Unabhängige Doppelprogrammierung - das Nonplusultra der Validierung?

Beate Hientzsch
Accovion GmbH
Helfmann-Park 10
65760 Eschborn
beate.hientzsch@accovion.com

Gabi Lückel
Accovion GmbH
Helfmann-Park 10
65760 Eschborn
gabi.lueckel@accovion.com

Jörg Müller
Accovion GmbH
Helfmann-Park 10
65760 Eschborn
joerg.mueller@accovion.com

Nicola Tambascia
Accovion GmbH
Helfmann-Park 10
65760 Eschborn
nicola.tambascia@accovion.com

Zusammenfassung

Unabhängige Doppelprogrammierung hat sich im Rahmen der Analyse von klinischen Daten als der Industriestandard entwickelt und etabliert. Im Alltag heißt das, dass SAS-Programme, die für die Erstellung von Analysedatensätzen, Tabellen, Listings und Graphiken entwickelt werden, durch Nachprogrammierung der generierten SAS-Outputs verifiziert werden. Implementierungsstrategien reichen von Doppelprogrammierung in der gleichen Arbeitsumgebung durch Kollegen bis zu Doppelprogrammierung in einer physikalisch anderen Umgebung, z.B. durch eine andere Firma. Die Qualität der Programmierung und Validierung zu Grunde liegenden Spezifikationsdokumente, sowie der Status der klinischen Daten sind für den Erfolg im Sinn von Zeit, benötigten Ressourcen und im Hinblick auf Qualität ausschlaggebend. Automatisierter Vergleich der Ergebnisse, sofern richtig und sinnvoll angewendet, z.B. durch die Prozedur COMPARE und eine benutzerfreundliche Dokumentation unterstützen Effizienz und Transparenz des Prozesses.
Dieser Beitrag wird Herausforderungen, Vorteile, Grenzen und Risiken der Validierung der Ergebnisse mittels unabhängiger Doppelprogrammierung beleuchten.

Schlüsselwörter: Unabhängige Doppelprogrammierung, Validierung, Spezifikationen, SAS-Programme, Analysedatensätze, COMPARE-Prozedur

1 Einführung

1.1 Regulatorischer Hintergrund

Die Ergebnisse klinischer Studien führen zur Zulassung oder Nichtzulassung von Medikamenten. Insofern ist die Validität der ermittelten Ergebnisse in Bezug auf Wirksamkeit und Sicherheit essentiell für die Pharma- oder Biotechnologieunternehmen, die Medikamente und Medizinprodukte entwickeln und zur Zulassung bringen wollen, aber auch für die Behörden (Food and Drug Administration (FDA), European Medicines

Agency (EMA), etc.), die die Zulassung von Medikamenten und Medizinprodukten verantworten. Letztendlich geht es einzig um das Wohl der Patienten.
Aufgrund der großen Menge unterschiedlicher Daten, die für die Analyse einer Studie in Zusammenhang gebracht und ausgewertet werden müssen, ist die Komplexität der Programme, die hierfür entwickelt werden, nicht zu unterschätzen. Adäquate Validierung der Programme und der Ergebnisse ist ein Muss, um die Forderung der Behörden nach Vollständigkeit, Genauigkeit, Zuverlässigkeit, Fehlerfreiheit und Rückverfolgbarkeit der eingereichten Daten und Analysen erfüllen zu können ([1], [2], [3]). Aber was ist adäquat und gleichzeitig leistbar? Hierzu hat die FDA im Jahr 2003 eine Empfehlung ausgesprochen:
"*... base your validation approach on a justified and documented* ***risk assessment*** *and the potential of the system to affect product quality and safety as well as record integrity*" [4].
Es geht also darum einzuschätzen, wie groß das Risiko ist einen Fehler zu machen und welche Folgen ein möglicher Fehler haben würde. Entsprechend der Risikobewertung kann dann das passende Validierungsmodell ausgewählt werden.

1.2 Überblick Validierungsmodelle

Die Minimalanforderung, d.h. das niedrigste Validierungslevel, ist der Check des SAS Programms, des Log-Files und der dazu gehörigen Outputs gegen die Spezifikationsdokumente und die Prüfung auf inhaltliche Richtigkeit durch den Programmierer selbst. Darauf aufbauende Validierungsmodelle involvieren nach dem Erstcheck durch den Programmierer einen zweiten Programmierer, den sogenannten Validierer, und folgen dem 4-Augenprinzip.
Die nachfolgend genannten Validierungsmodelle sind, teilweise mit geringen firmenspezifischen Modifikationen, die in der Pharmabranche etablierten Methoden für die Validierung sogenannter single-use / one-off Programme, also der Programme, die studienspezifisch / für eine spezielle Fragestellung / Anwendung entwickelt werden [5], [6]:

- Check durch den Programmierer
- Check durch den Programmierer + unabhängiger output review
- Check durch den Programmierer + unabhängiger source code / log / output review
- **Unabhängige Doppelprogrammierung**, das Thema dieser Veröffentlichung

Projektübergreifend einsetzbare Programme, zum Beispiel Standardmakros, werden in der Regel nach IT Standards und gemäß des Software Development Life Cycles (SDLC) sowie mittels Testfällen validiert und sind nicht Inhalt dieser Veröffentlichung.

