

Software Engineering in der Praxis

Praktische Übungen

Software-Metriken

David Föhrweiser Marc Spisländer

Lehrstuhl für Software Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

- 1 Inhalt

- 2 Software-Metriken
 - Definition
 - Allgemeine Betrachtungen zum Messen

- 3 Software-Metriken mit Together
 - Lines of Code
 - Halstead-Metrik
 - McCabe-Metrik
 - Messen mit Together

- 4 Zusatz: Findbugs als Codeanalyse-Werkzeug

Ab heute betrachten wir eine tiefere Abstraktionsstufe:
Code bzw. Implementierung

Software-Metriken

Definition

»Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.« (IEEE Standard 1061, 1992)

Zweck

Software messen, um sie vergleichbar zu machen, um:

- Qualität zu erhöhen
- Aufwand/Kosten abzuschätzen

Software-Metriken

Definition

»Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit.« (IEEE Standard 1061, 1992)

Zweck

Software messen, um sie vergleichbar zu machen, um:

- Qualität zu erhöhen
- Aufwand/Kosten abzuschätzen

Allgemeine Betrachtungen zum Messen

Kontext: Reale Welt

- Menge von Objekten
 - Jedes Objekt verfügt über gewisse Attribute
 - Zum Beispiel:
 - Menge von Menschen mit Attributsfunktion »Alter«
 - Menge von Software mit Attributsfunktion »Lines of Code«
 - Menge von Software mit Attributsfunktion »Qualität«
 - Menge von Software mit Attributsfunktion »Komplexität«

Allgemeine Betrachtungen zum Messen

Kontext: Reale Welt

- Menge von Objekten
- Jedes Objekt verfügt über gewisse Attribute
- Zum Beispiel:
 - Menge von Menschen mit Attributsfunktion »Alter«
 - Menge von Software mit Attributsfunktion »Lines of Code«
 - Menge von Software mit Attributsfunktion »Qualität«
 - Menge von Software mit Attributsfunktion »Komplexität«

Allgemeine Betrachtungen zum Messen

Kontext: Reale Welt

- Menge von Objekten
- Jedes Objekt verfügt über gewisse Attribute
- Zum Beispiel:
 - Menge von Menschen mit Attributsfunktion »Alter«
 - Menge von Software mit Attributsfunktion »Lines of Code«
 - Menge von Software mit Attributsfunktion »Qualität«
 - Menge von Software mit Attributsfunktion »Komplexität«

Messen

Messen bedeutet die experimentelle Bestimmung von Attributwerten.

Modell der realen Welt

Zur systematischen, wissenschaftlichen Untersuchung: Abbildung der realen Objekte auf mathematische Objekte (Mengen, Funktionen, Relationen, ...).

Kontext: Mathematische Welt

- Menge M von Objekten
- Attributfunktion $A : M \rightarrow W_A$
- Messfunktion $m_A : M \rightarrow W_A$ (Modelliert das Messverfahren und die Messergebnisse)

Korrekte Messung

Frage

- Wird das Richtige gemessen, d. h. ist die Messung korrekt?
- Bzw. gilt $A = m_A$?
- Oder abgeschwächer: Korrelieren A und m_A ?

Schwierigkeit

- Oft kein Konsens über die Definition der Attributsfunktion
- Beispiele: Qualität oder Komplexität von Software, Intelligenzquotient, ...

Korrekte Messung

Frage

- Wird das Richtige gemessen, d. h. ist die Messung korrekt?
- Bzw. gilt $A = m_A$?
- Oder abgeschwächer: Korrelieren A und m_A ?

Schwierigkeit

- Oft kein Konsens über die Definition der Attributsfunktion
- Beispiele: Qualität oder Komplexität von Software, Intelligenzquotient, ...

Korrekte Messung

Frage

- Wird das Richtige gemessen, d. h. ist die Messung korrekt?
- Bzw. gilt $A = m_A$?
- Oder abgeschwächer: Korrelieren A und m_A ?

