Kapitel 3

Komplexitätsmessung

Das Kapitel umfaßt die Überprüfungen der ausgewählten Meßverfahren. Um die Messungen anwenden zu können, muß die Verarbeitung von ABAP/4-Programmen vereinfacht werden. Im Kapitel 3.1 wird die Vereinfachung beschrieben. Auf dieser Grundlage erfolgt die ABAP/4-Interpretation für die zyklomatischen Zahl von McCabe im Kapitel 3.2 und für die vier Basisgrößen von Halstead im Kapitel 3.3. Im folgenden Kapitel 3.4 werden die ABAP/4-Beispiel-Programme gemessen und die Verfahren gegenüber der intuitiven Komplexität validiert.

3.1 Vereinfachte Betrachtung der ABAP/4-Ablaufsteuerung

Die ereignisorientierte Programmiersprache ABAP/4 verfügt neben einer internen Steuerung auch über eine externe Steuerung des Programmablaufs. Diese externe Steuerung wird durch zeitliche Ereignisse erreicht: Es erfolgt keine Bearbeitung der Verarbeitungsblöcke zu den Ereignissen wie sie im ABAP/4-Programm angeordnet sind. Zeitliche Ereignisse werden, wie im Kapitel 2.4.2 beschrieben, durch andere ABAP/4-Programme oder durch interaktive Anwendereingaben erzeugt. Eine Verbindung zwischen den zeitlichen Ereignissen und dem ABAP/4-Programm entsteht durch Ereignisschlüsselworte.

Folgende Ereignisse treten zur Laufzeit eines Reports, der logische Datenbanken verwendet, ein:

`INITIALIZATION`	Der Zeitpunkt vor dem Erscheinen des Selektionsbildes.
`START-OF-SELECTION`	Der Zeitpunkt nach der Verarbeitung des Selektionsbildes.
`GET <tabelle>`	Der Zeitpunkt der Bereitstellung eines Datenbankeintrages durch die logische Datenbank.
`GET LATE <tabelle>`	Der Zeitpunkt nach der Verarbeitung aller Datenbankeinträge aus den hierarchisch untergeordneten Datenstrukturen der verwendeten logischen Datenbank.

`END-OF-SELECTION`	Der Zeitpunkt nach der Selektion der Daten durch die logische Datenbank.

Weitere Ereignisse, die nicht in Verbindung mit logischen Datenbanken stehen, treffen während der Verarbeitung der Ausgabeliste eines Reports ein:

`TOP-OF-PAGE`	Der Zeitpunkt bei dem Beginn einer neuen Seite.
`END-OF-PAGE`	Der Zeitpunkt, wenn eine Seite beendet wird.

Nachfolgende Ereignisse können während der Anzeige einer Ausgabeliste eines Reports stattfinden:

`AT LINE-SELECTION`	Der Zeitpunkt, zu dem der Anwender eine Zeile auswählt.
`AT USER-COMMAND`	Der Zeitpunkt, zu dem der Anwender eine Funktionstaste drückt oder einen Befehl im Befehlsfeld eingibt.
`AT PF<nn>`	Der Zeitpunkt, zu dem der Anwender eine Funktionstaste mit dem Funktionscode PF<nn> drückt.

Die Meßverfahren von [McCabe76] und [Halstead77] berechnen die Modulkomplexität aus dem Programmtext. McCabe legt für sein Komplexitätsmaß Entscheidungsstrukturen zugrunde. Sein Verfahren mißt die strukturelle Komplexität. Die logische Struktur eines Moduls oder eines Verarbeitungsblocks wird in ABAP/4 durch die interne Steuerung beschrieben. Halstead legt Regeln zur Identifizierung von Operatoren und Operanden fest, um die Komplexität zu bestimmen. Das Verfahren mißt die berechende Komplexität. Alle ABAP/4-Schlüsselworte werden als Operatoren behandelt. Neben allen Anweisungen werden alle Metazeichen, wie Punkte, Doppelpunkte und Komma als Operatoren betrachtet. Konstanten und Variablen in ABAP/4-Programmen sind Operanden. Die zyklomatische Komplexität und das Verfahren von Halstead bilden die ABAP/4-Ereignis-Steuerung nicht ab.

Um die Verfahren von McCabe und Halstead auf die Programmiersprache ABAP/4 anwenden zu können, wird die durch die Ereignissteuerung vom ABAP/4-Prozessor erzeugte Struktur vernachlässigt. Die Abfolge der Ereignisse wird als sequentiell, wie im Programmtext angeordnet, aufgefaßt. Nur die interne Steuerung innerhalb eines Verarbeitungsblocks wird abgebildet. Die sequentielle Abfolge gilt für alle in diesem Abschnitt aufgeführten und im Kapitel 2.4.2 beschriebenen Ereignisse.

Die vereinfachte Betrachtung der ABAP/4-Ablaufsteuerung hat auf die beiden Meßverfahren von McCabe und Halstead folgende Auswirkung:

- Bei McCabe hat ein Ereignisschlüsselwort keine Auswirkung auf die logische Struktur des ABAP/4-Programms. Es wird wie eine sequentielle Anweisung betrachtet und hat keinen Einfluß auf die zyklomatische Zahl.

- Bei Halstead wird ein Ereignisschlüsselwort wie jede andere Anweisung als ein Operator betrachtet. Das Ereignisschlüsselwort fließt in die Anzahl eindeutiger Operatoren η_1 und in die Anzahl aller Operatoren N_1 ein.

3.2 ABAP/4-Kontrollgraphen für Messungen nach McCabe

In dem Meßverfahren von [McCabe76] werden struktur-erzeugende Anweisungen durch gerichtete Graphen dargestellt. Sie zeigen die logische Struktur der Software-Moudle. Die zyklomatische Komplexität kann aus dem gerichteten Graphen oder direkt aus den syntaktischen Konstrukten eines Programms ermittelt werden.

Nach McCabe beeinflussen syntaktische Konstrukte wie Ereignisse, bedingte Ausführungen, Iterationen und unstrukturierte Konstrukte, die Einfluß auf die Struktur eines Programms nehmen.

3.2.1 Symbole der gerichteten Graphen

In ABAP/4 werden zeitliche Ereignisse, Verarbeitungsschritte und Entscheidungen in Verzweigungen unterschieden. Verarbeitungsschritte verkörpern sequentielle Anweisungen und Entscheidungen sind bedingten Ausführungen und Iterationen. Der Aufruf von Funktionsbausteinen und Unterprogrammen wird differenziert. Nach [Pomberger93] können Funktionsbausteine ohne Vorbehalt, Unterprogramme nur eingeschränkt als Module angesehen werden. Die Deklaration eines Unterprogramms wird gesondert aufgeführt, um sie von zeitlichen Ereignissen abzugrenzen. Besonders gekennzeichnet werden Beginn und Ende eines Reports, um unter anderem unstrukturierte Konstrukte abbilden zu können. Die Knoten der gerichteten Graphen werden aus folgenden Symbolen aufgebaut:

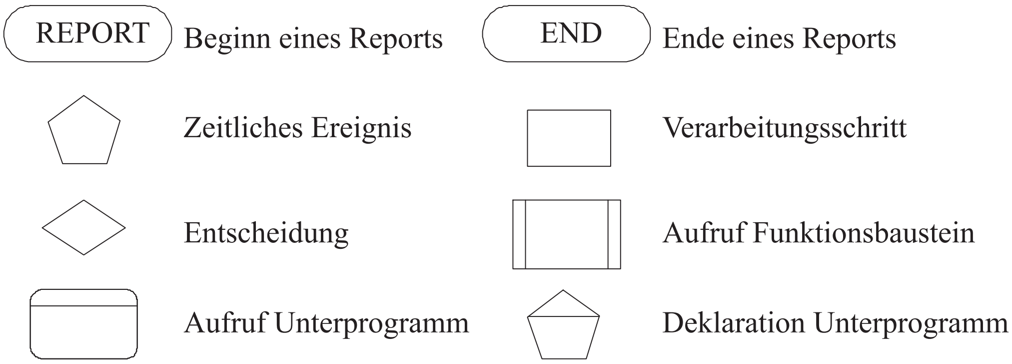


Abbildung 3.1: Symbole der Knoten in gerichteten Graphen für ABAP/4

3.2.2 Gerichtete Graphen von Kontrollstrukturen

Die minimale Anzahl von Konstrukten, die in der strukturierten Programmierung benötigt werden, sind Sequenzen, bedingte Ausführungen und Iterationen. In ABAP/4 werden zusätzlich Ereignisse betrachtet.

Ereignis

Wie in Kapitel 3.1 beschrieben, werden alle dort aufgeführten Ereignisse als sequentielle Anweisungen angesehen. Diese Betrachtung gilt auch für die Deklaration von Unterprogrammen, vgl. Kapitel 2.4.2. Das Ereignisschlüsselwort `GET LATE <tabelle>` wird immer im Zusammenhang mit dem jeweiligen `GET <tabelle>`-Ereignis angeführt.

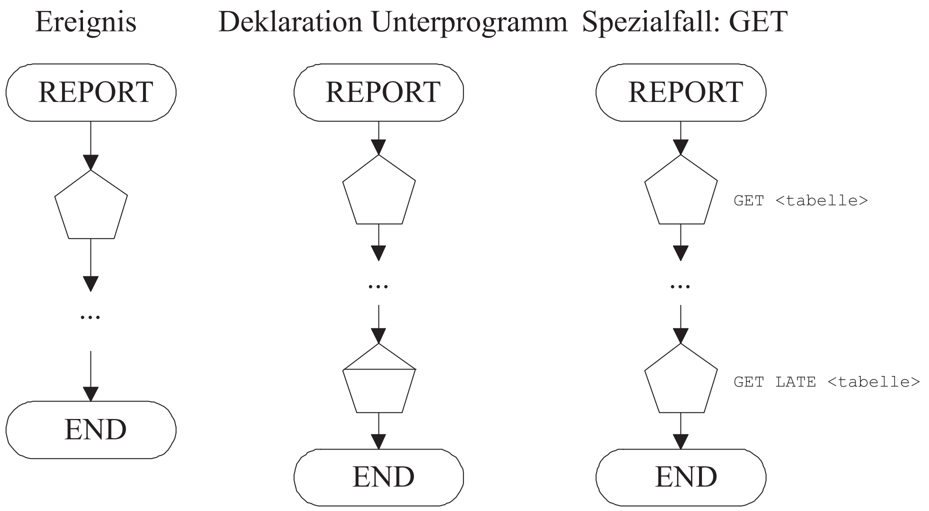


Abbildung 3.2: Ereignissteuerung in ABAP/4

Sequenz

Zusammengefaßt werden alle Ereignisse, allgemeine Verarbeitungsschritte Funktionsbaustein-, Unterprogrammaufrufe und Unterprogrammdeklarationen als Sequenz betrachtet. Deren Komplexität ist, unabhängig von der Anzahl von Anweisungen, immer 1. Allgemeine Verarbeitungsschritte sind alle operationalen ABAP/4-Schlüsselworte.

