



November 2011

PROJEKT-
KONFIGURATIONS-
MANAGEMENT
UNTER z/OS

Martin Wallrabenstein
A:gon Solutions GmbH

■ Abstract

Größere IT Projekte erfordern häufig Änderungen bzw. Erweiterungen der IT-Infrastruktur. Leider wird die Notwendigkeit solcher Änderungen erst in einer fortgeschrittenen Projektphase erkennbar. Dieses Dokument beschreibt anschaulich, welche Maßnahmen an der Infrastruktur ergriffen werden können, um den Projekterfolg sicherzustellen. Neben technischen und organisatorischen Maßnahmen werden auch die unter z/OS verwendeten Techniken beschrieben.

■ Inhalt

Allgemein wird unter Konfigurationsmanagement die Herstellung und Erhaltung von IT-Dienstleistungen verstanden. Ein System, welches diese Dienstleistungen zur Verfügung stellt, ist ein IT-Produkt. Größere Änderungen an IT-Produkten finden in der Regel in Projekten statt, deshalb sollte zu Beginn oder besser vor größeren Projekten der bestehende Konfigurationsprozess überprüft und gegebenenfalls erweitert werden, damit die bevorstehende Welle von Änderung am Produkt reibungslos produktiv eingesetzt werden kann. Im Rahmen des Konfigurationsmanagements für IT-Projekte sollte beispielsweise geprüft werden, ob es eine Qualitätsumgebung für den Performancetest unter produktiven Bedingungen gibt. Aber auch Prüfungen, ob zu jedem Programm die richtige Version des Quellcodes vorliegt, gehören dazu.

Schwerpunkt dieses Dokuments ist der Build- und Deploymentprozess von Software, die unter z/OS erstellt wird. Neben den zu beachtenden Techniken und den organisatorischen Maßnahmen beschreibt es mögliche Erweiterungen an einer z/OS Entwicklungsarchitektur und stellt die damit verbundenen Problematiken und Techniken dar.

In der Regel sind diese Prozesse aufgrund Ihrer Komplexität mehr oder weniger toolunterstützt. Das Whitepaper geht nicht auf ein spezielles Tool ein, sondern stellt die allgemeine Arbeitsweise eines Changemanagementtools vor.

In den einzelnen Kapiteln dieses Dokuments werden dargestellt:

1. **Erweiterung des Umgebungskonzepts**

In diesem Teil werden mögliche Erweiterungen an einer Entwicklungsumgebungsarchitektur (wie die Einbindung einer Quality Umgebung) vorgestellt.

2. **Techniken**

In diesem Kapitel werden Techniken beschrieben, die im Rahmen des Konfigurationsmanagement Systems zu berücksichtigen sind. (Ablauf eines Compile, die Arbeitsweise eines Changemanagementtools und die Verkettung von Dateibibliotheken)

3. **Organisation von Einsätzen**

Zusätzlich zu den technischen Maßnahmen sind organisatorische Maßnahmen erforderlich (Wie beispielsweise die Sicherstellung einer Synchronisation bei parallelen Änderungen an einer Entität). Diese Maßnahmen werden hier beschrieben.

4. **Anhang**

Im Anhang werden die im Dokument verwendeten Fachbegriffe beschrieben.

■ Einführung in die Praxis

Das in diesem Dokument Beschriebene wird zur anschaulichen Darstellung an Beispielen gezeigt. Diese Textstellen sind links durch einen senkrechten Strich gekennzeichnet. Für die Beispiele wird folgender Fall angenommen:

Die Firma Maxstein GmbH hat Ihren langjährigen Konkurrenten Wiesendorf übernommen. Nach der Übernahme sollen die IT-Systeme zusammengeführt werden. Im Anschluss an die Analysephase wurde entschieden, die IT-Systeme der Firma Maxstein funktional und technisch zu erweitern und anschließend die Daten der Firma Wiesendorf in das neue Maxstein System zu integrieren.

Der Geschäftsführer der Firma Maxstein Dr. Mayer beschließt das Projekt in 3 Teilprojekte aufzuteilen:

1. Erweiterung der bestehenden Infrastruktur.
2. Funktionale Erweiterung der Software.
3. Datenmigration der Daten der Firma Wiesendorf.

Für die Umsetzung der Teilprojekte wurde die dargestellte Grobplanung aufgestellt.

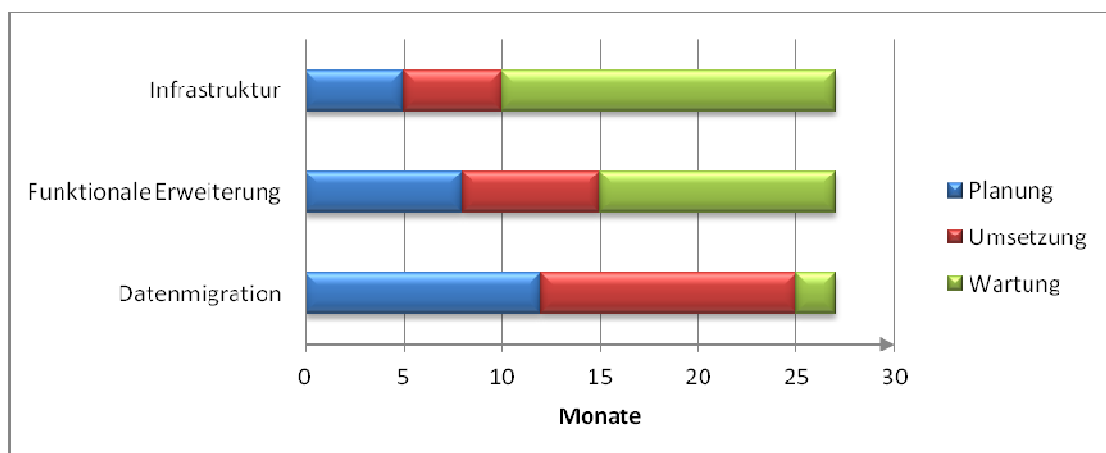


Abbildung 1: Bitte immer eine aussagekräftige Bildunterschrift

Die Firmen Maxstein und Wiesendorf haben wenig Erfahrung mit der Umsetzung von größeren Projekten. Aus diesem Grund, wird beschlossen für die Teilprojekte externe Projektleiter einzusetzen. Die Rolle des Gesamtprojektleiters übernimmt der Projektleiter Herr Müller der Firma Wiesendorf. Herr Müller wird von einem erfahrenen Projektleiter eines IT Dienstleistungsunternehmens unterstützt.

■ 1 Umgebungsanpassung

Werden in einem Unternehmen überwiegend Wartungsarbeiten und kleinere Projekt durchgeführt, ist eine einfache Umgebungsarchitektur ausreichend und sinnvoll. Eine solche Umgebungsarchitektur kann wie in folgender Darstellung aussehen.

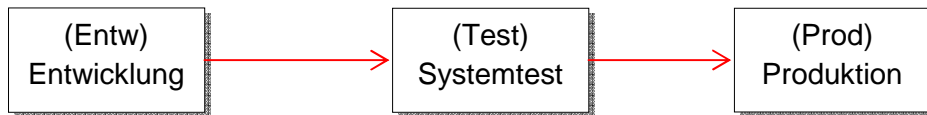


Abbildung 2: Umgebungsarchitektur 1-1 (Ausgangsumgebungsarchitektur)

Beispiel: Ausgangs Umgebungsarchitektur der Firma Maxstein

In unserem Beispiel wird diese Umgebungsarchitektur als Ausgangssituation (Umgebungsarchitektur der Firma Maxstein) unterstellt. Im weiteren Verlauf wird diese Architektur für die projektspezifischen Anforderungen erweitert.

■ 1.1 Hinzufügen einer Umgebung für den Qualitätstest

Eine Qualitätstestumgebung wird in der Regel zwischen der Test- und der Produktionsumgebung hinzugefügt, um die Softwaremigration nach Produktion und das Verhalten der Software produktionsnah zu testen. Bei diesen Tests wird davon ausgegangen, dass die Software fachlich korrekt ist.

Eine Quality Umgebung wird vorgesehen, um:

- die Migration des Softwarepakets zu testen

Nach Abnahme der Software in der Systemtestumgebung wird das Softwarepaket in die Quality Umgebung migriert und ein Regressionstest durchgeführt. Dies ist erforderlich weil, im Laufe des Systemtests oft noch Änderungen an der Software stattfinden. Gründe hierfür können sein: Fehler aus dem Systemtest, das Softwarepaket wird in mehreren Teilen geliefert, Teile der Software sollen nicht migriert werden.

Jede Änderung im Systemtest kann jedoch ungewünschte Nebeneffekte auf die Migration und den Test haben. Deshalb ist es sinnvoll, nach Abnahme der Software in der Systemtestumgebung diese zunächst in die Quality Umgebung zu migrieren und einen Regressionstest durchzuführen.

- die Software unter produktiven Bedingungen zu testen

In der Systemtestumgebung steht die Funktion der Software im Vordergrund. Diese Tests finden oft mit reduzierten Daten und einem vereinfachten Umgebungssetup statt. Last Tests finden aus diesem Grund verstärkt in der Quality Umgebung statt.

Beispiel Hinzufügen einer Quality Umgebung in die Umgebungsarchitektur

Der Chefentwickler des IT-Beratungsunternehmens Dipl. Ing. Schultz schlägt nach einer Analyse der IT-Infrastruktur vor, eine Qualitätstestumgebung aufzubauen. Diese Umgebung wird im ersten Schritt für Performancetests unter produktiven Bedingungen vom Teilprojekt Infrastruktur benötigt. Gleichzeitig soll der Aufbau dieser Umgebung gut dokumentiert werden, damit weitere Umgebungen schnell aufgesetzt werden können. Nach einer längeren Diskussion kann Herr Schultz die Mitarbeiter der Firma Maxstein überzeugen, trotz der vorgetragenen Bedenken, diese Umgebung aufzubauen. Diese Umgebung wird im Migrationspfad zwischen Systemtest und Produktion eingefügt. Der direkte Migrationspfad zwischen Systemtest und Produktion bleibt unverändert bestehen, damit Problemfixes ohne Umweg nach Produktion eingesetzt werden können. Die Tatsache, dass an dem bestehenden Migration Pfad zunächst festgehalten wird, erleichterte die Zustimmung zu dieser Änderung. Zur Dokumentation des Umgebungsaufbaus stellt Herr Schultz eine Vorlage (Diese Vorlage wird in Kapitel 3 organisatorische Maßnahmen vorgestellt) zur Verfügung.

