# Neuerungen in SASUnit, insbesondere Ermittlung der Testabdeckung

Patrick René Warnat
HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
patrick.warnat@analytical-software.de

**Zusammenfassung**

Gründliches und umfassendes Testen ist eine wichtige Voraussetzung, um neu erstellte oder geänderte SAS-Makros möglichst fehlerfrei einsetzen zu können. SAS-Entwickler können zur Unterstützung des Testens das Framework SASUnit einsetzen. Bei Verwendung von SASUnit werden die Testaufrufe der SAS-Makros als ausführbare Quelltexte erstellt und verwaltet. Dadurch lassen sich Tests ohne nennenswerten Aufwand jederzeit wiederholen, und negative Seiteneffekte von Software-Änderungen lassen sich leicht feststellen.
Dieser Beitrag gibt eine generelle Einführung in das bestehende und frei verfügbare Unit-Test-Framework SASUnit, erstellt von HMS Analytical Software. Mit SASUnit lassen sich Unit-Tests definieren, ausführen und Testergebnisse dokumentieren. Im Beitrag wird ein Überblick zu den aktuellen Entwicklungen zum SASUnit-Framework gegeben und insbesondere das neue Feature zur Ermittlung der Testabdeckung vorgestellt.
Denn um zu gewährleisten, dass der Quelltext eines SAS Makros korrekt definiert ist, sollte man prüfen, dass alle Teile des SAS-Makros mindestens einmal während der Tests ausgeführt werden. Dazu ist es sinnvoll, die so genannte Testabdeckung zu ermitteln, welche bemisst, wie gut der Quelltext eines SAS-Makros durch Tests abgedeckt ist.

**Schlüsselwörter:** Makro, SASUnit, Testen, Testabdeckung

# 1 Einleitung

SAS-Makros werden genutzt um die Wiederverwendbarkeit von SAS-Programmquelltexten zu vereinfachen.
Ein wichtiger Teil bei der Entwicklung eines Makros ist die Sicherstellung, dass das Makro die gegebenen Benutzeranforderungen korrekt implementiert.
Dazu ist es notwendig das Makro gründlich zu testen, zunächst in der grundlegenden Entwicklung, sowie später auch nach jeder Änderung/Erweiterung des Makros.
Dieser Prozess kann durch die Verwendung des Unit-Test-Frameworks SASUnit vereinfacht werden.
SASUnit ist ein Unit-Test-Framework für die Entwicklung, Ausführung und die Dokumentation der Tests von SAS-Makros. Mit SASUnit werden auch die Tests als ausführbare SAS-Quelltexte implementiert. Tests werden in sogenannten Testszenarien gruppiert. Testszenarien werden in sogenannte Test-Suiten zusammengefasst.

SASUnit ist im Grunde eine Sammlung von SAS-Makros, welche die Ausführung der Testszenarien ermöglichen und automatisch einen Report erstellen, der die ausgeführten Tests und Ihre Ergebnisse dokumentiert.

Eine SASUnit Test-Suite besteht aus einem oder mehreren Testszenarios. Ein Testszenario ist ein ausführbares SAS-Programm, welches Testfälle definiert. Jeder Testfall testet einen bestimmten Aspekt des zu testenden Makros (Prüfling). SASUnit wird von einem SAS-Programm kontrolliert, üblicherweise run_all genannt, welches eine Test-Suite repräsentiert.
In einer Test-Suite wird mit dem Aufruf des Makros %initSASUnit eine Testdatenbank geöffnet (in der Form von SAS-Dateien). Das Makro %runSASUnit führt jedes Testszenario in einer eigenen SAS-Session aus.
Pro Testfall in den Testszenarien werden statische Testdaten definiert, das zu prüfende SAS-Makro aufgerufen, und schließlich spezielle „Assertion"-Makros (Prüfungs-Makros) aufgerufen, um die erzeugten Ergebnisse mit zuvor festgelegten, erwarteten Ergebnissen zu vergleichen. Die Testergebnisse werden in der Testdatenbank gespeichert. Als letzter Schritt in einer Test-Suite-Definition wird mit dem Makro %reportSASUnit ein Report aus den Testergebnissen in der Testdatenbank generiert.
Abbildung 1 gibt einen Überblick zum Aufbau von Unit-Tests mit SASUnit.

