code etc.) | WorkProducts (Dokumente, Code, Modelle, Programme etc.) |
| Anwenderunterstützung | Prozessbeschreibung, Beispiele, Vorlagen | Dokumentation, Vorlagen, automatisierte Arbeitsschritte |
| Dokumentation | V-Modell XT | HTML-basiert |
| Werkzeugunterstützung | OSS, komerziell für Prozessauthoring und Projektdurchführung | integriert im Visual Studio 2005, einige OSS Add-ons und Utilities |

**Tabelle 1.1.:** Vergleichskriterien für die beiden Prozessmodelle zur Ermittlung potenzieller Intergationskapazitäten

Das V-Modell XT ist als generisches Vorgehensmodell umfassender als der MSF. Es besteht auch explizit darauf, möglichst vielen Anwendungsbereichen gegenüber offen zu sein, wofür es dann aber auf konkrete Methoden verzichtet. Das V-Modell XT ist somit auf einer eher abstrakten Ebene anzusiedeln. Ein Blick auf die Tabelle 1.1, die die gerade aufgestellten Kriterien abfragt und gegenüberstellt, unterstreicht dies. Der MSF auf der anderen Seite ist ein viel konkreteres Vorgehensmodell. Es orientiert sich an genau einer Zieldomäne – der Softwareentwicklung. Hierfür bietet es aber dann auch entsprechende Methoden und Werkzeuge an. Wichtig an dieser Stelle: Obwohl der MSF nicht explizit in einer rein dokumentarischen Form wie das V-Modell vorliegt, kann er auch mit Abstrichen ohne Werkzeuge – also als Leitfaden – verwendet werden.

## 1.2. Bewertung von Integrationsmöglichkeiten

Dieser Abschnitt stellt die Integrationsoptionen beider Prozesse zusammen. Grundlage und Ausgangspunkt hierfür ist die Analyse des MSF im Anhang A. Unter Berücksichtigung der Ergebnisse aus [KT06] und Bezug nehmend auf Tabelle 1.1 leiten wir hier zwei Integrationsschritte ab und motivieren die weiteren Arbeiten zur Prozessintegration des MSF im V-Modell (Kapitel 2) und weitergehend auch die Potenziale des Einsatzes des V-Modells im Kontext der Microsoft Werkzeuge als Alternative, bzw. Ergänzung zum MSF (Kapitel 3).

### 1.2.1. Identifikation von Integrationsverfahren

Zunächst soll jedoch die Analyse mit der Art der erwarteten Ergebnisse in Relation gesetzt werden. Grundlage hierfür bilden die möglichen Ergebnistypen einer Integration des V-Modells und des Microsoft Solutions Framework. Generell sind drei Wege der Integration beider Vorgehensmodelle in Betracht zu ziehen:

1. Integration des MSF als **konkrete Softwareentwicklungsmethode** in das V-Modell, das heißt also Ausgestaltung, bzw. Verfeinerung des V-Modells durch MSF-Inhalte.
2. Integration unter Beibehaltung der Zuständigkeitstrennung, das heißt: Das V-Modell bildet den **Mangementrahmen** für die Methode MSF.
3. Integration/Bereitstellung des V-Modell XT in die TFS Infrstruktur mithilfe eines **Process Templates**.

Die gerade aufgezählte Option der Intergation des V-Modells als Managementrahmen für den MSF ist hier nur eine mögliche Spielweise. Unter der Annahme, dass eine angemessene Abbildung zwischen den Einzelelementen von V-Modell und MSF existiert, steht einer fachlichen Modellierung des V-Modells in der Sprache des MSF nichts im Wege. Als direkte Konsequenz ist eine Integration des V-Modells in das Visual Studio Team System, insbesondere als Prozess in den Team Foundation Server, als Option in Betracht zu ziehen. Kapitel 3 befasst sich mit dieser Thematik.

### 1.2.2. Variantendiskussion

Die erste Variante bedarf genauer Betrachtung, da hier frühzeitig der Handlungsrahmen abzustecken und somit die Integration zu begrenzen ist. Die Betrachtungen in Anhang A haben dies anhand des problematischen Abgleichs der Strukturelemente gezeigt. Dort wurde ein mögliches Modellierungsverfahren abgeleitet, das auf der fachlichen Ebene die Überführung von Strukturen des MSF in die Sprache des V-Modells gestattet. Der zu erwartende **Informationsverlust** dieser interpretativen Abbildung wurde dabei noch nicht abgeschätzt.

Die zweite Variante bietet eine einfache, struktur- und vorgehenserhaltende Integrationsvariante an. Auf der einen Seite bleibt der MSF (unabhängig welches Derivat) weitestgehend unberührt. Das V-Modell XT wird hier jedoch um seine Systemerstellungsanteile reduziert. Es verbleibt ein allgemeiner Projektmanagementrahmen vergleichbar mit Prince 2 [Köh06]. Problematisch an dieser Stelle ist die Art und Weise, wie das V-Modell eine Sicht über der Systemerstellung aufspannt. Hier ist genau zu prüfen, wie die Kopplung der Managementanteile des V-Modells mit den Systementwicklungsanteilen des MSF herzustellen ist. Weiterhin muss auf der Seite des MSF wieder die Diskussion geführt werden, wie mit potenziellen Informationsverlusten umzugehen ist. Im Gegensatz zur Variante eins, ist das Verfahren zur Integration potenziell einfacher (es findet an Stellen statt, an denen der MSF vergleichsweise wenig Aussagen macht), jedoch ist die „Streichmasse" beim V-Modell XT derartig hoch, dass hier unter Umständen eine V-Modell Konfomität nicht mehr gewährleistet werden kann (Entfall des Systemerstellungsanstzes = Philosophiebruch).

Die dritte Variante beinhaltet das größte Potenzial, da hier die Infrastruktur des MSF nicht als „Anhängsel" an das V-Modell fungiert, sondern als **Zieldomäne** dafür. Analog zum ersten Punkt muss hier jedoch auch wieder bewertet werden, wie

Umfangreich die Integration werden soll. Dies stellt sich zum aktuellen Diskussionszeitpunkt als notwendig dar, da es wenig sinnvoll ist, das V-Modell in seiner generischen Form im Team Foundation Server (TFS) abzubilden, was der Intention dieser Arbeit auch widersprechen würde. Vielmehr ist es erforderlich, eine angemessene „Menge V-Modell" im TFS abzubilden. Die inhaltliche Ausgestaltung kann dabei zu Teilen aus den Erkenntnissen der Variante eins einfließen.

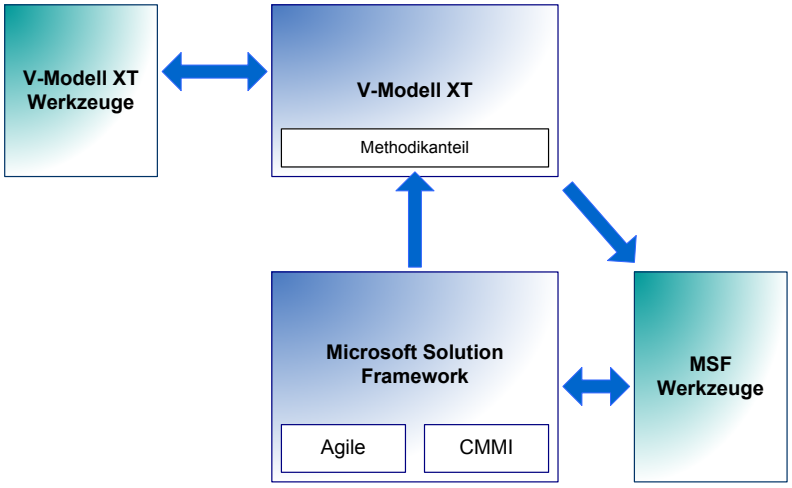



**Abbildung 1.9.:** Mögliche Integrationswege des Microsoft Solution Frameworks mit dem V-Modell XT

In Abbildung 1.9 stellen sich die möglichen Pfade noch einmal dar. Der MSF – unabhängig, ob Agile- oder CMMI-Derivat, geht als Methodikanteil in das V-Modell ein (Variante 1 der obigen Liste). Verwendung finden hier die Standardwerkzeuge des V-Modell XT, also der **V-Modell XT Editor** und der **V-Modell XT Projektassistent**. Bei dieser Verfahrensweise erfolgt die Analyse und Integrationskonzeption ausgehend vom V-Modell mit einer geeigneten Modellierung des MSF „in der Sprache" des V-Modells (Kapitel 2). Gegenstand der Integration sind hier weniger die Konzepte des MSF, sondern viel mehr seine Ergebnistypen.

Der zweite durch Abbildung 1.9 vorgesehene Pfad schlägt die **Integration des V-Modells in den Team Foundation Server** unter Berücksichtigung der gerade geführten Diskussion vor.

Die Variante zwei (V-Modell XT als Managementrahmen für den MSF) wird im Folgenden aufgrund der zu erwartenden Probleme bei nicht optimaler Aufwand-/Nutzenbilanz nicht weiter verfolgt.

## 1.3. Integrationskonzeption

Nach der gerade geführten Diskussion um mögliche, sinnvolle Integrationskonzepte, soll dieses Kapitel mit einer finalen Auswahl und Zieldefinition schließen. Ergänzend zu den Zielen werden gleichzeitig die bestehenden Rahmenbedingungen (als Anforderungen) beschrieben.

### 1.3.1. Integrationskonzept 1: Der MSF als Methode im V-Modell XT

**Beschreibung:** Das Microsoft Solutions Framework wird als konkrete Methode im V-Modell XT modelliert. Es stellt somit eine Konkretisierung des (generischen) Systementwicklungsprozesses dar, wie er durch das V-Modell beschrieben wird.

**Ergebnisse:** Als Ergebnis wird der MSF mit den Mitteln des V-Modells modelliert. Die Modellierung wird dabei so weit getrieben, dass einer Implementierung mit dem V-Modell XT Editor nichts im Wege steht. Dabei entstehen V-Modell XT Elemente für den MSF als Methoden. Konkret entstehen Vorgehensbausteine, Projektdurchführungsstrategien, Produkt-, Aktivitäts- und Rollendefinitionen.

**Mittel:** Zur Umsetzung stehen die Quellen des MSF und des V-Modell XT in der Version 1.2Bw (reduziert um die Inhalte Bw) zur Verfügung. Die Modellierung erfolgt auf der Basis der Analysearbeiten aus Anhang A. Zur Umsetzung werden die Werkzeuge des V-Modells verwendet.

### 1.3.2. Integrationskonzept 2: Das V-Modell XT als Prozess im Team Foundation Server

**Beschreibung:** Das V-Modell XT ist in geeigneter Form in den Team Foundation Server zu integrieren und somit als Entwicklungsprozess im Umfeld Visual Studio 2005 Team System zur Verfügung zu stellen.

**Ergebnisse:** An dieser Stelle entsteht ein Integrationskonzept mit prototypischen Betrachtungen. Dieses kann als Vorstufe einer prototypischen Implementierung dienen und ist somit grundlegende Voraussetzung für eine Umsetzung im TFS.

**Mittel:** Zur Umsetzung stehen die Quellen des MSF (Strukturquellen aus dem Visual Studio SDK), die Quellen des V-Modell XT sowie die aktuellen Versionen der Authoringtools für den MSF zur Verfügung.

## 1.4. Lebenszyklus der Konzepte und Anpassungen

Beide Integrationsverfahren müssen sich im Lebenszyklus des V-Modells passend integrieren. Abbildung 1.10 zeigt hierzu ein vereinfachtes Modell des V-Modell Lebenszyklus, das bereits um Anteile für die Integration des MSF enthält. Die Abbil-

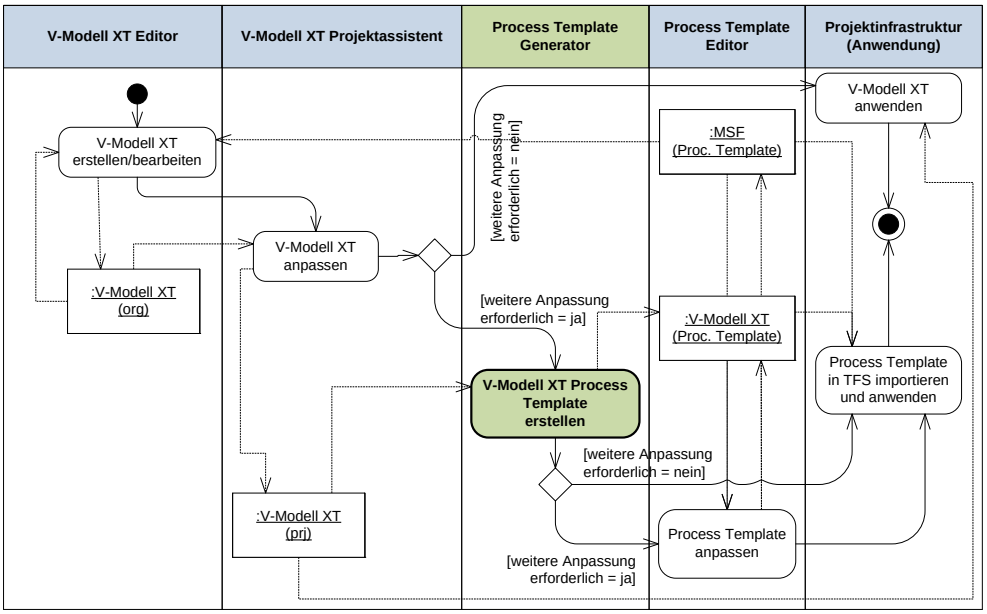



**Abbildung 1.10.:** Einordnung der Anpassungs- und Integrationsprozesse in den V-Modell Lifecycle

dung 1.10 ordnet die einzelnen Vorgänge grob den dazu verwendeten Werkzeugen zu. In der Authoring- und Tailoring-Phase sind dies die Standardwerkzeuge des V-Modells. Im Anschluss sind Generatoren vorzusehen (siehe auch Kapitel 3.1), die die Resultate des Tailorings weiter verarbeiten. Ziel dieser Generatoren ist der **Process Template Editor**, der einerseits das Zielformat für den Team Foundation Server festlegt, andererseits aber bereits das Format für den MSF definiert hat.

## 1.5. Ergebnisse

Im weiteren Verlauf vertiefen wir die Betrachtung der gerade aufgeführten Integrationskonzepte. Wir skizzieren in Kapitel 2 einen Prototyp für die Integration der MSF Anteile in das Systementwicklungsmodell des V-Modells. Im Kapitel 3 beschreiben wir darauf aufbauend die Optionen der Integration des V-Modells in die Werkzeuglandschaft des MSF, insbesondere im Team Foundation Server.

Als Ergebnistypen beschreibt dieser Bericht einen Architekturprototyp für die Integration des MSF in das V-Modell XT sowie ein Konzept zur Integration des V-Modell XT in das Visual Studio 2005 Team System, beziehungsweise den Team Foundation Server als Management- und Entwicklungsmodell.

# 2. Entwurf des MSF für das V-Modell XT

Dieses Kapitel stellt einen Architekturdurchstich für die Integration des MSF als Methode im V-Modell XT vor. Die grundlegenden Überlegungen hierzu entstammen [KT06]. Das Ziel dieses Durchstichs[1] ist es, den MSF mit den Mitteln des V-Modells zu beschreiben. Konkret bedeutet das, dass der MSF in Vorgehensbausteinen remodelliert wird. Vor dem Hintergrund dieser Modellierung, wurde bereits in Anhang A Wert darauf gelegt, sinnvoll modellierbare Elemente zu identifizieren und in strukturell sinnvollen Einheiten zusammenzufassen.

Ein weiteres Ziel des Durchstichs ist es, die Verträglichkeit der beiden Vorgehen auf der Metamodellebene zu zeigen. Unabhängig von automatischen oder formal korrekten Verfahren [Fri06] wird hier ein möglichst pragmatischer Ansatz verfolgt, der auf eine unmittelbare Überführbarkeit und Anwendbarkeit zielt. Umfangsmäßig umfasst der Durchstich das allgemeine Konzept und Inhalte im Volumen des MSF Agile-Derivats. um die weitergehende Anwendung die Möglichkeiten des V-Modells zu zeigen.

## 2.1. Modellierung der Strukturen

Zunächst befassen wir uns mit der Modellierung der Strukturen des MSF im Kontext des V-Modells. Im Anhang A haben wir hierzu schon weit reichende Untersuchungen getätigt. Als Ergebnis ist festzuhalten, dass der MSF sehr wohl mit den Mitteln des V-Modells darstellbar ist – hierzu sind jedoch Entscheidungen hinsichtlich der strukturellen Abbildung und Überführung der Inhalte zu treffen. Die Ab-

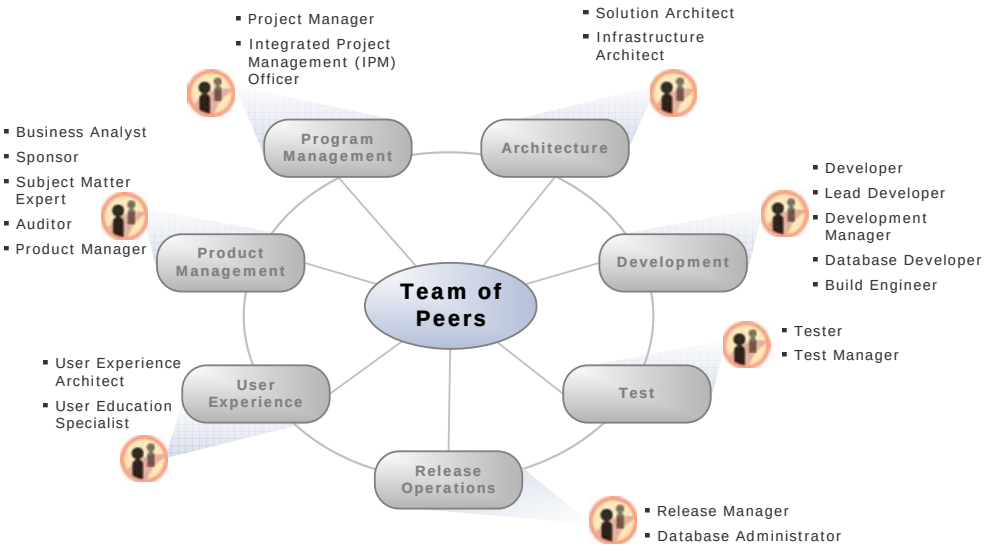



**Abbildung 2.1.:** Rollenmodell des MSF unter Berücksichtigung der Advocacy Groups nach [Tur06]

bildung 2.1 zeigt die so genannten **Advocacy Groups** des MSF nach [Tur06]. Die Advocacy Groups sind dabei im Wesentlichen mit den einzelnen **Projektdisziplinen** vergleichbar, die die grundlegenden Vorgehensbausteinstrukturen und weiter gehend die thematischen Gruppierungen in den Vorgehensbausteinen bilden. Sie dienen uns daher als Ausgangspunkt für die Modellierung.