2 Prozess der unabhängigen Doppelprogrammierung

2.1 Was ist „unabhängige Doppelprogrammierung“?

Unter „Unabhängiger Doppelprogrammierung“ versteht man, dass zwei Programmierer dieselbe Programmieraufgabe unabhängig voneinander mit dem Ziel erledigen schlussendlich zum gleichen Ergebnis zu kommen, d.h. den gleichen Output zu erzeugen. Dabei müssen nicht beide Programmierer die gleiche Programmiersprache, also SAS einsetzen, sondern es ist durchaus möglich, für die Zweitprogrammierung eine andere Programmiersprache, z.B. R zu verwenden.
Es geht also darum, dass auf Basis existierender Spezifikationen, zwei Programmierer unabhängig voneinander die gewünschten Outputs erstellen. Sobald beide ihre Programme, Log-Files und Outputs auf Richtigkeit geprüft haben und für fertig erachten, werden die erzeugten Outputs programmatisch, in der Regel mit der Prozedur COMPARE verglichen. Auftretende Unterschiede werden analysiert und korrigiert bis beide Programme die gleichen Outputs liefern.

Unabhängige Doppelprogrammierung hat sich im Rahmen der Auswertung von klinischen Studien in der Pharmabranche zunehmend zum Standard für studienspezifisch entwickelte Programme bzw. deren Outputs etabliert. Andere Validierungsmodelle (siehe 1.2) treten in den Hintergrund.
Unabhängige Doppelprogrammierung wird für die Validierung von Analysedatensätzen (ADS), aber auch von Tabellen, Listings und Graphiken (TLGs) eingesetzt.
ADS (SAS Datensätze) können problemlos mit der Prozedur COMPARE verglichen werden. Doppelprogrammierung ist aus diesem Grund bei der Erstellung von ADS besonders nahe liegend. Wird Doppelprogrammierung bei TLGs (meist Lst-Files) eingesetzt, werden die Inhalte der Outputs entweder durch selektive Doppelprogrammierung einzelner Ergebnisse mit der geeigneten SAS-Prozedur nachprogrammiert und visuell verglichen, oder aber die in dem Output dargestellten Ergebnisse werden vor der Ausgabe in einen SAS-Datensatz geschrieben, der dann wiederum mit PROC COMPARE verglichen werden kann.

Diese Veröffentlichung fokussiert sich beispielhaft auf die Doppelprogrammierung von ADS. Allerdings können die Schlussfolgerungen uneingeschränkt auf die Doppelprogrammierung von TLGs übertragen werden.
Für die Unabhängigkeit bei der Doppelprogrammierung sind entscheidend:
1) die der Programmierung zugrunde liegenden Spezifikationen
2) die Unabhängigkeit der Programmierer
3) der Abgleich der Ergebnisse und die daraus resultierenden Maßnahmen

Die Frage, ob die schlussendliche Gleichheit der Ergebnisse sicherstellt, dass die Outputs korrekt sind bzw. die Limitierungen dieses Prozesses, werden in den nachfolgenden Kapiteln besprochen.

2.2 Programmierung der Studienanalyse - Datenfluss

Im Rahmen einer klinischen Studie werden die gesammelten Daten in einer klinischen Datenbank erfasst. Für die Analyse der klinischen Rohdaten werden gemäß der Empfehlung der FDA zunächst Analysedatensätze programmiert, die die Basis für die Darstellung der Ergebnisse in Tabellen, Graphiken und Listings bilden.
In den "Study Data Specifications" erklärt die FDA die Relevanz von Analysedatensätzen folgendermaßen: *"Analysis datasets are created to support results presented in study reports, the Integrated Summary of Safety (ISS) and the Integrated Summary of Efficacy (ISE) and to support other analyses that enable a thorough regulatory review. Analysis datasets contain both raw and derived data."* [7].