Bei der Erstellung von Unit-Tests ist es sinnvoll sicherzustellen, dass alle Quelltext-Bestandteile mindestens in einem Test ausgeführt wurden. Dadurch wird vermieden, dass Teile des Quelltextes ungetestet bleiben. Die automatisierte Ermittlung welche Quellcode-Passagen in den Tests ausgeführt wurden nennt man Ermittlung der Testabdeckung. Ab der Version 1.2.1 unterstützt SASUnit unter SAS 9.3 die Ermittlung der Testabdeckung.

Die folgenden Abschnitte geben zunächst noch mehr Informationen zu SASUnit sowie einen Überblick zu den allgemeinen Neuerungen ab der Version 1.2. Danach wird die Möglichkeit der Ermittlung der Testabdeckung vorgestellt, die in der aktuellen SASUnit Version 1.2.1 unterstützt wird.

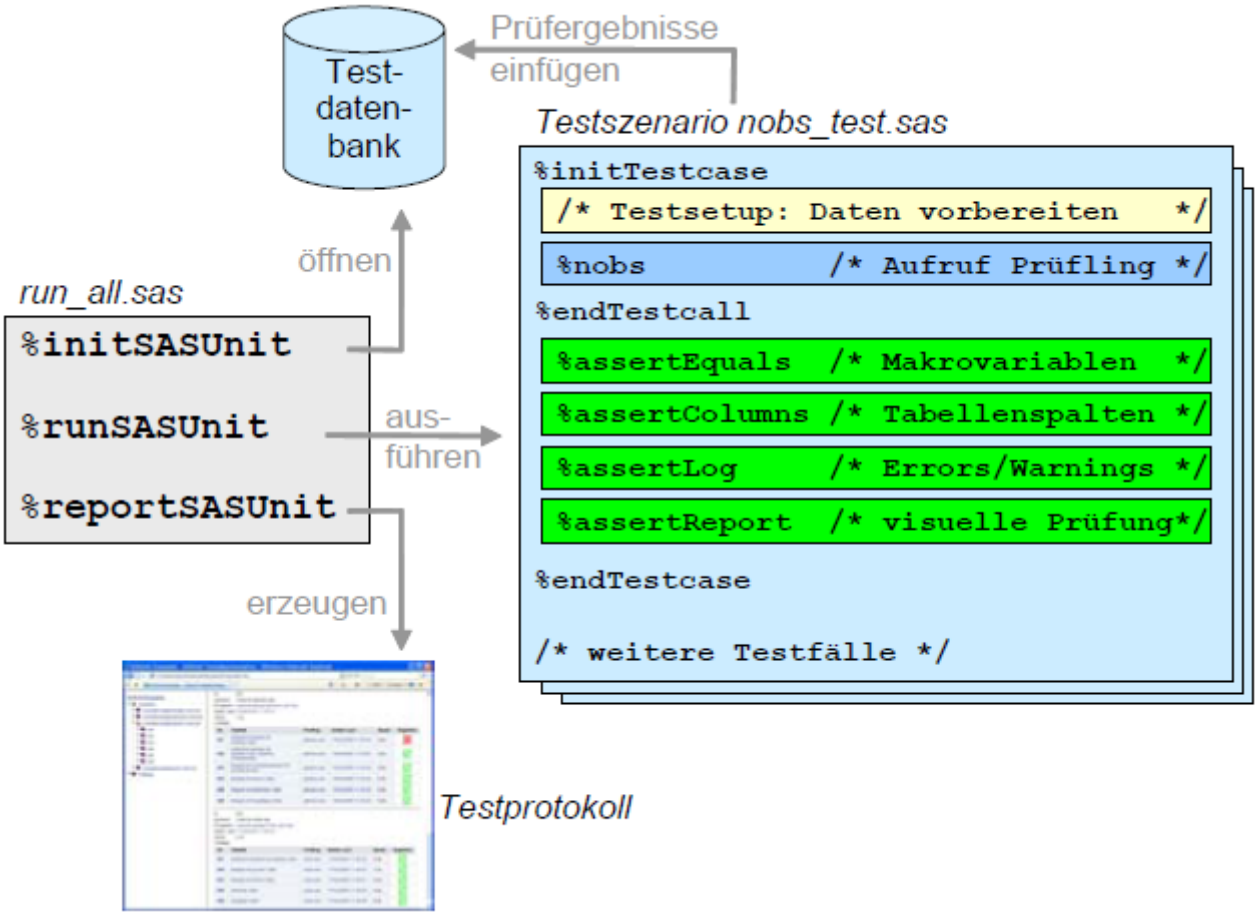


**Abbildung 1:** Aufbau von Unit-Tests mit SASUnit. Diagramm aus der Dokumentation von SASUnit, verfügbar über [1].

# 2 SASUnit

## 2.1 Bezugsmöglichkeit und Vorteile von SASUnit

SASUnit ist eine Open-Source-Software und über sourceforge.org [1] frei beziehbar. Die Vorteile von SASUnit können wie folgt zusammengefasst werden:

- Testszenarien und Tests für SAS-Makros werden als SAS-Quelltexte geschrieben.
- "Assertion"-Makros unterstützen die Prüfung von Ergebnissen: Werte von Makro-Variablen, Inhalte von SAS-Dateien, Existenz externer Dateien oder Auftauchen bestimmter Log-Meldungen.
- Testausführung und Dokumentation der Testergebnisse können im Batch-Modus ausgeführt werden.
- Die automatisch entstehende Testergebnis-Dokumentation ist klar und einheitlich gegliedert und mit der Dokumentation der SAS-Makros kombiniert.
- Laufzeit-Messungen werden zusätzlich ausgeführt.

- Nicht-automatisierbare Prüfungen können in Testszenarien aufgenommen werden. In der Testergebnisdokumentation erscheint dann ein Hinweis, dass z. B. noch manuelle Prüfungen von Reports gemacht werden müssen.
- Verfügbar für 8.2, 9.1, 9.2 und 9.3 unter Windows, Linux und Solaris

## 2.2 Neuerungen in SASUnit

Mit der Version 1.2 wurden die folgenden Änderungen an SASUnit durchgeführt (Beschreibung aus der SASUnit Dokumentation):

- Version 1.2 ist das erste Release von SASUnit, dessen Entwicklung vollständig auf der Open-Source Projektmanagement-Plattform SourceForge verwaltet wurde. Dadurch können neue Releases ab jetzt schneller bereitgestellt werden, und das Entwicklungsteam kann besser mit den Anwendern zusammenarbeiten.
- Die Unterstützung für unterschiedliche Ausführungsumgebungen (SAS-Versionen, Betriebssysteme etc.) ist systematisiert worden. Es gibt jetzt zwei Kommandodateien pro unterstützter Ausführungsumgebung: eine für die komplette Ausführung einer Test-Suite und eine, die nur geänderte Szenarien ausführt.
- Projekte können jetzt ohne weiteres an eine andere Stelle im Dateisystem verschoben werden, ohne dass Pfade angepasst werden müssen.
- Kommandodateien pausieren, falls dies nötig ist, um Fehlermeldungen anzuzeigen.
- Die Testverfahren wurden verbessert. SASUnit wird in einer speziellen Selbst-Test-Umgebung getestet, und es wird ein detaillierter Testplan abgearbeitet.
- Die Hauptseite der Testdokumentation enthält nun mehr Informationen über die Ausführungsumgebung (SAS-Version, Konfiguration, Benutzer-Id etc.).
- Das SASUnit-Logo wurde in ein grünes OK-Symbol geändert und enthält nun einen Link auf SourceForge.
- Es gibt eine neue Assertion für Performance-Tests (assertperformance.sas).
- Das Beispiel regression_test.sas wurde verbessert, um es besser verständlich zu machen (siehe regression.xls) und ist nun auch unter SAS auf Linux verfügbar.
- Testfälle, die Logdateien umleiten, werden nun unterstützt. Dafür gibt es die neuen globalen Makrovariablen g_logfile und g_printfile.
- Einige Fehler wurden bereinigt, insbesondere ein Fehler, der die inkrementelle Testausführung (nur geänderte Tests ausführen) in englischen Konfigurationen von Windows beeinträchtigt hat.

In der SASUnit-Version 1.2.1 ist zusätzlich noch die Möglichkeit hinzugekommen, eine Ermittlung der Testabdeckung durchzuführen. Dies wird in den folgenden Abschnitten beschrieben.

# 3 Ermittlung der Testabdeckung von Unit-Tests

## 3.1 Um was geht es?

Die Bewertung der Quelltextabdeckung von Unit-Tests ist ein formaler Ansatz um festzustellen, wie gut ein Quelltext getestet wurde. Die Ermittlung der Testabdeckung hilft dabei die Qualität von Testfällen zu bewerten. Es existieren verschiedene Ansätze, um die Abdeckung von Quelltexten durch Unit-Tests zu ermitteln.
Beispiele sind:

- „statement coverage“: Beschreibt ob jedes Statement/jede Quelltextzeile während der durchgeführten Tests ausgeführt wurde.
- “decision coverage”: Bewertet ob alle boolschen Ausdrücke, die in Ausführungs-Kontrollbefehlen (z. B. %IF-statements) verwendet wurden, in der Menge der durchgeführten Tests jeweils mindestens einmal zu wahr und einmal zu falsch aufgelöst wurden.
- “path coverage”: Prüfung, ob alle logisch möglichen Ausführungspfade während der Tests ausgeführt wurden.