1 Im Rahmen von [Aly06] wurden ebenfalls einige Erprobungen zur Integration des MSF in das V-Modell XT gemacht.

### 2.1.1. MSF-Architektur im V-Modell XT

Ein weiterer Punkt, der bei der Abbildung zu beachten ist und bereits im Anhang A berücksichtigt wurde, ist die grundlegende Architektur des MSF im Kontext V-Modell XT. Der MSF selbst stellt neben einem abstrakten Basismodell zurzeit zwei konkrete Ausprägungen bereit: das Agile- und das CMMI-Derivat. Beide Prozessederivate verfügen dabei über gewisse (strukturelle und inhaltliche) Gemeinsamkeiten. Mit Berücksichtigung auf die anstehende Abbildung ins V-Modell wurde die Analyse in Anhang A bereits so durchgeführt, dass beide Deriavate angemessen berücksichtigt werden. Abbildung 2.2 zeigt die Zielarchitektur. Der MSF wird in Form

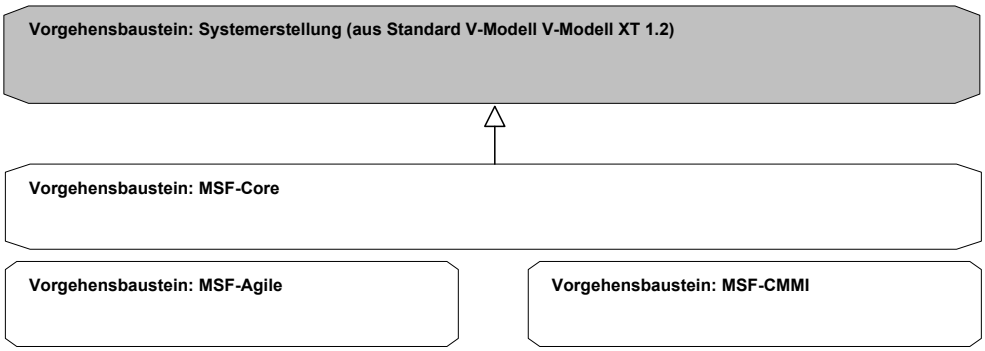



**Abbildung 2.2.:** Integrationsvorgehen und Zielarchitektur für die Integration des MSF in das V-Modell XT

von drei Vorgehensbausteinen entworfen und im V-Modell integriert:

**MSF-Core:** Der Vorgehensbaustein **MSF-Core** enthält alle Elemente, die beiden MSF-Derivaten gemein sind. Er legt weiterhin den Grundstein für die Strukturelle Abbildung des MSF im V-Modell. Dieser Vorgehensbaustein wird in diesem Bericht detailliert entworfen und umgesetzt.

**MSF-Agile:** Dieser Vorgehensbaustein enthält die Differenz zwischen MSF-Core und dem Agile-Derivat. Dieser Vorgehensbaustein wird im Kontext dieser Arbeit zwar weitgehend vollständig entworfen, jedoch nicht in derselben Detailtiefe, wie der Core-Baustein.

**MSF-CMMI:** Dieser Vorgehensbaustein enthält die Differenz zwischen MSF-Core und dem CMMI-Derivat. Dieser Vorgehensbaustein wir nur skizziert. In den meisten der folgenden Diskussionen betrachten wir ihn auch als Blackbox.

### 2.1.2. Vorgehensbausteine

Die Abhängigkeitstruktur der Vorgehensbausteine ist so gestaltet, dass der MSF-Core-Baustein auf dem V-Modell Baustein **Systemerstellung** basiert. Die beiden konkreten Derivate wiederum basieren auf dem Core-Baustein. Mit diesem Verfahren lässt sich Redundanz weitgehend vermeiden, jedoch nicht vollständig ausschließen. Insbesondere im Architekturbereich tritt Redundanz auf der Ebene der konkreten Derivate auf. Anhang A.1.2 und A.3.3 widmen sich dieser Thematik.

Im Folgenden entwerfen wir die beiden Vorgehensbausteine **MSF-Core** und **MSF-Agile**. Das CMMI-Derivat lassen wir außen vor.

#### Der Vorgehensbaustein MSF-Core

Der Vorgehensbaustein MSF-Core enthält alle wesentlichen Elemente, die beiden Prozessderivaten gemein sind. Er basiert inhaltlich auf der Modellierung und Analyse aus Anhang A. Original übernommen aus den MSF-Derivaten sind die Produkte (basierend auf den MSF *Work Products*) und die Rollen. Die Zuordnungen, die in Anhang A, insbesondere in den Tabellen A.4, A.6 und A.7 sind im Kontext des V-Modells jedoch noch einmal genauer zu betrachten. Abbildung 2.3 zeigt den Vorge-

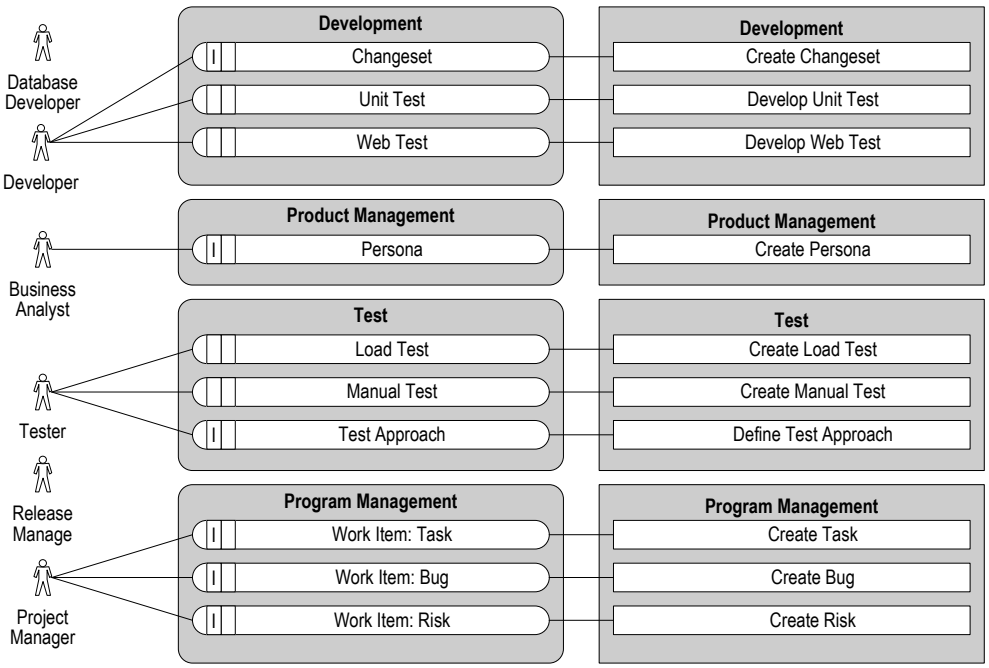



**Abbildung 2.3.:** Der Vorgehensbaustein MSF-Core

hensbaustein **MSF-Core** in der V-Modell XT Notationsform für Vorgehensbausteine (Kapitel 1.1.1). Problematisch ist insbesondere der Unterschied in den Philosophien der beiden Vorgehensmodelle. Der MSF definiert als aktivitätsorientiertes Vorgehen **Produktflüsse**, in denen Artefakte in verschiedenen Aktivitäten bearbeitet werden können (vgl. Anhang A.3.2, Tabelle A.5). Dieses Verfahren kann im V-Modell XT nicht abgebildet werden.

Als Lösungsansatz wurde hier ein Verfahren gewählt, in dem „Placebo-Aktivitäteten" auf der Produktebene eingeführt werden. Das V-Modell dekomponiert top-level Aktivitäten in Teilaktivitäten, die in verschiedenen Ablaufformen durchlaufen werden können. Beispielhaft sei hier die Aktivität **Develop Unit Test** genannt. Diese Aktivität ist nicht durch den MSF definiert, sondern durch die V-Modell Integration. Der MSF ordnet dem Work Product **Unit Test** die drei Aktivitäten: Fix a Bug, Implement a Development Task und Implement a Database Development Task zu. Jede dieser Aktivitäten darf gleichberechtigt mit einem Unit Test interagieren. Die Intergration in das V-Modell geht deshalb folgenden Weg:

```
Unit Test (A)--> Develop Unit Test
                 (TA)--> Fix a Bug
                 (TA)--> Implement a Development Task
                 (TA)--> Implement a Database Development Task
```

Dieses Verfahren wird für alle Produkte bis auf die **Work Items** und Aktivitäten homogen angewendet. Die Work Items verlangen nach einer besonderen Behandlung, auf die wir im Anschluss noch einmal detailliert eingehen.

```
Changeset      (A)--> Create Changeset
                  (TA)--> Fix a Bug
                  (TA)--> Implement a Development Task
                  (TA)--> Implement a Database Development Task

Unit Test      (A)--> Develop Unit Test
                  (TA)--> Fix a Bug
                  (TA)--> Implement a Development Task
                  (TA)--> Implement a Database Development Task

Web Test       (A)--> Develop Web Test
                  (TA)--> Test a Quality of Service Requirement
                  (TA)--> Test a Scenario
```

```
Persona        (A)--> Create Persona
                  (TA)--> Test a Quality of Service Requirement
                  (TA)--> Test a Scenario
                  (TA)--> Capture Project Vision

Load Test      (A)--> Create Load Test
                  (TA)--> Test a Quality of Service Requirement

Manual Test    (A)--> Create Manual Test
                  (TA)--> Test a Quality of Service Requirement
                  (TA)--> Test a Scenario

Test Approach  (A)--> Define Test Approach
                  (TA)--> Test a Quality of Service Requirement
                  (TA)--> Test a Scenario
                  (TA)--> Guide Project
```

Wie der oben stehenden Liste schon zu entnehmen ist, finden sich die Teilaktivitäten zum Teil gar nicht angemessen im Core-Baustein wieder. Aus diesem Grund, modellieren wir im Vorgehensbaustein MSF-Core nur bis auf die Ebene der Aktivitäten und verlagern die Definition der Teilaktivitäten in die konkreten Methodenbausteine. Das V-Modell gestattet dies durch weitreichende Verknüpfungsverfahren.

### Work Items

Noch nicht berücksichtigt sind an dieser Stelle die Work Items. Work Items sind durch die Kombination MSF und Team Foundation Server als Datenbankeinträge, bzw. Datenelemente definiert. Work Items stellen einen Artefakttyp dar, der über

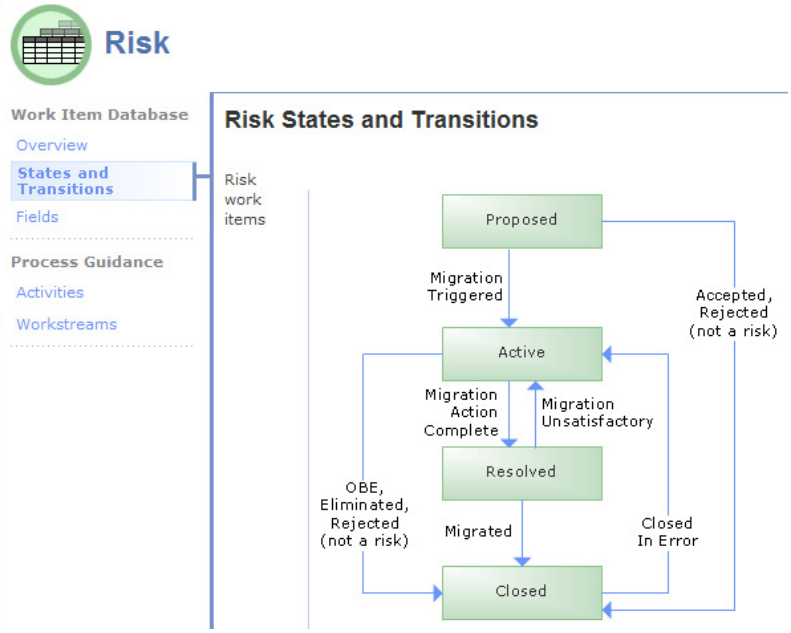



**Abbildung 2.4.:** Statusmaschine des Work Items „Risk" – entnommen der MSF CMMI Dokumentation

die Möglichkeiten des V-Modells hinaus geht. Jedes Work Item besteht aus einer Datenstruktur und einer Menge definierter Operationen die Übergänge zwischen verschiedenen Stati erlauben (siehe Abbildung 2.4 und Anhang B). Sie tragen somit eine eigene Methodik und definieren quasi im „Vorbeigehen" Projektdisziplinen wie **Risiko-** oder **Aufgabenmanagement**. Aufgrund des rein deskriptiven Charakters der vorliegenden Integration gehen dieses Informationen verloren und wandern in die jeweiligen Beschreibungen von Produkt und Aktivität. Work Items stellen einen echten Mehrwert gegenüber V-Modell Produkttypen dar. Bei der Konzeption einer V-Modell-Integration für den TFS (Kapitel 3) sind diese Fähigkeiten besonders von Wert.

### Der Vorgehensbaustein MSF-Agile

Der Vorgehensbaustein **MSF-Core** allein definiert keine Methode und liefert fast keinen wesentlichen Nutzen. Um ein Projekt nach MSF durchführen zu können, sind

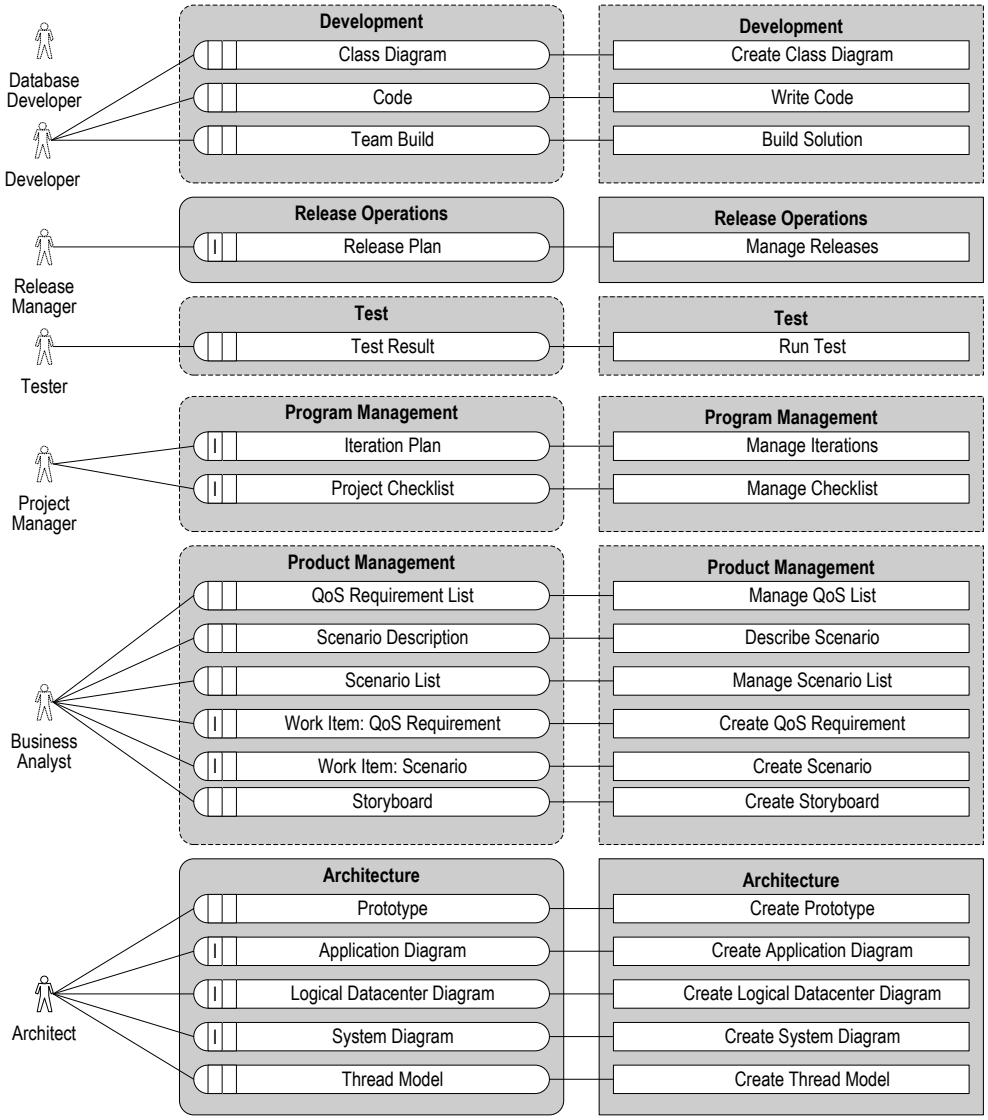



**Abbildung 2.5.:** Der Vorgehensbaustein MSF-Agile

weitere Elemente notwendig, die in Form der beiden Derivate Agile und CMMI gegeben werden. Dieser Abschnitt stellt den Vorgehensbaustein **MSF-Agile** vor. Ergänzend zum Core-Baustein zeigt Abbildung 2.5 die Inhalte des Agile-Bausteins. Die Gruppen- und Rollendefinitionen werden aus dem Core-Baustein übernommen (Beziehung: *basiert auf*). Neu hinzukommen die Produkt- und Aktivitätsgruppen:

- Release Operations und
- Architecture

Zur Rolle der Architektur siehe auch Anhang A.1.2 und A.3.3. Die Verfahrensweise zur Handhabung der Abbildung von Work Products auf Produkte (nach V-Modell XT), der Behandlung von Aktivitäten und Work Items ist identisch mit dem Verfahren im Core-Baustein. Ergänzend dazu definiert dieser konkretere Baustein die oben ausgeschlossenen Teilaktivitäten.