Bedingte Ausführungen

Bedingte Ausführungen sind in ABAP/4 die `IF`-Abfrage und die, in der Verarbeitung innerhalb einer Iteration bei Gruppenwechseln, `ON CHANGE OF`-Anweisung. Bei der `CASE`-Anweisung ist die zyklomatische Komplexität abhängig von der Anzahl der Verzweigungen.

Iteration

Allen Iterationen in ABAP/4 kann der gleiche Kontrollgraph zugrunde gelegt werden. Der Entscheidungsknoten ist bei der `WHILE`-Schleife ein logischer Ausdruck

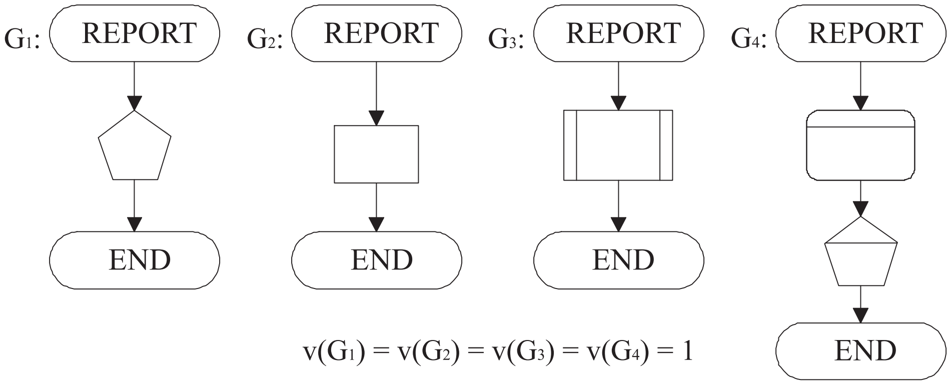


Abbildung 3.3: Kontrollgraphen von Sequenzen und ihre Komplexität

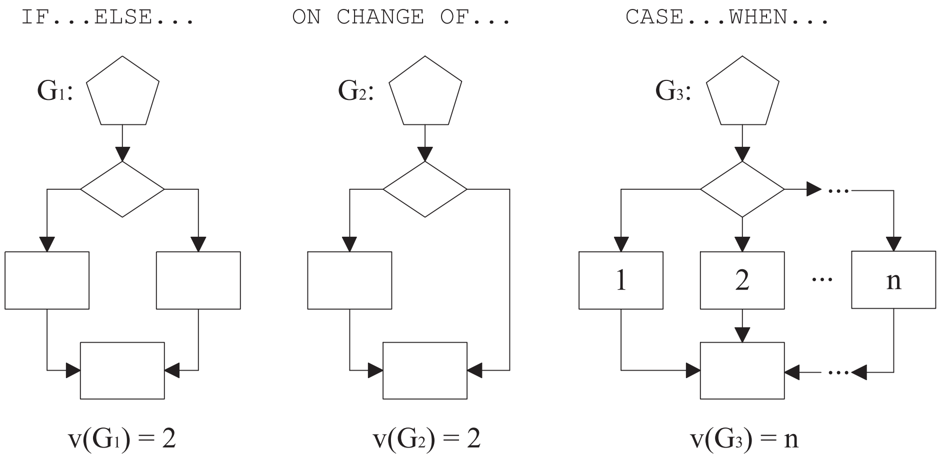


Abbildung 3.4: Kontrollgraphen der bedingten Ausführungen und ihre Komplexität

oder bei den anderen Schleifen ein Parameter. Dieser Parameter kann eine numerische Größe oder eine Tabelle sein. Die zyklomatische Komplexität der Iterationen ist mit Ausnahme bei der `WHILE`-Schleife, 2. In Abhängigkeit einer Verkettung der Binäroperationen im logischen Ausdruck steigt die zyklomatische Zahl.

3.2.3 Unstrukturierte Konstrukte

Als unstrukturierte Konstrukte sind nach McCabe die ABAP/4-Schlüsselworte `CHECK`, `EXIT`, `STOP` und `REJECT` anzusehen. Da die Abfolge der Ereignisse als Sequenz betrachtet wird, ist die zyklomatische Komplexität bei Sprüngen aus Ereignissen 2. Liegt einer der vier Basis-Typen der unstrukturierten Konstrukte vor, ist die Komplexität mindestens 3, vgl. Kapitel 2.2.1.

Die oben aufgezählten Konstrukte werden verwendet, um zum Beispiel einen Per-

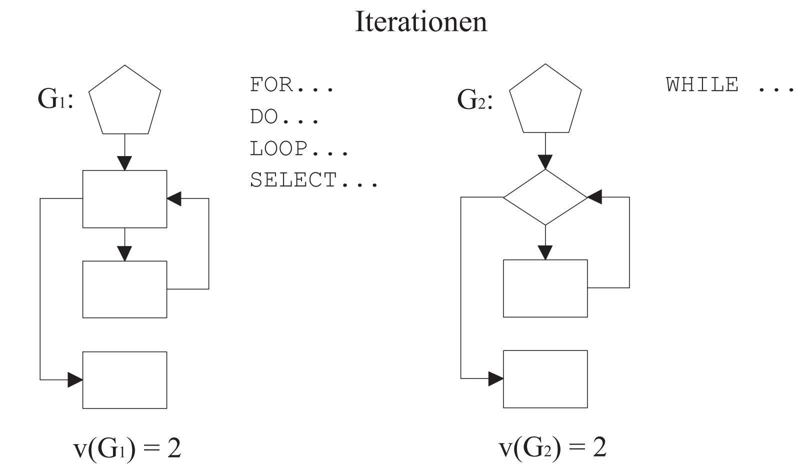


Abbildung 3.5: Kontrollgraph der Iteration und ihre Komplexität

formancegewinn bei der Datenselektion zu erreichen.

`CHECK`-Anweisung

Innerhalb von Iterationen ermöglicht `CHECK` den Sprung an den Schleifenanfang. Unterprogramme können mittels `CHECK` vorzeitig verlassen werden. Bei der Datenselektion wird der nächste Tabelleneintrag auf gleicher Hierarchiestufe verarbeitet.

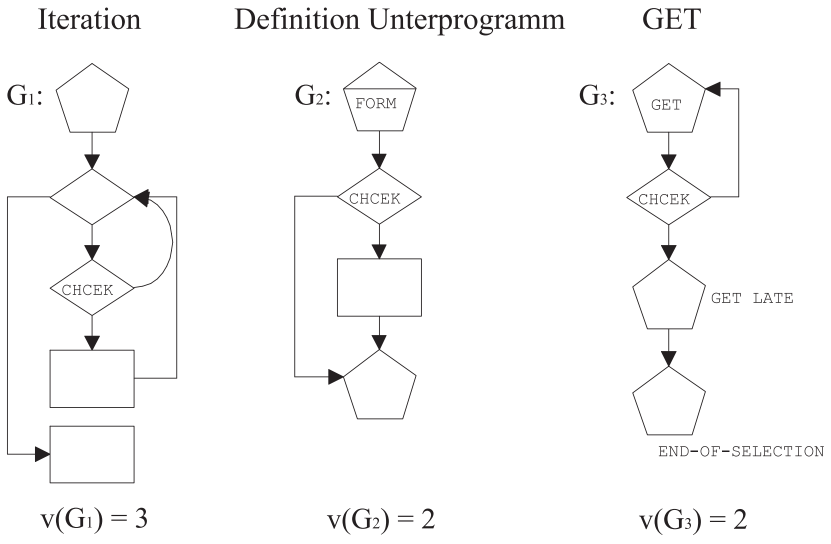


Abbildung 3.6: Kontrollgraphen der CHECK-Anweisung in verschiedenen Konstellationen mit ihrer Komplexität

`EXIT`-Anweisung

Die `EXIT`-Anweisung verhält sich in Schleifen und Unterprogrammen wie eine `CHECK`-Anweisung. Außerhalb von Iterationen und Unterprogrammen sowie bei der Datenselektion wird die Verarbeitung abgebrochen und die Liste angezeigt. Das Ereignis `END-OF-SELECTION` wird nicht prozessiert.

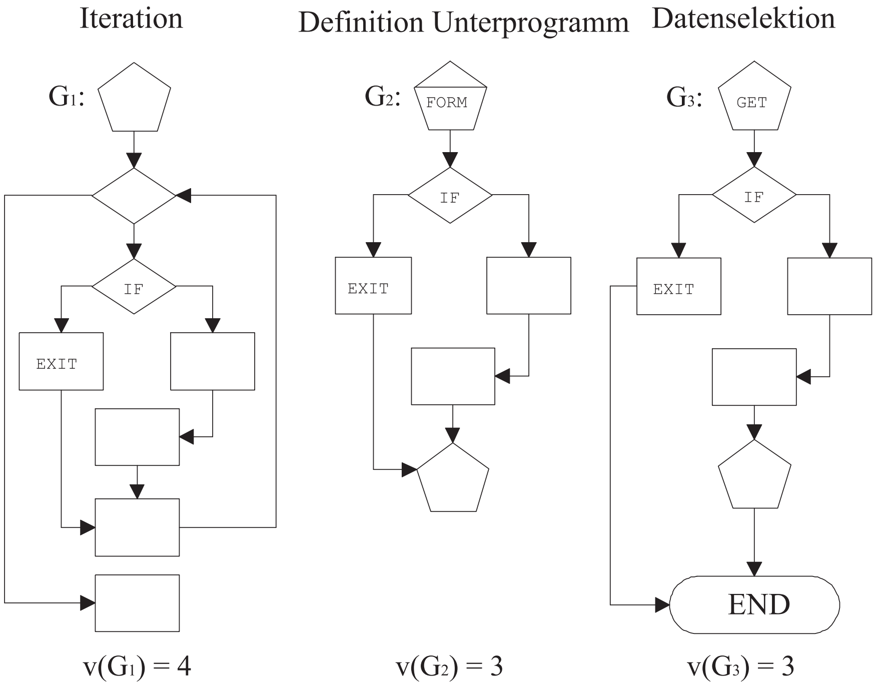


Abbildung 3.7: Kontrollgraphen der EXIT-Anweisung in verschiedenen Konstellationen mit ihrer Komplextät

`STOP`- und `REJECT`-Anweisung

Ein beliebiger Verarbeitungsblock kann sofort mit `STOP` verlassen werden. In diesem Fall führt die Verzweigung direkt zum Ereignis `END-OF-SELECTION`. Innerhalb der Datenselektion beginnt nach `REJECT` die Verarbeitung des nächsten Tabelleneintrages auf gleicher Hierarchiestufe.