Für eine detailierte Beschreibung und Diskussion der verschiedenen Ansätze zur Ermittlung der Testabdeckung, siehe [2].

## 3.2 Warum ist die Ermittlung der Testabdeckung sinnvoll?

Durch die Ermittlung der Testabdeckung kann festgestellt werden, ob es Quelltext-Bestandteile gibt, die in den zugehörigen Unit-Tests nie ausgeführt werden. So wie Unit-Tests eine Möglichkeit sind, um die Qualität von Quelltexten zu sichern, ist die Ermittlung der Testabdeckung hilfreich, um die Qualität der Testfälle selber zu prüfen. Durch die Ermittlung der Testabdeckung kann leicht verhindert werden, dass Teile von Quelltexten ungetestet und auf diese Weise unentdeckte Fehler in Quelltexten bleiben.

## 3.3 Implementierung in SASUnit

### 3.3.1 Generelles Konzept

In der aktuellen Version 1.2.1 von SASUnit ist die Ermittlung der Testabdeckung durch eine Variante der „statement coverage“ realisiert: Es wird die Testabdeckung jeder Quelltextzeile geprüft. Die Ermittlung der Testabdeckung wird im Aufruf des SASUNIT-Makros initSASUnit über den Parameterwert i_testcoverage = 1 eingeschaltet (Voraussetzung: SAS 9.3). Zur Implementierung der Ermittlung der Testabdeckung wird die SAS-Option MCOVERAGE verwendet, die ab der SAS-Version 9.3 in SAS verfügbar ist.
Mit der MCOVERAGE-Option und der zugehörigen MCOVERAGELOC Option kann während einer SAS-Sitzung protokolliert werden, welche Quelltextzeilen ausgeführt werden und welche Quelltextzeilen sogenannter „non contributing code“ sind (vor allem

Kommentarzeilen, aber auch z. B. Sprungzielmarkierungen für %GOTO-Befehle). Da von SAS-Unit pro Testszenario eine eigene SAS-Sitzung erzeugt wird, wird pro Testszenario eine Text-Datei erzeugt, in der mit einer für MCOVERAGE spezifischen Syntax protokolliert wird, welche Quelltextzeilen ausgeführt wurden und welche Quelltextzeilen „non-contributing code“ darstellen.
Nach Ausführung aller Testszenarien werden alle MCOVERAGE-Protokolldateien ausgewertet, um schließlich pro zu prüfendem Makro die Vereinigungsmenge der in allen Testfällen ausgeführten Quelltextzeilen zu ermitteln. Dies wird mit der Anzahl an Quelltextzeilen verglichen, die potenziell ausführbaren Quelltext darstellen. Dadurch kann die Prozentzahl ausgerechnet werden, die die Testabdeckung repräsentiert: Anzahl ausgeführter Zeilen / Anzahl aller ausführbaren Zeilen. Abbildung 2 zeigt ein Beispiel für den Teil des SASUnit-Ergebnisberichts, der die Testabdeckung zeigt.

Zusätzlich wird eine HTML-Repräsentation des geprüften Quelltexts erstellt, in der die einzelnen Quelltextzeilen entsprechend eingefärbt werden.

| Prüfling | Szenario | Testfälle | Prüfungen | Testabdeckung [%] | Ergebnis |
|---|---|---|---|---|---|
| boxplot.sas | 001 | 22 | 45 | 96 | ☐ |
| generate.sas | 002 | 6 | 9 | 100 | ☑ |
| getvars.sas | 003 | 6 | 12 | 100 | ☑ |
| nobs.sas | 004 | 6 | 12 | 100 | ☒ |
| regression.sas | 005 | 1 | 6 | 100 | ☐ |

**Abbildung 2:** Beispiel für den Teil des SASUnit-Ergebnisberichts der pro getestetem Makro (Prüfling) die Testabdeckung zeigt.