### 2.1.3. Projektdurchführungsstrategien

Das Vorgehen des MSF Agile ist ebenfalls geeignet in einen V-Modell kompatiblen Ablauf zu überführen. Im Wesentlichen sind dabei die iterativen Modelle des MSF (Anhang A.1, Seite 37) auf das V-Modell zu übertragen. Einen Ansatz hierzu skizziert Anhang A.4 und [KT06]. Jedoch ist die Standard Projektdurchführungsstrategie des V-Modells für Inkrementelle Systementwicklung zu umfangreich und schwergewichtig für den MSF Agile[2]. Um die aktuellen Integrationsbestrebungen konsequent weiter zutreiben, wird an dieser Stelle eine eigens an den MSF Agile angepasste Projektdurchführungsstrategie **Sw-Development using MSF 4 Agile** (Abbildung 2.6) entworfen[3].

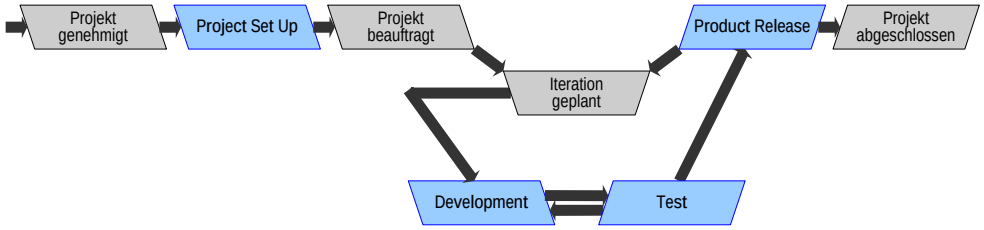



**Abbildung 2.6.:** Die Projektdurchführungsstrategie: Sw-Development using MSF 4 Agile

Diese Projektdurchführungsstrategie beachtet die wesentlichen Elemente des V-Modells (Projektstart, Abschnittsplanung und Projektabschluss). Weiterhin wird ergänzend der Vertragsschluss aus dem V-Modell XT übernommen, da der MSF diesen Abschnitt eines Projekts nicht konsequent umsetzt. Dann werden vier neue Entscheidungspunkte definiert:

- Project Set Up
- Development
- Test
- Product Release

**Entscheidungspunkt: Project Set Up** *Project Set Up* ersetzt den Entscheidungspunkt *Projekt definiert*, da der MSF zwar ähnliche Projektdefinitionsmechanismen vorsieht wie das V-Modell, hier jedoch wesentlich kompakter zu Werke geht (Work Product: **Project Checklist**). Der auf *Projekt definiert* eigentlich folgende Entscheidungspunkt *Angebot abgegeben* entfällt ersatzlos. Lediglich die Beauftragung mittels *Projekt beauftragt* bleibt erhalten, muss jedoch gegebenenfalls noch ergänzt werden.

**Entscheidungspunkte: Development und Test** Ebenfalls neu hinzugekommen sind die beiden Entscheidungspunkte *Development* und *Test*. Diese beiden Entscheidungspunkte bilden die im MSF-Agile Interationsmodell explizit ausgewiesenen Entwicklungs- und Testabschnitte ab. Sämtliche entwicklungsbezogenen Work Products beziehen sich auf diese beiden Entscheidungspunkte. Um jedoch weiterhin die Flexibilität des MSF zu erhalten, wird hier nicht weiter reglementiert, sondern einfach ein Wechsel zwischen Entwicklungs- und Testabschnitt gestattet, ohne die aktuelle Iteration zu verlassen (vgl. Abbildung A.1, Seite 37). Der gesamte Systementwicklungsanteil des V-Modells wird somit durch die MSF-Agile Methode ersetzt. Die durch das V-Modell XT erzeugte hierarchische Systemstruktur entsteht ebenfalls[4].

2 Für eine Implementierung des CMMI-Derivats hingegen ist diese Projektdurchführungsstrategie nahezu perfekt und muss lediglich bei der Instanziierung mit zusätzlichen freien Meilensteinen versehen werden

3 Um die konkreten Koppelstellen hervorzuheben, werden die nachgenutzten Elemente des V-Modells unverändert übernommen, während die neuen Element sich an den englischen Originalbezeichnern orientieren. Englische Elemente sind somit Erweiterungen.

4 Im CMMI-Derivat bleibt die Systemerzeugnisstruktur weit gehend erhalten.

**Entscheidungspunkt: Product Release** Ebenfalls neu hinzugekommen ist der Entscheidungspunkt *Product Release*, der im Wesentlichen die beiden Entscheidungspunkte *Lieferung durchgeführt* und *Abnahme erfolgt* zusammenfasst. Der MSF-Agile sieht hier ein weniger formales, einfaches Modell vor, das verlangt, dass am Ende einer jeden Iteration ein lauffähiges Produkt vorliegen muss. Der Releasebau (also Build, Paketierung, Lieferung etc.) ist somit immer Bestandteil einer Iteration. In der letzten Iteration liegt das finale Release vor, sodass hier wieder in den vom V-Modell übernommenen Projektabschluss gewechselt werden kann.

### 2.1.4. Produktzuordnungen

Nachdem die neue Projektdurchführungsstrategie definiert wurde, sind nun noch die Produkte aus Abschnitt 2.1.2 geeignet an die Entscheidungspunkte zu binden. Die Folgende Liste gibt die Zuordnungsvorschrift der einzelnen Entscheidungspunkte an:

```
Projekt genehmigt
                    --> Bewertung der Ausschreibung
                    --> Projektfortschrittsentscheidung

Project Set Up
                    --> Project Checklist (ersetzt PHB!)
                    --> Iteration Plan
                    --> Release Plan
                    ----------------------------------
                    --> Projektfortschrittsentscheidung

Projekt beauftragt
                    --> Vertrag (von AG)
                    --> Projektplan
                    --> Projektfortschrittsentscheidung

Iteration geplant
                    --> Project Checklist
                    --> Iteration Plan
                    --> Release Plan
                    --> Changeset
                    --> Test Approach
                    ----------------------------------
                    --> Projektplan
                    --> Projektfortschrittsentscheidung

Development
                    --> Changeset
                    --> Unit Test
                    --> Web Test
                    --> Class Diagram
                    --> Code
                    --> Team Build
                    --> Prototype
                    --> Application Diagram
                    --> Logical Datacenter Diagram
                    --> System Diagram
                    --> Thread Model
                    --> Persona
                    --> QoS Requirement List
                    --> Scenario List
                    --> StoryBoard
                    ----------------------------------
                    --> Projektfortschrittsentscheidung

Test
                    --> Changeset
                    --> Unit Test
                    --> Web Test
                    --> Team Build
                    --> Prototype
                    --> Load Test
                    --> Manual Test
                    --> Test Approach
                    --> Test Result
```

```
                    ----------------------------------
                    --> Projektfortschrittsentscheidung

Product Release
                    --> Release Plan
                    --> Changeset
                    ----------------------------------
                    --> Projektfortschrittsentscheidung

Projekt abgeschlossen
                    --> Projektabschlussbericht
                    --> Projektfortschrittsentscheidung
```

**Produktzuordnung und Vorlage zur Qualitätssicherung** In der oben stehenden Abbildungsvorschrift sind alle relevanten Zuordnungen von Produkten zu Entscheidungspunkten gezeigt. Problematisch daran ist, dass durch das V-Modell selbst noch weitere Produkte hinzukommen, beziehungsweise auf der anderen Seite nicht alle MSF-Produkte, die oben aufgezählt sind auch zur Qualitätssicherung vorgelegt werden müssen.

## 2.2. Integration der Strukturen in das V-Modell XT

Der Entwurf von Vorgehensbausteinen und Projektdurchführungsstrategien genügt, um die Strukturen des MSF zu modellieren, nicht jedoch, um ihn als Methode angemessen im V-Modell zu integrieren. Hierfür ist es erforderlich, die Abhängigkeitsstrukturen aufzulösen und die Position des MSF in den einzelnen gültigen Tailoringprofilen zu definieren.

### 2.2.1. Tailoring

Als erstes legen wir dafür die Eigenschaften im Kontext des Tailorings fest und erstellen ein entsprechendes Profil, das den MSF adressiert. Die für uns relevanten Merkmale, beziehungsweise Kriterien sind in Tabelle 2.1 aufgezählt. Wir lassen den MSF als Option in der Softwareentwicklung für Auftragnehmerprojekte[5] zu. Hardwareentwicklung wird nicht abgedeckt, jedoch unterstützen wir durch den Merkmalswert „eingebettetes System" auch die Erstellung von Programmen, beispielsweise mit dem .NET Compact Framework, für so genannte Smart Devices.

| Merkmal/Kriterium | Wert |
|---|---|
| Projekttyp | Systementwicklungsprojekt AN |
| Projektrolle | AN ohne UAN |
| Gegenstand | SW-System |
| Gegenstand | eingebettetes System |
| Lebenszyklusausschnitt | Entwicklung |

**Tabelle 2.1.:** Wertebelegungen und Profileigenschaften/Auswahlkriterien für den MSF im V-Modell

Es ist mit diesem Entwurf erforderlich, die beiden in Tabelle 2.1 genannten Projekttypen den Vorgehensbaustein **MSF-Core** als optionalen hinzuzufügen. Ebenso sind die beiden Vorgehensbausteine für die Derivate „Agile" und „CMMI" zu behandeln. Beachtet werden muss jedoch die exklusive Auswahl der Vorgehensbausteine. So sollen nur die beiden Paare $(Core, Agile)$ und $(Core, CMMI)$ verfügbar sein. Eine Kombination beider oder das alleinige Verwenden von „Core" ist nicht erwünscht. Es ist an dieser Stelle also nicht nur erforderlich, die entsprechenden Projekttypen zu

5 AG/AN-Projekte sehen wir in der aktuellen Modellierung nicht vor. Jedoch sind diese durch kleinere Anpassungen ebenfalls denkbar.

erweitern, sondern darüber hinaus auch noch ein neues Projektmerkmal **MSF-Derivat** einzuführen. MSF-Derivat ist dabei die Wertemenge $\{--, Agile, CMMI\}$ zugewiesen, die zu folgenden Kombinationen führt:

```
--     -> MSF-Core, Agile und CMMI        := Deaktiviert
       -> PDS: SW-Development MSF...      := Deaktiviert

Agile -> MSF-Core, Agile                  := Aktiviert
       -> MSF-CMMI                        := Deaktiviert
       -> PDS: SW-Development MSF...      := Aktiviert
       -> Standard VM-PDSe                := Deaktiviert

CMMI   -> MSF-Core, CMMI                  := Aktiviert
       -> MSF-Agile                       := Deaktiviert
       -> PDS: SW-Development MSF...      := Deaktiviert
       -> PDS: Inkrementelle SE-AN        := Aktiviert
       -> Standard VM-PDSe                := Deaktiviert
```

Ziel ist es, durch dieses Merkmal einzusteuern, in welcher Kombination dei MSF-Komponenten in ein V-Modell Projekt eingebracht werden. Prinzipiell ist lediglich sicher zustellen, dass bei der Auswahl des Agile-Derivats die richtige Kombination aus Vorgehensbausteinen und die Agile-PDS gewählt wird. Analog gilt das für das CMMI-Derivat, wobei hier standardmäßig jeweils die Inkrementellen Projektdurchführungsstrategien vorausgewählt werden sollen.

**System- und Produkterzeugung** Weitere strukturelle Integrationsaufgaben sind in Hinblick auf das „Einklinken" der neuen Inhalte in das Abhängigkeits- und Systemerzeungungsnetz des V-Modells. Einen ersten Eindruck haben wir bereits beim Entwurf der Entscheidungspunkte und der entsprechenden Produktzuordnung gegeben (Abschnitte 2.1.3 und 2.1.4). Darauf aufbauend sind aber beispielsweise noch die erzeugenden oder inhaltlichen Produktabhängigkeiten zu modellieren. Insbesondere die strukturellen Produktabhängigkeiten, die für die Systementwicklung durch den MSF vorgesehen sind, müssen noch abgeglichen werden. Da es sich hier um die Abbildung der jeweiligen Entwicklungsmodellen mit ihren verschiedenen Sichten und Abstraktions- beziehungsweise Kompositionsebenen handelt, wollen wir dies hier nicht weiter vertiefen. Aus der Erfahrung geht jedoch hervor, dass eine Abbildung intuitiv erfolgen kann.

## 2.2.2. Produktvorlagenerzeugung

Als nächstes wollen wir die Produktvorlagen, beziehungsweise die Erzeugung der Templates durch den Editor besprechen. Nicht alle Produkte, die wir in Abschnitt 2.1.2 modelliert haben, sind sinnvoll in die vom V-Modell gewohnten Templates zu überführen. Es eignen sich hierfür im Wesentlichen:

```
Changeset            --> VB: MSF-Core
Persona              --> VB: MSF-Core
Test Approach        --> VB: MSF-Core
Release Plan         --> VB: MSF-Agile -> !RTF
Iteration Plan       --> VB: MSF-Agile -> !RTF
Project Checklist    --> VB: MSF-Agile -> Ersetzt PHB des V-Modells
QoS Requirement List --> VB: MSF-Agile
Scenario Description --> VB: MSF-Agile
Scenario List        --> VB: MSF-Agile -> !RTF
Storyboard           --> VB: MSF-Agile
Thread Model         --> VB: MSF-Agile
```

Obwohl die Work Items (Abschnitt 2.1.2, Anhang B) als stark strukturierte Datensätze darstellbar sind, haben wir auf einen expliziten Export als Template verzichtet. Vielmehr wird an dieser Stelle angeregt, auf der Grundlage der Typdefinition eines Work Items eine Liste, zum Beispiel in Excel zu betreiben anstatt explizite RTF-Templates zu erzeugen. Wie oben gezeigt, ist es auch sinnvoll, beispielsweise die Pläne für den Vorlagenexport vorzusehen; jedoch mach der RTF-Export eines Projektplans nicht viel Sinn, sodass hier ggf. ein anderes Format zu bevorzugen ist. Durch die Möglichkeit, externe Dokumentvorlage im V-Modell zu referenzieren, könnten hier zum Teil sogar die Originalvorlagen verwendet werden.

## 2.3. Bewertung

Es wurde gezeigt, dass eine Remodellierung des Microsoft Solutions Framework mit den Mitteln des V-Modells möglich ist. Aufgrund der unterschiedlichen Philosophien der beiden betrachteten Prozesse (V-Modell: produktorientiert; MSF: aktivitätsgetrieben), kann eine $1 : 1$ Abbildung nicht erfolgen. Fehlende, bzw. unterschiedlich interpretierte Konzepte müssen im Rahmen der Abbildung ineinander überführt werden. Entsprechende Überlegungen hierzu sind in Kapitel 2.1 und Anhang A zu finden. Trotz der prinzipiellen Abbildbarkeit sind Abstriche zu machen, die sich insbesondere auf die unterschiedlichen Konzepte beziehen. So kennt beispielsweise der MSF das Konzept der **Work Items**, das über die „einfachen" Produktvorlagen des des V-Modells weit hinaus geht. Eine Abbildung und Remodellierung der Work Items hat einen signifikanten Verlust an verschiedenen Fähigkeiten (zum Beispiel automatisches Reporting) zur Folge.

Der gravirenste Punkt ist jedoch, dass bei einer Remodellierung und Integration des MSF in das V-Modell die weit reichende Werkzeugintegration und -unterstürzung des MSF verloren geht. Die zur Zeit verfügbaren Werkzeuge für das V-Modell stellen keine Alternative zum Visual Studio und zum TFS dar. Der MSF würde somit wieder auf die reine Dokumentation reduziert und würde sich somit wieder auf der Ebene der Version 3.x zurückfallen.

# 3. Vorschläge zur V-Modell XT Implementierung im Visual Studio Team System

Im letzten Kapitel haben wir festgestellt, dass das Microsoft Solutions Framework im V-Modell XT abbildbar ist. Jedoch sind an vielen Stellen Kompromisse zu schließen, die die relative freizügige Prozessdefinition auf den formalen Rahmen des V-Modells abbilden. Eines der größten Probleme dabei ist der Verlust der guten Werkzeugeinbindung des MSF. Die Arbeit auf der Strukturebene und insbesondere die Analyse des MSF mit dem Ziel der V-Modell-Integration (Anhang A) hat jedoch Potenzial für die „andere Richtung" gezeigt (vgl. Kapitel 1.3.2).