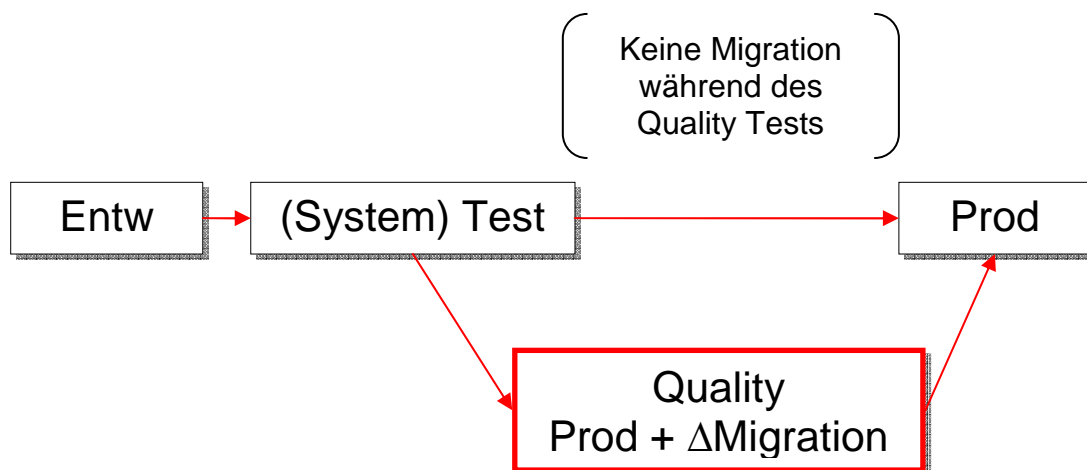


Abbildung 3:

Für die Qualitätstestumgebung wird folgendes festgelegt

- Der Zeitraum zwischen Migration von Systemtest in die Quality Umgebung und anschließend nach Produktion sollte kurz gewählt werden, damit Fix Einsätze nach Produktion das Testergebnis nicht beeinträchtigen.
- Zum Zeitpunkt eines Qualitätstest sollten sich nur Änderungen in der Qualitätsumgebung befinden, die auch nach Produktion gebracht werden sollen.
- Datenbanktabellen und Dateien haben die gleiche Größe wie in Produktion
- Programme werden mit den Compileroptionen für Produktion umgewandelt

■ 1.2 Hinzufügen eines Wartungspfads

Befinden sich größere Projektergebnisse im Test, existieren auch viele von Produktion abweichende Unterprogramme in der Testumgebung. Werden vor Einführung dieser Neuerungen Anpassungen in Produktion benötigt (z.B. ein Produktionsfix), müssen jedoch zum Test dieser Änderungen die Unterprogrammversionen aus Produktion aufgerufen werden. Dies ist jedoch in der „Projekttestumgebung“ nicht (ohne weiteres) möglich.

Damit notwendige Änderungen weiterhin vor Einsatz der Projektergebnisse in Produktion (auch mit den produktiven Unterprogrammversionen) getestet werden können, ist es sinnvoll einen parallelen Pfad für Wartung und kleinere Projekte einzurichten.

Im Folgenden ist ein Beispiel für die Erweiterung einer bestehenden Umgebungsarchitektur (schwarz) um einen Wartungspfad (rot) dargestellt.

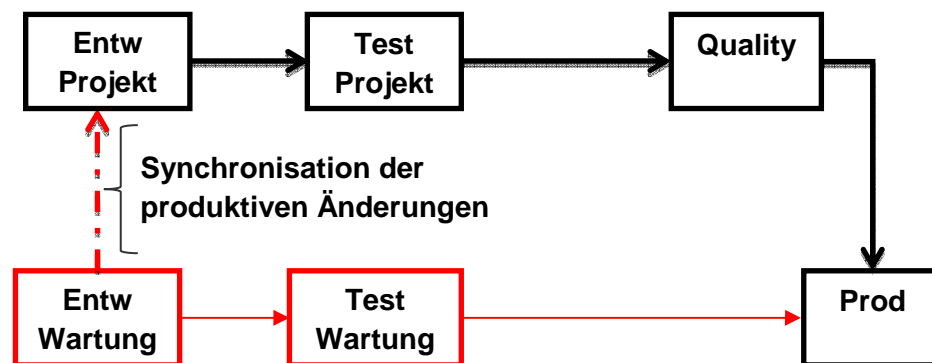


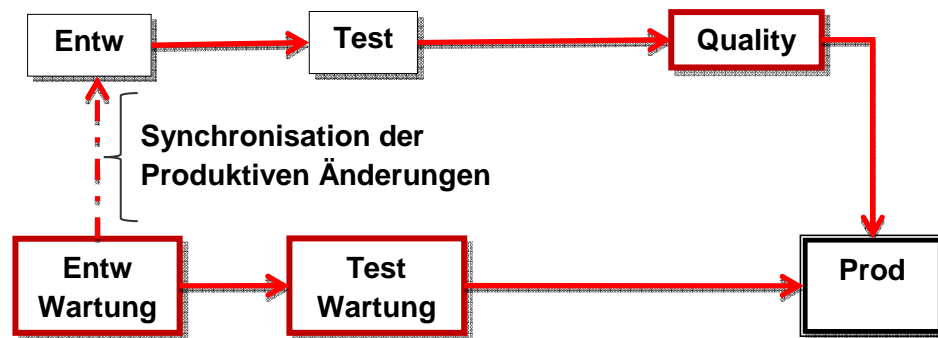
Abbildung 4: Umgebungsarchitektur 1-2 (Erweiterung um einen Wartungspfad)

Änderungen, die über den Wartungspfad nach Produktion gebracht werden, müssen in die Projektentwicklungsumgebungen nachgezogen (synchronisiert) werden, damit mit Einführung der Projektergebnisse in der produktiven Umgebung diese Änderungen nicht rückgängig gemacht werden. Die Verantwortung für die Synchronisation kann den Entwicklern, die für die Durchführung der produktiven Änderung verantwortlich sind, übertragen werden.

Beispiel - Erweiterung der Umgebungsarchitektur um einen Wartungspfad

Nachdem sich die neue Quality Umgebung bewährt hat (die Migration von der Systemtest in die Quality Umgebung und die Migration aus der Quality Umgebung nach Produktion funktionieren). Der Umgebungsaufbau wurde zwar widerwillig aber gut

dokumentiert.) stellt Herr Schultz auf dem wöchentlichen Projektmeeting sein Konzept für einen Wartungspfad vor.



Zunächst wird bezweifelt, ob eine solche Umgebung notwendig ist. In dem Meeting fallen Aussagen wie „unsere 20 zu ändernden Programme werden wir ja noch ohne Wartungspfad händeln“ - „also für meinen Bereich sind es mindesten 30“. Herr Schultz berichtet aus seinem früheren Projekt: Dort wurden mitten in der Umsetzungsphase wichtige Mitarbeiter abgezogen um im Hau Ruck Verfahren neue Testumgebungen aufzubauen. Für die Migration wurden Behelfsverfahren, die einen erheblichen manuellen Nachbearbeitungsaufwand zur Folge hatten, aufgesetzt. Insgesamt führte dies zu einer erheblichen Projektverzögerung und Kostensteigerung.

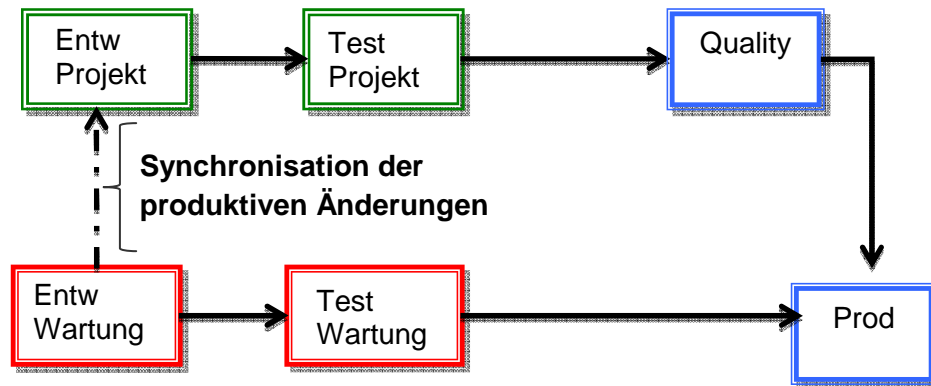
Der Gesamtprojektleiter Herr Müller übernimmt die Aufgabe das Konzept von Herrn Schultz im Lenkungsausschuss vorzustellen. In diesem Ausschuss berichtet Herr Müller auch über den erfolgreichen Aufbau der Quality Umgebung. Vom Lenkungsausschuss wird Herr Müller beauftragt, die Änderungen an der Umgebungsarchitektur durchzuführen. Die notwendigen organisatorischen Maßnahmen werden von den Bereichsleitern der Firma Maxstein durchgeführt.

Als Ergebnis wird der folgende Umgebungsplan ausgearbeitet (inklusive der Zuordnung der Umgebungen zu Teams). Dieser Umgebungsplan wird allen IT Mitarbeitern zu Verfügung gestellt.

Team Zuordnung der Umgebungen

Damit die Zusammenführung der IT – Systeme der Firmen Wiesendorf und Maxstein zügig umgesetzt werden kann, wird ein eigenes Wartungsteam (rot) gegründet. Dieses Wartungsteam ist für die Wartungsumgebungen zuständig und soll die Projektmitglieder (grün) von Wartungsaufgaben befreien und auch für die Synchronisation der Wartungsänderungen in die Entwicklung sorgen. Neben dem Wartungsteam und der Entwicklungsmannschaft für die Zusammenführung der IT – Systeme gibt es noch die Produktionsbetreuung (blau). Die Produktionsbetreuung ist auch für die Quality Umgebung verantwortlich. Damit haben die Produktionsbetreuer

die Möglichkeit die Migration der Projektänderungen vor Einsatz nach Produktion zu testen.



■ 1.3 Umgebungen für Fachtests und Schulungen

Werden weitere Umgebungen z.B. für Schulungen oder Fachtester benötigt, muss zunächst geklärt werden, auf welchem Softwarestand aufgesetzt werden soll, über welchen Zeitraum die Umgebungen benötigt werden und wie lange der Aufbau einer Umgebung dauert.

Softwarestände auf die aufgesetzt werden kann sind:

- Die Produktionsumgebung
- Die Quality Umgebung
- Die Systemtestumgebung

Die zusätzlich benötigten Umgebungen können in den Migrationspfad aufgenommen werden oder als unabhängige Umgebungen aufgebaut werden. Eine Umgebung im Migrationspfad hat den Vorteil, dass hier die gleichen Migrationsprozesse verwendet werden können, wie für die übrigen Umgebungen auch. Die Quell und Zielumgebungen sind ebenfalls festgelegt.

Im Gegensatz dazu muss für eine unabhängige Umgebung definiert werden, was wann von wo propagiert wird und gegebenenfalls müssen für diese speziellen Umgebungen geänderte Objekte in andere Umgebungen zurücksynchronisiert werden. Dafür kann dieser Umgebungstyp schnell an die Bedürfnisse der Tester angepasst werden.