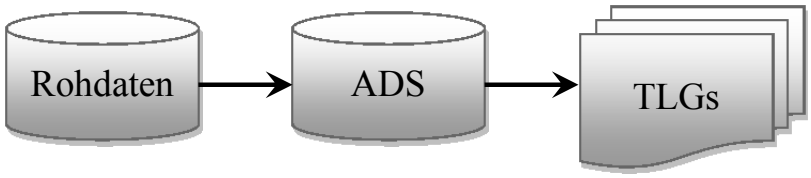

Abbildung 1: Datenfluss

Die Analysedatensätze enthalten dementsprechend alle auswertungsrelevanten Daten und Herleitungen, so dass in den Tabellen-, Listing- oder Grafikprogrammen keine weiteren Analysen, sondern lediglich die tabellarische oder graphische Ausgabe gemacht werden muss.
Struktur (Metadaten) und Inhalt von Analysedatensätzen werden in den sogenannten Analysedatensatzspezifikationen beschrieben. Dazu gehören auch alle Herleitungsregeln. Diese ADS Spezifikationen sind die Grundlage für die Programmierung der Analysedatensätze.

Tabelle 1: Auszug aus der Spezifikation eines Demographie-Analysedatensatzes

Variable Name	Label	Type	Code	Format	Source	Variable Source Derivation
SUBJECT	Subject	Num 8			[DMG]	
DOB1D	Birth (date)	Char 9		$9.	[DMG]	
AGE	Age (years)	Num 8		3.	Derived	Calculate Age at visit date (VIS1D)
AGECLAS	Ageclas	Num 8	1:Young 2:Adults	$20.	Derived	Young:0-18=>ageclas=1 Adults: ab 18=>ageclas=2
VIS1D	Visit Date	Char 9		$9.	[DMG]	
SEX1C	Sex	Num 8	1:Male 2:Female	SEXF.	[DMG]	

2.3 Erstellung der Spezifikationen

Ziele und Ablauf einer klinischen Studie sind im klinischen Studienprotokoll (Clinical Study Protocol, CSP) fixiert. Im statistischen Analyseplan (Statistical Analysis Plan,

SAP) werden auf Basis des Studienprotokolls alle auswertungsrelevanten Methoden definiert. Der statistische Analyseplan wiederum ist die Grundlage für alle weiteren, für die Programmierung relevanten Spezifikationsdokumente, d.h. den Tabellenvorlagen (Table Shells) und ADS Spezifikationen.

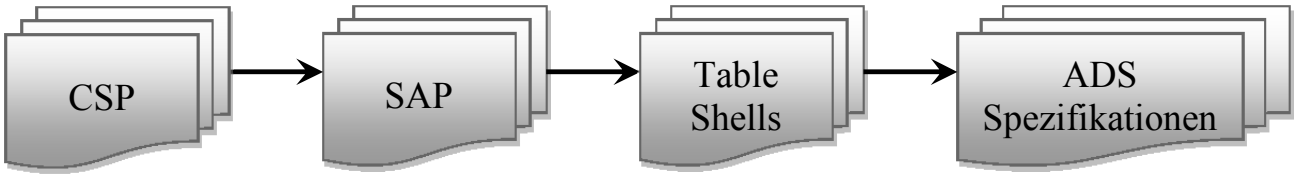

Abbildung 2: Spezifikationsfluss

2.4 Prozessablauf der Doppelprogrammierung

Basierend auf den ADS-Spezifikationen beginnen die Programmierer mit der Entwicklung von Programm A und B. Dies kann zeitgleich und frühzeitig beginnen, nämlich sobald ausreichend Daten in der klinischen Datenbank vorhanden sind. Wenn beide Programmierer der Meinung sind, dass die Datensätze (ADS A und ADS B) fertig und richtig sind, werden diese mit Hilfe von PROC COMPARE verglichen.