Dieses Kapitel befasst sich mit Optionen zur geeigneten Integration des V-Modell XT in die Microsoft Werkzeugkette für Softwareentwicklung. Ziel dieses Kapitels ist es, die Frage zu beantworten, welche Integrationsverfahren für das V-Modell XT im Visual Studio 2005 Team System und dem Team Foundation Server sinnvoll umzusetzen sind und entsprechende Lösungen zu skizzieren.

## 3.1. Ziel einer Implementierung und Umfangsdiskussion

Eine sinnvolle Umsetzungsvariante ist in Abbildung 3.1 dargestellt. Ausgehend von einem projektspezifisch angepassten V-Modell XT kann das Exportergebnis weiter verarbeitet werden. Zum Einsatz kommt hierbei ein **PTE-Generator**[1]. Dieser erzeugt ein Process Template, das entweder direkt in den Team Founation Server importiert werden kann oder in weiteren Bearbeitungsschritten mit dem standardmäßig vorgesehenem Process Template Editor weiter bearbeitet werden kann (siehe auch Abbildung 1.10).

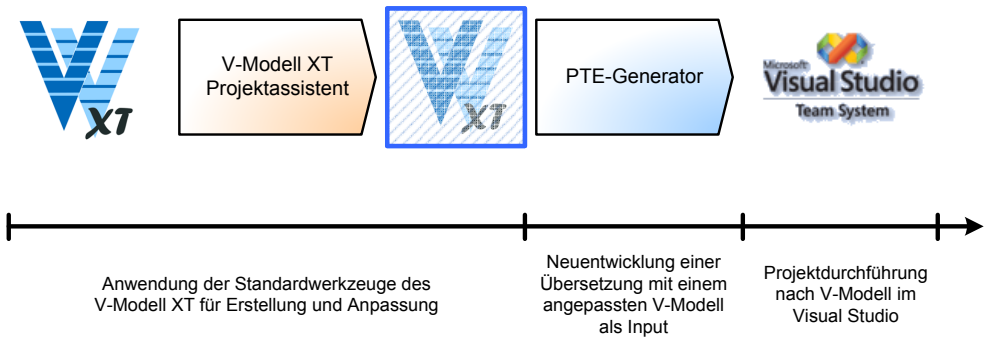



**Abbildung 3.1.:** Zielvorstellung des Gesamtitegrationskonzeptes

Der PTE-Generatorkonzept ist das Ergebnis dieses Abschnitts. Es wird die notwendige Funktionalität bieten, ein V-Modell XT Projekt in ein Process Template für den Team Foundation Server zu überführen. Hierzu muss er einerseits die Strukturen des V-Modells adäquat erfassen und abbilden, weiterhin muss er die Abläufe und bis zu einem gewissen Grad Methodik hinzufügen (vgl. [KK06]). Die folgenden Abschnitte geben Einblick in die Überlegungen zur Konzeption. Sie verstehen sich jedoch noch nicht als vollständiges Realisierungskonzept.

1 PTE: Process Template Editor – Ein Werkzeug, das dazu verwendet wird, um die Process Templates für den Team Foundation Server zu bearbeiten.

## 3.2. Modellierung der Strukturen

Zunächst muss überlegt werden welche Strukturen in welchem Umfang zu modellieren sind. Hierzu definieren wir zuerst die Domäne, beziehungsweise die Teilbereiche des V-Modells, die in der Modellierung zu berücksichtigen sind. Wir greifen dabei auf den Projektassistenten (Abbildung 3.2) zurück und führen ein „Vortailoring" durch, um das V-Modell projekttypspezifisch einzuschränken.

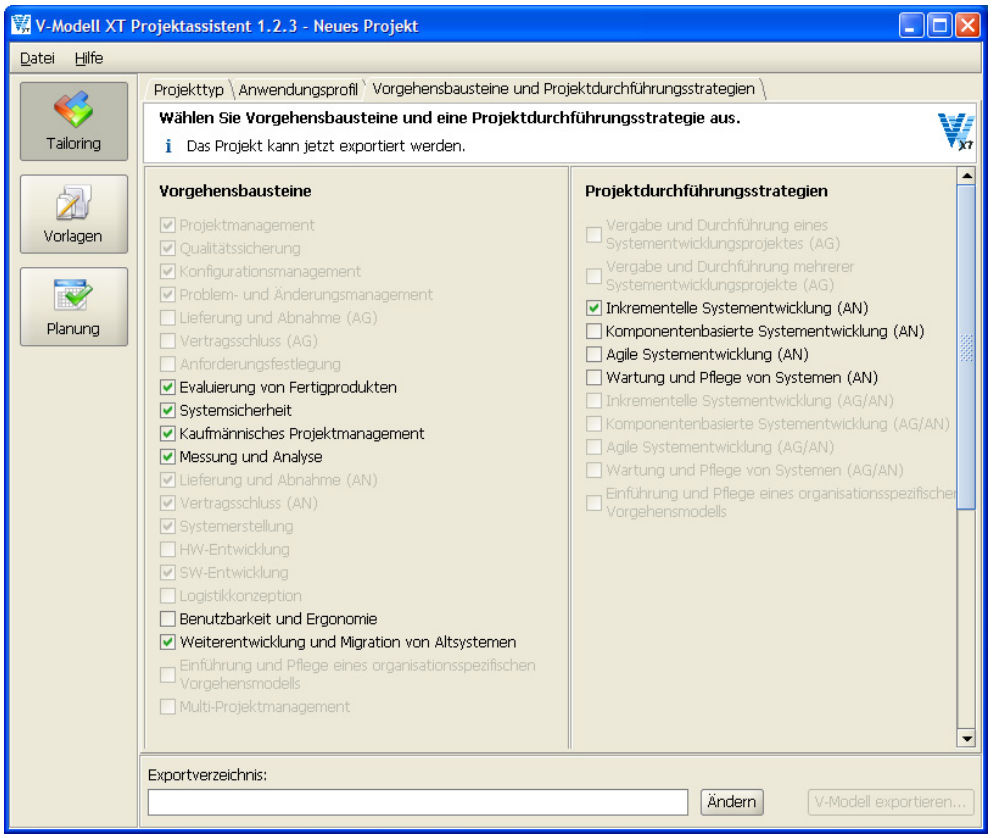



**Abbildung 3.2.:** Tailoringprofil des betrachteten V-Modell XT Teils

Betrachtet werden für die Modellierung die Standardanteile des V-Modell für Auftragnehmerprojekte. Dies betrifft:

- den V-ModellXT Kern (Vorgehensbausteine: Projektmanagement, Qualitätssicherung, Konfigurationsmanagement, Problem- und Änderungsmanagement.)
- die AG/AN-Schnittstelle (Vorgehensbausteine: Lieferung und Abnahme (AN), Vertragsschluss (AN))
- die für den Projekttyp und -gegenstand verpflichtenden Anteile (Vorgehensbausteine: Systemerstellung und SW-Entwicklung)
- die Projektdurchführungsstrategie: Inkrementelle Systementwicklung (AN)
- die zusätzlichen Komponenten des V-Modells, die im Kontext der Integration mit der Visual Studio Plattform relevant sind (verbleibende Vorgehensbausteine, siehe Abbildung 3.2)

### 3.2.1. Produkte zur Integration

Das V-Modell XT ist ein **produktzentriertes** Vorgehensmodell. Dementsprechend stehen auch Produkte im Zentrum der Betrachtung. In [KK06] präsentieren wir einen Ansatz, der aus einem mittels Tailoring projektspezifisch angepassten V-Modell XT ein SharePoint 2007 Portal erzeugt. Eine der zentralen Fragen dort ist (wie auch hier):

Welche der Produkttypen des V-Modells sind für eine automatische Verarbeitung geeignet?

Im Kontext der V-Modell-Integration rücken hier die Work Items ( Anhang A.3.1 und B) des MSF in den Fokus. Insbesondere im Bereich Aufgaben- oder Risikomangement, sowie im Bereich Systemmodellierung und -erstellung bietet das Visual Studio-Backend bereits weit reichende Möglichkeiten an. Wie bereits angemerkt sind Work Items ein Konzept, das weit über die Produkttypdefinitionen des V-Modells hinausgeht. Im Rahmen der Integration sind somit folgende Punkte/Fragen zu klären:

- Welche Produkttypen sind für AN-Softwareprojekte durch das V-Modell XT definiert?
- Welche dieser Produkttypen eignen sich für einen Export, sodass mithilfe von formularbasierten Bearbeitungsprozessen eine Überführung auf Work Items erforderlich ist?
- Wie werden diese Produkttypen als Work Items modelliert? (Formularmodellierung, Zustandsmodellierung und Prozessmodellierung)
- Wie weit kann dieser Prozess automatisiert werden? Wie viel externe Logik ist hier mit einzubringen/wie viele Annahmen sind hier zu treffen?

Insbesondere der letzte Punkt verlangt Aufmerksamkeit. Das V-Modell definiert eine Reihe von Aufgaben und Produkten in einer sehr abstrakten Form. Werden diese als geeignet für die automatische Integration und formularbasierte Verarbeitung im Team Foundation Server identifiziert, muss der PTE-Generator hier gegebenenfalls methodische Anteile „hinzuzaubern".

Ein offener Punkt bleibt jedoch noch zu klären: Was passiert mit dem Rest? Wo finden sich V-Modell Spezialitäten, wie zum Beispiel die Systemerzeugung wieder? Hierfür ist es aber erforderlich, einen Überblick über die zu betrachtenden Mengen des V-Modells zu bekommen.

### 3.2.2. Ergebnisse des V-Modells für die Integration

In [Kuh06] haben wir bereits das Thema der über das Tailoring hinausgehenden Anpassung des V-Modells dikutiert. Dort war der Fokus jedoch Redundanzen zu Hausstandards zu vermeiden und entsprechende Abbildungen für Standardszenarios zu ermitteln. Dieses Verfahren kann auch hier wieder zur Anwendungkommen. Bei einer Integration des V-Modells in die Visual Studio Infrastruktur sind nämlich nicht nur die Eigenheiten des Team Foundation Server zu berücksichtigen, das heißt also die Rahmenbedingungen für die Prozesse, sondern auch die Paradigmen der Systementwicklung. Konkret bedeutet das, dass bei der Integration des V-Modells als Projektrahmen für Visual Studio Projekte adäquate Abbildungen beispielsweise für die Systemstruktur gefunden werden müssen. Auch hier ist die automatisierte Übersetzung zu prüfen.

Das Tailoring (wie in Abbildung 3.2 skizziert) liefert eine Menge von 69 Produkttypen. Diese können klassifiziert werden in:

1. Modellierungs-/Entwicklungsbezogene Produkttypen
2. Managementprodukttypen
3. Sonstiges

#### Modellierungs-/Entwicklungsbezogene Produkttypen

Unter der ersten Rubrik fassen wir alle Produkttypen zusammen, die einen unmittelbaren Bezug zur Systementwicklung mit dem Visual Studio haben. Dies umfasst:

- Datenbankentwurf
- Externe Einheit
- Externe-Einheit-Spezifikation
- Externes SW-Modul

- Externes-SW-Modul-Spezifikation
- Implementierungs-, Integrations- und Prüfkonzept SW
- Implementierungs-, Integrations- und Prüfkonzept System
- Implementierungs-, Integrations- und Prüfkonzept Unterstützungssystem
- Segment
- SW-Architektur
- SW-Einheit
- SW-Komponente
- SW-Modul
- SW-Spezifikation
- System
- Systemarchitektur
- Systemspezifikation
- Unterstützungssystem
- Unterstützungs-Systemarchitektur

Diese Produkttypen des V-Modells müssen auf angemessene Elemente des Visual Studio und des Team Foundation Servers abgebildet werden. Die Notwendigkeit ergibt sich aus dem Punkt, dass das Visual Studio mit der Integration so genannten **Domain Specific Langauges** spezifische Modellierungswerkzeuge anbietet. Diese decken ein Spektrum ab, das vom grob granularen Systemdesign über die Datenbankmodellierung bis hin zum Entwurf von Klassenstrukturen mitsamt Codegenerierung reicht. Somit sind insbesondere die V-Modell Produkttypen:

- Datenbankentwurf
- Segment
- SW-Architektur
- SW-Einheit
- SW-Komponente
- SW-Modul
- SW-Spezifikation
- System
- Systemarchitektur
- Systemspezifikation

näher zu untersuchen. Die Entwurfsmethode, die dem Visual Studio zugrunde liegt impliziert ebenso wie das V-Modell eine Hierarchisierung des Systementwurfs und der anschließenden Realisierung. Die konkrete Abbildung, sowie die Verteilung beziehungsweise Repräsentation der Einzelelemente (insbesondere mit Hinblick auf die explizite Nennung von Architektur- und Spezifikationsdokumenten im V-Modell) ist noch zu prüfen.

#### Managementprodukte

Der zweite oben aufgeführte Punkt sind die Managementprodukttypen. Hierzu zählen wir alles, was in den Bereichen Berichtswesen, Aufgaben- oder Risikomanagement, oder auch Anfroderungsmanagement etc. zu finden ist. Im Kontext des Tailoringresultats betrachten wir hier insbesondere:

- Änderungsentscheidung
- Änderungsstatusliste
- Arbeitsauftrag
- Besprechungsdokument
- Kaufmännische Projektkalkulation
- Käufmännischer Projektstatusbericht

- Messdaten
- Metrikauswertung
- Problem-/Änderungsbewertung
- Problemmeldung/Änderungsantrag
- Projektabschlussbericht
- Projektfortschrittsentscheidung
- Projekthandbuch
- Projektmanangement-Infrastruktur
- Projektplan
- Projektstatusbericht
- Projekttagebuch
- QS-Bericht
- QS-Handbuch
- Risikoliste

Den Bereich „Sonstige" betrachten wir an dieser Stelle nicht weiter. Wir übernehmen die Modellierung der verbleibenden Produkttypen vorerst direkt vom V-Modell und nehmen an, dass sie in einem weiterverwendbaren Exportformat zur Verfügung stehen.

Betrachten wir die oben stehende Liste an Produkttypen, die das V-Modell für das skizzierte Szenario als Managementprodukte vorsieht. Allen Produkten gemein ist, dass sie entweder in regelmäßigen Intervallen in immer derselben Form erstellt werden müssen (z. B. ein Bericht) oder einen einmaligen, globalen Charakter haben (z. B. die Risikoliste). Wie auch bei den Produkttypen für die Softwareentwicklung greifen wir hier soweit möglich auf die bereits vorhandenen Fähigkeiten des Team Foundation Server zurück. Wir bedienen uns dabei intensiv des **Work Item**-Konzepts (siehe Anhang A.3 und insbesondere auch B.2). Work Items sind wie bereits erläutert Datenbankeinträge. Dem entsprechend lassen sich Satzdaten (z. B. ein Risiko oder ein Arbeitsauftrag) auch hervorragend in Form von Work Items modellieren. Durch die Verwendung des Microsoft SQL Servers und seiner Fähigkeiten zum Data Warehousing und diverser Business Intelligence-Kapazitäten sind die derartig modellierten Satzdaten verfolgbar und vor allem messbar[2]. Es findet sich somit bereits an dieser Stelle eine einfache Abbildung:

```
Messdaten --> Ergebnisse einer SQL-Query
    Metrikauswertung --> Report über den Messdaten
```

**Abbildung: Messdaten und Metrikauswertung**

Bei der Integration des V-Modell XT in den Team Foundation Server müssen Metriken definiert und als SQL-Abfragen realisiert dem Team Foundation Server übergeben werden. Die Produkte Messdaten und Metrikauswertung müssen nicht mehr gesondert betrachtet werden.

Messdaten und Metriken sind Produkttypen des V-Modells, die vollständig durch den TFS realisiert werden. Betrachten wir als nächstes den Bereich des **Aufgabenmanagements**. Für die initiale Konzeption versuchen wir anhand unserer Erfahrungen aus [KK06] einen breiteren Blickwinkel zu finden. Wir betrachten aus dem Anwendungsprofil resultierenden Produkttypen die folgenden für das Aufgabenmanagement:

- Änderungsentscheidung
- Änderungsstatusliste
- Arbeitsauftrag
- Problem-/Änderungsbewertung

2 Der Team Foundation Server kann verschiedene Messgrößen über Datenbankabfragen modellieren und diese automatisch ausführen und so detaillierte Reports erstellen.

- Problemmeldung/Änderungsantrag
- Risikoliste

Wir sehen diese Punkte als zentral an, da sie im unmittelbaren Zusammenhang mit Aufgaben im Projekt stehen. Der Arbeitsauftrag ist dabei sehr einfach abzubilden:

```
Arbeitsauftrag --> Task (Work Item)
```

Analog verfahren wir:

```
Risikoliste (Eintrag) --> Risk (Work Item)
Risikoliste (gesamt)  --> Abfrage des TFS nach WI:Risk
```

Ausgehend davon lassen sich Workflows definieren, die unmittelbar mit den verbleibenden Produkttypen in Verbindung steht. Die Verfahrensweise mit Arbeitsaufträgen und Risiken, kann für das Problem- und Änderungsmanagement analog übernommen werden. Dazu ist es erforderlich:

1. Problemmeldung/Änderungsantrag muss als Work Items modelliert werden
2. Problem-/Änderungsbewertung muss als Work Items modelliert werden
3. Änderungsentscheidung muss als Work Item modelliert werden
4. Änderungsstatusliste muss analog zur Risiko- und Taskliste modelliert werden
5. Workflows und Abhängigkeiten zwischen den einzelnen Work Items modelliert werden

Die technischen und konzeptionellen Fähigkeiten stehen zur Verfügung. Der TFS liefert mit den technischen Fähigkeiten die notwendige Infrastruktur; das V-Modell steuert die inhaltlichen Aspekte bei.