Beispiel 1 – Anwendertestumgebung mit im Migrationspfad eingebunden

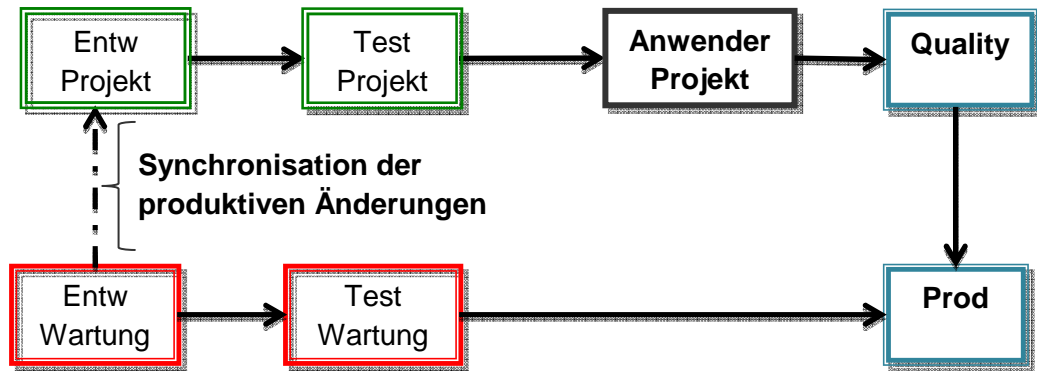


Abbildung 6: Umgebungsarchitektur **Fehler! Kein Text mit angegebener Formatvorlage im Dokument.**-3 - Umgebung für den Fachbereich (Bild 1)

Beispiel 2 – Anwendertestumgebung als unabhängige Umgebung

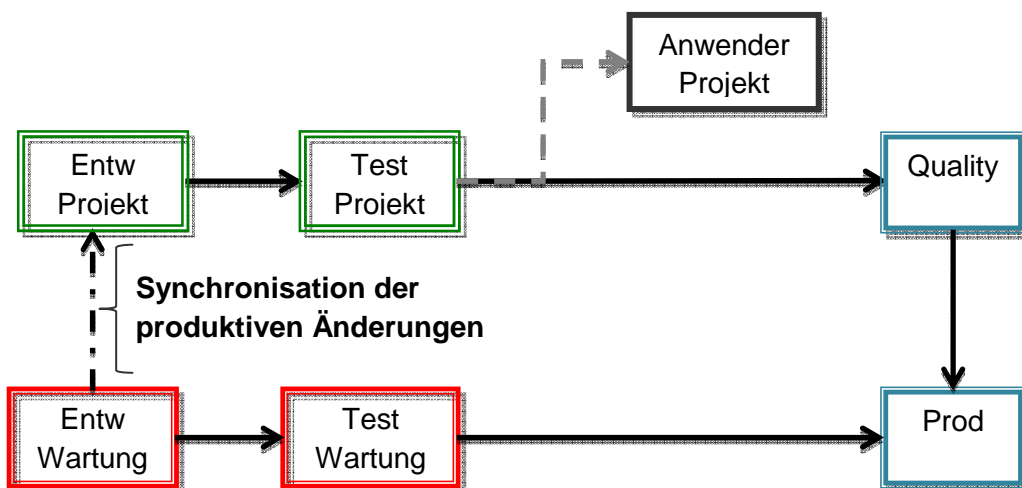
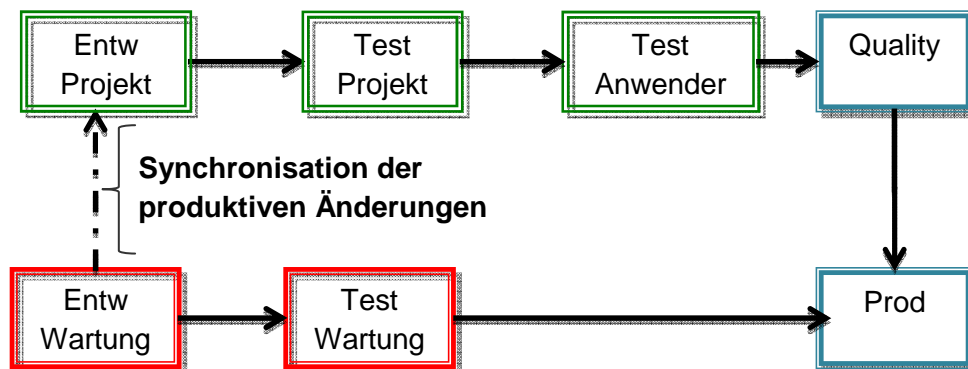


Abbildung 7: Umgebungsarchitektur **Fehler! Kein Text mit angegebener Formatvorlage im Dokument.**-4 - Umgebung für den Fachbereich (Bild2)

Beispiel Hinzufügen weiterer Testumgebungen

Nach der Vorstellung des neuen Projekts bei der Firma Wiesendorf, auf dem Herr Müller das Teilprojekt Infrastruktur vertritt, äußern die Mitarbeiter den Wunsch die Software der Firma Maxstein kennen zu lernen. Daraufhin entsteht die Idee, die Mitarbeiter der Firma Wiesendorf auch für die Prüfung der funktionalen Erweiterungen einzuplanen. Hierfür wird eine eigene Umgebung im Entwicklungspfad zwischen Test und Qualitätsumgebung eingefügt. In dieser Umgebung ist, bis zur Migration der geänderten Software aus der Testumgebung, der produktive Softwarestand aktiv. Die

Migration der angepassten Software findet nach Abnahme der Software in der Test Umgebung statt.



„Wenn Sie gerade dabei sind neue Umgebungen aufzubauen – Wir benötigen auch eine Umgebung für uns, um den Test der Übernahme der Kundendaten der Firma Wiesendorf zu testen – Derzeit werden von uns in der Testumgebung erstellte Daten von anderen manipuliert und das Testergebnis wird unbrauchbar.“ heißt es auf dem folgenden Projektmeeting , auf dem Herr Müller die neue Anwendertestumgebung vorstellt. Auf die Frage, welcher Softwarestand wird benötigt heißt es „Zunächst die Programme aus Produktion plus ein paar Programme aus der Entwicklung - ach ja - und die Tabellen aus der Entwicklung in Produktionsgröße“.

Herr Schultz entscheidet für das Teilprojekt Datenmigration eine eigene unabhängige Umgebung zum Test aufzubauen. In dieser Umgebung können die Entwickler die Datenmigration von und zum IT System der Firma Wiesendorf testen. Diese Umgebung wird nach Umsetzung des Projekts nicht weiter benötigt. Die neue Umgebungsarchitektur für die Durchführung des Projekts sieht wie folgt aus:

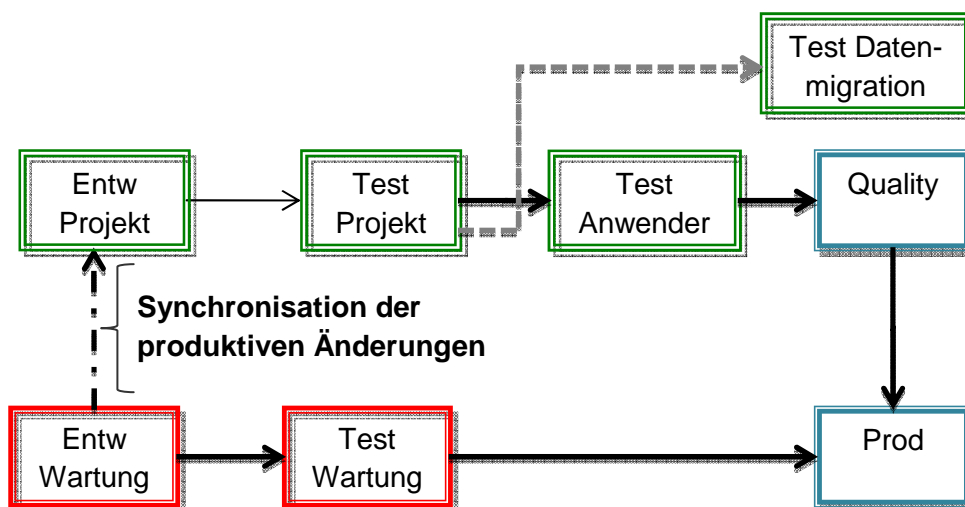


Abbildung 9:

Farbliche Zuordnung zu den Organisationseinheiten: Produktionsbetreuung (blau), Wartungsmannschaft (rot), Entwicklungsteam (grün)

■ 2 Techniken

In diesem Kapitel werden Techniken beschrieben, die im Rahmen des Konfigurations-Managements benötigt werden.

Die folgenden Unterkapitel bestehen aus:

- 2.1 Verkettung und Synchronisation von Umgebungsbibliotheken.
- 2.2 Synchronisation von Umgebungen.
- 2.3 Erstellung von ausführbaren Programmen.
- 2.4 Migration von Programmänderungen.
- 2.5 Arbeitsweise eines Change Managementsystems.

■ 2.1 Verkettung und Synchronisation

Eine einfache und optimale Möglichkeit neue Umgebungen aufzubauen und diese auf dem aktuellen Stand zu halten, ist die *Verkettung*² der *Umgebungsbibliotheken*¹⁷. Bei dieser Technik wird beim Neuaufbau einer Umgebung eine leere Bibliothek vor die Bibliotheken des gewünschten Softwarestands verkettet. In diese Bibliothek gehören nur *Entitäten*¹⁹, die vom Softwarestand der nächsten Umgebung abweichen. Durch diese Technik ist eine Überprüfung der Migration relativ einfach (z.B. muss nach einer vollständigen Migration in die nächste Umgebung diese Bibliothek leer sein). Oftmals ist es nicht möglich Bibliotheken aller *Entitätstypen*¹⁹ zu verketteten. Gründe hierfür sind: Nicht alle verwendeten Tools unterstützen die Verkettung (z.B. unterstützen die meisten Job Scheduler keine Verkettung von JCL Bibliotheken) oder in den Entitäten befinden sich umgebungsspezifischen Anpassungen.

Ist die Verkettung nicht möglich, müssen diese Bibliotheken technisch *synchronisiert*³ bzw. synchron gehalten werden. Neben der technischen Synchronisation gibt es noch eine logische Synchronisation, diese ist jedoch unabhängig von der verwendeten Technik und wird im nächsten Kapitel (Organisatorische Maßnahmen) behandelt.

Entitäten, bei denen die Verkettung in der Regel von den verwendeten Tools bzw. vom Betriebssystem unterstützt / nicht unterstützt wird:

Entität	Tools und Objekte
Entitäten, bei denen die Verkettung in der Regel von den Tools unterstützt wird	
Copybooks	Der Compiler unterstützt die Verkettung von Copybook Libraries.

	Load Module REXX Prozeduren Clist Prozeduren ISPF Entitäten	Bei diesen Entitäten durchsucht das Betriebssystem die Verkettung beim Starten und nachladen.
Entitäten, bei denen die Verkettung in der Regel nicht unterstützt wird		
	DB2 Packages	DB2 Packages enthalten Umgebungsspezifika (z.B. den Tabellen –Qualifier). Der Aufruf des Produktions Package führt zu einem Zugriff auf die Produktionstabellen, dies ist nur in Produktion gewünscht.
	JCL Datenkarten	Datenkarten und die JCL werden vom Betriebssystem als sequenzielle Dateien behandelt. Bei einer Verkettung werden die Inhalte der Verkettung zu einer gemeinsamen Datei zusammenkopiert und nicht wie gewünscht die erste Datenkarte bzw. JCL verwendet.

Tabelle 1:

2.1.1 Verkettung von Bibliotheken

Das Prinzip einer Verkettung wird hier am Beispiel Verkettung der Umgebungsbibliotheken für Test und Produktion gezeigt. In diesem Beispiel befinden sich die Programme PGM3 und PGM4 in der Testumgebung.

- PGM3 ist eine Weiterentwicklung des Programms aus Produktion. Dieses Programm befindet sich in einer älteren Version bereits in Produktion.
- PGM4 ist eine Neuentwicklung.
- PGM1 und PGM2 wurden nicht geändert, können aber durch die *Verkettung*² aus der Testumgebung aufgerufen und verwendet werden.

Beispiel: Verkettung der *Loadlibraries*¹¹ der Umgebungen Test und Produktion. In dieser Abbildung sind die Dateien in der Reihenfolge der Verkettung in einer JCL dargestellt:

Verkettung in der JCL	
Im Test	In Produktion
//STEPLIB DD DSN=Testlib,...	//STEPLIB DD DSN=Prodlib,...
// DD DSN=Prodlib,...	

Tabelle 2:

Anpassungen am PGM1 oder PGM2, die in Produktion durchgeführt werden, sind sofort in der Testumgebung wirksam, weil sich diese Programme nicht in der Testumgebung befinden und somit aus Produktion genommen werden. Bei einer Lösung durch Synchronisation ist dies nicht der Fall.

Änderungen am PGM3 müssen analysiert und in den Projektumgebungen nachgezogen werden, weil sich durch die Anpassung des Programms PGM3 in Produktion das Programm PGM3 in der Testumgebung nicht ändert.