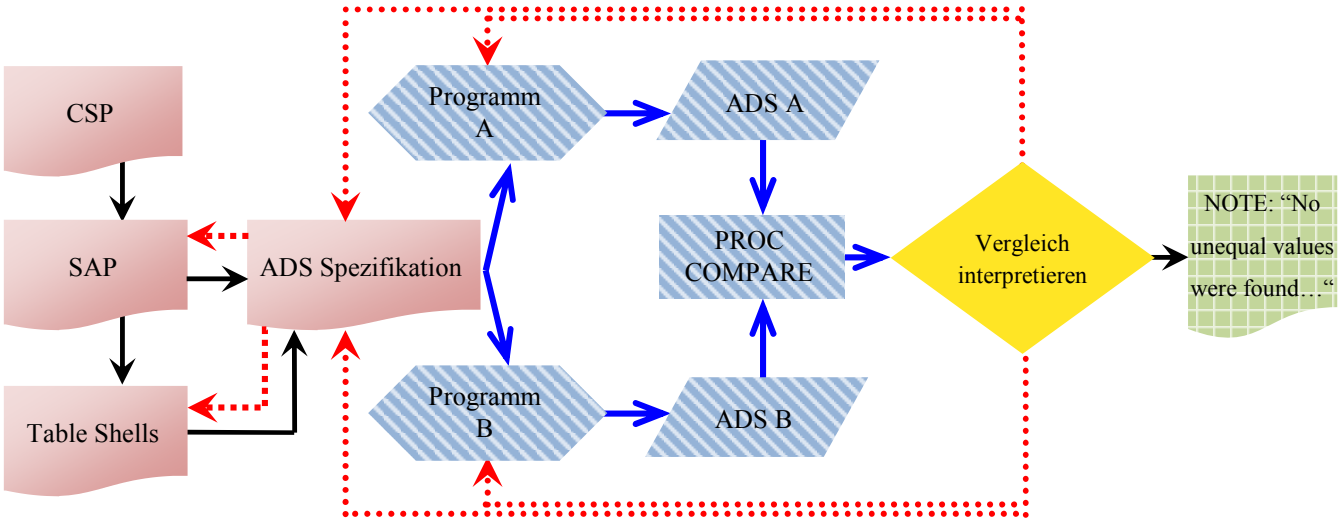

Abbildung 3: Prozessablauf der Doppelprogrammierung

Unterschiede, die sich aus dem Vergleich der Datensätze ergeben, werden anschließend identifiziert und interpretiert, um festzulegen, an welchen Stellen Korrekturen vorgenommen werden müssen. Hierzu kann und sollte bei Unklarheiten ein unabhängiger Dritter, z.B. der verantwortliche Statistiker, zu Rate gezogen werden. Korrekturen können in jedem der Spezifikationsdokumente und/oder in einem oder beiden Programmen notwendig sein. Der Prozess wird so lange wiederholt, bis der Abgleich keine Unterschiede mehr zeigt.

3 Kernelemente und Herausforderungen der unabhängigen Doppelprogrammierung

3.1 Spezifikationsdokumente

Die ADS-Spezifikationsdokumente sind die Basis für die Programmierung der Analysedatensätze und spielen aus diesem Grund eine entscheidende Rolle. Bevor die Programmierung beginnen kann, müssen stabile und gute Spezifikationen vorliegen. Aber was ist „gut“?
Zunächst stellt sich die Frage, wer die Spezifikationen erstellt. Will man von Anfang an jegliche Beeinflussung während der Programmierung ausschließen, sollten die Spezifikationen nicht von einem der beiden Programmierer erstellt worden sein. Allerdings gehen dadurch Synergien verloren, weil das während der Erstellung der Spezifikationen erworbene Wissen nicht bei der Programmierung eingesetzt werden kann.
Weiterhin ist die Frage, wie genau die Spezifikationen sein sollten. Je genauer, umso geringer der individuelle Interpretationsspielraum der einzelnen Programmierer. Ungleichheiten durch unterschiedliches Verständnis der Spezifikationen können so auf ein Minimum reduziert werden. Wer aber hat die Spezifikationen auf Richtigkeit geprüft? Die Überlegung drängt sich auf, ob diese nicht auch einem unabhängigen Review unterzogen werden müssten. Oder kann und sollte dieser Review durch die Programmierer, die während der Programmierung die Spezifikationen auf den Prüfstand stellen, ersetzt werden und sollten die Programmierer die Spezifikationen nicht in jedem Fall kritisch hinterfragen?