### 3.3.2 Probleme mit der SAS-Option MCOVERAGE

Leider ist das von der MCOVERAGE-Option erzeugte Protokoll fehlerbehaftet. Die Markierung von „non-contributing code“ ist in manchen Fällen uneinheitlich. So werden z. B. ausgeführte %LOCAL, %IF, %ELSE Statements in den meisten Fällen nicht als „non-contributing code“ markiert, in manchen Fällen werden diese Statements allerdings inkorrekt als „non-contributing code“ markiert, obwohl sie ausgeführt wurden. Auch werden in manchen Fällen PROC- oder DATA- Steps oder Teile davon als „non-contributing code“ markiert, obwohl der entsprechende Quelltextabschnitt definitiv ausführbar ist und auch ausgeführt wurde. Diese Probleme wurden bereits an den SAS-Support gemeldet und vom SAS-Support als Fehler der MCOVERAGE-Option anerkannt. Leider ist noch nicht bekannt wann dieses Problem behoben wird. Innerhalb der Quelltextzeilen, die von MCOVERAGE als „contributing code“ eingestuft werden, arbeitet die Markierung von ausgeführtem Code allerdings fast immer korrekt (auch hier wurden Fehler beobachtet, allerdings sehr selten). Da also in der Regel die große Mehr-

zahl an Quelltextzeilen korrekt markiert werden, kann man schon jetzt gut mit der MCOVERAGE Option arbeiten, allerdings muss man die Ergebnisse, insbesondere die Markierungen von „non-contributing code“, derzeit immer noch auf Plausibilität prüfen.

## 3.4 Beispiel für die Ermittlung der Testabdeckung

Ein sehr einfaches Beispiel-Makro soll zur Veranschaulichung der Ermittlung der Testabdeckung dienen:

```
/*Code Coverage Test Macro*/
%MACRO ccTestMacro1(binaryInput);
   %LOCAL printTxt;
   %IF &binaryInput EQ 1 %THEN %DO;
      %LET printTxt = A value equal to 1 was given.;
   %END;
   %ELSE %DO;
      %LET printTxt = A value not equal to 1 was given.;
   %END;
   %PUT &printTxt;
%MEND  ccTestMacro1;
```

Nehmen wir zunächst an es existiere nur ein Testfall mit dem folgenden Makro-Aufruf:

```
%ccTestMacro1(0);
```

In diesem Fall wird ein Report wie folgt erzeugt, der den Quelltext des Makros entsprechend der Ausführung der einzelnen Quelltextzeilen einfärbt (Zur Übersichtlichkeit sind in der Darstellung Zeilennummern hinzugefügt worden). Die orange markierte Zeile (Zeile 5) ist nicht ausgeführter Quelltext, grau markiert werden Kommentare (Zeile 1), grün markiert wird ausgeführter Quelltext (restliche Zeilen). Zeile 1 beinhaltet keinen ausführbaren Quelltext und wird daher nicht zur Ermittlung der Testabdeckung verwendet. Bleiben 10 Zeilen, wovon eine nicht im Test ausgeführt wird, dadurch ergibt sich eine Testabdeckung von 9/10 = 90 %.

```
000001   /*Code Coverage Test Macro*/
000002   %MACRO ccTestMacro1(binaryInput);
000003      %LOCAL printTxt;
000004      %IF &binaryInput EQ 1 %THEN %DO;
000005         %LET printTxt = A value equal to 1 was given.;
000006      %END;
000007      %ELSE %DO;
000008         %LET printTxt = A value not equal to 1 was given.;
000009      %END;
000010      %PUT &printTxt;
000011   %MEND ccTestMacro1;
```

Wenn ein Testfall mit dem Makro-Aufruf `%ccTestMacro1(1)` hinzugefügt wird, werden alle Quelltextzeilen (außer der Kommentarzeile) grün eingefärbt und es ergibt sich eine Testabdeckung von 100%.

# 4 Fazit

Der Einsatz von SASUnit bedeutet zwar etwas Mehraufwand bei der Erst-Entwicklung von SAS-Makros, erspart allerdings viel Arbeit bei der Erstellung von geänderten Makro-Versionen, da Tests einfach wiederholt ausgeführt werden können. Der Einsatz der Ermittlung der Testabdeckung trägt zur Qualitätssicherung für die implementierten Testfälle bei und dadurch auch zur Verbesserung der eigentlichen Makro-Codes. Zusätzlich wird die Dokumentation der Testergebnisse automatisch generiert, was alleine schon die Integration von SASUnit in den Entwicklungsprozess von SAS-Makros lohnenswert macht.

## Literatur

[1] SASUnit download über sourceforge.org, http://sourceforge.net/projects/sasunit/

[2] Wikipedia: Code Coverage, http://en.wikipedia.org/wiki/Code_Coverage, zuletzt abgerufen am 08.02.2013.