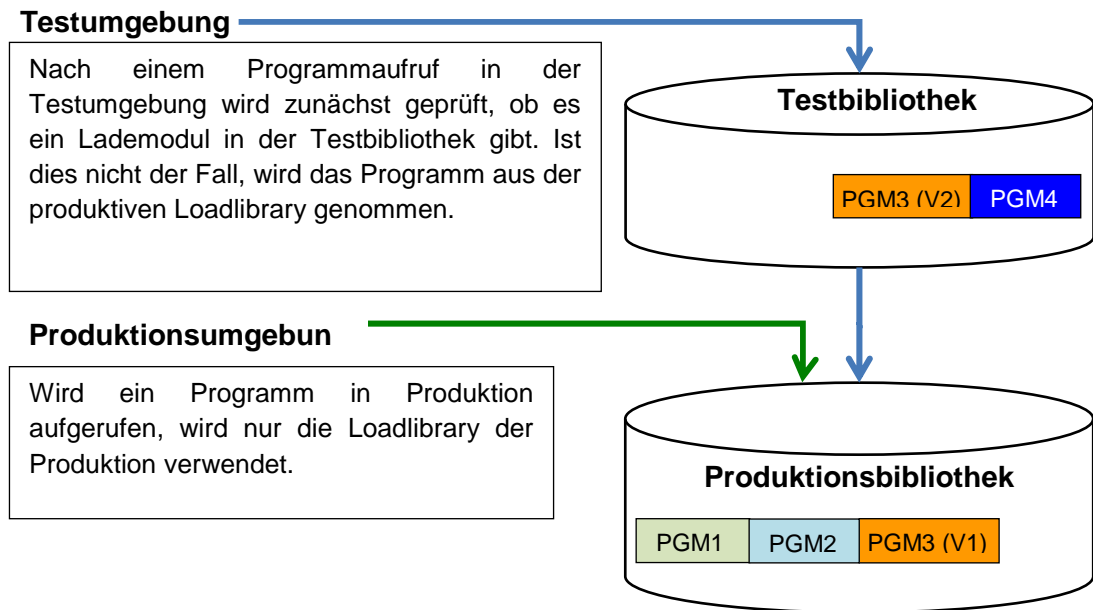


Abbildung 11: Techniken 1 - Verkettung von Bibliotheken

2.1.2 Verkettung von Bibliotheken

Im Unterschied zur Verkettung werden die Entitäten bei der Synchronisation nach Migration in die nächste Umgebung nicht gelöscht. Beim Neuaufbau einer Umgebung werden alle Entitäten der nächsten Umgebung in die neue Umgebungsbibliothek kopiert.

Probleme mit diesem Konzept treten auf, wenn z.B. ein Wartungsfix nicht vollständig in den Projektumgebungen nachgezogen wird. Dieses Problem stellt sich bei der Verkettung von Umgebungsbibliotheken oftmals nicht. Bei verketteten Bibliotheken müssen nur Entitäten synchronisiert werden, die sich in der Weiterentwicklung befinden.

Beispiel: Synchronisierte *Loadlibraries*¹¹ der Umgebungen Test und Produktion :

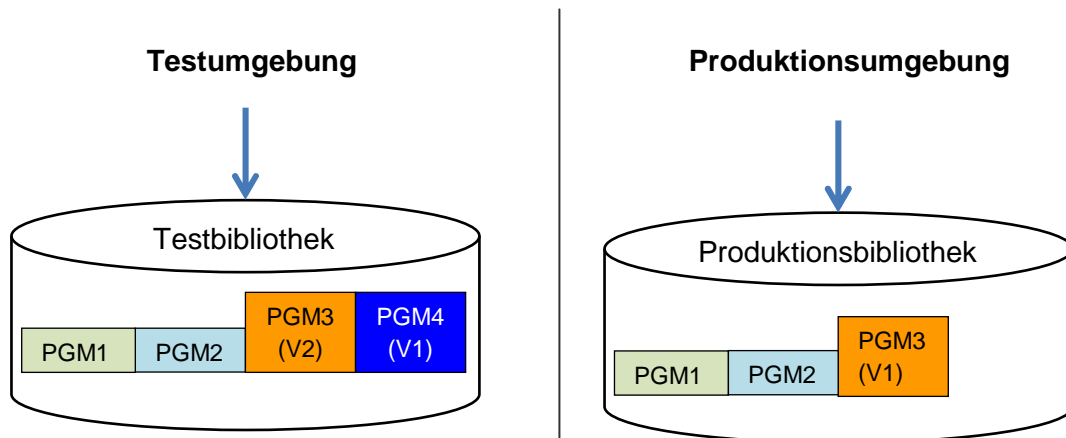


Abbildung 12: Techniken 2 – Synchronisation von Umgebungsbibliotheken

Im Beispiel zu sehen: Anpassungen am PGM1 oder PGM2, die in Produktion durchgeführt werden, müssen zur Synchronisation in die Testumgebung kopiert werden. Bei der Verkettung wird automatisch das Lademodul aus der Produktion verwendet.

Änderungen am PGM3 in Produktion müssen unabhängig, ob Verkettung oder Synchronisation, analysiert und in den Projektumgebungen nachgezogen werden.

2.1.3 Gegenüberstellung Verkettung und Synchronisation

Bei den Softwarekomponenten muss unterschieden werden, zwischen Komponenten deren Versionen miteinander verkettet werden können und welchen die synchronisiert werden müssen.

*Loadmodulebibliotheken*¹⁷ können miteinander verkettet werden. Diese Bibliotheken werden der Reihe nach durchsucht, bis das gewünschte Loadmodule gefunden wird. Bei den Umgebungen sind die *Loadmodulebibliotheken*¹⁷ mit den aktuelleren Versionen voranzustellen. Vorteil dieses Verfahrens: nur geänderte Loadmodule sind in den Umgebungen vorhanden (mit Ausnahme von Produktion)

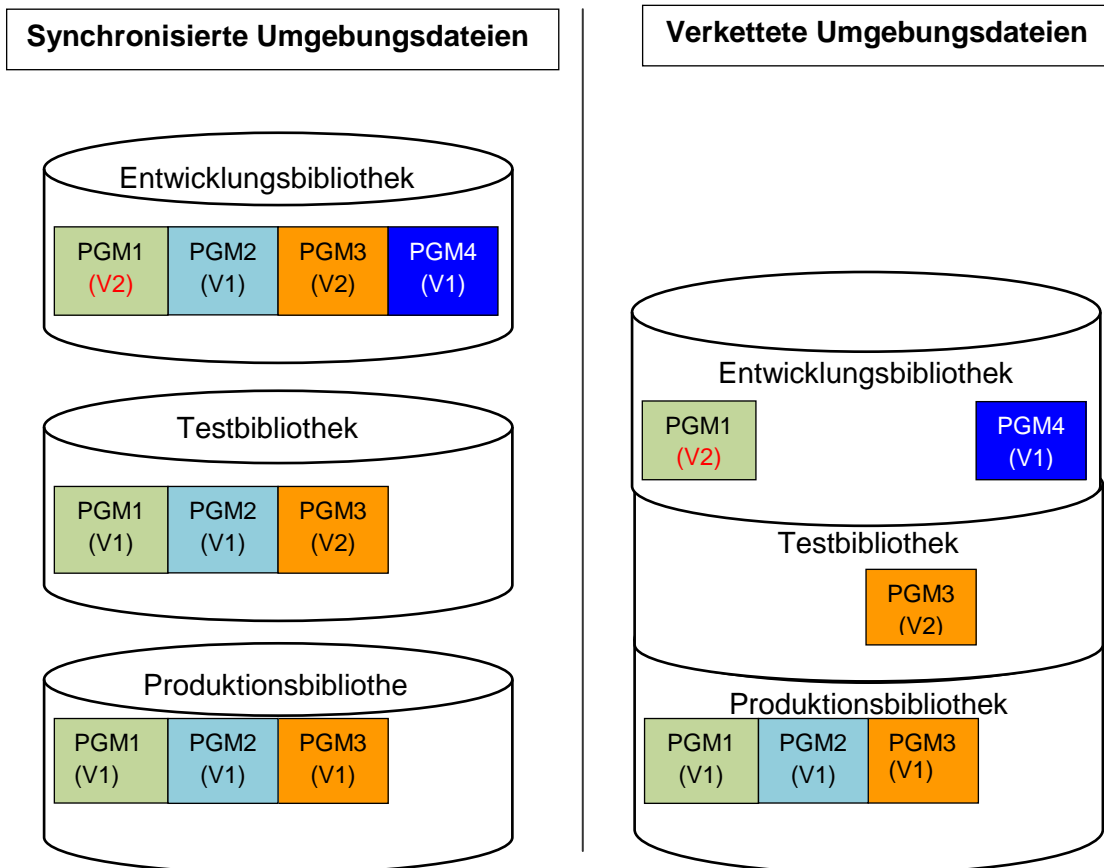


Abbildung 13: Techniken 3 - Vergleich Synchronisation und Verkettung

▪ **Vorteile Verkettung**

- Zur Speicherung der Entitäten¹⁹ wird weniger Speicherplatz benötigt.
- Einfach erweiterbare Umgebungsarchitektur.
- Einfacheres Löschen von Änderungen (keine Rücksynchronisation erforderlich).
- Keine technischen Synchronisationsfehler.

▪ **Vorteile Synchronisation**

- Aufruf ist geringfügig schneller.
- Umgebungen sind voneinander unabhängig.
- Einfachere Erstellung von Testaufträgen (Der Entwickler muss die Verkettung nicht kennen).

■ 2.2 Synchronisation von Umgebungen

Zur Durchführung eines Migrationstests in der Quality Umgebung (dieser kann erst dann durchgeführt werden, wenn feststeht was migriert werden soll) muss sichergestellt werden, dass die Quality Umgebung mit der Produktionsumgebung *synchronisiert*⁸ ist. Das heißt: Alle Programme, die nicht nach Produktion migriert werden sollen, müssen mit den Programmständen in Produktion übereinstimmen (abgesehen von umgebungsspezifischen Anpassungen). Damit wird sichergestellt, dass die produktive Umgebung nach der Migration mit der Quality Umgebung synchron ist.

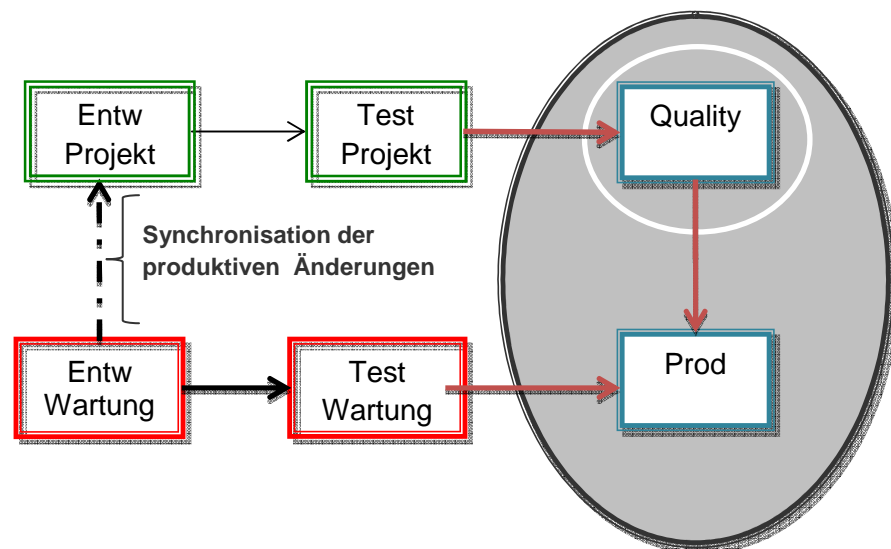


Abbildung 14: Techniken 4 - Synchronisation von Umgebungen

Eine Synchronisation der Quality Umgebung kann erreicht werden, indem der Produktionsstand vor Migration der Projektänderungen in die Quality Umgebung gebracht wird. Dies kann durch Leeren der Quality Bibliotheken durchgeführt werden, im Fall von synchronisierten Bibliotheken sind diese zusätzlich mit dem Produktionsstand zu füllen. Während des Tests in der Quality-Umgebung sind alle Wartungsänderungen, die über den Wartungspfad nach Produktion gebracht werden, in der Quality Umgebung nachzuziehen.