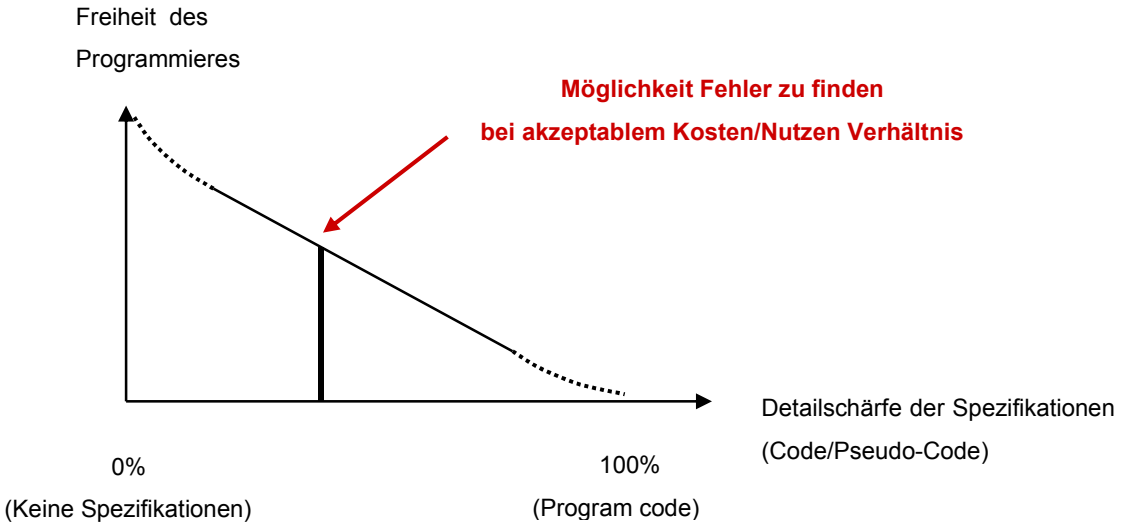

Abbildung 4: Detailschärfe der Spezifikationen

Die Detailschärfe der Spezifikationen steuert die Möglichkeit, Fehler in den Spezifikationen zu finden, oder schnell auf ein einheitliches, aber nicht zwingend richtiges Ergebnis zu kommen. Enthalten Spezifikationen z.B. einen Fehler in einer Herleitung, die keinen Raum zur Interpretation lässt, können beide Programmierer schnell zum glei-

chen, aber falschen Ergebnis kommen. Enthalten die Spezifikationen nicht viel mehr Details als der SAP, bleibt viel Raum für Interpretationen und es kann sehr lange dauern und vieler Absprachen bedürfen, bis Gleichheit der Ergebnisse erreicht ist. Es gilt also ein gesundes Mittelmaß bezüglich der Genauigkeit der Spezifikationen zu finden, um bei akzeptablem Kosten-Nutzen-Verhältnis das Finden von Fehlern zu ermöglichen.
Wichtig ist für diesen Prozess allerdings auch, wie der Status der Spezifikationen beurteilt wird. Sind diese verhandelbar, d.h. während der Programmierung als noch nicht final angesehen, bleibt den Programmierern die Chance, Unklarheiten und Lücken zu hinterfragen, sowie auch mögliche Fehler zu adressieren. Oft ergibt sich auch durch bestimmte Datenkonstellationen die Notwendigkeit der Anpassung der Spezifikationen während der Studienanalyse. Gibt es firmeninterne Vorgaben, dass diese z.B. zum Zeitpunkt des Datenbankschlusses finalisiert sein müssen, birgt dies die Gefahr nachträglich notwendiger Änderungen oder aber unterschiedlichster Work-Arounds. Die Prozesse der Studienanalyse werden unterstützt, wenn Spezifikationen bis zur Finalisierung der Analyse lebende Dokumente bleiben.

3.2 Unabhängigkeit

Wie ist Unabhängigkeit definiert: „Der eine sieht und weiß nicht, wie der andere die Aufgabe löst“. Unterschiedliche Modelle sind hier denkbar:

Modell A: Programmierer A und B arbeiten mit gleichen Zugriffsrechten in derselben physikalischen Programmierumgebung, entwickeln aber unabhängig voneinander ihre Programme. Die Programme und Datensätze des anderen Programmierers sind theoretisch jederzeit einsehbar. Hier basiert die Unabhängigkeit auf Vertrauen und auf der Verabredung, dass keiner den SAS-Code und die Ergebnisse, in diesem Fall den Analysedatensatz, des anderen zu Hilfe nimmt. Dies ist häufig der Ansatz der Wahl, da jederzeit automatisch gewährleistet ist, dass beide Programmierer mit den gleichen Rohdaten arbeiten. Gleiche Rohdaten sind die Grundvoraussetzung für gleiche Ergebnisse und bei einer laufenden Studie ändern sich die Daten ggf. täglich.

Modell B: Bei dieser Variante wird auf ein und demselben Server, aber durch Beschränkung der Zugriffsrechte sichergestellt, dass Programmierer A und B keinen Zugriff auf Programme und Output des andern Programmierers haben. Die Rohdaten liegen an einer Stelle, auf die beide Zugriff haben. Die Programme und die erstellten Analysedatensätze werden in unterschiedlichen, durch Zugriffsrechte geschützten Ordnern erstellt. Erst wenn der Prozess der Programmierung abgeschlossen ist, werden die Analysedatensätze in eine Umgebung verschoben, für die Leserechte vergeben sind, so dass sie verglichen werden können. Notwendige Änderungen werden wiederum getrennt voneinander implementiert.
Die Vergabe spezieller Zugriffsrechte bedeutet administrativen Zusatzaufwand, verringert die Flexibilität des Personaleinsatzes, verhindert frühe Einsicht in die Analysedatensätze z.B. durch den Statistiker und verzögert die Erstellung weiterer ADS, die von

den doppelt programmierten ADS abhängig sind und unterbindet den frühen Beginn der Programmierung von Tabellen, Graphiken und Listings.