3.3 ABAP/4-Interpretation für Halstead

Das Meßverfahren von [Halstead77] basiert auf der Bestimmung der Anzahl von Operatoren und Operanden eines Moduls oder Programms. Aus diesen Größen entwickelt Halstead Grundmessungen wie die Programmlänge N, die Programm-Größe V, den Informations-Inhalt I und die Programm-Leistung E. Die Berechnung der

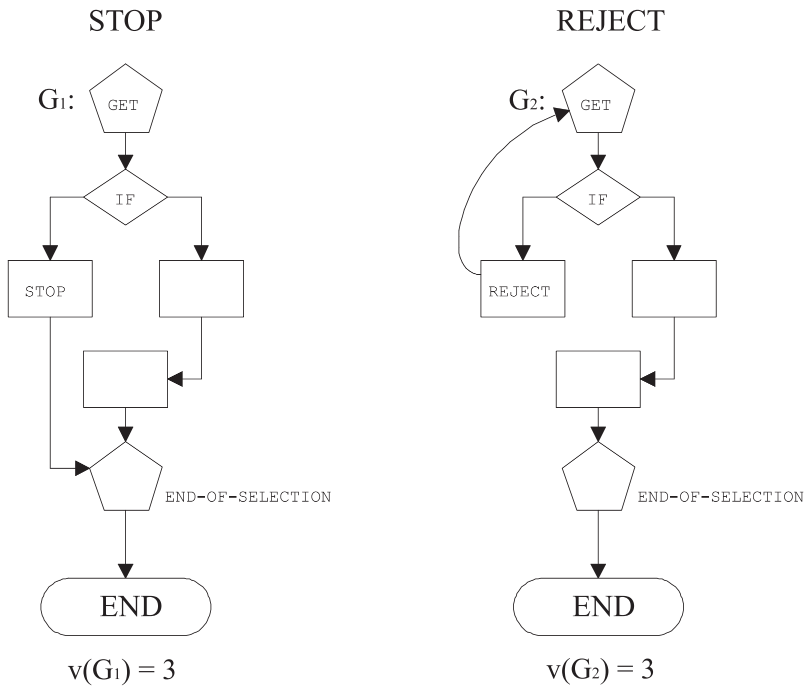


Abbildung 3.8: Kontrollgraphen der STOP- und REJECT-Anweisung und ihre Komplexität

Messungen aus den Operatoren und Operanden stützt sich auf einfache mathematische Operationen. Der Schwerpunkt der Halstead-Methode liegt in der berechenden Komplexität.

Halstead führt keine eindeutigen Angaben über die Identifizierung von Operatoren und Operanden an. Eindeutige Regeln zur Identifizierung von Operatoren und Operanden sind Voraussetzung, um nachvollziehbare Ergebnisse zu erhalten.

3.3.1 Regeln zur Identifzierung von Operatoren

Im allgemeinen bezeichnet Halstead Metazeichen und Schlüsselworte als Operatoren, die Operationen auf Operanden ausüben und Einfluß auf den Programmablauf nehmen. Ausgeschlossen sind deklarative Schlüsselworte und Kommentare.

In ABAP/4 sind die Schlüsselworte in vier Kategorien unterteilt: Ereignis-, Steuerungs-, operationale und deklarative Schlüsselworte. Eine vollständige Auflistung aller als Operator geltenden Schlüsselworte und deren Gewichtung als Operator befindet sich in Anhang A. Ergänzt wird die Liste durch die Anzahl der Operanden, die im Zusammenhang mit dem jeweiligen Operator stehen.

Ereignisschlüsselworte

Die Ereignisschlüsselworte werden jeweils als ein Operator betrachtet:

- `AT LINE-SELECTION.`
- `AT USER-COMMAND.`
- `AT PF<nn>.`
- `AT SELECTION-SCREEN.`
- `AT SELECTION-SCREEN OUTPUT`
- `END-OF-SELECTION.`
- `END-OF-PAGE.`
- `FORM ... ENDFORM.`
- GET <segment>.
- GET LATE <segment>.
- `INITIALIZATION.`
- `START-OF-SELECTION.`
- `TOP-OF-PAGE.`
- `TOP-OF-PAGE DURING LINE-SELECTION.`

Steuernde Schlüsselworte

Alle steuernden Schlüsselworte wie Funktionsbaustein-, Unterprogrammaufrufe, bedingte Ausführungen, Iterationen, Gruppenwechsel und unstrukturierte Konstrukte werden jeweils als ein Operator gezählt, auch wenn sie klammernde Operatoren sind.

In bedingten Ausführungen und Iterationen erhöht jeder Verknüpfungs- und boolsche Vergleichsoperator die Gewichtung des Operators um 1.

- `AT FIRST. ... ENDAT.`
- `AT NEW ... ENDAT.`
- `AT END OF ... ENDAT.`
- `AT LAST. ... ENDAT.`
- `CALL FUNCTION <name> ... .`
- `CASE ... ENDCASE.`
- `CHECK.`
- `DO. ... ENDDO.`
- `EXIT.`
- `FOR. ... ENDFOR.`
- `IF ... ENDIF.`
- `LEAVE.`
- `LOOP AT ... ENDLOOP.`
- `ON CHANGE OF ... ENDON.`
- `PERFORM <name> ... .`
- `REJECT.`
- `SELECT ... ENDSELECT.`
- `STOP.`
- `WHILE ... ENDWHILE.`

Metazeichen

Folgende Metazeichen werden in ABAP/4 zum Abschluß von Programmsätzen und Kettensätzen verwendet. Sie werden jeweils als ein Operator gezählt und sind nicht wie andere Metazeichen operationale Schlüsselworte.

- . (Abschluß ABAP/4-Satz)
- , (Verbindung von Kettensatzgliedern)
- : (Leitet Kettensatz ein)

Keine Operatoren

Alle operationalen Schlüsselworte, die als Operator interpretiert werden, werden in diesem Abschnitt nicht aufgeführt. Hier ist auf den Anhang A verwiesen.

Es wird eine sogenannte Negativliste aufgestellt, die alle Schlüsselworte beinhaltet, die nicht als Operator angesehen werden.

Kommentarzeilen:

- `*` in erster Spalte
- `"` in beliebiger Spalte

Allgemeine Schlüsselworte:

- `BREAK-POINT`
- `INCLUDE`
- `LINE-COUNT`
- `LINE-SIZE`
- `MESSAGE-ID`
- `NO STANDARD PAGE HEADING`
- `PAGE COUNT`
- `REPORT`

Schlüsselworte in Deklarationen:

- `BEGIN OF`
- `BEGIN OF COMMON PART`
- `CONDITION`
- `DATA`
- `DECIMALS`
- `DEFAULT`
- `DYNPROS`
- `END OF`
- `END OF COMMON PART`
- `FIELD-GROUPS`
- `FIELD-SYMBOLS`
- `LIKE`
- `LOCAL`
- `OCCURS`
- `PARAMETERS`
- `SEGMENTS`
- `SELECT-OPTIONS`
- `TABLES`
- `TEXTS`
- `TYPE`
- `VALUE`

Besonderheiten

Einige ABAP/4-Metazeichen müssen kontextabhängig behandelt werden. Es ist möglich sie in verschiedenen Kontexten als verschiedene Operatoren einzusetzen. Entsprechend zählen sie als zwei eindeutige Operatoren η_1.

- Der Operator "`=`" fungiert kontextabhängig entweder als Vergleichs- oder als Zuweisungsoperator.
- Das Zeichen "`+`" beschreibt zum einen den Offset bei Feldern und Variablen und ist zum anderen ein arithmetischer Additionsoperator.
- Die Klammern "`( .. )`"legen bei Feldern und Variablen die Ausgabelänge fest und dienen außerdem zur Klammerung von arithmetischen Ausdrücken und logischen Vergleichsoperationen.

Mittels der beiden letzten Operatoren ist es in ABAP/4 möglich, Teilbereiche von Feldern durch Offset- und Längenangaben anzusprechen und zu verändern: Die Operatoren "`+`" und "`( .. )`" in dem Konstrukt `<feld>+<pos>(<länge>)` werden bei der Ausgabe des Feldes oder bei einer Zuweisung vorher auf das Feld angewendet. Sie ändern das Feld entsprechend ab. Die Gesamtanzahl der Operatoren N_1 bei einem derartigen Konstrukt erhöht sich zusätzlich um 2.

Sätze in einem ABAP/4-Programm, die mit demgleichen Schlüsselwort beginnen, können mit “:” und “,” zu einem Kettensatz zusammengefaßt werden. Es erfolgt keine Auflösung des Kettensatzes in r Konstrukte, bestehend aus r einzelnen Operatoren mit entsprechenden Operanden. Die Verkettung bleibt erhalten und wird entsprechend bewertet, z.B. WRITE: <vari1>, <vari2>, <vari3>. Dieses Konstrukt besteht insgesamt aus fünf Operatoren und drei Operanden. Eine Auflösung in WRITE <vari1>. WRITE <vari2>. WRITE <vari3>. hat sechs Operatoren und drei Operanden.

Bei den beiden bedingten Ausführungen in ABAP/4, bei der CASE- und bei der IF-Anweisung, gilt folgende Zählung: Die komplette CASE-Anweisung ist ein Operator, da sich die Fallunterscheidung auf ein Feld bezieht. Bei der IF-Anweisung erhöhen weitere ELSEIF-Fälle jeweils die Anzahl der Operatoren um 1.

3.3.2 Regeln zur Identifizierung von Operanden

Halstead bezeichnet Variablen und Konstanten als Operanden, die von Operatoren verwendet und verändert werden. Für die Sprache ABAP/4 wird die Definition von Operanden um Datenobjekte wie z.B. Datenbanksegmente und Tabellen erweitert.

Operanden sind zum einen alle im Deklarationsteil eines ABAP/4-Programms aufgeführten Variablen, Konstanten und Tabellen. Zum anderen zählen alle referenzierten Felder im Programmtext als Operanden. Verwendete Ziffern- und Textliterale wie z.B. 1, 34, 'TEXT', SPACE gelten auch als Operanden.

Operanden im Deklarationsteil

Mögliche Operanden im Deklarationsteil eines ABAP/4-Programms sind:

- Alle mittels DATA sowie in Funktionsbausteinen und Unterprogrammen mittels LOCAL deklarierten Variablen und Konstanten.
- Mit dem Zusatz BEGIN OF in der DATA-Anweisung deklarierten internen Tabellen und deren Felder.
- Datenbanksegmente in SEGMENTS-Definitionen
- SAP-Tabellen in TABLES-Deklarationen.
- Texte der SAP-Textverarbeitung mittels TEXTS.
- Dynpros für die Batch-Input[1]-Verarbeitung in DYNPORS-Definitionen.
- Reportinterne Felder, über eine PARAMETERS-Deklaration. Die Felder können vom Anwender gesetzt werden.