Schwierigkeit

- Oft kein Konsens über die Definition der Attributsfunktion
- Beispiele: Qualität oder Komplexität von Software, Intelligenzquotient, ...

Korrekte Messung

Frage

- Wird das Richtige gemessen, d. h. ist die Messung korrekt?
- Bzw. gilt $A = m_A$?
- Oder abgeschwächer: Korrelieren A und m_A ?

Schwierigkeit

- Oft kein Konsens über die Definition der Attributfunktion
- Beispiele: Qualität oder Komplexität von Software, Intelligenzquotient, ...

Korrekte Messung

Frage

- Wird das Richtige gemessen, d. h. ist die Messung korrekt?
- Bzw. gilt $A = m_A$?
- Oder abgeschwächer: Korrelieren A und m_A ?

Schwierigkeit

- Oft kein Konsens über die Definition der Attributsfunktion
- Beispiele: Qualität oder Komplexität von Software, Intelligenzquotient, ...

Definition von Attributsfunktionen

Mögliche Auswege

- Definiere Attributsfunktion A durch Messfunktion m_A :

$$A := m_A$$

Korrektheit der Messung per Definition gegeben

- Untersuche Korrelation zwischen A und m_A :
 - Unterstelle jedem Menschen eine *intuitive* Vorstellung von A
 - Definiere m_A
 - Prüfe durch statistischen Test, ob m_A und die »intuitive Funktion« A korrelieren

Definition von Attributsfunktionen

Mögliche Auswege

- Definiere Attributsfunktion A durch Messfunktion m_A :

$$A := m_A$$

Korrektheit der Messung per Definition gegeben

- Untersuche Korrelation zwischen A und m_A :
 - Unterstelle jedem Menschen eine *intuitive* Vorstellung von A
 - Definiere m_A
 - Prüfe durch statistischen Test, ob m_A und die »intuitive Funktion« A korrelieren

Beispiel *Lines of Code*

Definition der Attributsfunktion

Für alle Programme p :

$$A(p) := \text{»Anzahl der Zeilen von } p\text{«}$$

Definition der Messfunktion

Definition der Messfunktion m_A durch Algorithmus:

- Eingabe: Programm p
- Berechnung: Zählen der Zeilenendmarkierungen in p
- Ausgabe: Anzahl der Zeilen in p

Beispiel *Lines of Code*

Definition der Attributsfunktion

Für alle Programme p :

$$A(p) := \text{»Anzahl der Zeilen von } p\text{«}$$

Definition der Messfunktion

Definition der Messfunktion m_A durch Algorithmus:

- Eingabe: Programm p
- Berechnung: Zählen der Zeilenendmarkierungen in p
- Ausgabe: Anzahl der Zeilen in p

Beispiel *Lines of Code*

Korrektheit der Messung

Durch mathematischen Beweis folgerbar:

$$A = m_a$$

Beispiel *Software-Komplexität*

Problem

Keine allgemein anerkannte, genaue Definition der Attributsfunktion *Software-Komplexität*.

Hoffnung

- Intuitive Vorstellung der Attributsfunktion *Software-Komplexität* korreliert mit einer Messfunktion (z. B. *Lines of Code*)
- Nur durch statistischen Test (Experiment) feststellbar

Beispiel *Software-Komplexität*

Problem

Keine allgemein anerkannte, genaue Definition der Attributsfunktion *Software-Komplexität*.

Hoffnung

- Intuitive Vorstellung der Attributsfunktion *Software-Komplexität* korreliert mit einer Messfunktion (z. B. *Lines of Code*)
- Nur durch statistischen Test (Experiment) feststellbar

Beispiel *Software-Komplexität*

Problem

Keine allgemein anerkannte, genaue Definition der Attributsfunktion *Software-Komplexität*.