---

**Abbildung: Managementmechanismen des V-Modells**

Die Modellierung von Managementaufgaben, die das V-Modell XT vorsieht, im Kontext des Team Foundation Servers erfolgt durch massive Anwendung des Work Item Konzepts. Die Produktabhängigkeiten des V-Modells liefern hier weite Teile der benötigten Workflow Definition; die zugrunde liegenden Statusmaschienen unterstützen die Ermittlung des Projektfortschritts.

---

Die Konzentration auf Work Items, genau wie in [KK06] betrachtet, bietet die Möglichkeit mithilfe von Formularen wesentliche Aufgaben abzudecken. Dies betrift z. B. auch die Möglichkeiten, Problemmeldungen als InfoPath-Formular via Outlook per Email zu versenden und diese somit direkt in die Workflows des TFS einzubringen[3].

Bislang nicht berücksichtigte Produkttypen, wie Handbücher oder Berichte, können auf ähnliche Weise abgebildet werden. Für Berichte bieten sich wieder Formulare, gegebenenfalls sogar Work Items an. Für Handbücher steht mit den Team Webseiten eine einfache Möglichkeit zur Verfügung, Richtlinien und Dokumentation direkt im Portal zu hinterlegen. Die Projektdokumentation könnte somit im Idealfall komplett papierlos realisiert werden (siehe Abbildung 1.6).

## 3.3. Modellierung von Abläufen

Das V-Modell XT sieht mit dem Konzept der **Projektdurchführungsstrategie** ein Konzept zur prinzipiellen Beschreibung von grob granularen Projektabläufen vor. Projektdurchführungsstrategien definieren über einer Menge von Entscheidungspunkten mögliche Ablaufpfade. Dabei gibt es aber kein Konzept, das etwas zur Phase vergleichbares darstellt. Das V-Modell verzichtet auf den Phasenbegriff und setzt statt dessen auf so genannte Projektfortschrittsstufen, die eben durch Entscheidungspunkte abgeschlossen werden (Kapitel 1.1.1).

Obwohl das V-Modell nicht explizit von Phasen spricht, sind diese dennoch in den Projektdurchführungsstrategien aufzufinden. So gibt es beispielsweise Analyse- und

3 Eine Modellierung der Workflows sowie deren Umsetzung während der Generierung muss noch erfolgen.

Designphasen, es gibt Releases, Implementierung und Test. Eine entsprechende Einteilung/Strukturierung eines V-Modell Projekts ist für die Überführung auf den TFS ratsam, da dieser ein Strukturierung des Projektablaufs unterstützt. Der TFS bietet weitgehende Unterstützung an, wie ein Blick auf die Ablaufmodelle des MSF (Anhang A.2.1, Abbildungen A.1 und A.2) zeigen. In der Tat ist es so, dass die Standard-PDS des V-Modells, also die inkrementelle Systementwicklung, sich sehr weit mit dem Ablauf des MSF CMMI-Derivats (Abbildung A.2) überdeckt. Eine stark vereinfachte Darstellung ist Abbildung A.4 im Anhang A.4 zu entnehmen. In [KT06] haben wir uns ebenfalls schon einmal mit der Abbildung der Prodjektdurchführungsstrategien befasst. Ausgehend von der Modellierung aus Anhang A.2.1 geben wir im Folgenden auf der Grundlage der PDS für die inkrementelle Systementwicklung eine Abbildungsvorschrift an, die sich auf das Areas[4]-Modell des TFS abbilden lässt.

```
EP: Projekt genehmigt
EP: Projekt definiert
EP: Angebot abgegeben
EP: Projekt beauftragt        --> Area: Projektinitialisierung

EP: System spezifiziert
EP: System entworfen
EP: Feinentwurf abgeschlossen
EP: Systemelemente realisiert
EP: System integriert
EP: Lieferung durchgeführt
EP: Abnahme erfolgt           --> Area: Systementwicklung

EP: Projekt abgeschlossen     --> Area: Projektabschluss
```

Diese drei Areas lassen sich Top-Level über eine Projektstruktur legen. Innerhalb dieser Areas besteht die Möglichkeit, mehrere Iterationen zu durchlaufen. Die Iterationen richten sich dabei nach Abläufen, wie sie das V-Modell vorsieht. So z. B. kann in der Area Systementwicklung das Modell der Iterationen für Entwicklung, Einkauf von Fertigprodukten oder ähnliches abgebildet werden.

Eine Sonderrolle nimmt bei den Abläufen die Projektplanung und -steuerung ein. Diese ist auch im MSF als kontinuierlicher Prozess modelliert, der in verschiedenen Workflows zur Anwendung kommt. Eine explizite, diskrete Steuerung über ausgewiesene Meilensteine ist dort nicht vorgesehen und erschwert auch die Modellierung/Behandlung des Entscheidungspunktes *Iteration geplant*.

Insbesondere zu beachten ist aber unabhängig von der Projektstruktur die Gestaltung der Abläufe. Der Rahmen, den der Team Foundation Server hier aufspannt ist dafür augenscheinlich zu starr. In [KK06] haben wir uns mit dieser Problematik schon einmal auseinander gesetzt, wobei dort eine einfache, linearisierte Liste von Aufgaben mitsamt der erforderlichen Verknüpfungen zur Produkten und Aktivitäten genügte. Ein ähnliches Verfahren sollte auch bei der Implementierung des V-Modells im TFS gewählt werden; die Möglichkeiten des TFS für die Anwendung von Sicherheitsinstrumenten durch die Hierarchisierung des Projekts sollten ebenfalls berücksichtigt werden.

---

**Abbildung: Projekplanex- und -import**

Für den Import eines V-Modell XT Projekts in den TFS genügt die initiale Planung des V-Modell XT Projektassistenten und dessen Planungskomponente. Es werden hier (automatisiert) dieselben Weiterverarbeitungsschritte durchgeführt, wie bei der Verwendung des Exports in Microsoft Project. Der TFS ist in der Lage, Projektpläne extern zur Verfügung zustellen (z. B. via Project) und diese auch wieder zu synchronisieren.

---

4 Areas entsprechen im Wesentlichen Projektphasen, verfügen aber durch das Werkzeug TFS zusätzlich noch über Sicherheitseigenschaften, die es somit gestatten, einzelne Bereiche eines Projekts nicht hierarchisch zu strukturieren, sondern gleichzeitig auch über Zugangsbeschränkungen zu sichern.

## 3.4. Automatisierung und Werkzeugunterstützung

Abschließend befassen wir uns noch mit der Werkzeugunterstützung dieses Konzepts. Prinzipiell gibt es mindestens zwei Wege, wie eine Integration des V-Modells im Team Foundation Server aussehen kann.

Der erste mögliche Weg ist eine direkte Integration in den Projektassistenten des Visual Studios. In den Abbildungen 3.3 bis 3.6 sind die vier Standardschritte aufgeführt.

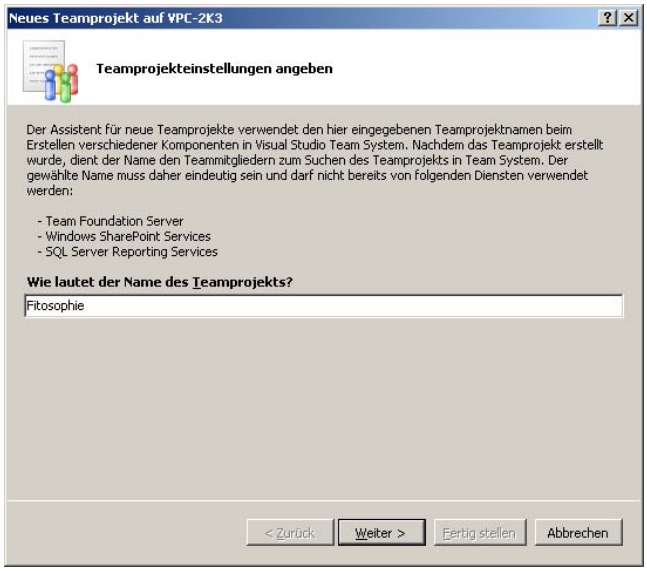



**Abbildung 3.3.:** Schritt 1 – Das Projekt wird benannt.

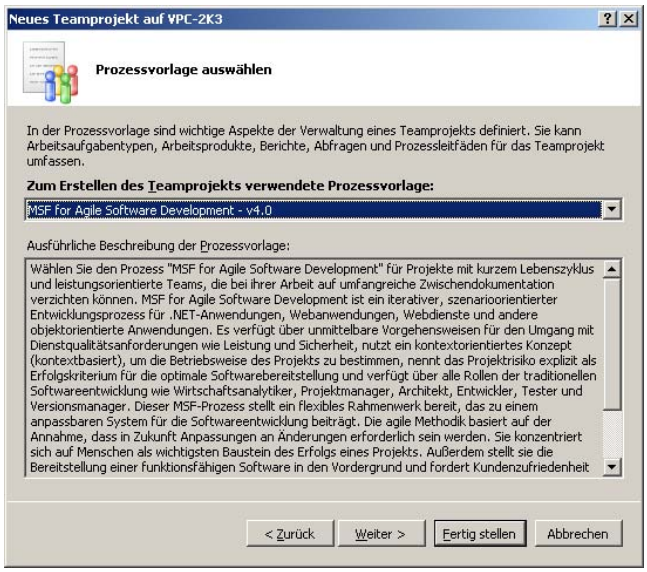



**Abbildung 3.4.:** Schritt 2 – Das für das Projekt zu verwendende Vorgehensmodell wird anhand eine Process Templates ausgewählt.

Der zweite Schritt (Abbildung 3.4) wird unabhängig von der Art der Integration durchlaufen. Um ein Vorgehensmodell im Team Foundation Server verfügbar zu machen, muss es als **Process Template** vorliegen. Im Abschnitt 3.1 haben wir den Weg vom V-Modell zu einem adequaten Template bereits skizziert. Im Anschluss, oder auch vor dem Schritt 4 (Abbildung 3.6) müsste bei einer direkten Integration

des V-Modells in das Visual Studio, beziehungsweise den TFS mindestens ein weiterer Schritt eingefügt werden. Das Tailoring, genauer das Ableiten des Anwendungsprofils (vgl. Abbildung 3.2) muss hier noch eingeschoben werden. Weiterhin müsste der gesamte Export (Dokumentation, Produkttemplates, Plan) ebenfalls integriert werden.

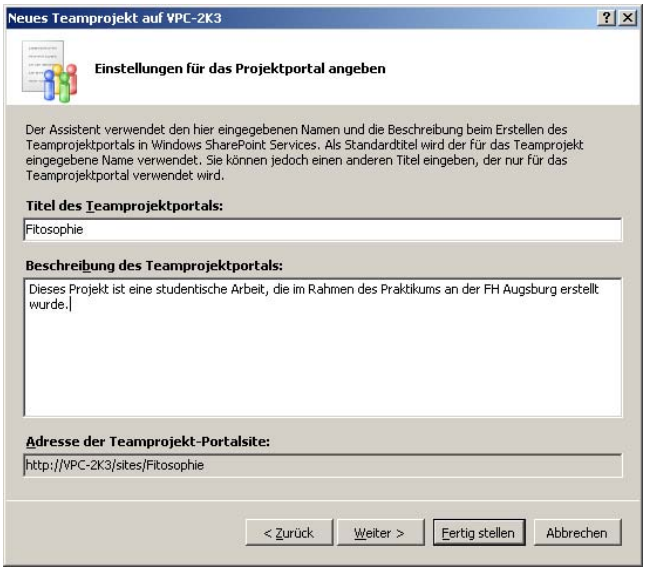



**Abbildung 3.5.:** Schritt 3 – Eingabe der Informationen für das Teamportal.

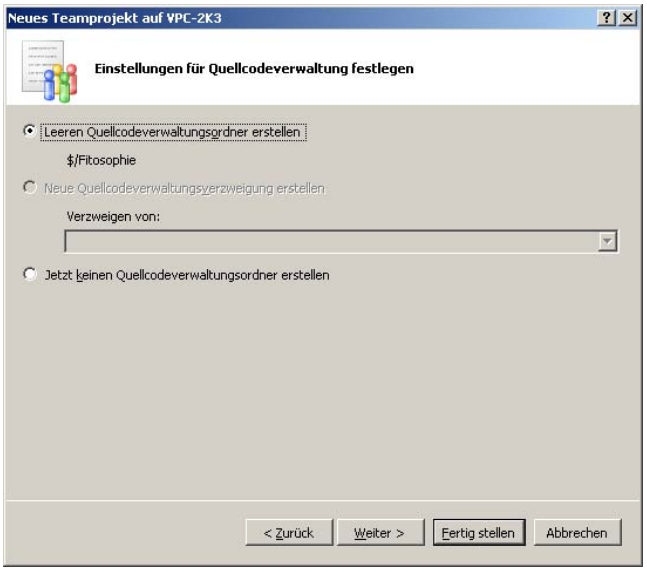



**Abbildung 3.6.:** Schritt 4 – Festlegen weiterer Projekteigenschaften; hier am Beispiel des Anlegens/Konfigurierens der Quellcodeverwaltung.

Dieser Schritt ist sicherlich möglich und verspricht eine gute Anwendbarkeit für den Projektmanager. Jedoch muss die Sinnhaftigkeit hinterfragt werden: Nicht jeder vom V-Modell XT unterstützte Projekttyp ist auch sinnvoll im Visual Studio System umzusetzen.

Beispielhaft seien hier Auftraggeberprojekte genannt, die typischerweise keine unmittelbare Entwicklung zur Folge haben. Diese Projekte benötigen im Wesentlichen nur die Fähigkeiten aus [KK06], also Unterstützung des Managements. Es wäre hier nicht zielführend, für diesen Kontext, der nicht zwingend durch IT'ler erfüllt wird, eine komplexe Entwicklungsumgebung wie das Visual Studio anzubieten.

**Entwurfsentscheidung**

Aufgrund dieser Tatsache und dem zu erwartenden Aufwand wird dieser Weg verworfen.

Statt dessen deckt das hier entwickelte Integrationskonzept **nur Auftragnehmerprojekte**, und hier wieder eingeschränkt auf den Bereich **Softwareentwicklung**, ab.

### Erstellung eines Generators und Ausbau der Werkzeugkette

Eine dem Aufwand/Nutzen-Verhältnis angemessene Lösung ist der Einsatz eines Generators, der die Werkzeugkette des V-Modells hin zum Visual Studio System komplettiert. Mit einer ähnlichen Technik wurden in [KK06] schnell einsetzbare Ergebnisse erziehlt. Der Process Template Generator (Abschnitt 3.1) setzt dabei unmittelbar nach dem Tailoring des V-Modell XT an:

```
PTEG {
  in:
    v-modell-xt.xml
    plan.csv

  out:
    v-modell-pte.Directory
}
```

Der Generator nimmt analog zu [KK06] eine *v-modell-xt.xml* Datei entgegen. Diese ist das Ergebnis eines Tailorings mit dem Projektassistenten. Weiterhin nimmt der Generator einen durch das Planungsmodul des Projektassistenten erzeugten initialen Projektplan entgegen. Als Ergebnis liefert der Generator ein durch den TFS verarbeitbares Process Template.

Dieses Verfahren unterstützt auch weiterhin die beiden zur Verfügung stehenden Anpassungsverfahren (siehe Abbildung 1.10). Der V-Modell XT Editor dient weiterhin der Anpassung des V-Modells. Der Process Template Editor kann weiterhin für die Anpassung von Process Templates verwendet werden. Es ergibt sich unter Berücksichtigung des PTEG's folgende Werkzeugkette:

1. V-Modell XT Editor
2. V-Modell XT Projektassistent
3. Process Template Generator (PTEG)
4. Process Templare Editor (inkl. MSF WinBuild und weiterer Prozesswerkzeuge für das Visual Studio)
5. Visual Studio Team Foundation Server

Eine entsprechende zyklische (Weiter-)Entwicklung ist ebenfalls möglich. So ist es beispielsweise denkbar, dass das V-Modell mitsamt der Standardwerkzeuge explizit in jedem Projekt verwendet und somit immer ein aktuelles V-Modell genutzt wird. Andererseits ist es weiterhin auch möglich, mithilfe des V-Modells eine organisationsspezifische Anpasung vorzunehmen, diese in ein Process Template zu überführen und dort weiter zupflegen. An dieser Stelle greifen aber wieder (zum Zeitpunkt der Erstellung dieses Beitrags) noch offene Fragen hinsichtlich der Konsistenz der ausgeleiteten Prozesse bei der Weiterentwicklung des V-Modells.

# 4. Ausblick

Das mit diesem Beitrag vorgestellte Konzept beschreibt Potenziale und Optionen für die Implementierung des V-Modell XT im Visual Studio Team System, insbesondere im Visual Studio Team Foundation Server. Das V-Modell kann hier als Vorgehensmodell direkt im Backendwerkzeug der Entwicklungsumgebungen von Microsoft implementiert werden.

### Verwandte Arbeiten

Um zu den Ableitungsketten zu kommen, wurde über einen Zeitraum von ein einhalb Jahren intensive Vorarbeit geleistet. So wurde insbesondere das Microsoft Solutions Framework analysiert, das eine sehr Umfangreiche Implementierung der Fähigkeiten des Team Foundation Servers zeigt. In [KT06] haben wir uns das erste Mal intensiv damit auseinander gesetzt, wie man den MSF mit der Sprache des V-Modells modellieren kann. Als Ergebnisse lagen ein Verständnis des Anliegens des Prozesses, eine Implementierungsidee und Optionen zur wechselseitigen Integration der beiden Prozesse vor. Eine erste Evaluierung der Basiskonzepte wurde in [Aly06] durchgeführt. Mit dem vorliegenden Beitrag haben wir die entsprechende Modellierung verfeinert und als Grundlage zur Herleitung für eine Implementierung des V-Modells verwendet.