Zusätzlich zum Risiko von Synchronisationsfehlern, die beim Nachziehen von Wartungsaktivitäten auftreten können, beeinträchtigen Wartungseinsätze das Testergebnis des Tests in der Qualitätsumgebung. Aus den genannten Gründen sollen Wartungseinsätze nach Produktion im Zeitraum des Quality Tests nur durchgeführt werden, wenn diese unbedingt notwendig sind.

■ 2.3 Erstellung von ausführbaren Programmen

Programme bestehen in der Regel aus mehreren Teilen: den Quellteilen wie *Sourcecode*⁵ und *Copystrecken*⁶, sowie den ausführbaren Programmteilen wie *Lademodul*¹¹ und *DB2 Package*⁹. Eine wichtige Aufgabe des Konfiguration Managements ist, diese Teile zusammen zu halten. Soll z.B. ein Fehler in einem Programm korrigiert werden, benötigt der Programmierer den zum Lademodul passenden *Sourcecode*⁵.

Im nächsten Unterkapitel wird zunächst beschrieben, wie aus einer *Source*⁵ (Quellprogramm) ein *Lademodul*¹¹ erstellt wird. Leser, denen die Begriffe *Sourcecode*⁵, *Lademodul*¹¹, *Copystrecke*⁶, *DBRM*⁸, *DB2 Package*⁹, *Statischer*¹⁵ und *dynamischer Unterprogrammaufruf*¹⁴ keine Fremdwörter sind, können dieses Kapitel überspringen und mit dem Kapitel Migration von Programmen fortfahren.

2.3.1 Compile eines Programms

Programme werden vom Programmierer im *Quelltext*⁵ erstellt. Im Folgenden als *Sourcecode*⁵ bezeichnet. Der *Sourcecode*⁵ kann in der Regel nicht direkt von einem Computer ausgeführt werden (Ausnahmen sind interpretierbare Quellen z.B. Rexx oder CLIST Prozeduren, diese müssen nicht kompiliert werden).

Damit das Programm ausgeführt werden kann, ist zunächst ein *Umwandlungsprozess*⁴ erforderlich. Dieser Prozess besteht aus dem *Compileschritt*⁴ und weiteren von der Programmart abhängigen Schritten (Beispielsweise benötigt ein Programm, das die Datenbank DB2 verwendet, einen weiteren Schritt zur Übersetzung der SQL Befehle⁷).

2.3.1.1 Copystrecken

Die meisten Programmiersprachen bieten die Möglichkeit, Programmteile in *Copystrecken*⁶ auszulagern. Damit wird die Möglichkeit geschaffen, diese Programmteile in mehreren Programmen zu verwenden und zentral zu warten.

Nach Änderung von *Copystrecken*⁶ ist sicherzustellen, dass alle Programme, die diese Copystrecke verwenden, noch umgewandelt werden können. Die Funktionalität der Programme, die diese Copystrecke verwenden, aber die Änderung nicht benötigen (indirekt betroffene Programme), darf sich nicht ändern.

2.3.1.2 Prinzipieller Ablauf eines Compile

In diesem Abschnitt wird am schematischen Ablaufdiagramm eines *Compile- bzw. Build Prozess* die Problematiken für die Migration von Programmen gezeigt.

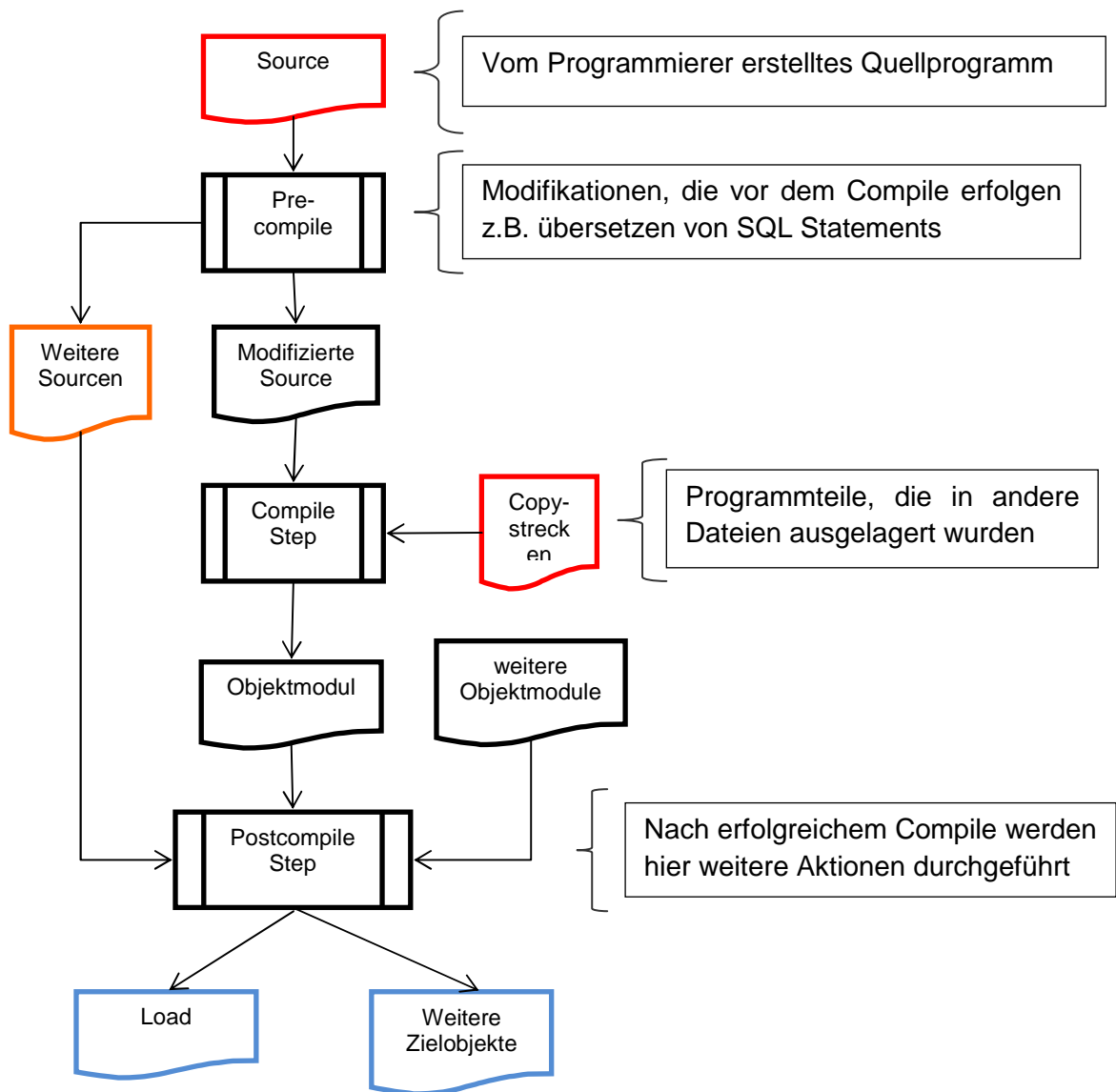


Abbildung 15: Techniken 5 - Prinzipieller Ablauf eines Compileprozess

Für das Konfiguration Management ergibt sich folgende Problematik: Vom Computer werden zur Ausführung nur die *Zielobjekte*¹⁰ (blau dargestellt) benötigt, die zugehörigen *Quellobjekte*⁵ (rot dargestellt) werden erst wieder benötigt, wenn Änderungen am Programm erforderlich sind. Aufgabe des Konfiguration Managements ist, sicherzustellen, dass die passenden *Quellobjekte*⁵ bei Bedarf zur Verfügung stehen.

2.3.1.3 Statische und Dynamische Unterprogrammaufrufe

Ein *Lademodul*¹¹ kann als eine Sammlung von *Zielprogrammen*¹⁰ verstanden werden. Wird das Hauptprogramm in einem Lademodul gestartet, werden alle im Lademodul enthaltenen Programme (dies sind statische Unterprogramme) in den *Adressraum*¹⁸

geladen und das Hauptprogramm gestartet. Nur Programme, die noch nicht geladen wurden, werden nach *einem dynamischen Unterprogrammaufruf*¹⁴ (Aufruf eines Programms, das sich nicht im selben Lademodul befindet), aus einem anderen *Lademodul*¹¹ nachgeladen und ausgeführt.

Vorteile eines *statischen Unterprogrammaufrufs*¹⁵: Programme müssen nicht nachgeladen werden und die Ausführung ist geringfügig schneller. Für das Konfiguration Management bedeutet dies aber, wird ein Programm geändert, müssen alle Lademodule, die dieses Programm enthalten, neu gelinkt werden. Dies führt regelmäßig zu Fehlern, weil oftmals zwar alle Programme neu kompiliert werden, aber die Reihenfolge nicht beachtet wird.

Reihenfolge des Bild Prozess, wenn statische Aufrufe im Programm benötigt werden.

- Objekte, die statisch aufgerufen werden.
- Objekte, die Programme statisch aufrufen.

Prinzipieller Ablauf eines statischen / dynamischen Unterprogrammaufrufs

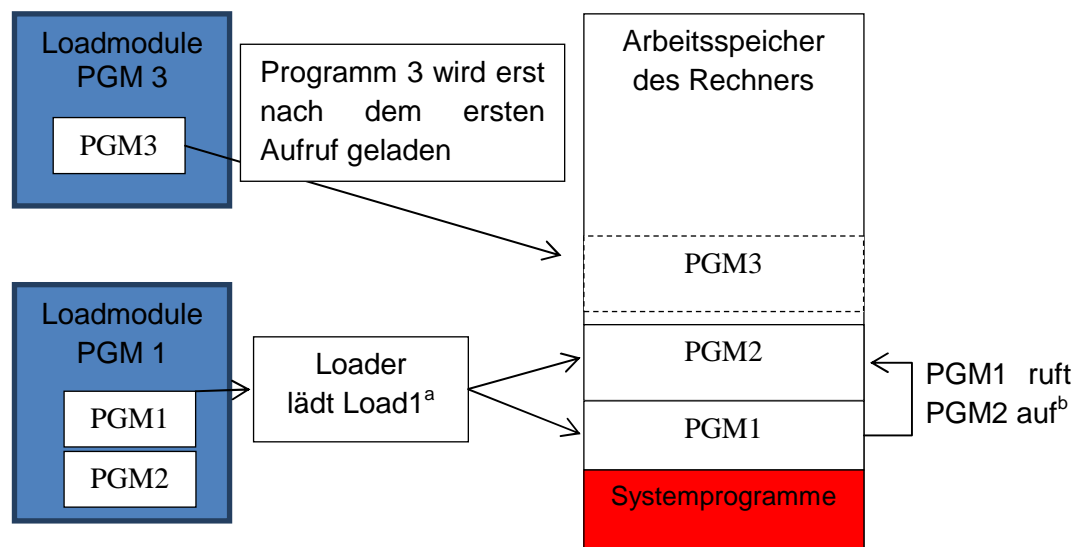


Abbildung 16: Techniken 6 - dynamischer und statischer Unterprogrammaufruf

- a) Beim Start des Hauptprogramms PGM1 werden alle Programme aus dem Lademodul PGM1 geladen (im Beispiel PGM1 und PGM2).
- b) Wird PGM2 als Unterprogramm aufgerufen, wird das PGM2 aus dem Lademodul PGM1 verwendet, selbst wenn es ein Lademodul mit dem Namen PGM2 gibt.
- c) Wird PGM3 (dynamisch) als Unterprogramm aufgerufen, wird das Programm PGM3 aus dem Lademodul PGM3 in den Speicher des Rechners geladen und ausgeführt.