Hoffnung

- Intuitive Vorstellung der Attributsfunktion *Software-Komplexität* korreliert mit einer Messfunktion (z. B. *Lines of Code*)
- Nur durch statistischen Test (Experiment) feststellbar

Software-Metriken mit Together

Das Werkzeug Together bietet verschiedene Metriken an, um Software zu messen. Für das Praktikum wichtig:

- Lines of Code (LOC)
- Halstead Program Length (HPLen)
- Halstead Program Volume (HPVol)
- Cyclomatic Complexity nach McCabe (CC)

LOC (Anzahl der Code-Zeilen)

Vorteile

- Einfach zu berechnen
- Einfach nachzuvollziehen

Nachteile

- Keine hohe Aussagekraft
- Abhängig vom Programmierstil

LOC (Anzahl der Code-Zeilen)

Vorteile

- Einfach zu berechnen
- Einfach nachzuvollziehen

Nachteile

- Keine hohe Aussagekraft
- Abhängig vom Programmierstil

Halstead-Metrik

- Betrachte Programm als Wort über zwei disjunkte Alphabete:

Σ_1 Operatoren-Alphabet (Schlüsselwörter, +, -, Funktionenbezeichner, ...)

Σ_2 Operanden-Alphabet (Variablen, Literale)

- Definiere dann:

N_1 := »Gesamtanzahl der vorkommenden Operatoren«

N_2 := »Gesamtanzahl der vorkommenden Operanden«

N := $N_1 + N_2$ (Programmlänge $HPLen$)

n := $|\Sigma_1| + |\Sigma_2|$ (Programmwortschatz $HPVoc$)

V := $N \log n$ (Programmvolumen $HPVol$)

Halstead-Metrik

- Betrachte Programm als Wort über zwei disjunkte Alphabete:

Σ_1 Operatoren-Alphabet (Schlüsselwörter, +, -, Funktionenbezeichner, ...)

Σ_2 Operanden-Alphabet (Variablen, Literale)

- Definiere dann:

N_1 := »Gesamtanzahl der vorkommenden Operatoren«

N_2 := »Gesamtanzahl der vorkommenden Operanden«

N := $N_1 + N_2$ (Programmlänge *HPLen*)

n := $|\Sigma_1| + |\Sigma_2|$ (Programmwortschatz *HPVoc*)

V := $N \log n$ (Programmvolumen *HPVol*)

Halstead-Metrik (Fortsetzung)

Vorteile

- Einfach zu berechnen
- Indikator für Fehleranfälligkeit und Wartungsaufwand
- Robust in der Wahl der Programmiersprache

Nachteile

- Nur auf vollständigen Code anwendbar
- Weiterführende Halstead-Metriken, wie *Mentaler Aufwand* fragwürdig

Halstead-Metrik (Fortsetzung)

Vorteile

- Einfach zu berechnen
- Indikator für Fehleranfälligkeit und Wartungsaufwand
- Robust in der Wahl der Programmiersprache

Nachteile

- Nur auf vollständigen Code anwendbar
- Weiterführende Halstead-Metriken, wie *Mentaler Aufwand* fragwürdig

McCabe-Metrik

- Annahme: Programm hat genau einen Eintritts- und genau einen Austrittspunkt (erfüllt beim strukturellen Programmieren)
- Aufbau des Kontrollflussgraphen G
 - Anzahl der Kanten: e
 - Anzahl der Knoten: n
- *Zyklomatische Komplexität:*

$$v[G] = e - n + 2 \quad (CC)$$

McCabe-Metrik

- Annahme: Programm hat genau einen Eintritts- und genau einen Austrittspunkt (erfüllt beim strukturellen Programmieren)
- Aufbau des Kontrollflussgraphen G
 - Anzahl der Kanten: e
 - Anzahl der Knoten: n
- *Zyklomatische Komplexität:*

$$v[G] = e - n + 2 \quad (CC)$$

McCabe-Metrik

- Annahme: Programm hat genau einen Eintritts- und genau einen Austrittspunkt (erfüllt beim strukturellen Programmieren)
- Aufbau des Kontrollflussgraphen G
 - Anzahl der Kanten: e
 - Anzahl der Knoten: n
- *Zyklomatische Komplexität:*

$$v[G] = e - n + 2 \quad (CC)$$

McCabe-Metrik (Fortsetzung)