In weiteren Arbeiten haben wir schrittweise Integrations- und Anwendungskonzepte für das V-Modell im Kontext der dem Team Foundation Server zugrunde liegenden Infrastruktur erarbeitet. Mit dem als „CollabXT" vorgestellten Ansatz [KK06] haben wir den hier weitergedachten generativen Ansatz entwickelt und erprobt.

### Weitere Arbeiten

Mit dem vorliegenden Beitrag kann der Grundstein für die Implementierung des V-Modells als Vorgehensmodell für den Team Foundation Server und somit als Entwicklungsmodell für .NET-basierte Entwicklung mit dem Visual Studio gelegt werden. Als nächster Schritt wird ein Durchstich eines V-Modell XT Process Templates erstellt, das einer weiteren Zwischenbewertung unterzogen wird. Hier aufgestellte Forderungen nach Abbildungen oder Addition von Fachlogik, wie z. B. weitergehende Workflowunterstützung müssen noch weiteren Bewertungsschritten unterzogen werden.

Es sind über die reine Erstellung von Templates und Generatoren/Transformatoren aber noch weitere, zum Teil auch grundlegende, konzeptionelle Arbeiten notwendig. So z. B. sind die Fragen nach dem Variantenmanagement, Update- und Releasezyklen des V-Modells ebenso zu beantworten, wie die Frage nach der Konformität des resultierenden V-Modell Derivats. Ein gegebenenfalls auftretender Informationsverlust ist zu ermitteln und zu bewerten. Weiterhin ist noch die Frage nach dem Volumen des Prozesses zu beantworten, der aus der Generierung aus dem V-Modell entsteht. Anders als beim „puren" V-Modell, ist diese Version bereits an vielen Stellen mit methodischen Komponenten ausgestattet, die weit über die Verpflichtungen des V-Modells hinausgehen. Auch dieser Bereich ist noch genauestens zu analysieren und zu bewerten.

Weitere, aktualisierte Informationen zu den aktuellen Arbeiten sind unter `http://www4.in.tum.de/~kuhrmann` zu finden.

# A. Ergebnisse und Elemente des MSF

Dieser Anhang fasst die Elemente zusammen, die in den beiden von Microsoft bereitgestellten MSF-Derivaten definiert sind. Die Auflistung, Gruppierung und Zuordnung unterstützt den methodischen Entwurf und die Integrationskonzeption des MSF ins V-Modell XT.

## A.1. Das Rollenmodell

In diesem Abschnitt wird das Rollenmodell des MSF näher betrachtet. Die Instanzen für die beiden Prozessderivate werden hier direkt gegenübergestellt und anhand ihrer Aufgabenbereiche, Zuständigkeiten und der Beteiligung an der Erstellung von Ergebnissen betrachtet. Ziel der Gegenüberstellung ist eine Möglichkeit einer Ableitung eines gemeinsamen *Basisrollenmodells*.

| Rolle | MSF CMMI | MSF Agile | Core |
|---|---|---|---|
| Auditor | √ | | |
| Architect | | √ | |
| Build Engineer | √ | | |
| Business Analyst | √ | √ | √ |
| Developer | √ | √ | √ |
| Database Developer | √ | √ | √ |
| Development Manager | √ | | |
| Infrastructure Architect | √ | | |
| IPM Officer | √ | | |
| Lead Developer | √ | | |
| Product Manager | √ | | |
| Project Manager | √ | √ | √ |
| Quality of Service Specialist | √ | | |
| Release Manager | √ | √ | √ |
| Solution Architect | √ | | |
| Sponsor | √ | | |
| Subject Matter Expert | √ | | |
| Test Manager | √ | | |
| Tester | √ | √ | √ |
| User Education Specialist | √ | | |
| User Experience Architects | √ | | |

**Tabelle A.1.:** Zusammenfassung uns Positionierung der Rollen

Die Tabelle A.1 zeigt die Rollen der beiden MSF-Derivate. Das Agile-Derivat ist im Vergleich zum formalen Modell nur spärlich mit Rollen ausgestattet, was ein Indiz auf kleine Teams ist. Für ein MSF-Basismodell bilden wir hier einfach die Schnittmenge (Spalte „Core" in Tabelle A.1), was dazu fürht, dass die folgenden Rollen Kandidaten für die Aufnahme ins Basismodell sind:

- Business Analyst
- Developer/Database Developer
- Project Manager
- Release Manager
- Tester

Nur aufgrund der Namen kann aber keine Zuordnung erfolgen. Die Kandidaten werden nun noch einmal näher bezüglich ihrer Zuständigkeiten und der Beteiligung bei der Erstellung der Arbeitsergebnisse untersucht. Weiterhin ist zu untersuchen, wie es sich mit den Verbleibenden Rollen verhält. Tabelle A.1 zeigt zum Beispiel im Bereich der Architektur (als eines der Zentralen Arbeitsgebiete) starke Abweichungen. Die Frage, ob und in wie weit die Architektur im Basismodell durch eine Rolle vertreten wird, ist ebenfalls noch zu beantworten.

### A.1.1. Abdeckung der Rollen im Basismodell

**Project Manager** Diese Rolle hat bezüglich der Workstreams und Aktivitäten in den beiden Derivaten keine Schnittmenge. Die Aufgaben, Fähigkeiten und Zuständigkeiten der Personen mit dieser Rolle sind jedoch trotzdem zu großen Teilen gleich (auch wenn die Beschreibungstexte differieren, was jedoch eher den unterschiedlichen Entwicklungsteams zuzuschreiben ist).

**Developer, Database Developer, Tester und Release Manager** Diese Rollen verfügen über Aktivitätszuordnungen, die in beiden MSF-Derivaten identisch vorkommen. In konkreten Details unterscheiden sie sich jedoch, was hier aber wieder den unterschiedlichen Ansprüchen und Adressaten der beiden MSF-Derivate zuzuordnen ist.

**Business Analyst** Diese Rolle kommt ebenfalls in beiden MSF-Derivaten vor. Auch hier gibt es Überschneidungen und Abweichungen.

Alle bis hierher identifizierten Rollen sind also geeignet, in einem MSF-Basismodell aufgenommen zu werden. Bleibt noch zu klären, wie mit dem Architekten zu verfahren ist.

### A.1.2. Rolle der Architektur im Basismodell

Das MSF Agile-Derivate definiert eine zentrale Rolle *Architect*. Dieser nimmt hier alle Architektur-bezogenen Aufgaben wahr. Dies umfasst ein breites Spektrum von der Lösungsarchitektur bis hin zu User Interface und Bedrohungsanalysen.

Im CMMI-Derivat ist die Architektenrolle aufgeteilt auf:

- Infrastructure Architect,
- Solution Architect und
- User Experience Architect.

Die Aufteilung der einzelnen Zuständigkeiten erfolgt hier also wesentlich fein granularer. Es ist daher nicht zielführend, die Architektur bereits im Basismodell durch eine Rolle zu hinterlegen, da der Architekt hier als Abstraktion dienen würde und in den konkreten Derivaten zu verfeinern wäre. Mit Hinblick auf die Anforderung einer Implementierbarkeit im V-Modell XT, wird daher davon Abstand genommen, da auch das V-Modell das Konzept der inhaltlichen Verfeinerung nicht ausreichend unterstützt.

## A.2. Aktivitätsmodell des MSF – Workstreams und Aktivitäten

Analog zu den Rollen- und Ergebnismodellen werden in diesem Abschnitt die aktiven Modellanteile des MSF näher untersucht. Aufgrund der engen Kopplung der drei Modelle untereinander ließe sich hier ein Basismodell auch rein argumentativ ableiten. Dennoch werden die aktiven Anteile separat untersucht.

Abschließen zu den Aussagen dieses und der vorangegangenen Abschnitte, dient dieser letzte Modellanteil dem Abschluss des „Basis-MSF". Es finden sich hier also gleichzeitig Aussagen zur Sinnhaftigkeit und Plausibilität des Basismodells.

### A.2.1. Tracks

Tracks sind im Wesentlichen mit den Projektfortschrittstufen des V-Modell XT vergleichbar. Im V-Modell XT werden einzelne Projektabschnitte definiert und durch einen Entscheidungspunkt abgeschlossen. Projektfortschrittsstufen im V-Modell XT und somit die Entscheidungspunkte folgen einen *Ende-Ende*-Semantik, das heißt dass die einzelnen Projektabschnitt ohne Weiteres parallel ausgeführt werden können.

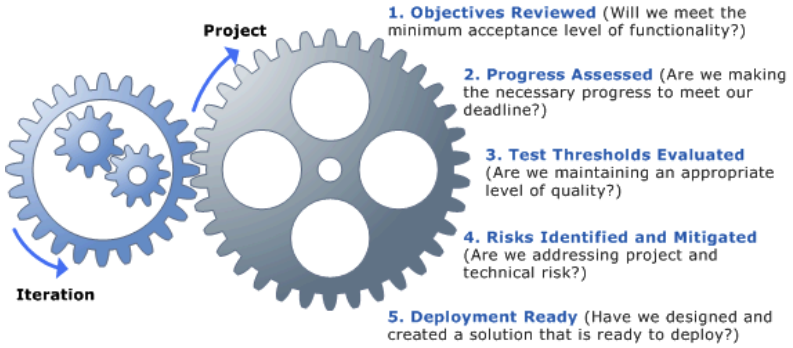



**Abbildung A.1.:** Tracks des Agile-Derivats des MSF aus der Online-Dokumentation

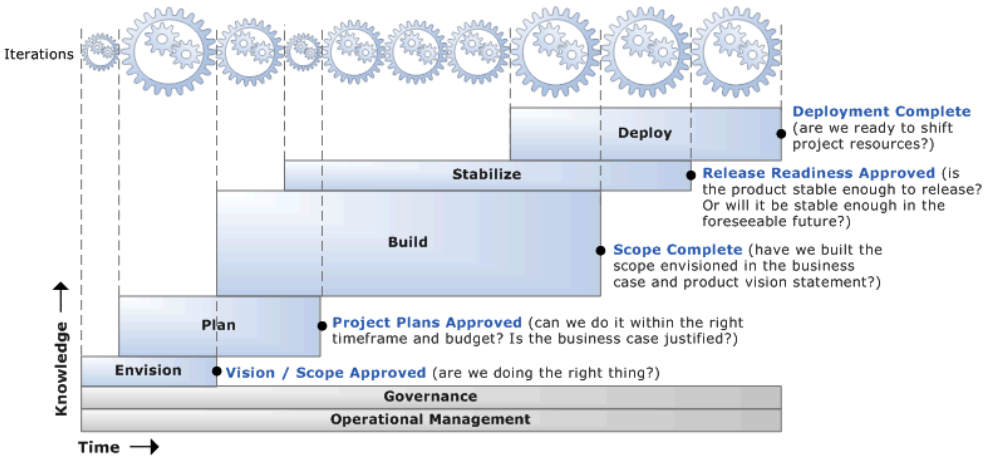



**Abbildung A.2.:** Tracks des CMMI-Derivats des MSF aus der Online-Dokumentation

Projektfortschrittsstufen, bzw. Abschnitte im MSF heißen **Tracks**. Tracks kapseln bestimmte Aufgaben wie Management oder Entwicklung und geben einem Projekt somit eine gewisse Ordnung. Die Abbildungen A.1 und A.2 zeigen Konzeptuelle Grafiken der Tracks (entnommen der Online-Dokumentation unter `http://msdn2.`

microsoft.com/en-us/teamsystem/aa718795.aspx). Abbildung A.1 zeigt dabei das vergleichsweise einfache Modell des Agile-Derivats. Es zielt mit wenigen Kontrollpunkten und schnell ablaufenden, einfachen Iterationen auf kleine (agile) Projekte. Die Abbildung A.2 zeigt hingegen ein stärker strukturiertes Vorgehen mit definierten Kontrollpunkten.

Wie bereits angedeutet müssen die Tracks nicht sequenziell ablaufen. Analog zum V-Modell XT sind also Überlagerungen in einzelnen Projektphasen durchauszugelassen (siehe Abbildung A.2). Nichts desto trotz definieren die Prozess nur in Teilen überschneidungen und definieren für bestimmte Bereiche eigene Tracks (Tabelle A.2). Die Überschneidungen befinden sich gut lokalisierbar in den Analyse- und Entwicklungsphasen eines Projekts (Envision, Build, Stabilize und Deploy). Mangementtätigkeiten werden in eigenen Tracks definiert. Beispielhaft seien hier Continous (MSF Agile) und Operational Management (MSF CMMI) genannt, die beide querschnittliche Projektdisziplinen wie Qualitätssicherung oder Risikomangement erfassen.

| Track | MSF CMMI | MSF Agile | Core |
|---|---|---|---|
| Envision | √ | √ | √ |
| Build | √ | √ | √ |
| Stabilize | √ | √ | √ |
| Deploy | √ | √ | √ |
| Planning | √ | | |
| Governance | √ | | |
| Operational Management | √ | | |
| Plan | | √ | |
| Continous | | √ | |

**Tabelle A.2.:** Positionierung der Tracks der beiden MSF-Derivate. Die Tracks der Entwicklungsaufgaben sind hier deckungsgleich während die Management- und Querschnittsdisziplinen sind zum Teil stark unterscheiden.

Die Tracks dienen im Folgenden zur Modellierung von MSF Projektabläufen. Genauere Betrachtungen dazu sind Abschnitt A.4 zu entnehmen.

### A.2.2. Workstreams

Der MSF fasst seine Arbeitspakete in Form von Workstreams zusammen. Workstreams sind Sammlungen von Aktivitäten, um Projektergebnisse zu erstellen. Workstreams sind im Wesentlichen mit den Produktflüssen des V-Modell 97 [DW99] vergleichbar. Aufgrund der Tatsache, dass es im V-Modell XT kein adäquates Mittel zur Abbildung gibt, wird an dieser Stelle von einer weitergehenden Analyse der einzelnen Workstreams abgesehen.

Erste Analysen zeigen, dass ÜBerdeckungen aufgrund des unterschiedlichen Schnitts der Workstreams in den beiden MSF-Derivaten zu keinen verwertbaren Ergebnissen führen würden. Produkt- und Aktivitätszuordnugn sind beispielsweise nicht eindeutig. In der Tabelle A.3 ist ein Ausschnitt dieser Analyse zu sehen, die diese Problematik verdeutlicht. Der Track **Planning** (aus MSF CMMI) enthält verschiedene Workstreams zur Anforderungsermittlung (*Create**). Das MSF Agile-Derivate verortet Workstreams mit vergleichbaren Inhalten in den **Build**-Track und erzeugt in diesen beispielsweise ein Produkt *Scenario* anstelle des Produkts *Requirement* (siehe Hervorhebung in Tabelle A.3).

Workstreams sind aus diesem Grund für die Strukturierung von Inhalten des MSF nicht geeignet. Der Weg einer Analyse möglicher Schnittmengenelemente zur Ableitung eines MSF-Basismodells über die Ableitung konkreter Elemente aus der Struk-

| Track | Workstream | MSF CMMI | MSF Agile | Core |
|---|---|---|---|---|
| *Planning* | | | | ? |
| | Baseline Configuration Mngt. | √ | | |
| | Create a Scenario | √ | | |
| | Create a QoS Requirement | √ | | |
| | **Create Product Requirements** | √ | | |
| | Create Solution Architecture | √ | | |
| | Plan Project | √ | | |
| | Establish Environments | √ | | |
| *Build* | | √ | √ | ? |
| | Manage Change Requests | √ | | |
| | Plan an Iteration | √ | √ | |
| | Test a Customer Requirement | √ | | |
| | Fix a Bug | √ | √ | |
| | Implement a Development Task | √ | √ | |
| | Analysis | √ | | |
| | Verify a Product Requirement | √ | | |
| | Create Solution Architecture | | √ | |
| | Test a Scenario | | √ | |
| | **Create a Scenario** | | √ | |
| | Test a QoS Requirement | | √ | |
| | Build a Product | | √ | |
| | Create a QoS Requirement | | √ | |
| | Guide Iteration | | √ | |

**Tabelle A.3.:** Analyse der Tracks und der Workstreams – Ausschnitt

tur: **Tracks** ↦ **Workstreams** ↦ **Activities** ↦ **Work Products/Work Items** führt zu keinen verwertbaren Ergebnissen.

## A.3. Ergebnistypen des MSF

In diesem Abschnitt werden die Ergebnistypen der beiden MSF-Derivate betrachtet. Die Analyse umfasst dabei sowohl die grob granularen *Work Items* als auch die fein granularere *Work Products*. Auch dieser Abschnitt stellt sich zum Ziel, ein Basismodell der Ergebnisse zu Formulieren.

### A.3.1. Analyse und Besonderheiten der Work Items

Besonders beachtet werden muss an dieser Stelle, dass Produkte für die Modellierung des MSF im Kontext V-Modell XT eigentlich nur unter den Work Products zu suchen sind. Work Items sind explizit **keine** Ergebnistypen, sondern dienen den integrierten Trackingmechanismen des MSF in Kombination mit dem Team Foundation Server. Dieses Muster greift aber nur so lange, wie man die Verwendung des Team Foundation Server als Ablaufumgebung für MSF-Projekte voraussetzen kann. Bei einer Modellierung des MSF im V-Modell XT ist dies jedoch nicht der Fall, sodass auch die Work Items wieder in das Produktmodell integriert werden müssen.