■ 2.4 Migration von Programmänderungen

Grundsätzlich gibt es zwei Möglichkeiten Programme zu migrieren.

1. Übernahme der *Quellobjekte*⁵ in die Zielumgebung mit anschließendem *Compile*⁴ in der Zielumgebung.

Vorteile:

- Hierdurch wird sichergestellt, dass die *Sourceobjekte*⁵ zu den Zielobjekten passen.
- *Compileprobleme*⁴ werden erkannt.
- *Compileoptionen*²⁰ können für die neue Umgebung geändert werden (z.B. Compile mit Test bzw. Optimierungsoption).

Nachteile:

- Ein zusätzlicher Compile⁴ kostet Zeit und Ressourcen.
- Ein bereits getestetes und abgenommenes Zielobjekt¹⁰ ändert sich durch den Compile⁴.

2. Übernahme der Zielobjekte¹⁰ in die Zielumgebung ohne separaten Compile⁴.

Vorteile:

- Kein Zusätzlicher Compile erforderlich
- Ein bereits getestetes und abgenommenes *Zielobjekt*¹⁰ ändert sich nicht

Nachteile:

- *Compileoptionen*²⁰ können nicht geändert werden.
- Für die Sicherstellung, dass die *Sourceobjekte*⁵ zu den Zielobjekten passen, sind weitere Maßnahmen erforderlich
- Mögliche *Compileprobleme*⁴ in der neuen Umgebung werden nicht erkannt.

In vielen Unternehmen werden bei der Programmübernahme in die Quality Umgebung die Programme neu *compiliert*⁴ (mit den *Compileoptionen*²⁰ für Produktion). Bei der Produktionsübernahme werden dann die Zielobjekte direkt in Produktionsumgebung übernommen.

Dadurch wird ermöglicht:

- in der Systemtestumgebung eigene *Compileoptionen*²⁰ zu verwenden (in der Quality Umgebung werden diese dann für Produktion optimiert),
- die Produktionsübernahme kurzfristig durchzuführen (Übernahme der Zielobjekte ist wesentlich schneller als die Übernahme der Quellobjekte mit anschließendem Compile),
- Überprüfung der Compilefähigkeit in der Quality Umgebung

2.4.1 Probleme bei Änderungen von Copybooks

Nach Änderung von *Copystrecken*⁶ ist sicherzustellen, dass alle Programme, die diese Copystrecke verwenden, noch *umgewandelt*⁴ werden können und sich die Funktionalität der nur indirekt betroffenen Programme nicht ändert. Mit indirekt betroffenen Programmen sind Programme gemeint, die die Copystrecke zwar zur Umwandlung benötigen, die funktionale Erweiterung der Copystrecke aber nicht (z.B. in eine Datei werden weitere Informationen in bereits vorgesehene unbenutzten Bereiche gestellt, oder es wird die Rechengenauigkeit für bestimmte Anwendungen erhöht).

Ob alle von der Copybookänderung betroffenen Programme in jeder Umgebung neu eingesetzt werden, muss im Projekt entschieden werden. Gründe, die Programme nicht einzusetzen, können sein: Größe des Migrationsfensters, Ressourcenverbrauch eines Massencompile, Risiko eines Neueinsatzes.

Mit folgenden Möglichkeiten wird sichergestellt, dass alle Programme, die diese Copystrecke verwenden, noch *umgewandelt*⁴ werden können und sich die Funktionalität der nur indirekt betroffenen Programme nicht ändert:

5. *Recompile*⁴ der betroffenen Programme mit anschließendem Test
6. Copybookänderung ist abwärtskompatibel, nur die Programme, die diese Copybookänderung benötigen werden umgewandelt.

Wünschenswert ist natürlich, dass alle Programme, die die geänderte Copystrecke enthalten, neu kompiliert und in allen Umgebungen neu eingesetzt werden. Dies erfordert allerdings eine hohe Disziplin bei den Entwicklern, eine zusätzliche Kontrollinstanz oder eine technische Lösung die dies sicherstellt.

■ 2.5 Arbeitsweise eines Changemanagementsystems

Durch ein Changemanagementsystem soll sichergestellt werden, dass *Quell*⁵ und *Zielobjekte*¹⁰ zueinander passen und die Umgebungen *technisch synchron*³ sind. Dies ist besonders für die Test-, Abnahme- und Produktionsumgebung wichtig. Des Weiteren werden Metadaten in einem Repository abgelegt (beispielsweise seit wann ist ein Objekt von wem in Bearbeitung) Solche Informationen können durch eine Auswertung ermittelt werden oder stehen internen Prozessen zur Verfügung (z.B. erfolgt beim Versuch, parallele Änderungen an einem Objekt durchzuführen, mindestens ein Hinweis).

Im optimalen Fall hat nur das Change Management System Schreibzugriffe auf alle *Zielobjekte*¹⁰ der Anwendung, alle Prozesse sind im Change Management System definiert und die passenden *Sourceobjekte*⁵ sind vollständig im Change Management vorhanden.

In der Regel ist dies jedoch nicht für alle Objekte der Fall, da:

- Altbestände aus Zeiten vor Einführung des Change Management System weiter existieren,
- es nicht wirtschaftlich ist, jeden Spezialfall im Change Management System abzubilden,
- sich häufig ändernde Zielobjekte, werden in der Regel nicht durch das Change Management System verwaltet,
- nicht alle Abteilungen arbeiten mit dem Change Management System,
- nicht alle eingesetzten Tools arbeiten mit dem Change Management System zusammen.

Auch bei Einsatz eines Change Management Systems sind zusätzliche organisatorische Maßnahmen erforderlich, um die Umgebungen synchron zu halten (z.B. kann das Changemanagementsystem zwar feststellen, dass paarallele Änderungen an einem Objekt stattgefunden haben. Ob sich aber die Entwickler abgesprochen haben und die parallelen Änderungen synchronisiert wurden, kann technisch nicht entschieden werden).

Prinzipieller Aufbau eines Changemanagementsystems:

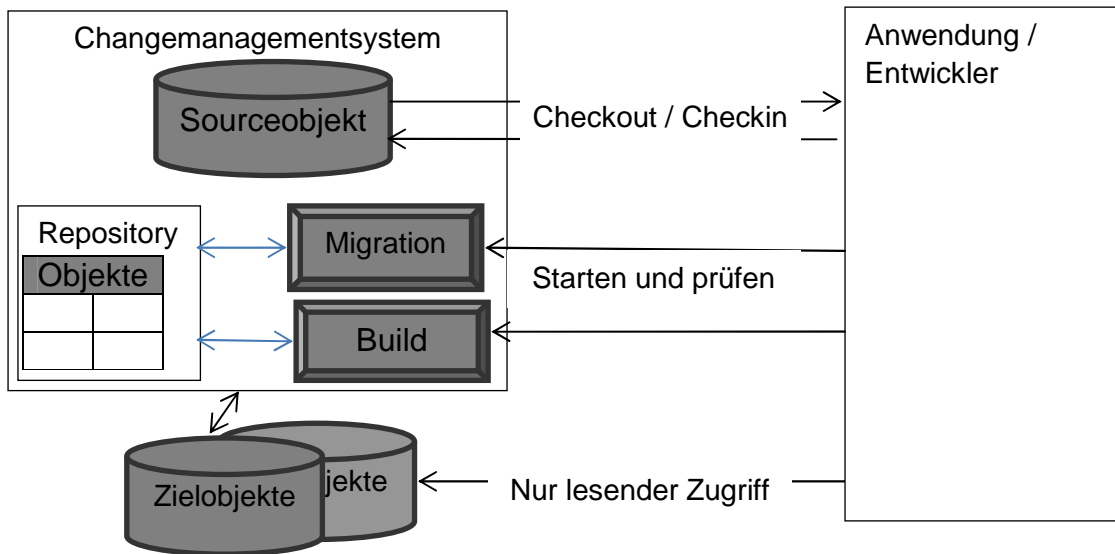


Abbildung 17: Techniken 7 - Arbeitsweise eines Change Management Systems

Repository

Das Repository ist ein Verzeichnis zur Verwaltung der Objekte im Changemanagementtool. Hier werden alle von den Prozessen benötigten und generierten Metadaten abgelegt.

Beispiele für Metadaten, die im Repository abgelegt werden:

- **Programmtyp**
Mithilfe dieser Information steuert der Build Prozess⁴ welcher Compilestep (z.B. Cobol, C, FORTRAN) verwendet werden soll und welche Pre- bzw. Postcompilesteps benötigt werden (z.B. DB2, CICS).
- **Checkoutinformationen**
Welcher Programmierer hat seit wann ein Programm in Bearbeitung.
- **Versionenverwaltung**
Welche Source ist seit wann in welcher Umgebung aktiv.
- **Parent-Child Beziehungen zwischen den Entitäten**
Z.B. welche Copybooks, Files, Datenbanktabellen werden von welchen Programmen verwendet. Welcher Job, welche Transaktion ruft welches Programm auf.

Sourceobjekte werden vom Change Management System verwaltet. Der Entwickler kann sich diese aus dem Change Management System „ausleihen“ (checkout) und die geänderte Source wieder zurückstellen. Das Change Managementsystem prüft hierbei, ob das Sourceobjekt schon in Bearbeitung ist. Im Repository werden die Userid des

Entwicklers, der Timestamp sowie die Umgebung, aus der die Source entliehen bzw. zurückgestellt wurde, festgehalten.

Zielobjekte werden vom Change Management System durch den **Build Prozess** erstellt. Hierzu werden Informationen aus dem Repository verwendet (beispielsweise welcher Compilesteps (z.B. Cobol, C, Fortran) verwendet werden soll und welche Pre- bzw. Postcompilesteps benötigt werden. Die im Build Prozess gesammelten Informationen (z.B. welche Copybooks und welche Unterprogramme benötigt werden) werden wieder in das Repository zurückgeschrieben. Im idealen Fall hat nur das Changemanagementsystem Schreibzugriff auf die Zielobjekte. Der ebenfalls im Change Management System integrierte **Migrationsprozess** sorgt für die Weitergabe der Source und Zielobjekte in die nächste Umgebung.

Buildprozess

Durch diesen Prozess wird aus einem Source Objekt ein Zielobjekt. Wie der Prozess abläuft ist vom Objekttype und den Definitionen im Changemanagementsystem abhängig. Wird eine Versionenverwaltung für Objekte, deren Source und Zielobjekt inhaltlich gleich sind, gewünscht, können diese Objekte als Source in das Changemanagementsystem gestellt werden. Mithilfe des Bildprozess (in diesem Fall ein einfaches Kopieren) wird das Sourceobjekt in eine Zielobjektbibliothek gestellt. Permanente Änderungen am Zielobjekt sollten dann nur noch über den Buildprozess des Changemanagementsystems erfolgen

Migrationsprozess

Der Migrationsprozess steuert die Übernahme von Software in die nächste Umgebung. In diesem Prozess können umfangreiche Prüfungen eingebaut werden (beispielsweise kann geprüft werden, ob bei einer Copybookänderung alle Programme mit übernommen werden oder ob die Version, von der aus Entwickelt wurde, noch mit der in der Zielumgebung übereinstimmt).