[1] Über die Schnittstelle Batch-Input können große Datenmengen in ein SAP-System übernommen werden. Dieses erfolgt mittels einer Batch-Input-Mappe, in der eine Folge von Transaktionen zusammengefaßt sind, die mit Anwenderdaten versehen wurden.

- Wertemengen zur Datenselektion über eine `SELECT-OPTIONS`-Definition. Diese Mengen können auch vom Anwender gesetzt werden.
- Deklarierte Feldgruppen für die Extraktionssortierung mittels `FIELD-GROUPS`.
- Deklarierte Feldsymbole[2] mittels `FIELD-SYMBOLS`.

Referenzierte Felder als Operand

Neben den direkt deklarierten Variablen kann in einem ABAP/4-Programm mit referenzierten Feldern über deklarierte Datenbanksegmente und Tabellen gearbeitet werden. Die Deklaration dieser Felder erfolgt indirekt über das Datenbanksegment oder die SAP-Tabelle. Folgende referenzierte Felder werden als Operanden betrachtet:

- Referenzierte Datenbankfelder aus deklarierten Segmenten über <segment>-<feldname>.
- Referenzierte Tabellenfelder aus deklarierten SAP-Tabellen über <tabelle>-<feldname>.
- Referenzierte Systemfelder über SY-<feldname>.

3.3.3 Ermittlung der Operatoren und Operanden

ABAP/4-Programmteile können in sogenannte `INCLUDE`-Programme ausgelagert werden. `INCLUDE`-Anweisungen zählen nicht als Operator. Die `INCLUDE`-Programme werden in das ABAP/4-Programm einbezogen.

Ein typischer Satzaufbau eines ABAP/4-Programms ist ein Wechsel von Operatoren und Operanden. Jeder Satz schließt immer mit einem Punkt "`.`" ab. Ein Kettensatz wird mit einem Doppelpunkt "`:`" eingeleitet und mit einem Punkt "`.`" beendet. Die einzelnen Bestandteile des Kettensatzes werden durch ein Komma "`,`" getrennt.

Die Anzahl der eindeutigen Operatoren η_1 und die Gesamtanzahl der Operatoren N_1 wird folgendermaßen aus dem Programmtext eines ABAP/4-Programms ermittelt: Eindeutige Operatoren η_1 sind alle mindestens einmal auftretenden Schlüsselworte und Metazeichen im Anweisungsteil eines ABAP/4-Programms. Mitgezählt werden die Anweisungen aus Funktionsbausteinen und Unterprogrammen. `INCLUDE`-Programme werden aufgelöst. Zur Bestimmung der Gesamtanzahl N_1 wird das r-fache Auftreten jedes vorkommenden Operators η_1 aufsummiert. Die Metazeichen "`.`", "`:`" und "`,`" werden als einfache Operatoren behandelt.

Eindeutige Operanden η_2 sind alle deklarierten Variablen, referenzierten Felder, Ziffer- und Textliterale. Deklarierte, aber nicht benutzte Variablen, Konstanten, Segmente, Tabellen und Felder fließen nicht in η_2 ein, da sie keinen Einfluß auf das Programm ausüben. Die Parameter der Schnittstelle von Funktionsbausteinen und

[2]Zeiger werden in ABAP/4 als Feldsymbole bezeichnet.

Unterprogrammen sowie lokal deklarierte Variablen werden ebenfalls als Operanden betrachtet.

Allgemein gilt, daß alle Operanden von Operatoren verwendet und verändert werden müssen, um für die Zählung relevant zu sein. Für die jeweiligen Messungen bedeutet dies, daß bei ABAP/4-Schlüsselworten, die nicht als Operator angesehen werden, auch eventuell verwendete Operanden nicht berücksichtigt werden.

3.4 Messung der ABAP/4-Beispiel-Programme

In diesem Kapitel wird die Komplexität der ABAP/4-Beispiel-Programme ermittelt. Die Ergebnisse der Messungen werden gegenüber der intuitiven Komplexität validiert. Die intuitive Einschätzung von 62 Beispiel-Programmen erfolgt im Kapitel 3.4.1. Sie bildet die Basis für die anschließende Untersuchung der zyklomatischen Zahl im Kapitel 3.4.2 und der Programmier-Leistung im Kapitel 3.4.3. Es wird geprüft, ob die beiden Meßverfahren die intuitive Bewertung der ABAP/4-Beispiele bestätigen. Das Kapitel 3.4.4 befaßt sich mit dem Zusammenhang zwischen der Verfahren. Im abschließenden Kapitel 3.4.5 werden die gezogenen Konsequenzen aus den Messungen behandelt.

3.4.1 Intuitive Komplexität

Die Intuition ist das „ganzheitliche Erkennen ... von Sachverhalten". Wirklichkeitszusammenhänge werden vollständig erfaßt [Brockhaus89].

Die intuitive Komplexität bezieht sich auf die psychologische Komplexität. Sie ist eine erste und wichtige Einschätzung der ABAP/4-Beispiel-Programme. Die intuitive Komplexität teilt die Programme in verschiedene Klassen ein und ist vom Maßstabstyp ordinal. Die Relationen *komplexer als* und *gleich komplex* ordnen die Beispiele relativ zueinander zu. Innerhalb einer Klasse sind die Programme intuitiv als gleich komplex anzusehen. Dabei spielt die, bei der Beispiel-Auswahl gemachte Eingruppierung in sequentielle, nicht-sequentielle, unstrukturierte, modularisierte und unmodularisierte Programme keine Rolle mehr.

Durch die Einordnung der ABAP/4-Beispiel-Programme mittels der intuitiven Komplexität können der Meßverfahren von [McCabe76] und [Halstead77] überprüft werden. Es wird der Zusammenhang zwischen der intuitiven Einschätzung und den Ergebnissen der Messungen untersucht. Neben der Möglichkeit der Analyse der Validität der Messungen schafft die intuitive Komplexität eine Basis, um die beiden Verfahren miteinander vergleichen zu können, vgl. Kapitel 2.1.4.

Im Gegensatz zu den Meßverfahren von McCabe und Halstead bildet die intuitive Komplexität viele Facetten von Software-Komplexität ab. Folgende Kriterien und Fragen sind in die intuitive Komplexität der ABAP/4-Beispiel-Programme eingegangen und haben zur dargestellten Klasseneinteilung geführt:

- Programmablauf: Liegt eine Batch-Input- oder Listverarbeitung vor?

Report	Bezeichnung	Intu. Kompl.	Report	Bezeichnung	Intu. Kompl.
RFSCHU04	S-UM-001	1	RFSCHU17	S-UM-008	1
RFSCHU07	S-UM-002	1	RFSCHU13	N-UM-001	1
RFSCHU06	S-UM-006	1	RFSCHU09	U-UM-002	1
RFSCHU34	S-UM-009	2	RFSCHU15	N-UM-004	2
RFSCHU18	S-UM-010	2	RFSCHU19	N-UM-005	2
RMSCHU18	S-UM-011	2	RFSCHU26	N-UM-006	2
RFSCHU18	S-M-001	2	RFSCHU27	N-UM-009	2
RFSCHU18	S-M-002	2	RFSCHU37	N-UM-010	2
RMSCHU18	S-M-003	2	RFSCHU41	N-UM-011	2
RFSCHU13	N-UM-002-1	2	RFSCHU72	N-UM-016	2
RFSCHU13	N-UM-002-2	2	RFSCHU10	U-UM-001	2
RFSCHU13	N-UM-002-3	2	RFSCHU16	U-UM-003	2
RFSCHU22	N-UM-003	2			
RMSCHU11	S-UM-004	3	RFSCHU65	N-UM-014	3
RMSCHU21	S-UM-005	3	RFSCHU38	N-UM-017	3
RFSCHU11	S-UM-007	3	RFSCHU71	N-UM-019	3
RFSCHU25	N-UM-008	3	RFSCHU20	N-M-001	3
RFSCHU20	N-UM-012	3	RFSCHU72	U-UM-009	3
RFSCHU30	S-UM-003	4	RFSCHU19	U-UM-011	4
RFSCHU23	N-UM-013	4	RFSCHU27	U-UM-013	4
RFSCHU60	N-M-002	4	RFSCHU28	U-UM-014	4
RMSCHU06	U-UM-004	4	RFSCHU27	U-M-001	4
RFSCHU25	U-UM-010	4	RFSCHU28	U-M-002	4
RFSCHU33	N-UM-007	5	RFSCHU39	U-UM-007	5
RFSCHU42	N-UM-015	5	RFSCHU42	U-UM-008	5
RFSCHU63	N-UM-018	5	RFSCHU24	U-UM-012	5
RFSCHU60	N-UM-020	5	RFSCHU64	U-UM-015	5
RFSCHU35	U-UM-006	5			
RFSCHU32	N-M-003	6	RMSCHU39	U-UM-005	6
RFSCHU32	N-UM-021	7	RMSCHU30	U-M-003	7
RFSCHU42	U-M-004	8	RFSCHU31	U-M-005	9
RFSCHU34	U-M-006	10	RFSCHU80	U-M-007	11

Tabelle 3.1: Intuitive Komplexität der ABAP/4-Beispiel-Programme

- Programm-Länge in LoC: Wie viele Zeilen umfaßt das Programm und wie hoch ist der Anteil der Kommentarzeilen?

- Deklarationen im Programm: Welche SAP-Segmente und SAP-Tabellen werden genutzt und wie viele interne Tabellen, sonstige Variablen und Konstanten sind deklariert?

- Referenzierte Felder: Welche Felder von genutzte SAP-Segmenten und SAP-Tabellen sowie welche SAP-System-Felder werden referenziert? Wie hoch ist ihre Anzahl?

- Externe Steuerung durch Ereignisse: Welche logischen Datenbanken werden verwendet? Kann interaktiv in den Programmablauf eingegriffen werden? Inwieweit wird das Seitenlayout verändert?

- Steuernde Schlüsselworte: Welche Schlüsselworte werden verwendet? Zu welchem Grad werden sie verschachtelt wie mit Binäroperationen verknüpft?

- Unstrukturierte Konstrukte: Welche Konstrukte werden eingesetzt und wo werden sie benutzt?

- Operationale Schlüsselworte: Welche sequentiellen Anweisungen werden benutzt? Wie hoch ist der prozentuale Anteil am Programm?

- Modularisierung: Wie hoch ist der Grad der Modularisierung? Werden Unterprogramme und/oder Funktionsbausteine verwendet?