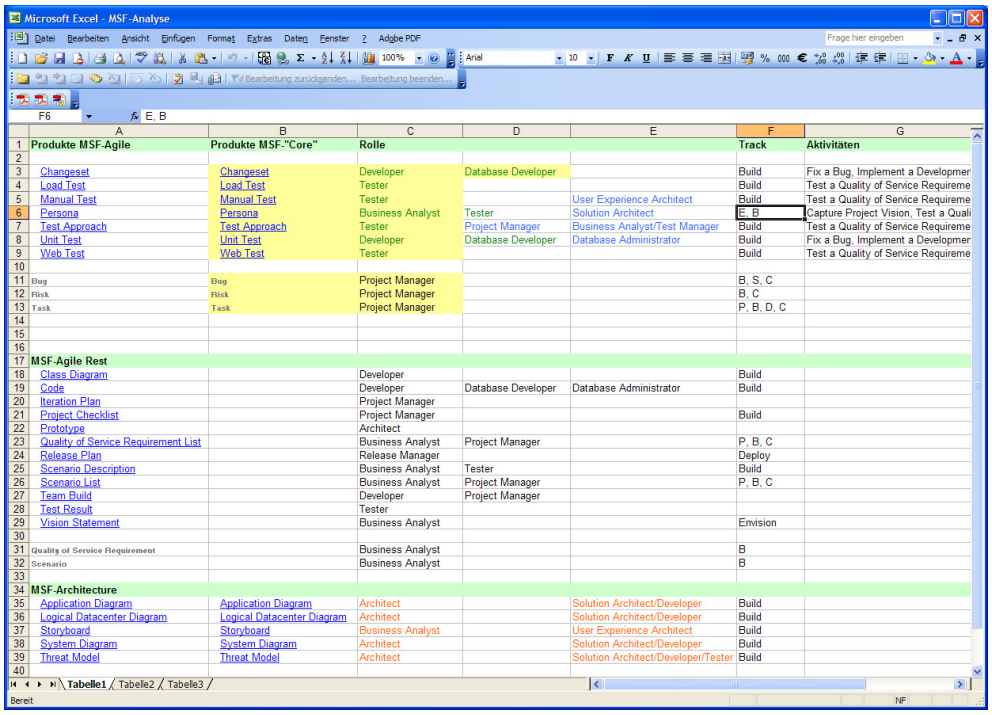
Deshalb modellieren wir auch die Work Items als Produkte. Die Tabelle A.4 listet eine Übersicht aller in beiden MSF-Derivaten definierten Work Item. Die Überschneidung zeigt, dass eine Modellierung analog zu Rollen ebenfalls geeignet ist. Auch die drei gemeinsamen Work Items (Task, Bug und Risk) ordnen sich in die bislang ermittelten Strukturen ein. Geeignete Mechanismen zur Integration und Modellierung im V-Modell sind hier noch während der detaillierten Konzeption (Kapitel **??**) zu untersuchen.

| Work Item | MSF CMMI | MSF Agile | Verantwortlich |
|---|---|---|---|
| Scenario | | √ | Business Analyst |
| Quality of Service Requirement | | √ | Business Analyst |
| Task | √ | √ | Project Manager |
| Bug | √ | √ | Project Manager |
| Risk | √ | √ | Project Manager |
| Review | √ | | Developer |
| Change Request | √ | | Business Analyst |
| Issue | √ | | Project Manager |

**Tabelle A.4.:** Übersicht der Work Items mit Zuordnung zu den einzelnen Prozessderivaten und der Feststellung verantwortlicher Rollen

## A.3.2. Analyse der Work Products

Work Products entsprechen weit gehend den Ergebnis-/Produkttypen des V-Modells. Beide Prozessderivate definieren eine unterschiedliche Menge an Produkten, die im Laufe eines Projekts zu erstellen sind. Das MSF Agile-Derivat definiert 24 Produkttypen; das CMMI-Derivat 51.




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **Produkte MSF-Agile** | **Produkte MSF-"Core"** | **Rolle** | | | **Track** | **Aktivitäten** |
| 2 | | | | | | | |
| 3 | Changeset | Changeset | Developer | Database Developer | | Build | Fix a Bug, Implement a Developmer |
| 4 | Load Test | Load Test | Tester | | | Build | Test a Quality of Service Requireme |
| 5 | Manual Test | Manual Test | Tester | | User Experience Architect | Build | Test a Quality of Service Requireme |
| 6 | Persona | Persona | Business Analyst | Tester | Solution Architect | E, B | Capture Project Vision, Test a Quali |
| 7 | Test Approach | Test Approach | Tester | Project Manager | Business Analyst/Test Manager | Build | Test a Quality of Service Requireme |
| 8 | Unit Test | Unit Test | Developer | Database Developer | Database Administrator | Build | Fix a Bug, Implement a Developmer |
| 9 | Web Test | Web Test | Tester | | | Build | Test a Quality of Service Requireme |
| 10 | | | | | | | |
| 11 | Bug | Bug | Project Manager | | | B, S, C | |
| 12 | Risk | Risk | Project Manager | | | B, C | |
| 13 | Task | Task | Project Manager | | | P, B, D, C | |
| 14 | | | | | | | |
| 15 | | | | | | | |
| 16 | | | | | | | |
| 17 | **MSF-Agile Rest** | | | | | | |
| 18 | Class Diagram | | Developer | | | Build | |
| 19 | Code | | Developer | Database Developer | Database Administrator | Build | |
| 20 | Iteration Plan | | Project Manager | | | | |
| 21 | Project Checklist | | Project Manager | | | Build | |
| 22 | Prototype | | Architect | | | | |
| 23 | Quality of Service Requirement List | | Business Analyst | Project Manager | | P, B, C | |
| 24 | Release Plan | | Release Manager | | | Deploy | |
| 25 | Scenario Description | | Business Analyst | Tester | | Build | |
| 26 | Scenario List | | Business Analyst | Project Manager | | P, B, C | |
| 27 | Team Build | | Developer | Project Manager | | | |
| 28 | Test Result | | Tester | | | | |
| 29 | Vision Statement | | Business Analyst | | | Envision | |
| 30 | | | | | | | |
| 31 | Quality of Service Requirement | | Business Analyst | | | B | |
| 32 | Scenario | | Business Analyst | | | B | |
| 33 | | | | | | | |
| 34 | **MSF-Architecture** | | | | | | |
| 35 | Application Diagram | Application Diagram | Architect | | Solution Architect/Developer | Build | |
| 36 | Logical Datacenter Diagram | Logical Datacenter Diagram | Architect | | Solution Architect/Developer | Build | |
| 37 | Storyboard | Storyboard | Business Analyst | | User Experience Architect | Build | |
| 38 | System Diagram | System Diagram | Architect | | Solution Architect/Developer | Build | |
| 39 | Threat Model | Threat Model | Architect | | Solution Architect/Developer/Tester | Build | |
| 40 | | | | | | | |

**Abbildung A.3.:** Analysematrix für die Ermittlung gemeinsamer MSF-Anteile und Abhängigkeitsfeststellung für die Modellierung des MSF-Agile im V-Modell XT

Eine Analyse kann hier nicht auf den aktiven Elementen des MSF erfolgen. Diese sind aufgrund ihrer nicht eindeutigen Zuordnung als Ausgangspunkt nicht verwertbar. Stattdessen erfolgt eine Analyse nun den Prinzipien des V-Modells entsprechend. Mithilfe einer Analysematrix (Abbildung A.3) werden Zuerst die sich überschneidenden Ergebnisse ermittelt (oberes Drittel der Abbildung[1]). Nachdem die „Schnittmenge" identifiziert wurde, wurde die Ergebnismenge anhand der Rol-

1 Als Durchstich dient hier der MSF Agile, sodass der Rest der Ergebnisse gesondert betrachtet wird, Abbildung A.3, Mitte.

len geprüft. Hier ergaben sich bereits die ersten Abweichungen (in Abbildung A.3 durch unterschiedliche Farben visualisiert). Die erste Rollenspalte enthält eindeutige Zuordnungen. Sind weitere Rollenzuordnungen identisch, stehen sie in der zweiten Spalte. Rollen, die in der zweiten Spalte stehen und ein andere Farbgebung haben (hier Blau), sind Rollen, die (unabhängig vom Derivat) in der Modellierung der einzelnen verfeinerten Derivate noch hinzukommen müssen (in V-Modell Terminologie: mirtwirkend).

Im unteren Drittel der Abbildung A.3 zeigt sich die Verteilung der Produkte und Rollen in der Architekturdisziplin. Hier wird noch einmal die Aussage aus Abschnitt A.1.2 unterstrichen, dass es im Rahmen eines Basismodells nicht sinnvoll ist, die Architektur bereits zu definieren. Die spätere Modellierung im V-Modell würde dies nicht angemessen unterstützen.

Die letzten beiden Spalten von Abbildung A.3 zeigen die „zeitliche" Einorndung der Produkte, das heißt wann sie erstellt werden sollen. Die Zuordnung zu einzelnen Tracks bereitet an dieser Stelle keine Schwierigkeiten in der späteren Modellierung. Problematisch hingegen ist jedoch, dass es zwischen den Produkten und Aktivitäten keine $1:1$-Beziehung gibt. Dies erschwert die Modellierung im V-Modell erheblich. In Kapitel **??** schlagen wir hierfür eine mögliche Verfahrensweise vor. Basierend auf der Analysematrix betrachten wir nun die entsprechenden Produkte und Aktivitäten des MSF, für das „MSF-Basismodell" (Tabelle A.5).

| Work Product | Aktivität | Work Item |
|---|---|---|
| Changeset | Fix a Bug<br>Implement a Development Task<br>Implement a Database Development Task | |
| Load Test | Test a Quality of Service Requirement | |
| Manual Test | Test a Quality of Service Requirement<br>Test a Scenario | |
| Persona | Capture Project Vision<br>Test a Quality of Service Requirement<br>Test a Scenario | |
| Test Approach | Test a Quality of Service Requirement<br>Test a Scenario<br>Guide Project | |
| Unit Test | Fix a Bug<br>Implement a Development Task<br>Implement a Database Development Task | |
| Web Test | Test a Quality of Service Requirement<br>Test a Scenario | |
| | | Task<br>Bug<br>Risk |

**Tabelle A.5.:** Übersicht der Work Products und Work Items für das Basismodell

Die Tabelle A.6 fasst gesamtheitlich noch die Zuordnungen der Rollen und Produkte zusammen. Sie betrachtet im oberen Tabellenanteil die Teile für ein Basismodell und im unteren Teil alle relevanten Elemente, die für den demonstrativen Durchstich relevant sind.

## A.3.3. Architekturdokumente

Tabelle A.6 enthält noch nicht die verbleibenden Architekturprodukte, die sowohl im Agile- als auch im CMMI-Derivat definiert sind (vgl. Abschnitt A.1.2). Diese

| Work Product | Developer | Database Developer | Tester | Business Analyst | Project Manager | Release Manager | |
|---|---|---|---|---|---|---|---|
| Changeset | √ | √ | | | | | |
| Load Test | | | √ | | | | |
| Manual Test | | | √ | | | | |
| Persona | | | √ | √ | | | |
| Test Approach | | | √ | | | | |
| Unit Test | √ | √ | | | | | |
| Web Test | √ | | | | | | |
| *Task* | | | | | √ | | |
| *Bug* | | | | | √ | | |
| *Risk* | | | | | √ | | |
| Class Diagram | √ | | | | | | |
| Code | √ | √ | | | | | |
| Iteration Plan | | | | | √ | | |
| Project Checklist | | | | | √ | | |
| Prototype | | | | | | | Architect |
| Quality of Service Requirement List | | | | √ | | | |
| Release Plan | | | | | | √ | |
| Scenario Description | | | √ | √ | | | |
| Scenario List | | | | √ | √ | | |
| Team Build | √ | | | | √ | | |
| Test Result | | | √ | | | | |
| Vision Statement | | | | √ | | | |
| *Quality of Service Requirement* | | | | √ | | | |
| *Scenario* | | | | √ | | | |

**Tabelle A.6.:** Übersicht der Work Products und Work Items mit Rollenzuordnung

Produkte sind alle der Rolle **Architect** (Agile) zugeordnet, beziehungsweise entsprechenden Rollen im CMMI-Derivat. Tabelle A.7 zeigt die entsprechenden Zusammenhänge.

| Produkt | MSF-Agile Rolle | MSF-CMMI Rolle |
|---|---|---|
| Application Diagram | Architect | Solution Architect<br>Developer |
| Logical Datacenter Diagram | Architect | Solution Architect<br>Developer |
| Storyboard | Business Analyst | User Experience Architect |
| System Diagram | Architect | Solution Architect<br>Developer |
| Thread Model | Architect | Solution Architect<br>Developer<br>Tester |

**Tabelle A.7.:** Übersicht der Work Items mit Zuordnung zu den einzelnen Prozessderivaten und der Feststellung verantwortlicher Rollen

## A.4. Projektablauf in MSF-Projekten

Um eine Integration des MSF in das V-Modell XT vollständig zu ermöglichen, ist es über die strukturelle Analyse hinaus notwendig, auch die Abläufe in einem MSF-Projekt zu untersuchen. Ziel ist es, ein Konzept zu ermitteln, das auf der Grundlage des Verständnisses der MSF-Projektabwicklung den Entwurf einer spezialisierten Projektdurchführungsstrategie erlaubt.

Das V-Modell XT definiert Projektfortschrittstufen, die jeweils mit Entscheidungspunkten abzuschließen sind. Der MSF definiert keine Entscheidungspunkte, jedoch ein mit der Projektfortschrittstufe vergleichbares Konzept: Tracks (siehe auch Abschnitt A.2.1).

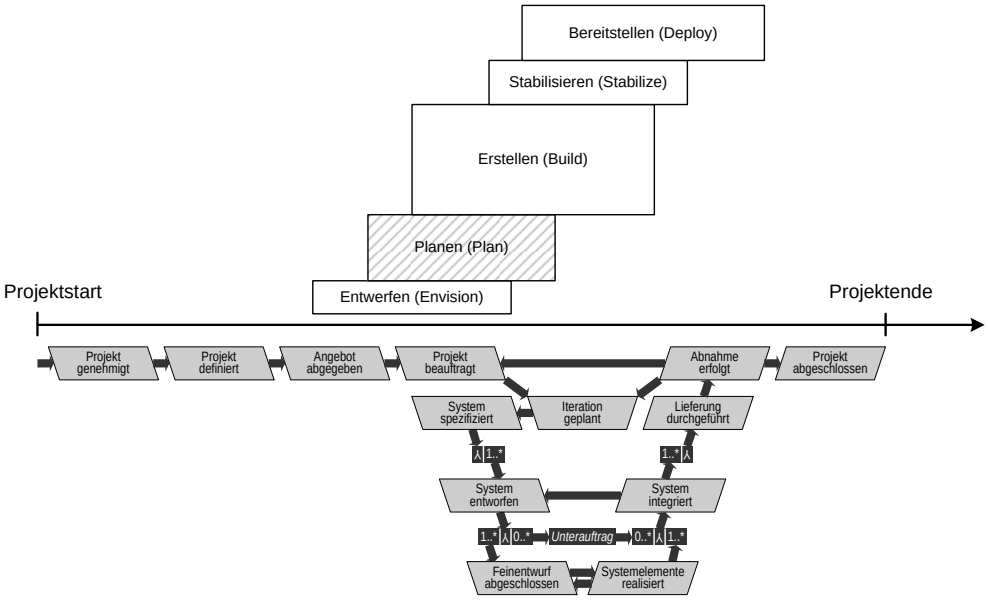



**Abbildung A.4.:** MSF Tracks im Vergleich zur inkrementellen Systementwicklung des V-Modells (Projektdurchführungsstrategie)

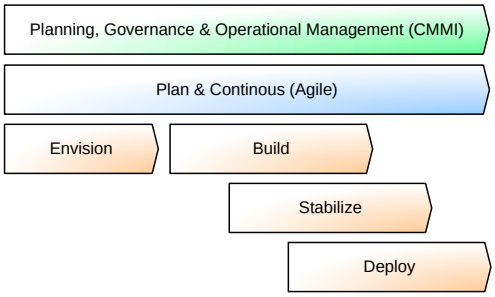



**Abbildung A.5.:** Vereinfachte Darstellung der Tracks und Herstellung der Bezüge Basismodell, Querschnittsdisziplinen

Die Abbildung A.4 gibt dabei in Anlehnung an [KT06] eine grobe Positionierung der Ablaufmodelle an. Sie stellt auch eine grobe zeitliche Einordnung dar. Eine weiter abstrahierte Darstellung gibt Abbildung A.5 an. Diese berücksichtigt das Basismodell, das neben den eigentlichen Workflows und Tasks jeweils noch die unterschiedlich ausgeprägten Managementmodelle berücksichtigt.

# B. Work Item Schema und Definition

Dieses Kapitel fasst einige Überlegungen zu **Work Items** zusammen. Dabei vertiefen wir zu nächst das Work Item XML-Schema und entwerfen im Anschluss Work Items, die die Integration des V-Modells, wie in Kapitel 3 erarbeitet, unterstützen.

## B.1. Schemadefinitionen von Work Items

Ein Work Item ist eine komplexe Datenstruktur, die in XML modelliert ist und durch den Process Template Editor verwendet wird. Das dem Work Item Typ zugrunde liegende Schema (vgl. Abbildung B.1) ist im Visual Studio SDK zu finden.