Auswertungen aus dem Repository

Für die Entwicklung bzw. für Kontrollen im Migrationsprozess können Auswertungen aus dem Repository erstellt werden. Wird z.B. eine Copystrecke geändert, können die Programme, die diese Copystrecke verwenden, ermittelt werden.

■ 3 Organisatorische Maßnahmen

■ 3.1 Synchronisation von Umgebungen

Unter *Synchronisation*³ wird die Abstimmung der Programmstände in den einzelnen Umgebungen verstanden. Für die vorgestellte Umgebungsarchitektur bedeutet dies, dass Änderungen, die über den Wartungspfad nach Produktion gebracht werden, in den Projektumgebungen nachgezogen werden müssen. Ziel der Synchronisation ist es, zu vermeiden, dass bereits behobene Fehler durch den Projekteinsatz wieder aktiviert werden.

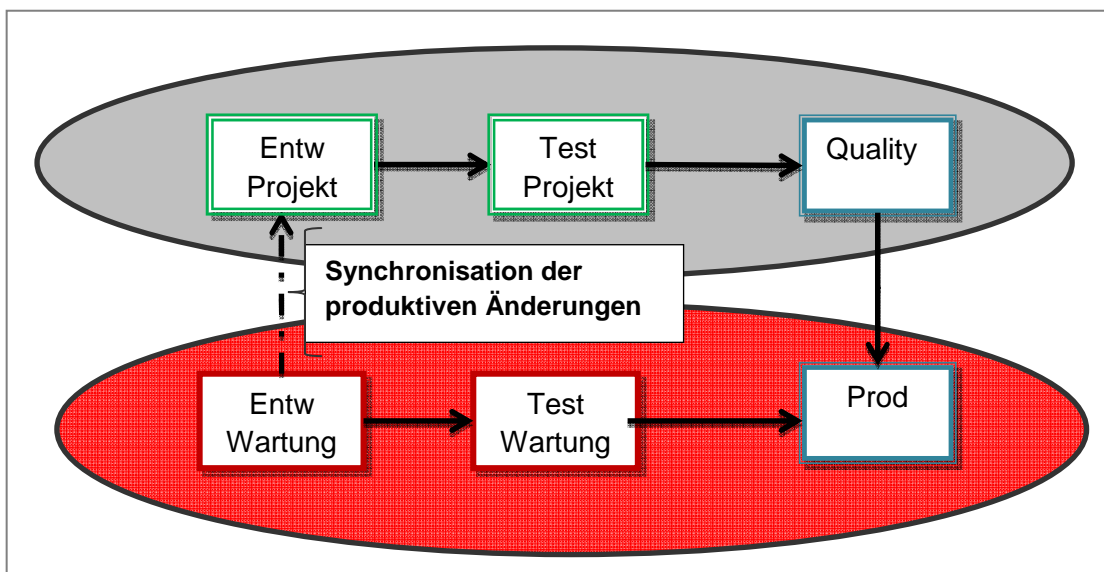


Abbildung 18: Organisatorische Maßnahmen 1 – Synchronisation Entwicklung und Wartung

Organisatorisch ist es sinnvoll, dass die Entwickler, die für den Wartungseinsatz verantwortlich sind, auch die Synchronisation in die Projektumgebungen sicherstellt. Zusätzlich sollten die Testfälle aus der Testumgebung für die Wartung (sofern vorhanden) in die Projekt-Testumgebung übernommen werden und die erfolgreiche Synchronisation dokumentiert werden.

Eine Prüfung auf nicht durchgeführte Synchronisationen kann durchgeführt werden, indem die Basisversion (Version, auf die bei der Entwicklung aufgesetzt wurde) mit der aktuellen Version in Produktion verglichen wird. Gibt es hier Abweichungen, sollten diese Abweichungen geprüft werden. Nach diesem Verfahren arbeitet auch das Changemanagementtool Endeavor. Eine weitere Möglichkeit ist, die Sourcen zu vergleichen. Beim Vergleich der Sourcen dürfen nur Änderungen erscheinen, die aus dem Projekt stammen.

■ 3.2 Bündelung und Splitten von Arbeitspaketen zu Einsatzstufen

In der Planungs- und Definitionsphase wird ein Projekt in Arbeitspakete unterteilt. Diese Arbeitspakete müssen gegebenenfalls in Realisierungsstufen gesplittet werden und mit anderen Arbeitspaketen zu Einsatzstufen gebündelt werden.

Die Migration findet dann in den verschiedenen Umgebungen statt. Die Migration nach Produktion sollte mindestens einmal produktionsnah getestet werden (z.B. Migration in die Quality Umgebung). Bei diesem Test können auch die Migrationszeiten abgeschätzt werden. Ziel ist, dass für den Produktionseinsatz alle wichtigen Informationen bezüglich des Einsatzes zur Verfügung stehen.

Stehen für die Migration keine professionellen Tools zur Verfügung, die alle Eventualitäten eines Einsatzes abdecken (in der Regel ist dies nicht der Fall), ist es sinnvoll die Informationen für die Einsätze in beispielsweise Excel Arbeitsmappen zu bündeln.

In einer Excel Arbeitsmappe können verschiedene Arbeitsblätter zusammengefasst werden, diese können schnell an die Bedürfnisse des Projekts oder Einsatz angepasst werden. An einem Einsatz sind in der Regel mehrere Teams beteiligt (z.B. Datenbankänderungen übernehmen die Datenbankadministratoren, JCL Änderungen die Arbeitsvorbereiter, Programmeinsätze das Operating). Für einen Einsatz wird neben einem Arbeitsblatt, aus dem übergreifende Details und der Status des Einsatzes hervorgeht, für jedes am Einsatz beteiligte Team ein eigenes Arbeitsblatt erstellt.

Möglicher Aufbau einer solchen Arbeitsmappe

Arbeitsblatt	Inhalt
Index	In diesem Arbeitsblatt werden allgemeine Informationen zum Einsatzsheet gespeichert wie <ul style="list-style-type: none"> • Versionsliste, • Beschreibung der einzelnen Arbeitsblätter
Einsatzstufen	In diesem Arbeitsblatt stehen: <ul style="list-style-type: none"> • Die Einsatztermine, • Der Änderungsumfang (wie Anzahl Programme, Datenbankänderungen), • Die Zielumgebung, • Die Ansprechpartner
Einsatzplanung	Detailplanung zu den Einsatzstufen, Liste der notwendigen Aktionen inklusive Abhängigkeiten, Ansprechpartner und Plandaten
Programme	Liste der einzusetzenden Quellobjekte, wie Programme und Copystrecken mit Angaben des Programmtype, Status der

	Programmänderung, aktuellen Version, Basisversion (Version von der aus Entwickelt wurde).
Datenbank	Aufführung der Datenbankänderungen wie Tabellen- und Indexerweiterungen
JCL Änderungen	Änderung an der Job Control wie <ul style="list-style-type: none"> • JCL – Änderungen, • GDG – Definitionsänderung, • Datenkartenänderungen, • Planänderung
Dokumente	Verweis auf die Dokumente, die Informationen zum Einsatz enthalten

:

Beispiel:

Den Aufbau der neuen Umgebungen möchte die Firma Maxstein im Excelsheet Umgebungserweiterung planen und dokumentieren.

Hierzu wurde die Excel Arbeitsmappe „Umgebungserweiterung“ angelegt. Diese Arbeitsmappe enthält die folgenden Arbeitsblätter:

1. Sheet Index

Das Sheet Index enthält neben allgemeinen Angaben zur Arbeitsmappe eine Tabelle der Versionsführung der Arbeitsmappe. Nach jeder erfolgreichen Migration wird (durch Kopieren der Excel Arbeitsmappe) eine neue Version der Arbeitsmappe erstellt und die alte Version archiviert.

	A	B	C	D	E	F
1	<i>Version</i>					
2		Version		Datum		
3	0.1			01.05.2011	Erstellung des Sheets für die neue Quality Umgebung	
4	0.2			01.06.2011	Schließen der Version nach Durchführung der Änderung in der Quality Umgebung	
5					Neue Version Hinzufügen der Planung für die Abnahmeumgebung	
6	<i>Im Sheet enthalten Arbeitsblätter</i>					
7		Index			Dieses Arbeitsblatt, allgemeine Informationen zum Einsatzsheet	
8	Einsatzstufen				In diesem Arbeitsblatt stehen: <ul style="list-style-type: none"> • Die Einsatztermine • Der Änderungsumfang (wie Anzahl Programme, Datenbankänderungen) • Die Zielumgebung • Die Ansprechpartner 	
9	Einsatzplanung				Detailplanung zu den Einsatzstufen Liste der notwendigen Aktionen inklusive Abhängigkeiten, Ansprechpartner, Plandaten	
10	Programme				Liste der einzusetzenden Programme: incl. Quell und Zielversion, Programmtype und Staus der Programmänderung	
11	Datenbank				Aufführung der Datenbankänderungen	
12	JCL Änderung				Änderung an der Job Control wie <ul style="list-style-type: none"> • JCL – Änderungen • GDG – Definitionsänderung • Datenkartenänderungen • Planänderung 	
13	Dokumente				Verweis auf die Dokumente, die Informationen zum Einsatz enthalten	
14						

2. Sheet Einsatzstufen

In dieses Arbeitsblatt gehören: eine Kurzbeschreibung des Einsatzes, sowie eine Liste der geplanten / durchgeführten Einsatzstufen mit Ansprechpartner, Kurzbeschreibung und Status.

Beschreibung des Einsatzes															
Mit diesem Einsatz soll für das Teilprojekt Infrastruktur unser Umgebungssetup um einen Wartungspfad und eine Quality Umgebung erweitert werden															
Einsatzstufen															
Stufe	Schritt	Datum			Umgebun	Anzahl Änderungen				Ansprechpartner		Kurzbeschreibung	Status	Kommentar	
		geplan	Revise	ist		PGM	DB	Jcl	Doc	Architekt	Durchführende				
1	1	26.08.11										2			Konzepterstellung
2	2	02.12.11			Quality					Hubert Müller					Neuaufbau der Quality Umgebung
3	2	30.09.11		30.09.11	Quality					Hubert Müller	Hubert Müller				Erstellung Security und Dataset
4	2	30.09.11	07.10.11	14.10.11	Quality					Hubert Müller					Konzept für die Quality Umgebung
5	2	30.09.11			Quality										Spacepools für Dateien und Datenbanktabellen
6	2	07.10.11			Quality						Space Management				Einrichtung eines neuen Space Pools für die Qualityumgebung
7	2	14.10.11			Quality						Security Office				Vergabe von Rechten an die Betreuer der Qualityumgebung
8	2				Quality		n				Datenbankbetreuer				Anlage der Datenbanken für die Quality Umgebung analog
9	2	6			Quality	2	1	n	1		Entwickler				Umgebungsparameter einsetzen
10	3				Wartung										Neuaufbau des Wartungspfad

Abbildung 20:

3. Sheet Einsatzplanung

In diesem Arbeitsblatt werden Details zu jeder Einsatzstufe, wie Abhängigkeiten, Zeitspannen, Verantwortliche festgehalten.