Modell C: Die Entwicklung der Programme findet in unterschiedlichen physikalischen Umgebungen statt. Die Implementierung dieses Modells ist z.B. denkbar, wenn die Doppelprogrammierung durch einen externen Anbieter auf dessen eigenem Server durchgeführt wird. Programmierer A und B haben dadurch keinen Zugriff auf Programme und Datensätze des Anderen. Erst wenn der Prozess auf beiden Seiten abgeschlossen ist, werden die Datensätze ausgetauscht und die Ergebnisse verglichen. Der Ablauf des Prozesses, d.h. wer zum Zeitpunkt der Vergleichs die beiden unabhängig voneinander programmierten ADS archiviert und anschließend den Vergleich durchführt, sollte vorab definiert werden. Dieses Verfahren erfordert, dass beiden Programmierern regelmäßig die gleichen Rohdaten sowie die Spezifikationsdokumente in der aktuellsten Version zur Verfügung gestellt werden. Dies ist ein zusätzlicher Administrationsaufwand, der eingeplant werden muss.

3.3 Vergleich der Ergebnisse

Die Ergebnisse, d.h. die unabhängig voneinander programmierten Analysedatensätze, werden mit der Prozedur COMPARE verglichen. Insbesondere beim ersten Vergleich muss selbst bei nicht komplexen ADS mit Unterschieden gerechnet werden. Diese können viele Ursachen haben.
In einer frühen Phase der Programmierung und Datenerfassung ist es durchaus normal und wünschenswert, dass die Datensätze A und B sich noch unterscheiden, weil beispielsweise „unsaubere“ Daten unterschiedlich behandelt wurden. Hier helfen Kommentare im Log-File diese speziellen Datenkonstellationen zu verfolgen bis schließlich bei sauberen, finalen Daten Gleichheit erzielt ist.
Ein weiterer Grund für Diskrepanzen könnte sein, dass die Spezifikationsdokumente nicht ausreichend genau waren, oder beide Programmierer diese unterschiedlich verstanden haben. In diesem Fall müsste die Herleitung hinterfragt, diskutiert und genauer beschrieben werden. Das kann zur Folge haben, dass mindestens eines der Programme angepasst werden muss.
Sollten die Spezifikationsdokumente ausreichend genau sein, muss eines der beiden Programme fehlerhaft sein.
In gegenseitiger Absprache wird dann festgelegt, was und wo geändert wird, und es wird ggf. ein Trackingsheet angelegt, in dem die Unterschiede und die daraus resultierenden Änderungen kurz beschrieben werden. An dieser Stelle fängt der Prozess wieder bei den Programmen A und B, oder bei den Spezifikationen an. Dieser Prozess wird so oft wiederholt, bis der programmatische Vergleich der Datensätze keine Unterschiede mehr zeigt.

Beim Vergleich der Datensätze mit PROC COMPARE ist es wichtig, die Analysedatensätze einheitlich zu sortieren. Die Sortierungsvariablen sollten im ID-Statement spezifiziert werden. Der Output der Prozedur COMPARE sollte komplett dargestellt werden,

damit eine kritische Prüfung möglich ist und alle möglichen Unterschiede dargestellt und entdeckt werden können.
Als vorteilhaft hat sich die Sortierung der Analysedatensätze mit dem unten stehenden PROC SQL-Schritt erwiesen, durch die sich der Vergleich automatisieren lässt. Die Prozedur SQL bietet nicht nur den Komfort der Sortierung, sondern außerdem die einfache Möglichkeit, die Reihenfolge der Variablen im Analysedatensatz zu steuern.

```
proc sql;
create table <output>(label="Analyse Datensatz")
  as select  SVAR1, VAR2, VAR3, ...
  from <input>
  order by VAR1, VAR2, VAR3;
quit;
```

Die Sortierung der Analysedatensätze lässt sich mit Hilfe von PROC CONTENTS auslesen und im ID-Statement von PROC COMPARE benutzen.
Um die Vollständigkeit des Vergleichs mit möglichst vielen Informationen zu gewährleisten, ist es empfehlenswert folgende Optionen bei PROC COMPARE zu verwenden:

```
proc compare
     base      = <ADS_A>
     comp      = <ADS_B>
     out       = <CMP_A_>
     method    = EXACT | ABSOLUTE criterion = 10E-12
     outall
     listall;
     id VAR1 VAR2 VAR3;
run;
```

Mit der Option LISTALL wird gewährleistet, dass auch Beobachtungen angezeigt werden, die keinen einheitlichen Key haben und somit in den Vergleich nicht einbezogen werden können.
Darüber hinaus wird mit der Option OUTALL ein Datensatz mit den Unterschieden des Vergleichs erstellt, in dem aus beiden verglichenen Datensätzen Beobachtung für Beobachtung untereinander gesetzt werden. Unter die beiden verglichenen Beobachtungen wird eine dritte eingefügt, welche zeigt, ob es Unterschiede in den vorherigen Beobachtungen gab.
Durch METHOD=EXACT werden die Werte mit einer absoluten Genauigkeit verglichen. Die Kombination METHOD=ABSOLUTE und die Option CRITERION erlaubt ein Nachjustieren der Genauigkeit auf beispielsweise die zwölfte Nachkommastelle. Wenn Rechengenauigkeit ins Spiel kommt, muss gegebenenfalls geklärt werden, was noch als Gleichheit betrachtet werden darf.