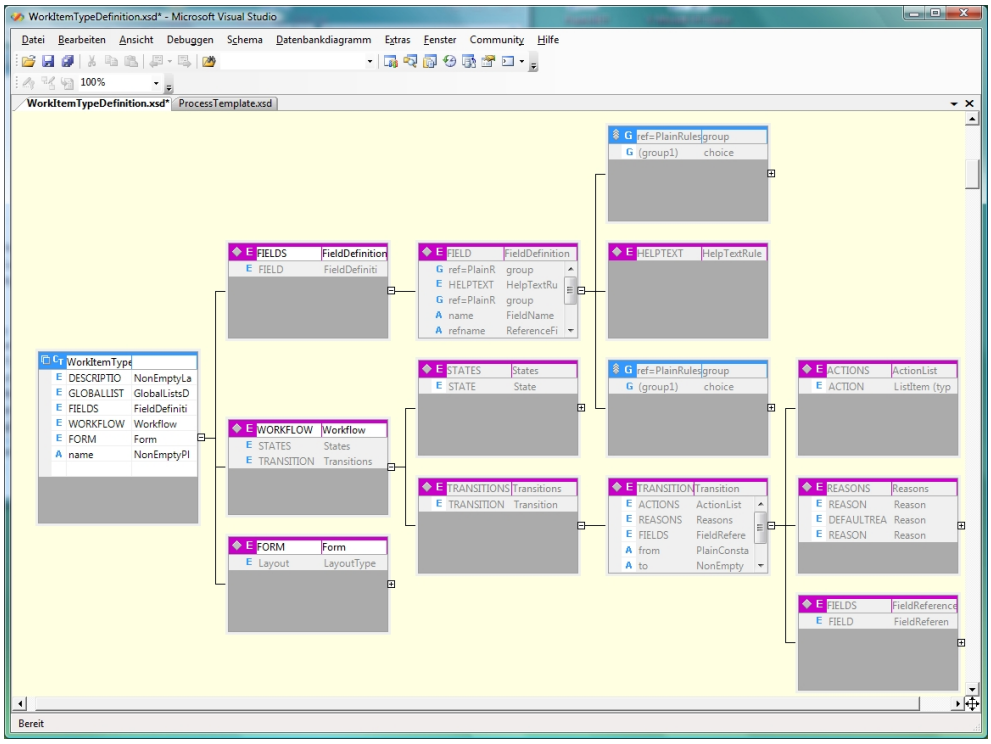



**Abbildung B.1.:** XML-Schemadefinition von Work Items (im Visual Studio XML Designer)

In Listing B.1 haben wir eine verkürzte und vereinfachte Form der Typdefnition dargestellt. Es handelt sich um das Root-Element, das die Grundstruktur eines Work Items darstellt. Ein Work Item besteht somit aus einer Menge von Variablen (Fields), einem Workflow (dargestellt durch einen Zustandsautomaten, Beispiel siehe Abbildung 2.4) und einer Beschreibung einer Benutzungsschnittstelle (Form).

Diese integrierte Struktur macht es im Kontext einer V-Modell Integration erforderlich, ausgewählte Produkttypen, die sich als Work Items anbieten, zu remodellieren (Kapitel 3.2). Insbesondere muss hier herausgearbeitet werden, dass es sich beim Zustandsautomaten der Work Items nicht zwangsweise um die Zustandasmaschine der V-Modell XT Produkttypen handeln muss. Mit den Workflows der Work Items gehen bis zu gewissen Grenzen (eigenständige) Prozesse einher, denen Methoden zugrunde liegen können. Ein Beispiel zeigt die Abbildung B.2, das den modellierten Workflow für das Work Item „Bug" darstellt. Das V-Modell sieht solche Methoden

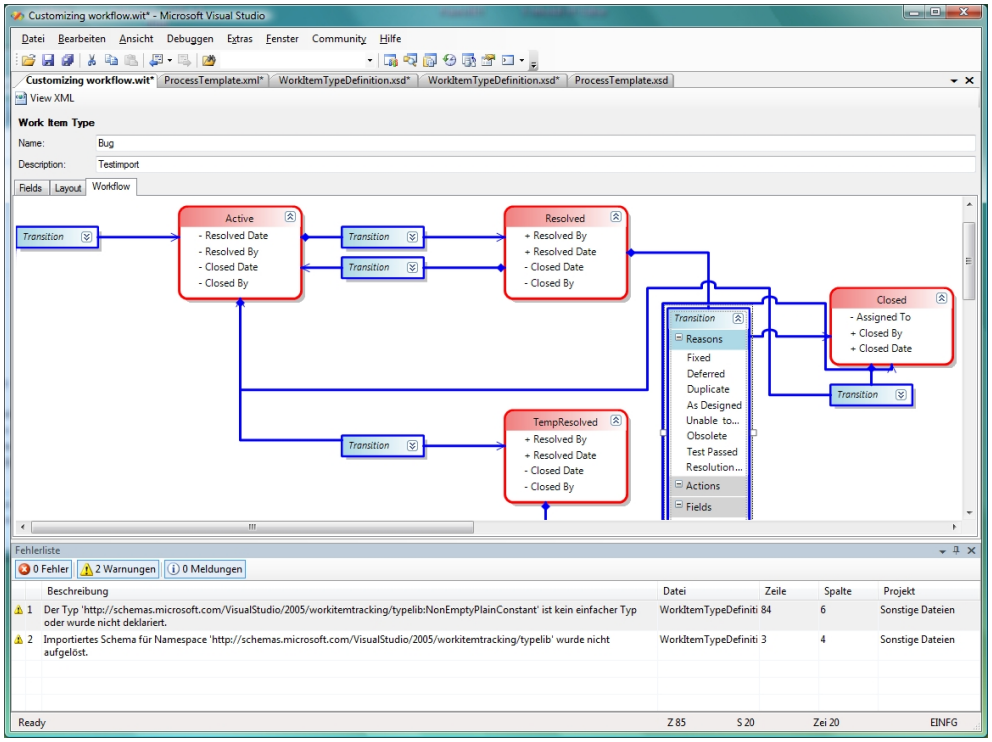



**Abbildung B.2.:** Workflow-Definition für ein Work Item mit Zuständen und Transitionen

nur begrenzt oder gar nicht vor. Im folgenden Abschnitt B.2 gehen wir darauf weiter ein. Für das high-level Design von Work Items aus dem V-Modell benötigen wir nur das Gesamtschema und weiterhin die Teile für die Definition von Workflows (Listing B.2).

**Listing B.1:** XML Typdefinition eines Work Items, Root-Elementdefinition (vgl. Abbildung B.1)

```
<!-- WORKITEMTYPE -->
  <xs:complexType name="WorkItemType">
    <xs:sequence>
      <xs:element name="DESCRIPTION" type="typelib:NonEmptyLargeText"
          minOccurs="0" />
      <xs:element name="GLOBALLISTS" type="typelib:GlobalListsDef" minOccurs=
          "0" />
      <xs:element name="FIELDS" type="FieldDefinitions" />
      <xs:element name="WORKFLOW" type="Workflow" />
      <xs:element name="FORM" type="Form" />
    </xs:sequence>
    <xs:attribute name="name" type="typelib:NonEmptyPlainConstant" use="
        required" />
  </xs:complexType>
```

**Listing B.2:** XML Typdefinition von Status- und Transition-Elemente für den Zustandsautomaten

```
  <!-- Nur ein Status-Element ohne Root -->
  <!-- State -->
  <xs:complexType name="State">
    <xs:sequence>
      <xs:element name="FIELDS" type="FieldReferences" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="value" type="typelib:NonEmptyPlainConstant" use="
        required" />
  </xs:complexType>

  <!-- Nur ein Transition-Element ohne Root -->
  <!-- Transition (TRANSITION) -->
  <xs:complexType name="Transition">
    <xs:all>
      <xs:element name="ACTIONS" type="ActionList" minOccurs="0" />
```

```
      <xs:element name="REASONS" type="Reasons" />
      <xs:element name="FIELDS" type="FieldReferences" minOccurs="0" />
    </xs:all>
    <xs:attribute name="from" type="typelib:PlainConstant" use="required" />
    <xs:attribute name="to" type="typelib:NonEmptyPlainConstant" use="
        required" />
    <xs:attribute name="for" type="typelib:IdentityName" />
    <xs:attribute name="not" type="typelib:IdentityName" />
  </xs:complexType>
```

## B.2. V-Modell XT Work Items

In Work Items sehen wir eine echten Mehrwert für einen Prozess, der bei einer Integration in das Visual Studio genutzt werden sollte. Für das V-Modell ist es hier jedoch erforderlich, Einzelne Produkttypdefinition zu remodellieren. Andere Aufgaben können beispielsweise aus den bereits existierenden Work Item Definitionen der MSF-Derivate (Anhang A.3.1, Tabelle A.4).

Aufgrund der höheren Ähnlichkeit zwischen V-Modell XT und MSF-CMMI greifen wir auf diesen zurück, um Work Items weiter zu verwenden. In Tabelle A.4 sind alle verfügbaren Work Item Typdefinitionen zusammengestellt, die für die beiden MSF-Derivate verfügbar sind. Unter `http://www.scrumforteamsystem.com/en/default.aspx` finden sich beispielsweise für Scrum weitere prozessspezifischen Typdefinitionen und ein entsprechendes Processtemplate. Standarmäßig definiert der MSF-CMMI die folgenden Work Items:

- Fehler (Bug)
- Aufgabe (Task)
- Problem (Issue)
- Änderungsanforderung (Change Request)
- Risiko (Risk)
- Anforderung (Requirement)
- Überprüfung (Review)

Von diesen decken sich Aufgaben, Probleme, Änderungsanforderungen und Risiken mit den Disziplinen, die auch das V-Modell (jedoch abstract) beschreibt. In Erweiterung der Analysen aus Kapitel 3.2 können diese vier Work Items als Ziel einer Abbildung dienen.

| Disziplin | V-Modell Produkt | Work Item |
|---|---|---|
| Aufgabenmanagement | Arbeitsauftrag | Aufgabe |
| Risikomanagement | Risikoliste | Risiko |
| PÄM | Änderungsstatusliste | Problem |
| PÄM | Änderungsentscheidung | Änderungsanforderung |
| PÄM | Problem-/Änderungsbewertung | |
| PÄM | Problemmeldung/Änderungsantrag | |

**Tabelle B.1.:** Projektdisziplinen, V-Modell XT Produkt und ihre Abbildung auf Work Items

Tabelle B.1 stellt die ausgewählten Prozesse, die im V-Modell dafür vorgesehenen Produkte und die passenden Work Items gegenüber. Die Work Items können bereits als Vorlage dienen, müssen jedoch in Teilen ergänzt werden. Auch ist es erforderlich, bestimmte Definitionen im V-Modell durch einen Generator (Kapitel 3.1) zu ergänzen. Auf der Ebene des Process Templates lassen sich aber nicht nur Work Item Typdefinitionen erstellen, sondern auch schon Standardaufgaben hinterlegen, also eine Liste von initialen Aufgaben analog zu [KK06] (siehe auch Abbildung B.3).

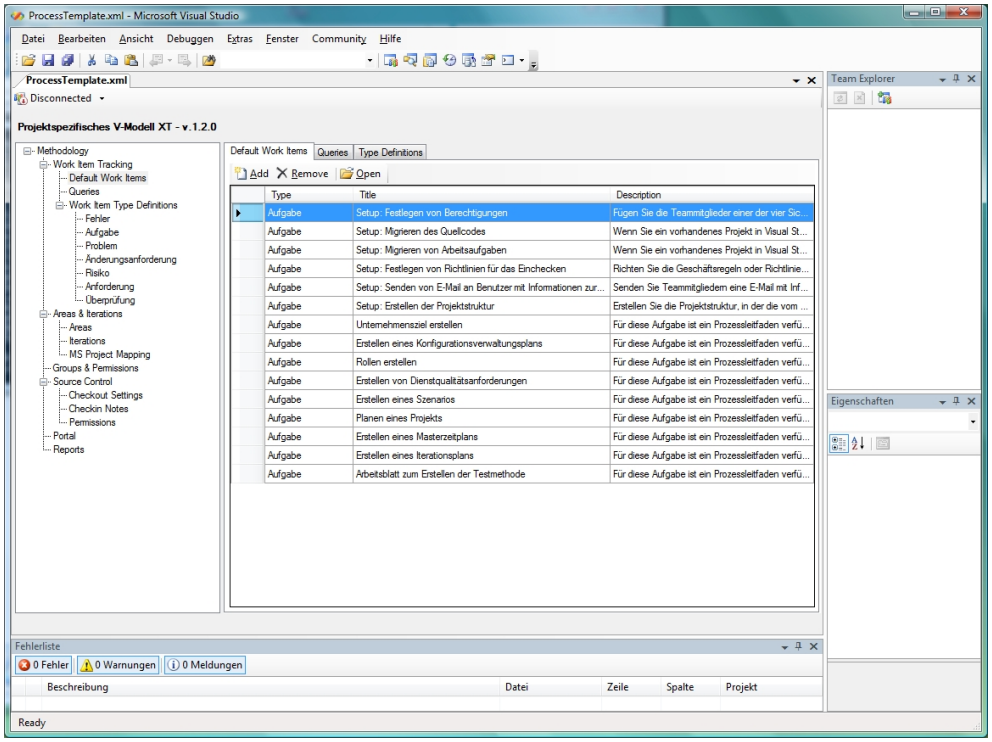



**Abbildung B.3.:** Standardinstanzen von Work Items für die initialen Projektaufgaben

Für diese Option muss das Schema/die Typdefinition der „Standardaufgabe" so angepasst werden, dass eine angemessene Abbildung der V-Modell Aktivitäten (Projektspezifisches V-Modell Referenz Teil 6) integriert werden kann. Ein direkte Anbindung der bearbeiteten Produkte kann so gewährleistet werden. beachtet muss an der Stelle jedoch werden, dass nicht alle Produkte des V-Modells als Template verwendet werden (Kapitel 3.2.2). Daher soll das Standard-Work Item nicht verändert und weiterhin im Bereich Aufgabenmanagement eingesetzt werden. Zur Angleichung der Teminologie wird es jedoch als **Arbeitsauftrag** bezeichnet. Für den oben skizzierten Anwendungsfall wird ein entsprechendes Work Item **Aktivität** eingeführt, das die oben beschriebenen Eigenschaften erfüllt (Tabelle B.2).

| Work Item (alt) | Einsatzzweck | Work Item (neu) |
|---|---|---|
| Aufgabe | Aufgabenmanagement | Arbeitsauftrag |
| Aufgabe | Aktivitätsmodell (XT) | Aktivität |

**Tabelle B.2.:** V-Modell XT-bezogene Work Items für das Aufgabenmanagement

Aktivitäten sind hier also spezielle Aufgaben. Einschränkungen sind hier hinsichtlich der Datenstruktur und deren Umfang zu machen. Ein passender Zustandsautomat (passend zum V-Modell XT Produktzustandsmodell) ist analog zu [KK06] umzusetzen.

Für das Problem- und Änderungsmanagement des V-Modells sind nach aktuellem Analysestand die Work Item Typen Problem und Änderungsanforderung ausreichend. Sie müssen jedoch hinsichtlich der formalen Kriterien des V-Modells angepasst werden, sodass Bewertungen und vor allem Entscheidungen reproduzierbar hinterlegt werden können.

Die Entwicklungs-/Codebezogenen Work Items übernehmen wir in diesem Konzept vollständig, da das V-Modell auf eine konkrete Entwicklungsmethode angewiesen ist und diese in Form der Standard-Work Items und deren Abstimmung mit dem Visual Studio vorliegt.

# Literaturverzeichnis


[Aly06] ALYOKHINA, A.: *Konzeption einer V-Modell XT Erweiterung zur Integration des Microsoft Solution Frameworks*. Diplomarbeit, Technische Universität München, 2006.

[DW99] DRÖSCHEL, W. und M. WIEMERS: *Das V-Modell 97*. Oldenburg, 1999.

[Fri06] FRIEDRICH, J.: *Technische und semantische Transformation von Vorgehensmodellen*. Diplomarbeit, Technische Universität München, 2006.

[Gna06] GNATZ, M.: *V-Modell XT: Meta-Modell und Konsistenzbedingungen*. Online, jan 2006. Version 1.1.

[GP06] GUCKENHEIMER, S. und J. J. PEREZ: *Software Enginnering with Microsoft Visual Studio Team System*. Addison Wesley, 2006.

[KB05] KUHRMANN, M., NIEBUHR D. und C. BARTELT: *Anwendung des V-Modell XT - Stand und Erfahrungen aus der Pilotierungsphase*. In: CREMERS, ARMIN B., RAINER MANTHEY, PETER MARTINI und VOLKER STEINHAGE (Herausgeber): *Informatik 2005 - Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V.(GI)*, Band P-67, Seiten 259–263. Gesellschaft für Informatik, sep 2005.

[Köh06] KÖHLER, P.-T.: *Prince 2 - Das Projektmanagement-Framework*. Nummer ISBN: 3-540-29181-4 in *xpert.press*. Springer, 2006.

[KK06] KALUS, G. und M. KUHRMANN: *CollabXT - Ein Ansatz zur automatischen Erzeugung von Kollaborationsportalen aus dem V-Modell XT*. (to appear), 2006.

[KT06] KUHRMANN, M. und T. TERNITÉ: *Implementing the Microsoft Solution Framework for Agile Software Development as concrete Development-Method in the V-Modell XT*. International Transactions on Systems Science and Applications, 2006.

[Kuh06] KUHRMANN, M.: *Projektspezifische Anpassungen nach dem Tailoring des V-Modell XT durchführen*. In: BISKUP, HUBERT und RALF KNEUPER (Herausgeber): *13. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e.V. (GI) zum Thema: Nutzen und Nutzung von Vorgehensmodellen*, Seiten 27–42. Shaker Verlag, mar 2006.

[NK05] NIEBUHR, D. und M. KUHRMANN: *Projekt-Tüv*. iX - Magazin für professionelle Informationstechnik, 1(1/2006):126–129, dec 2005.

[RH06] RAUSCH, A., HÖHN R. BROY M. BERGNER K. und S. HÖPPNER: *Das V-Modell XT*. eXamen.press. Springer, 2006.

[Tur06] TURNER, M.: *Microsoft Solutions Framework Essentials*. Nummer ISBN: 0735623538. Microsoft Press, 2006.

[vB04] BON, JAN VAN (Herausgeber): *MSF, a pocket guide*. Van Haren Publishing, jan 2004.