Die intuitive Komplexität bezieht sich nur auf die Auswahl der ABAP/4-Beispiel-Programme und ordnet diese relativ zueinander ein. Durch Hinzunahme weiterer Programme, die in die vorhandenen Klassen nicht einzuordnen wären, würde sich die Klasseneinteilung verschieben. Eine Verschiebung wäre bei Programmen mit einer höheren intuitiven Komplexität zu erwarten. Dieser Bereich ist durch die Beispiel-Auswahl kaum abgedeckt.

3.4.2 Zyklomatische Komplexität der Beispiel-Programme

Die Ermittlung der zyklomatischen Komplexität der ABAP/4-Beispiel-Programme erfolgt direkt aus den syntaktischen Konstrukten im Programmtext. Die zyklomatische Zahl eines strukturierten Programms ist gleich der Anzahl der Verzweigungen plus eins, vgl. Kapitel 2.2.1. Bei den Messungen wird zwischen ABAP/4-Verzweigungen und unstrukturierten Konstrukten keine Unterscheidung, mittels einer Gewichtung vorgenommen. Jedes unstrukturierte Konstrukt erhöht die zyklomatische Zahl im jeweiligen Kontext um eins. In Kapitel 3.2.3 sind die unstrukturierten Konstrukte in der entsprechenden Umgebung mit ihrer zyklomatischen Komplexität dargestellt. Bei binär verketteten `CHECK`-Anweisungen wird zusätzlich die zyklomatische Zahl um die Anzahl der Binäroperationen hinaufgesetzt. Das gleiche gilt auch für verkettete Verzweigungen. Ereignisse werden als sequentielle Anweisungen betrachtet, vgl. Kapitel 3.1.

In diesem Kapitel wird der Zusammenhang zwischen der intuitiven Einschätzung der Beispiel-Programme und den Ergebnissen der Messung nach McCabe untersucht. Eine hohe Korrelation besagt, daß die zyklomatische Zahl die intuitive Bewertung bestätigt.

Ergebnisse der Komplexitätsmessung

In der nachfolgenden Tabelle sind die Ergebnisse der Komplexitätsmessungen nach McCabe zusammengefaßt. Die Strukturierung der zyklomatischen Zahlen erfolgt nach der Klasseneinteilung durch die intuitive Komplexität. Eine vollständige Übersicht der Messungen ist im Anhang B zu finden. Sie beinhaltet neben der zyklomatischen Komplexität der einzelnen Module bei modularisierten Programmen auch die essentielle Komplexität von Beispielen mit unstrukturierten Konstrukten.

Int.[a] Kompl.	Report	Zykl.[b] Zahl	Report	Zykl. Zahl	Report	Zykl. Zahl
1	S-UM-001	1	S-UM-006	1	N-UM-001	2
	S-UM-002	1	S-UM-008	1	U-UM-002	3
2	S-UM-009	1	N-UM-010	2	S-M-002	4
	S-UM-010	1	U-UM-001	2	N-UM-002-1	4
	S-UM-011	1	U-UM-003	2	N-UM-002-2	4
	N-UM-004	2	N-UM-003	3	N-UM-002-3	4
	N-UM-005	2	N-UM-011	3	S-M-003	7
	N-UM-006	2	N-UM-016	3		
	N-UM-009	2	S-M-001	4		
3	S-UM-004	1	N-UM-019	2	N-UM-014	8
	S-UM-005	1	U-UM-009	4	N-M-001	8
	S-UM-007	1	N-UM-012	5		
	N-UM-008	2	N-UM-017	5		
4	S-UM-003	1	N-UM-013	6	U-M-001	11
	U-UM-004	2	U-UM-010	7	N-M-002	14
	U-UM-011	5	U-M-002	7		
	U-UM-014	5	U-UM-013	9		
5	N-UM-007	4	U-UM-007	7	U-UM-008	10
	U-UM-006	5	N-UM-015	8	N-UM-018	12
	N-UM-020	7	U-UM-015	8	U-UM-012	12
6	U-UM-005	12	N-M-003	15		
7	N-UM-021	9	U-M-003	19		
8	U-M-004	23				
9	U-M-005	(55)				
10	U-M-006	(77)				
11	U-M-007	(89)				

Tabelle 3.2: Die Ergebnisse der Komplexitätsmessung nach McCabe pro intuitiver Komplexitätsklasse

[a]Intuitive Komplexität
[b]Zyklomatische Zahl

Auswertung der Ergebnisse

Folgende Frage soll in diesem Abschnitt beantwortet werden: Bestätigt die zyklomatische Komplexität die intuitive Einschätzung der Beispiel-Programme?

Es wird untersucht wie hoch die Korrelation zwischen der zyklomatischen Zahl und der intuitiven Bewertung ist. Voraussetzung für die Messungen ist die Interpretation von McCabe für die Programmiersprache ABAP/4 in Kapitel 3.1 und 3.2.

Die intuitive Komplexität der Beispiele und die resultierende Klasseneinteilung liefern die Basis der Auswertung. Die Auswertung unterteilt sich in zwei Schritte:

1. Die zyklomatischen Zahlen werden pro Klasse zusammengefaßt und sind Ausgangspunkt für die eigentliche Auswertung.

2. Mit Hilfe der Korrelationsrechnung wird überprüft, ob zwischen der intuitiven Komplexität und der zyklomatischen Zahl ein Zusammenhang besteht.

Die Beispiel-Programme U-M-005, U-M-006 und U-M-007 werden in der Auswertung nicht betrachtet. Sie würden in der vorliegenden Klasseneinteilung die Auswertungsergebnisse verfälschen. Die statistische Sicherheit, abgelesen aus der Höhe des Korrelationskoeffizienten, würde sinken. Ein Grund ist, daß die Programme alleinige Vertreter ihrer Klassen sind. Es gibt keine Vergleichsmöglichkeiten für die Bewertung der intuitiven Komplexität. Ein weiterer Aspekt ist, daß die Klasseneinteilung für eine höhere intuitive Komplexität nicht gesichert ist. Die Klasseneinteilung bezieht sich nur auf die Auswahl der ABAP/4-Beispiel-Programme. Durch Hinzunahme weiterer ABAP/4-Programme in die Beispiel-Auswahl können u.a. die nicht-betrachteten Programme wahrscheinlich in höhere Klassen rutschen.

Intuitive Kompl.	Mittelwert Zykl. Zahl	Standard-abweichung	Intuitive Kompl.	Mittelwert Zykl. Zahl	Standard-abweichung
1	1,5	0,8	7	14,0	7,1
2	2,8	1,5	8	23,0	
3	3,7	2,8	9	(55,0)	
4	6,7	3,9	10	(77,0)	
5	8,1	2,8	11	(89,0)	
6	13,5	2,1			

Tabelle 3.3: Mittelwert der zyklomatischen Zahl nach McCabepro intuitiver Komplexitätsklasse

Im ersten Schritt sind die Ergebnisse der Komplexitätsmessungen aus Tabelle 3.2 in Tabelle 3.3 zusammengefaßt. Die intuitiven Komplexitätsklassen bilden die Grundlage. Die Standardabweichung gibt die Streuung um den berechneten Mittelwert der zyklomatischen Zahl an. Die intuitive Komplexität und die zyklomatische Zahl sind in der folgenden Abbildung graphisch dargestellt.

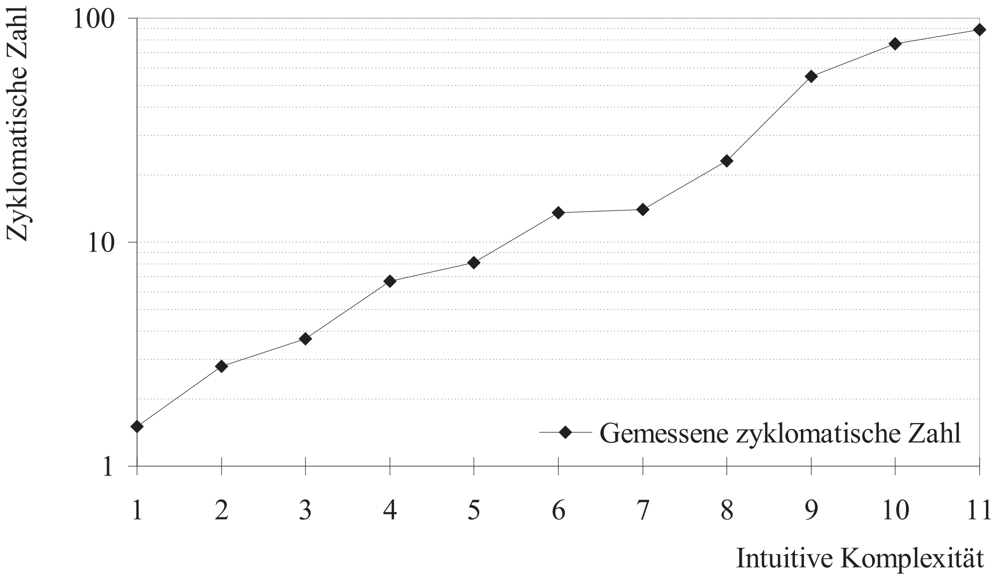


Abbildung 3.9: Graphische Darstellung der zyklomatischen Zahl auf Basis der intuitiven Komplexitätsklassen

Aus der Tabelle 3.3 und ihrer graphischen Darstellung, Abbildung 3.9 läßt sich intuitiv schließen, daß die zyklomatische Zahl ein Komplexitätsmaß für ABAP/4-Programme ist. Dies wird im folgenden statistisch überprüft und bestätigt.

Mittels der Korrelationsrechnung wird im zweiten Schritt kontrolliert, wie hoch der Zusammenhang zwischen der intuitive Komplexität und der zyklomatische Zahl ist. Die Tabelle 3.3 dient als Ausgangspunkt für die Korrelationsrechnung. Es handelt sich um eine einfache lineare Korrelation. Die ersten acht Komplexitätsklassen umfassen die Probenanzahl n. Bei sechs Freiheitsgraden $FG = n - 2$ und einem Korrelationskoeffizienten von $r = +0,952^{***}$ liegt die Irrtumswahrscheinlichkeit P unter $0,1\%$.

Das Ziel der Auswertung der Meßergebnisse ist, zu überprüfen, inwieweit es sich bei der Zunahme der zyklomatischen Komplexität um eine echte Steigerung handelt. Das Resultat der Korrelationsrechnung sagt aus, daß die Steigerung der zyklomatischen Komplexität über die acht untersuchten intuitiven Komplexitätsklassen mit einer Irrtumswahrscheinlichkeit P unter $0,1\%$ statistisch gesichert ist.

Wird für die Komplexitätsmessung nach McCabe die Interpretation für die Programmiersprache ABAP/4 vorausgesetzt, bestätigt die zyklomatischen Komplexität die intuitive Einschätzung der Beispiel-Programme. Das heißt, daß die zyklomatische Zahl zur Messung der Komplexität von ABAP/4-Programmen eingesetzt werden kann.