4 Vor- und Nachteile der unabhängigen Doppelprogrammierung

Beim Einsatz von Doppelprogrammierung als Validierungsmodell kann bereits zu einem sehr frühen Zeitpunkt mit der Validierung begonnen werden, nämlich sobald einige Rohdaten zur Verfügung stehen und die Spezifikationen erstellt sind. Diese Validierungsmethode ist insofern gerade bei länger laufenden Studien und bei Fertigstellung der Analyse in engen Timelines von Vorteil, weil sich die Ergebnisse durch erneutes Laufenlassen der Programme A und B einfach revalidieren lassen. Natürlich kann es auch bei validierten Programmen vorkommen, dass obwohl bereits zu einem frühen Zeitpunkt Gleichheit erzielt wurde, aufgrund von neuen und geänderten Daten, Änderungen an Spezifikationen und / oder Programmen nötig werden.
Bezüglich der Programmierumgebung für die Doppelprogrammierung ist es schwierig eine Wertung abzugeben. Bei Modell A könnte theoretisch der Status des anderen Programms eingesehen werden und durch Zwischenvergleiche könnte es zu einer ungewollten, evtl. auch fehleranfälligen „Annäherung“ beider Programme kommen. Allerdings sind Absprachen bei den eingeschränkteren Modellen B und C theoretisch ebenso möglich. Sorgfältige und fachgerechte Vorgehensweise und Einhaltung der definierten Prozesse sind unerlässlich.
Häufig kommt es vor, dass während der Programmierung Ungenauigkeiten oder Fehler in den Spezifikationen gefunden werden. Dies bedeutet aber nicht, dass alle Fehler in den Spezifikationen durch Doppelprogrammierung zwingend gefunden werden. Doppelprogrammierung ist kein Beweis für die Güte der Validierung. Sie macht den Validierungsvorgang „sichtbar“, da am Ende ein zweites Programm und der dokumentierte Vergleich existierten. Somit ist die durchgeführte Validierung nachvollziehbar und „beweisbar“.
Die Frage, ob die Datensätze A und B identisch sind, wird durch den sinnvollen Einsatz von PROC COMPARE beantwortet. Das Ergebnis: *„NOTE: No unequal values were found. All values compared are exactly equal.”* besagt nicht mehr, als dass alle Observations, die vergleichbar waren, keine Unterschiede aufweisen. Erst durch die Anwendung aller beschriebenen Optionen von PROC COMPARE wird gewährleistet, dass alle Zeilen und alle Variablen eines Datensatzes in den Vergleich mit einbezogen werden und Zeilen, die nicht in Einklang gebracht werden können (z.B. wegen unterschiedlicher IDs) oder Variablen, die nicht verglichen werden können (z.B. wegen ungleicher Variablennamen), ebenfalls im COMPARE-Ergebnis aufgezeigt werden.
Auch wenn die Validierung durch Doppelprogrammierung erfolgt ist und das Ziel Ergebnisgleichheit erreicht wurde, sollten der SAS-Programmcode und der Log-File auf Einhaltung industrieüblicher Regeln von „Good Programming Practice“ geprüft werden. Dies ist ein weiterer Arbeitsschritt, der notwendig ist, um sicherzustellen, dass der SAS-Code und der Log-File fehlerfrei und sauber sind.

5 Zeitaufwand

Der Zeitaufwand und damit die Kosten für die Doppelprogrammierung sind nicht immer gleich, sondern von verschiedenen Faktoren abhängig. Diese sind

- Implementierung der unabhängigen Programmierumgebung (siehe 3.3)
- Datenqualität und Vollständigkeit der Rohdaten
- Komplexität des Analysedatensatzes
- Stabilität und Qualität der Spezifikationen
- Klärung von Rückfragen
- Erfahrung der Programmierer