Stufe	Schritt	Teilschritt	Referenz		Datum & Uhrzeit		Verantwortlich	Durchführung	Beschreibung der Aktion	Kommentar
			ID	gänger	geplant	ist				
2 Neuaufbau der Quality Umgebung										
2	2		2-2-				Projekt		Erstellung eines Mengengerüst für die Quality Umgebung	
2	2	1	2-2-1						Mengengerüst für Flat files	
2	2	2	2-2-2				DBA		Mengengerüst für DB2 Tabellen	
2	2		2-2-1,	2-2-2			Projekt	Controlling	Anforderung des benötigten Space	

4. Sheet Programme

In diesem Arbeitsblatt ist die Liste der einzusetzenden Programme incl. Copystrecken zu finden. Neben den Programmnamen wird auch die Quell- oder Basisversion festgehalten. Synchronisationen während der Entwicklung sind ebenfalls in diesem Arbeitsblatt zu dokumentieren.

Stufe	Schritt	Teilschritt	Entität		Version		Änderungsbeschreibung	Status
			Name	Type	Ziel	Quelle		
2	2	6	BatchParm	BHP	1.4	1.3	Hinzufügen Umgebungsparameter für Quality	erl.
2	2	6	OnlineParm	OHP	1.4	1.2	Hinzufügen Umgebungsparameter für Quality	in Arbeit
Produktionfix (V1.3) vom 03.08 wurde nachgezogen								

■ 4 Projektabschlussbesprechung der Firma Maxstein

In der Projektabschlussbesprechung lobt der Geschäftsführer Dr. Mayer die gute Zusammenarbeit der Kollegen beider Firmen. Trotz anfänglicher Bedenken und Widerstände gegen die neuen Verfahren wurden die IT Systeme erfolgreich im Zeitplan zusammengeführt. Gründe hierfür waren unter anderem die rechtzeitige Erweiterung der IT Umgebungsarchitektur und eine gute Dokumentation der Einsätze.

Als besonders positiv wurde der Vorschlag empfunden, die im Projekt eingeführten Verfahren weiter beizubehalten. Schließlich ist die neue Firma um einiges größer und leistungsfähiger geworden.

Auf dem Projektabschlussmeeting fasst Herr Schultz die Lessons Learned zusammen:

- Jeder gesparte Euro, der am Anfang eines Projekts nicht in erforderliche Maßnahmen investiert wird, kostet im Laufe des Projekts ein Vielfaches.
- Sind Aufgaben mit großen Risiken verbunden (wie die Änderung der Umgebungsarchitektur) sollten diese so früh wie möglich begonnen werden.
- Der rechtzeitige Einsatz von Tools unterstützt das Konfigurationsmanagement, kann dieses jedoch nicht ersetzen.
- Dokumentation ist zwar unbeliebt und teuer, aber für größere Änderungen unerlässlich.

■ 5.0 Anhang

■ 5.1 Beschreibung von Fachbegriffen

Wichtige Fachbegriffe werden in der nachfolgenden Tabelle erklärt. Im Dokument wird durch eine hochgestellt Zahl nach dem Fachbegriff auf die Beschreibung in dieser Tabelle hingewiesen z.B. z/OS¹.

Nr.	Begriff	Beschreibung
1	z/OS	z/OS ist ein Betriebssystem für IBM Großrechner
2	Verkettung bzw. Konkatenation	<p>Bei dieser Technik werden mehrere Bibliotheken¹⁷ zu einer gemeinsamen Bibliothek¹⁷ verkettet. Diese Technik wird unter z/OS u.a. dazu verwendet, vom Betriebssystem zu startende Programme, auf mehrere Bibliotheken zu verteilen.</p> <p><i>Beispiel: Festlegung in welchen Bibliotheken die Programme (Lademodule) gesucht werden:</i></p> <pre>// STEPLIB DD DSN=Bibliothek1 <= 1. Position // DD DSN=Bibliothek2 <= 2. Position // DD DSN=Bibliothek3 <= 3. Position</pre> <p>STEPLIB := Name der aus den Bibliotheken 1-3 gebildeten Bibliothek.</p> <p><i>Kommen Programme mit gleichem Namen in mehreren Bibliotheken vor, wird das Programm aus der Bibliothek mit der kleinsten Position genommen.</i></p>
3	Synchronisation	<p>Unter dem Begriff Synchronisation wird die Abstimmung der Programmstände in den einzelnen Umgebungen verstanden. Wichtig ist hierbei, dass keine Anpassungen (z.B. Fehlerbehebungen) verloren gehen.</p> <ul style="list-style-type: none">• Technische Synchron Oft ist es möglich, Anpassungen automatisch zu synchronisieren. Diese Form der Synchronisation wird als technische Synchronisation bezeichnet.• Logische Synchronisation Bei einer parallelen Änderung kann es vorkommen, dass es bei der technischen Synchronisation zu Konflikten kommt. In diesem Fall muss ein Entwickler die Synchronisation durchführen. Diese Form der Synchronisation wird als logische Synchronisation bezeichnet.
4	Compile- / Build Prozess	Mit dem Build Prozess wird aus einem Quellprogramm ⁵ ein von Computer ausführbares Lademodul ¹¹ erstellt. Der Compiler ist in diesem Prozess der wichtigste Teilschritt. Umgangssprachlich wird der Build Prozess als Compile bezeichnet.

Nr.	Begriff	Beschreibung
5	Source, Sourcecode, Quellprogramm	Mit dem Begriff Quellprogramm, wird das vom Programmierer erstellte Programm bezeichnet. Dieses Programm muss in der Regel kompiliert ⁴ werden, damit es von einem Computer ausgeführt werden kann. In der Regel werden die englischen Begriffe Source oder Sourcecode verwendet.
6	Copystrecke	Teile eines Quellprogramms ⁵ , die in eigene Datei ausgelagert werden. Diese Programmteile werden im Build Process ⁴ dem Quellprogramm hinzugefügt und können in mehreren Programmen verwendet werden. Wird eine Copystrecke geändert, wirkt sich dies erst mit dem nächsten Build Prozess auf das Zielprogramm aus.
7	SQL Abfragen / SQL Statement	Abfragesprache für relationalen Datenbanken. Mit diesen SQL Befehlen werden Daten aus einer oder mehreren Tabellen extrahiert und in einer temporären Ergebnistabelle dem Programm zur Verfügung gestellt. In den Quellprogrammen sind die SQL Befehle eingebettet. Die SQL Befehle müssen im Build ⁴ Prozess in zusätzlichen Teilschritten separat behandelt werden. Sind im Quellprogramm SQL Befehle enthalten, entsteht zusätzlich ein DBRM ⁸ und ein Db2-Package ⁹
8	DBRM	SQL Statements ⁷ , die aus einem Quellprogramm extrahiert wurden. Zur Ausführung der SQL Statements müssen diese noch in ein DB2 System gebunden werden (Bind Prozess). Neben den SQL Befehlen wird der Timestamp (Consistency Token) des Build Prozess im DBRM und Lademodul gespeichert. Mit diesem Timestamp stellt DB2 sicher, dass Lademodul und DB2 Package ⁹ zusammen passen.
9	DB2 Package	Zielprogramm für SQL Statements ⁷ . Diese, aus einem Quellprogramm ⁵ extrahierten SQL Statements ⁷ , werden von der relationalen Datenbank DB2 ausgeführt.
10	Zielprogramm	Kompiliertes ⁴ Quellprogramm ⁵ . Das Zielprogramm wird aus einem Lademodul ¹¹ von einem Computer ausgeführt.
11	Lademodule	Sammlung von Zielprogrammen ¹⁰ , von denen ein Programm als Einstiegsprogramm gekennzeichnet ist. Das Einstiegsprogramm wird nach Laden der Zielprogramme aus dem Lademodul gestartet.
12	Hauptprogramm	Zielprogramm ¹⁰ , das vom Betriebssystem (z.B. z/OS ¹) gestartet wird
13	Unterprogramm	Programm, das von anderen Programmen (Haupt ¹² - oder Unterprogrammen) gestartet wird
14	Dynamischer Unterprogramm Aufruf	Aufruf eines Unterprogramms ¹³ , das sich nicht im Lademodul ¹¹ befindet
15	Statischer Unterprogramm Aufruf	Aufruf eines Unterprogramms ¹³ , das sich im Lademodul ¹¹ befindet
16	Linker	Programm, das mehrere Zielprogramme ¹⁰ in einem Lademodul ¹¹ zusammenfasst.
17	Library oder Bibliothek	Sammlung von Members (Ein Member ist z.B. ein Lademodule ¹¹ , oder eine Copystrecke ⁶)
18	Adressraum bzw. Address Space	Speicher eines Computers. Programme müssen zunächst in den Speicher geladen werden, bevor sie ausgeführt werden können.

Nr.	Begriff	Beschreibung
19	Entität	Mit dem Begriff Entität wird ein für den Migrationsprozess relevantes physisches Objekt bezeichnet. Die Entitäten werden in Entitätstypen unterteilt, Beispiele für Entitätstypen sind Lademodule ¹¹ , Quellprogramme ⁵ , Copystrecken ⁶ . Gespeichert werden die Entitäten in der Regel als Member ¹⁷ in Bibliotheken ¹⁷ , dabei werden Entitäten eines Entitätstypen in der Regel gemeinsam in einer eigenen Bibliothek gespeichert.
20	Compileoptionen	Dem Compile (Build) Prozess ⁴ können Optionen mitgegeben werden. Diese Optionen wirken sich auf das Zielobjekt aus (z.B. stellt die Option DYNAM sicher, dass alle Unterprogramme dynamisch ¹⁴ aufgerufen werden).

■ Der Autor

Nach seiner Berufsausbildung als Informationselektroniker und dem Studium zum Ingenieurinformatiker arbeitet Herr Wallrabenstein mit dem Betriebssystem z/OS seit 1993 bis heute ohne Unterbrechung. Sein Berufsumfeld umfasst die Anwendungsentwicklung, Produktionsplanung und die Betreuung von z/OS Softwarearchitekturen. Seine Freizeit verbringt Herr Wallrabenstein überwiegend mit seiner Familie.



Email: martin.wallrabenstein@agon-solutions.de

■ A:gon Solutions GmbH

Die A:gon Solutions GmbH, 2004 gegründet, ist ein unabhängiges IT-Dienstleistungsunternehmen mit Firmensitz in Eschborn bei Frankfurt und weiteren Standorten in Hamburg und Berlin. Das branchenübergreifende Dienstleistungsportfolio von A:gon umfasst das „A:gon-proven IT-Consulting“, eine bewährte, herstellerneutrale IT-Beratung; sowie die „A:gon-tailored IT-Solutions“, zu denen maßgeschneiderte, individuelle Softwareentwicklung, passgenaue Softwareintegration und effiziente Business Intelligence Lösungen gehören. A:gon stellt sich mit professionellem Projektmanagement, proaktivem Anforderungsmanagement und änderbaren Softwarearchitekturen auf die individuellen Bedürfnisse seiner Kunden ein. In ausgewählten Branchen wie Banken, Versicherungen, Aviation und Health Care bietet A:gon gemeinsam mit seinen Partnern Lösungen, die auf fundiertem Geschäftsprozess-Know-how beruhen und speziell auf die jeweilige Branche zugeschnitten sind: die „A:gon-tailored Business Solutions“. Die plattformübergreifende technologische Kompetenz bei A:gon reicht von klassischen Mainframe-Architekturen bis hin zu modernen Java/JEE Web- und Portal-Architekturen. Zu den Referenzkunden von A:gon gehören unter anderen die AOK Berlin-Brandenburg, die Commerzbank, die Deutsche Bank, die Deutsche Bank Bauspar, die Deutsche Börse, die Finanz Informatik und die Deutsche Lufthansa.

Copyright:

A:gon Solutions GmbH

Frankfurter Strasse 71-75
D-65760 Eschborn
Telefon : +49 6196 80269 0
Telefax : +49 6196 80269 11
<http://www.agon-solutions.de>

Handelsregister Frankfurt HRB 58185
St.-Nr. 4022826171
Geschäftsführer: Udo Peters