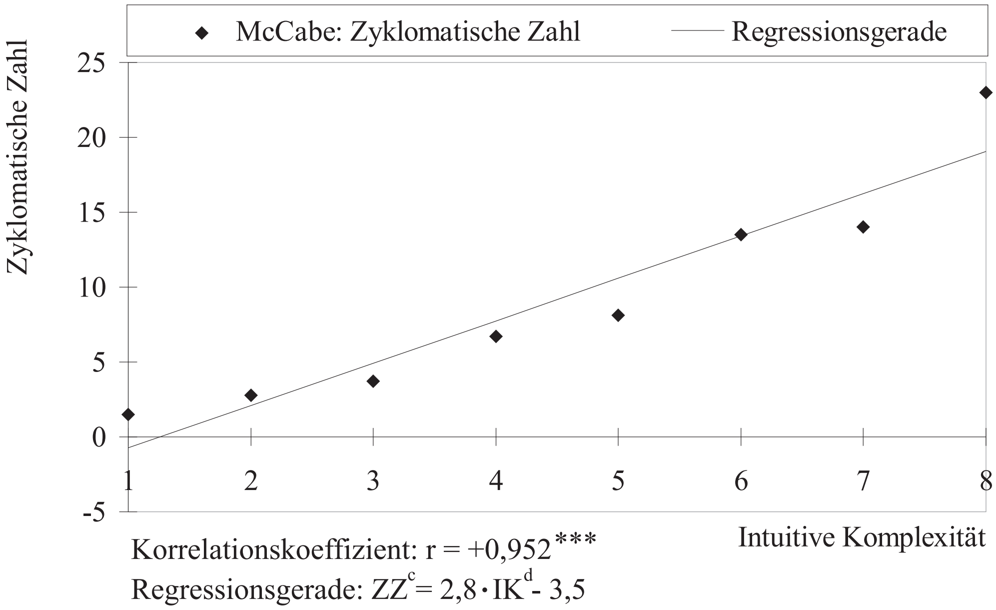


Abbildung 3.10: Graphische Darstellung der Korrelation der intuitiven Komplexität und der zyklomatischen Zahl

Die Regressionsanalyse hat verfolgt Ziel eine funktionale Beziehung zwischen den zwei Variablen x und y zu finden, die es gestattet, aus den vorgegebenen Werten der Einflußgröße x die jeweilige Zielgröße y zu schätzen. Die Beziehung kann durch eine Regressionsgerade beschrieben werden. Die Schätzung der Kennzahlen der Regressionsgerade geht von dem Prinzip aus, das die Gerade eine möglichst gute Anpassung an sämtliche empirisch ermittelten y-Werte darstellen soll. Die Regressionsgerade, die den Zusammenhang zwischen der intuitven Komplexität und der zyklomatischen Zahl beschreibt, ist: $\mathrm{ZZ} = 2,82 \cdot \mathrm{IK} - 3,55$. Auf Basis der ABAP/4-Beispiel-Auswahl kann die zyklomatische Zahl aus der intuitiven Komplexität berechnet werden.

3.4.3 Komplexitätsmessung nach Halstead

Halsteads Meßverfahren bestimmt die Komplexität eines Moduls oder Programms aus den Operatoren und Operanden im Programmtext. Die Grundmessungen sind die Programm-Länge N, die Programm-Größe V und die Programmier-Leistung E, wobei die Programmier-Leistung als Komplexitätsmaß angesehen wird. Die Basis der Messungen bilden die eindeutigen Operatoren η_1 und Operanden η_2 sowie die Gesamtanzahl der Operatoren N_1 und Operanden N_2. Aus diesen vier Größen werden die Grundmessungen berechnet. Eine Definition der Regeln zur Identifizierung von Operatoren und Operanden ist in Kapitel 3.3 beschrieben. Eine Aufstellung

[c]Zyklomatische Zahl
[d]Intuitive Komplexität

aller ABAP/4-Schlüsselworte mit der genauen Angabe der Zählung der Operatoren und Operanden befindet sich im Anhang A.

Zum einen unterscheidet Halstead, wie McCabe auch, nicht zwischen verschiedenen Verzweigungen untereinander sowie zwischen Verzweigungen und unstrukturierten Konstrukten. Zum anderen wird in der Programmier-Leistung E die Modularisierung eines Programms nicht abgebildet. Bei der Messung der modularisierten ABAP/4-Programme werden die eindeutigen Größen η_1 und η_2 ermittelt. Sie beinhalten alle einmalig vorkommenden Operatoren und Operanden im Hauptprogramm und in den Modulen. Die Gesamtzahl der Operatoren N_1 und Operanden N_2 setzt sich aus der Anzahl im Hautprogramm und in den Modulen zusammen. Aus diesen vier ermittelten Größen wird die Programmier-Leistung E für das komplette modularisierte Programm berechnet. Es wird nicht für jedes einzelne Modul eine Programm-Leistung E_m bestimmt, um diese zu einer gesamten Programmier-Leistung E zusammenzufassen.

Das Ziel in diesem Kapitel ist, den Zusammenhang zwischen der intuitiven Einschätzung und den Ergebnissen der Messung nach Halstead für die Beispiel-Programme zu untersuchen. Auch hier besagt eine hohe Korrelation, daß die Programmier-Leistung die intuitive Bewertung bestätigt.

Ergebnisse der Komplexitätsmessung

In bezug auf die Strukturierung der Messungen stimmt Halstead mit McCabe überein. Die Programmier-Leistung E jedes Beispiel-Programms ist in der folgenden Tabelle aufgeführt. Eine vollständige Darstellung der Messungen befindet sich im Anhang C. Die Aufstellung enthält neben der eindeutigen Anzahl und der Gesamtzahl der Operatoren η_1, N_1 und Operanden η_2, N_2 alle berechneten Messungen nach Halstead. Dies sind u.a. die Programm-Länge N, die Programm-Größe V und die Programmier-Leistung E.

Validation von Programm-Kriterien

Die errechnete Programm-Länge $\widehat{N}$ und der Sprachlevel λ werden von Halstead validiert. Der errechneten Programm-Länge $\widehat{N}$ stellt er die beobachtete Programm-Länge N gegenüber und überprüft die eventuelle Übereinstimmung. Beim Sprachlevel λ mit $\lambda = L^2 \cdot V$ wird der Exponent 2 durch einen beliebigen Exponent b ersetzt. Es wird kontrolliert wie dicht b an 2 liegt.

Überprüfung der errechneten Programm-Länge $\widehat{N}$

Die errechnete Programm-Länge $\widehat{N}$ wird durch Anwendnung einer einfachen Gleichung aus dem Vokabular η ermittelt: $\widehat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$. In welchen Zusammenhang die Programm-Längen N und $\widehat{N}$ stehen, wird mit der Korrelationsrechung überprüft. Nach Einteilung der ABAP/4-Beispiel-Programme in 7 Klassen auf Basis der Programm-Länge N, werden die jeweiligen Mittelwerte und Standardabweichun-

Int.[e] Kompl.	Report	Prog.[f] Leist.	Report	Prog. Leist.	Report	Prog. Leist.
1	U-UM-002	268	S-UM-001	317	S-UM-006	697
	N-UM-001	231	S-UM-002	415	S-UM-008	1431
2	U-UM-001	302	N-UM-005	786	N-UM-009	1666
	N-UM-010	527	U-UM-003	847	S-M-001	2005
	N-UM-011	541	N-UM-004	605	S-M-002	2663
	N-UM-002-1	431	N-UM-016	1011	S-UM-011	3344
	N-UM-002-2	431	N-UM-006	1088	S-UM-010	4724
	S-UM-009	709	N-UM-003	1325		
	N-UM-002-3	520	S-M-003	1562		
3	U-UM-009	1085	N-UM-017	3206	S-UM-007	(9013)
	N-UM-008	2091	S-UM-004	3310	S-UM-005	(48064)
	N-UM-019	2515	N-UM-014	4249		
	N-M-001	2431	N-UM-012	4969		
4	U-UM-004	2109	N-UM-013	3025	N-M-002	6341
	U-UM-014	2316	U-UM-013	3421	S-UM-003	8521
	U-UM-011	2505	U-M-001	4218		
	U-M-002	2803	U-UM-010	4250		
5	U-UM-007	2763	N-UM-015	4999	U-UM-012	6988
	U-UM-006	3505	U-UM-008	5382	N-UM-018	10526
	N-UM-007	4056	U-UM-015	5839	N-UM-020	14462
6	U-UM-005	4909	N-M-003	11287		
7	U-M-003	11770	N-UM-021	21466		
8	U-M-004	28694				
9	U-M-005	44809				
10	U-M-006	24967				
11	U-M-007	30437				

Tabelle 3.4: Die Ergebnisse der Komplexitätsmessung nach Halstead pro intuitiver Komplexitätsklasse

gen berechnet.

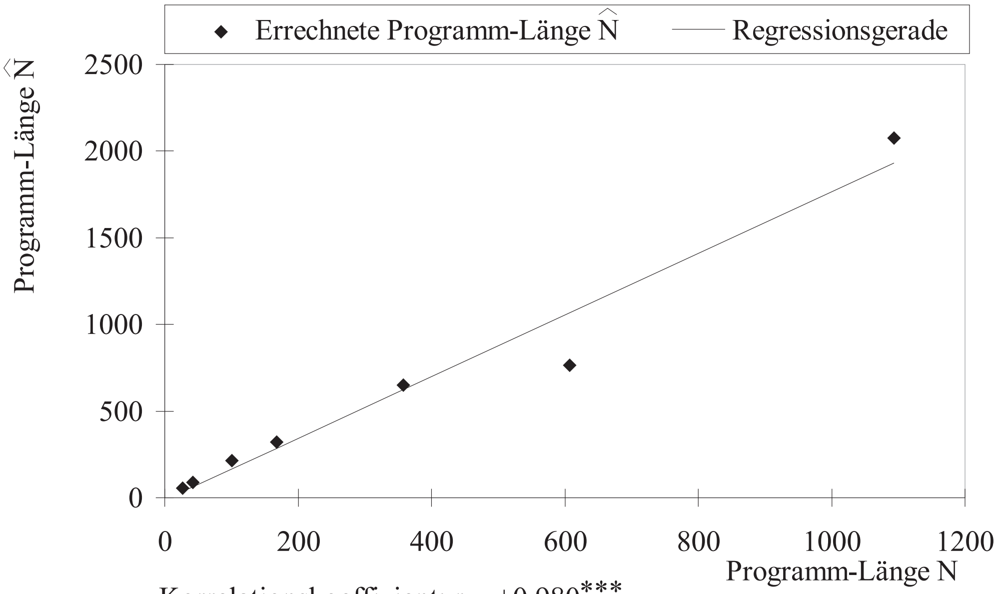
Der errechnete Korrelationskoeffizient r beträgt $+0,980^{***}$. Bei fünf Freiheitsgraden liegt die Irrtumswahrscheinlichkeit unter $0,1\%$. Das Ergebnis besagt, daß die Steigerung der errechneten Programm-Länge $\widehat{N}$ in bezug auf die beobachtete Programm-Länge N mit einer Irrtumswahrscheinlichkeit $< 0,1\%$ statistisch gesichert ist.