Die Vollständigkeit und die Qualität der Rohdaten haben einen entscheidenden Einfluss auf den Zeitaufwand der Doppelprogrammierung. Je schlechter die Daten, umso eher sind zunächst Programme nicht lauffähig und umso mehr Diskrepanzen zeigen sich in der Regel beim Vergleich, die dann jeweils beurteilt und dokumentiert werden müssen, auch wenn sie nicht gelöst werden sollten.
Desweiteren ist die Komplexität des Analysedatensatzes entscheidend für den Zeitaufwand. Üblicherweise werden gerade die schwierigen Analysedatensätze, d.h. Datensätze mit komplexen Herleitungen und vielen Variablen doppelt programmiert, um mögliche Fehler auszuschließen.
Das richtige Maß in der Spezifikationsgenauigkeit ist ein weiterer entscheidender Zeit- und somit auch Kosten-Nutzen-Faktor. Sind die Angaben zu ungenau, bleibt zu viel Interpretationsspielraum und es ist unwahrscheinlich, dass beide Programmierer zeitnah zum gleichen Ergebnis kommen, sodass mehrfach Programmanpassungen durchgeführt werden müssen. Die Beurteilung von Unterschieden und die dann notwendige Klärung der geeigneten Maßnahmen können zeitaufwändig sein, aber auch zu zeitlichen Verzögerungen führen, je nachdem, wer die Klärung herbeiführen kann. Es ist durchaus denkbar, dass ein komplexes Problem Rücksprache mit der Medizin oder anderen Fachfunktionen außerhalb von Statistik und Programming erfordert.
Auch im Fall der Doppelprogramming ist die Erfahrung der involvierten Programmierer entscheidend für den Zeitaufwand und die Qualität der Validierung. Das bezieht sich zwar auch auf die Effizienz der Programmierung, aber vor allem darauf, dass Spezifikationen richtig interpretiert und implementiert bzw. mögliche Fehler entdeckt werden.

6 Fazit

Zusammenfassend kann man sagen, dass „unabhängige Doppelprogrammierung“ in sehr unterschiedlicher Art und Weise implementiert werden kann und die Ergebnisse demzufolge entsprechend mehr oder weniger belastbar sind.
Das dokumentierte „all values are exactly equal“ heißt nicht, dass alles richtig ist. Die Fehlerquellen während der Analyse einer Studie sind vielfältig und können durch qualifizierte Validierung minimiert, aber auch durch unabhängige Doppelprogrammierung nicht ausgeschlossen werden.

Unabhängige Doppelprogrammierung kann, auf der anderen Seite, zu sehr großem Zeiteinsatz und damit zu sehr hohen Kosten führen und bleibt doch an einigen Stellen einflusslos, denn die Qualität der Rohdaten und die Validität der Spezifikationsdokumente kann Doppelprogrammierung ebenso wenig ersetzen, wie der Review des Programmcodes und des Log-Files. Der Einsatz von qualifiziertem und erfahrenem Personal ist letztendlich ausschlaggebend.
Insofern stellt sich die Frage, ob Doppelprogrammierung immer das adäquate Mittel der Wahl ist, oder ob nicht, gemäß der Empfehlung der FDA, das Validierungsmodell risikobasiert gewählt und wo angebracht, durchaus auch andere Validierungsmodelle, wie Output- und/oder Sourcecode-Review, eingesetzt werden sollten.

Literatur

[1] Keith C. Benze, SEC Associates, Inc., Risk-Based Approach to SAS Program Validation, http://www.pharmasug.org/2005/FC04.pdf

[2] Nikos Tsokanas, Roche Ltd, Welwyn Garden City, United Kingdom, PhUSE 2007, Paper IS02, Validation of Programs developed Using SAS, http://www.phusewiki.org/docs/2007/PAPERS/IS02.pdf

[3] Dipl.- Dok. Christoph Ziegler, F. Hoffmann – La Roche AG, Basel, Switzerland, PhUSE 2008, Paper RA02, Risk Based Approach Applied to the Validation of Report Objects, http://www.phusewiki.org/docs/2008/PAPERS/RA02.pdf

[4] FDA Guidance for Industry Part 11, Electronic records; Electronic signatures – Scope and Application, August 2003, http://www.fda.gov/regulatoryinformation/guidances/ucm125067.htm

[5] Claudia Meurer, Accovion GmbH, Validierung von SAS-Programmen für die Auswertung und Dokumentation klinischer Daten - Prozess, Umfang und Dokumentation der Validierung, http://saswiki.org/images/0/02/11.KSFE-2007-Meurer-Validierung_der_Auswertung_klinischer_Faten.pdf

[6] Christoph Ziegler, Dipl.-Dok., F. Hoffmann – La Roche AG, Basel, Switzerland, Beate Hientzsch, Accovion GmbH, Eschborn, Germany, PhUSE 2010, Paper RG04, Validation Strategies for Report Objects and Programs - Challenges and Obstacles in a Risk Based Reporting World, http://www.phusewiki.org/docs/2010/2010%20PAPERS/RG04%20Paper.pdf

[7] FDA Study Data Specifications 2.0, July 2012, http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM312964.pdf