In fast allen Klassen erreicht die errechnete Programm-Länge $\widehat{N}$ der ABAP/4-Beispiel-Programme das Doppelte von der beobachteten Programm-Länge N. Wie

[e]Intuitive Komplexität
[f]Programmier-Leistung E

Klasse $2^i \leq N < 2^{i+1}$ i	Mittelwerte Programm-Länge N	Mittelwerte Programm-Länge $\widehat{N}$	Standardabweichung Programm-Länge N	Standardabweichung Programm-Länge $\widehat{N}$
4	26,5	55,2	3,5	15,8
5	42,6	88,2	11,0	37,2
6	100,5	215,8	18,8	45,7
7	167,6	319,8	31,3	67,0
8	357,7	650,9	68,8	179,1
9	607,0	765,0	33,8	411,1
10	1094,0	2076,0		

Tabelle 3.5: Beobachtete Programm-Länge N gegen errechnete Programm-Länge $\widehat{N}$ für ABAP/4-Programme



Korrelationskoeffizient: r = +0,980***
Regressionsgerade: $\widehat{N} = 1{,}77 \cdot N - 11{,}28$

Abbildung 3.11: Graphische Darstellung der Korrelation der Programm-Längen N und $\widehat{N}$

in Kapitel 2.3.1 beschrieben, setzt sich die Programm-Länge N aus der Gesamtanzahl von Operatoren N_1 und Operanden N_2 zusammen. Dies gilt auch für die berechnete Programm-Länge $\widehat{N}$: die Gesamtzahl der Operatoren $\widehat{N}_1 = \eta_1 \log_2 \eta_1$ und Operanden $\widehat{N}_2 = \eta_2 \log_2 \eta_2$ eines Programms können, wie angegeben, berechnet werden.

Eine differenzierte Untersuchung zwischen den beobachteten und errechneten Operatoren und Operanden zeigt, daß die Berechnung für Operatoren akzeptable Werte liefert. Die Abweichungen liegen zwischen 70% über und 40% unter den beobachteten Operatoren. Demgegenüber ist die berechnete Anzahl von Operanden $\widehat{N}_2$ in allen Klassen mindestens doppelt so groß. Die einzelnen Ergebnisse sind im Anhang C zu finden.

Klasse $2^i \leq N < 2^{i+1}$ i	Mittelwerte Programm-Länge N	Programm-Länge $\widehat{N}$	Differenz aus $N - \widehat{N}$	$\frac{N-\widehat{N}}{N}$
4	26,5	55,2	-28,7	-1,084
5	42,6	88,2	-45,5	-1,069
6	100,5	215,8	-115,3	-1,148
7	167,6	319,8	-152,2	-0,908
8	357,7	650,9	-293,2	-0,820
9	607,0	765,0	-158,0	-0,260
10	1094,0	2076,0	-981,5	-0,897

Tabelle 3.6: Differenzen zwischen der beobachteten Programm-Länge N und der errechneten Programm-Länge $\widehat{N}$

Die Erklärung dieser Resultate ist in einer Eigenschaft von ABAP/4 und in der Interpretation von Halstead zu finden, vgl. Kapitel 3.3. In ABAP/4 besteht die Möglichkeit Felder von deklarierten Datenbanksegmenten und SAP-Tabellen zu referenzieren, ohne die Felder explizit zu deklarieren. ABAP/4-Systemfelder[3] können ebenfalls referenziert angesprochen werden. Bei der Ermittlung der eindeutigen Anzahl η_2 und der Gesamtzahl N_2 der Operanden fließen alle referenzierten Felder ein. In der Regel werden derartige Felder nur ein- bis zweimal im Programm angesprochen. Demnach üben sie keinen großen Einfluß auf die Gesamtanzahl der Operanden N_2 aus. Ihre Wirkung bezüglich der errechneten Anzahl der Operanden $\widehat{N}_2$ ist dagegen umso größer.

Validation des Sprachlevel λ

Der Sprachlevel λ beschreibt die Beziehung zwischen dem Programm-Level L und der Potentiellen-Größe V^* und ist definiert durch: $\lambda = L^2V$. Er ist für jede Sprache konstant. Zur Validation wird der Exponent 2 durch den Exponent b ersetzt. Bei gegebenen λ kann der Exponent b berechnet werden. Er sollte dicht an 2 liegen.

Die ABAP/4-Beispiel-Programme sind in Klassen auf Basis der Programm-Größe V eingeteilt, um Differenzen in den Beispielen zu minimieren.

Der Mittelwert des Sprachlevels λ lautet 57,54. Die zugehörige Standardabweichung

[3]In Systemfeldern werden Informationen vom System zur Verfügung gestellt.

Klasse $2^i \leq V < 2^{i+1}$ i	Mittelwerte Programm-Größe V	Programm-Level L
6	110	0,44
7	165	0,34
8	378	0,36
9	751	0,26
10	1403	0,21
11	2893	0,17
12	4340	0,15

Tabelle 3.7: Mittelwerte der Programm-Größe V und des Programm-Level L

beträgt 31,59. Aufgrund der großen Streuung von λ gilt für den ermittelten Exponent $b = 3,38$. Der Sprachlevel λ für die Sprache ABAP/4 wäre bei der vorliegenden Interpretation experimentell bestätigt, wenn der Exponent b nah bei 2 gelegen hätte. Da dies nicht der Fall ist, sollte der Sprachlevel als Messung nicht eingesetzt werden.

Auswertung der Ergebnisse

Die Auswertung der Messung verläuft gleich der von McCabe in Kapitel 3.4.2: Die von Halstead aufgestellte Programmier-Leistung E wird als Komplexitätsmaß angesehen. Die Frage ist, ob die Programmier-Leistung die intuitive Einschätzung der Beispiel-Programme bestätigt.

Dabei wird vergleichbar wie bei McCabe untersucht, wie hoch die Korrelation zwischen der Programmier-Leistung und der intuitiven Bewertung ist. Die Interpretation von Halstead für die Programmiersprache ABAP/4 in Kapitel 3.1 und 2.3 gilt als Voraussetzung für die Komplexitätsmessung.

Die, durch die intuitive Komplexität erhaltene Klasseneinteilung stellt auch bei Halstead die Grundlage der Auswertung dar. Die Zusammenfassung der Programmier-Leistungen E pro Komplexitätsklasse ist der Ausgangspunkt für die anschließende Korrelationsrechnung. Mit deren Hilfe wird der Zusammenhang zwischen der intuitiven Komplexität und der Programmier-Leistung E geprüft. Die drei Beispiel U-M-005, U-M-006 und U-M-007 werden bei der Auswertung der Ergebnisse von Halstead nicht betrachtet, vgl. Kapitel 3.4.2.

Die intuitive Komplexität und die Programmier-Leistung E aus Tabelle 3.8 sind in der folgenden Abbildung graphisch aufbereitet.

Es läßt sich auch bei Halstead intuitiv schließen, daß die Programmier-Leistung E ein Komplexitätsmaß für ABAP/4-Programme darstellt. Dies wird mittels der Korrelationsrechnung überprüft. Sie sagt aus, wie hoch der Zusammenhang zwischen der intuitiven Komplexität und der Programmier-Leistung E ist. Die Probenzahl

Intuitive Kompl.	Mittelwert Prog.-Leist.	Standard-abweichung	Intuitive Komplex.	Mittelwert Prog.-Leist.	Standard-abweichung
1	560	459	7	16618	6856
2	1320	1159	8	28694	
3	2982	1232	9	(44809)	
4	3951	2038	10	(24967)	
5	6502	3753	11	(30437)	
6	8098	4510			

Tabelle 3.8: Mittelwert der Programmier-Leistung E nach Halstead pro intuitiver Komplexitätsklasse

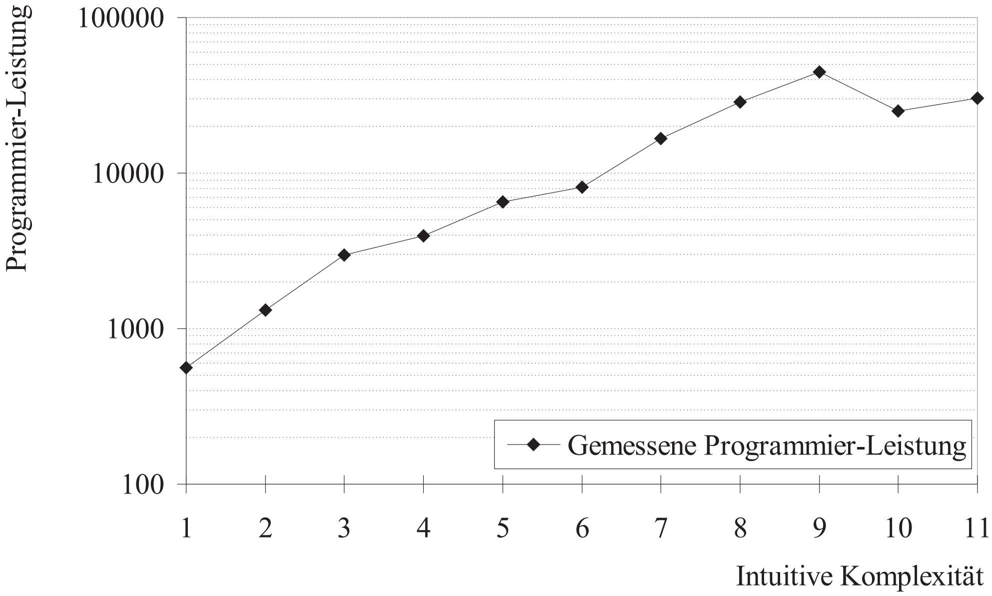


Abbildung 3.12: Graphische Darstellung der Programmier-Leistung E pro intuitiver Komplexitätsklasse

umfaßt die ersten acht Komplexitätsklassen, der resultierende Korrelationskoeffizient r ist $+0,886^{**}$. Bei sechs Freiheitsgraden liegt die Irrtumswahrscheinlichkeit P bei $< 1\%$.

Das Ergebnis der Korrelationsrechnung besagt, daß es sich bei der Zunahme der Programmier-Leistung E um eine echte Steigerung handelt. Sie ist über die acht untersuchten Komplexitätsklassen mit einer Irrtumswahrscheinlichkeit P unter 1% statistisch gesichert.

Die Programmier-Leistung E bestätigt die intuitive Einschätzung der Beispiel-Programme. Die Voraussetzung ist die Interpretation von Halstead für die Programmiersprache ABAP/4. Auch die Meßverfahren von Halstead kann zur Messung der Komplexität von ABAP/4-Programmen eingesetzt werden.

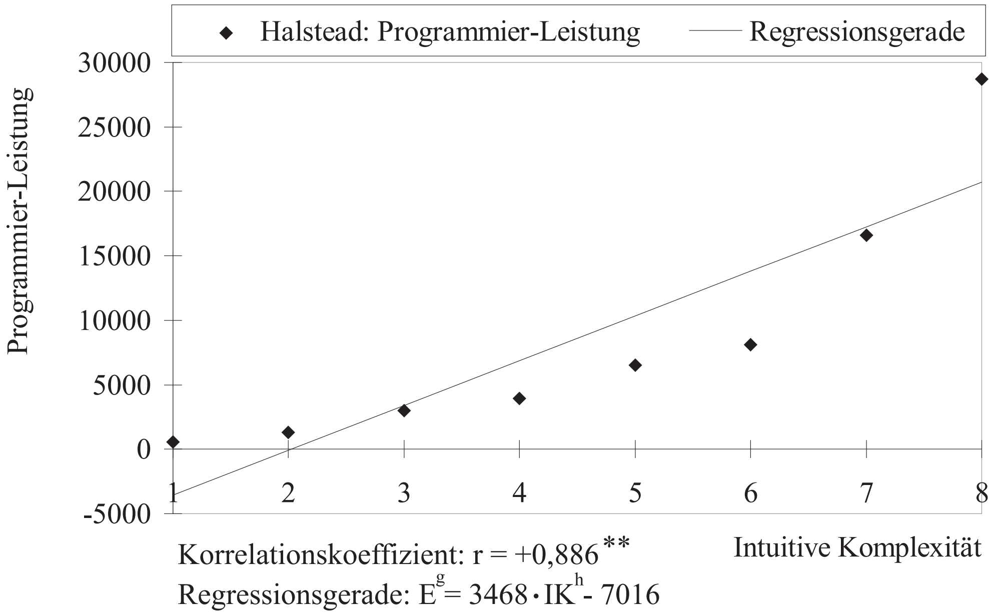


Abbildung 3.13: Graphische Darstellung der Korrelation der intuitiven Komplexität und der Programmier-Leistung E

Die Programmier-Leistung E kann für die Beispiel-Auswahl auch aus der intuitiven Komplexität ermittelt werden. Die Regressionsgerade, die die Beziehung beschreibt, lautet: $E = 3468 \cdot \mathrm{IK} - 7016$.

3.4.4 Zusammenhang zwischen McCabe und Halstead

Die Meßverfahren von McCabe und Halstead sind als Komplexitätsmaß für ABAP/4-Programme geeignet. Die Untersuchungen der beiden Methoden bestätigten jeweils die intuitiven Einschätzungen der ABAP/4-Programme, vgl. Kapitel 3.4.2 und 3.4.3. Die Verfahren bilden jeweils unterschiedliche Komplexitätskategorien ab: die zyklomatische Zahl von McCabe mißt die strukturelle Komplexität; die Programmier-Leistung E von Halstead ist ein Maß für die berechende Komplexität. Beide Kategorien fließen u.a. in die aufgestellte intuitive Komplexität ein.

Aufgrund der Ergebnisse der Auswertung treten folgende Fragen auf: Was für ein

[g] Programmier-Leistung E
[h] Intuitive Komplexität

Zusammenhang besteht zwischen den Komplexitätsmaßen von McCabe und Halstead? Inwieweit werden ABAP/4-Programme von der zyklomatische Zahl und der Programmier-Leistung E gleich eingeordnet?

Einfache Korrelation zwischen McCabe und Halstead

Die Abhängigkeit zwischen der zyklomatischen Zahl und der Programmier-Leistung E wird mittels einer einfachen Korrelation geschätzt. Aus der folgenden Tabelle, die sich aus den Tabellen 3.3 und 3.8 gewonnen ist, wird der Korrelationkoeffizient berechnet. Die drei Beispiel-Programme U-M-005, U-M-006 und U-M-007 bleiben bei der Berechnung unberücksichtigt.

Intuitive Komplexität	Zyklomatische Zahl	Programmier-Leistung E
1	1,5	560
2	2,8	1320
3	3,7	2982
4	6,7	3951
5	8,1	6502
6	13,5	8098
7	14,0	16618
8	23,0	28694
9	(55,0)	(44809)
10	(77,0)	(24967)
11	(89,0)	(30437)

Tabelle 3.9: Zyklomatische Zahl nach McCabe und Programmier-Leistung E nach Halstead

Die Probenzahl n beträgt 8. Der Korrelationskoefizient r wird auf $+0,957^{***}$ geschätzt. Mit einer Irrtumswahrscheinlichkeit unter $0,1\%$ ist die Steigerung der Programmier-Leistung E bei einer Zunahme der zyklomatischen Zahl statistisch gesichert. Dies gilt auch für den umgekehrten Fall. Schlußfolgerung: Die Meßverfahren ordnen die ABAP/4-Programme gleich ein.

Sowohl die zyklomatische Zahl als auch die Programmier-Leistung E sind keine festen Einflußgrößen wie es die intuitive Komplexität in den Auswertungen, Kapitel 3.4.2 und 3.4.3, ist. Folglich sind zwei Regressionen möglich: die zyklomatische Zahl kann aus der Programm-Leistung E geschätzt werden und umgekehrt. Die Regressionsgeraden lauten: $E = 1262 \cdot \mathrm{ZZ} - 2973$ und $\mathrm{ZZ} = 7,257 \cdot 10^{-4} + 2.928$.

Partielle Korrelation

[i] Programmier-Leistung E
[j] Zyklomatische Zahl

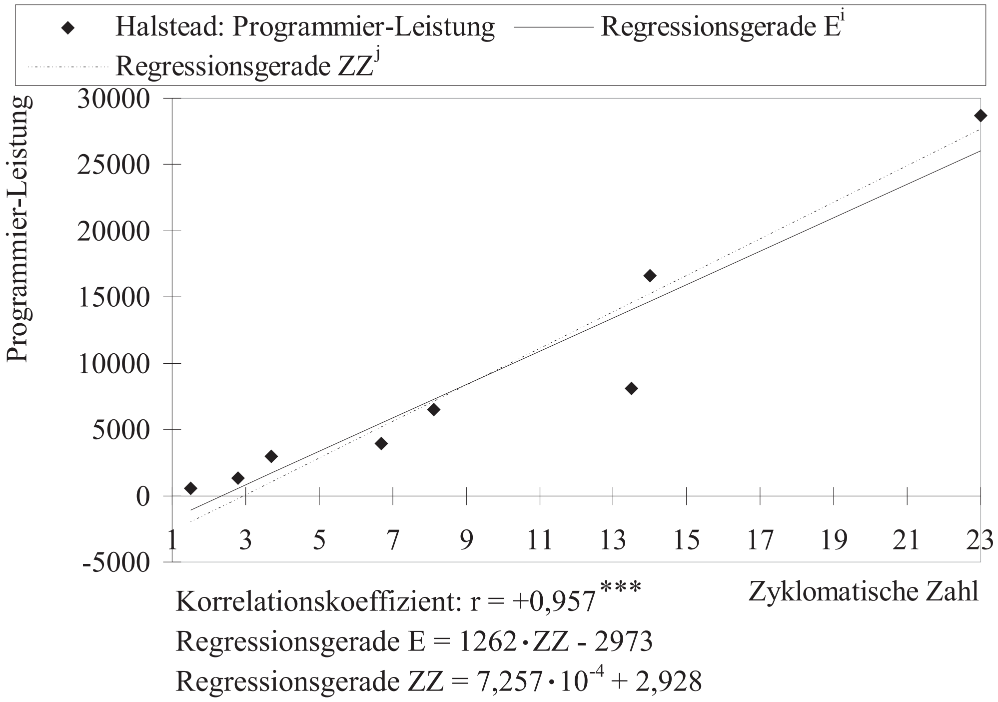


Abbildung 3.14: Graphische Darstellung der Korrelation der zyklomatischen Zahl und der Programmier-Leistung E

Die intuitive Komplexität beeinflußt die Korrelationen zwischen der zyklomatischen Zahl und der Programm-Leistung E. Um den tatschlichen Zusammenhang zwischen den beiden Komplexitätsmaßen zu erhalten, wird der partielle Korrelationskoeffizient bestimmt. Dieser gibt den Grad der Abhängigkeit der beiden Maßen an. Die intuitive Komplexität wird konstant gehalten und deren Einfluß ausgeschaltet. Da die oben genannten Größen mit steigender intuitiven Komplexität zunehmen, ergibt sich die Frage, ob der starke Zusammenhang nur auf die intuitive Komplexitat zurückzuführen ist oder ob bei jeder Komplexitätsklasse ein echter Zusammenhang existiert. Der partielle Korrelationskoeffizient lautet: $r = +0,800^{*}$. Bei fünf Freiheitsgraden ist der Zusammenhang mit einer Irrtumswahrscheinlichkeit unter 5% gesichert.

Die Beziehung der Komplexitätsmaße von McCabe und Halstead sind für ABAP/4-Programme auf dem 5%-Niveau statistisch gesichert. Dies gilt unter der Voraussetzung, daß der Einfluß der intuitiven Komplexität eliminiert ist.

3.4.5 Konsequenz

Die Komplexitätsmessungen von McCabe und Halstead bestätigen jeweils die intuitive Bewertung der ABAP/4-Programme. Die zyklomatische Zahl und die Programmier-Leistung können zur Messung der Komplexität eingesetzt werden. Voraussetzungen sind die einfachte ABAP/4-Ablaufsteuerung und die beschriebenen Interpretationen für die Programmiersprache ABAP/4. Die Validation erfolgte bisher nur für Schulungsprogramme. Eine Untersuchung für große Programme steht nun aus. Dies kann in der Diplomarbeit nicht erbracht werden. Das Verhalten der Messungen am Beispiel der alten und neuen Version der Lieferscheinerstellung wird analysiert. Die Überprüfung zeigt, ob die Meßverfahren die intuitive Einschätzung beiden Fassungen bestätigen.

Der partielle Korrelationskoeffizient zwischen der zyklomatischen Zahl und der Programmier-Leistung weist auf eine starke Beziehung zwischen den Komplexitätsmaßen hin. Das bedeutet, daß die Messungen ABAP/4-Programme gleich relativ zueinander einordnen. Eine Ausnahme sind sequentielle Programme, da das Verfahren von McCabe diese nicht abbilden kann. Diese Kenntnis wird für den Vorschlag einer erweiterten Komplexitätsmessung im Kapitel 4 ausgenutzt.