Eigenschaften

- $\nu[G]$ ist die maximale Anzahl linear unabhängiger Zyklen im *erweiterten Kontrollflussgraphen* (= Kontrollflussgraph + Kante vom Austrittspunkt zum Eintrittspunkt).
- Verfeinerung unter folgender Annahme:
 - Nur vier Klassen von Knoten im Kontrollflussgraphen:
 - x Funktionsknoten Knoten, die Berechnung ausführen
 - y Sammelknoten Knoten, die Verzweigung zusammenführen
 - z Prädikatenknoten Knoten mit 2 Ausgangskanten
 - 2 Zusatzknoten Eingangs- bzw. Ausgangsknoten
 - Dann gilt:

$$\nu[G] = e - n + 2 = (1 + x + y + 2z) - (x + y + z + 2) + 2 = z + 1$$

McCabe-Metrik (Fortsetzung)

Eigenschaften

- $\nu[G]$ ist die maximale Anzahl linear unabhängiger Zyklen im *erweiterten Kontrollflussgraphen* (= Kontrollflussgraph + Kante vom Austrittspunkt zum Eintrittspunkt).
- Verfeinerung unter folgender Annahme:
 - Nur vier Klassen von Knoten im Kontrollflussgraphen:
 - x Funktionsknoten Knoten, die Berechnung ausführen
 - y Sammelknoten Knoten, die Verzweigung zusammenführen
 - z Prädikatenknoten Knoten mit 2 Ausgangskanten
 - 2 Zusatzknoten Eingangs- bzw. Ausgangsknoten
 - Dann gilt:

$$\nu[G] = e - n + 2 = (1 + x + y + 2z) - (x + y + z + 2) + 2 = z + 1$$

McCabe-Metrik (Fortsetzung)

Eigenschaften

- $\nu[G]$ ist die maximale Anzahl linear unabhängiger Zyklen im *erweiterten Kontrollflussgraphen* (= Kontrollflussgraph + Kante vom Austrittspunkt zum Eintrittspunkt).
- Verfeinerung unter folgender Annahme:
 - Nur vier Klassen von Knoten im Kontrollflussgraphen:
 - x Funktionsknoten Knoten, die Berechnung ausführen
 - y Sammelknoten Knoten, die Verzweigung zusammenführen
 - z Prädikatenknoten Knoten mit 2 Ausgangskanten
 - 2 Zusatzknoten Eingangs- bzw. Ausgangsknoten
 - Dann gilt:

$$\nu[G] = e - n + 2 = (1 + x + y + 2z) - (x + y + z + 2) + 2 = z + 1$$

McCabe-Metrik (Fortsetzung)

Vorteile

- Indikator für Wartungsaufwand
- Anhaltspunkt für die Zahl benötigter unabhängiger Testfälle für Kontrollflussabdeckung
- Früh in der Entwicklung einsetzbar

Nachteile

- Keine Datenkomplexität
- Gleiche Gewichtung geschachtelter wie ungeschachtelter Schleifen

McCabe-Metrik (Fortsetzung)

Vorteile

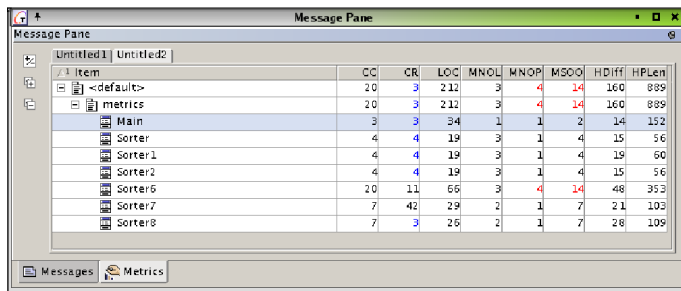
- Indikator für Wartungsaufwand
- Anhaltspunkt für die Zahl benötigter unabhängiger Testfälle für Kontrollflussabdeckung
- Früh in der Entwicklung einsetzbar

Nachteile

- Keine Datenkomplexität
- Gleiche Gewichtung geschachtelter wie ungeschachtelter Schleifen

Auswertungsmöglichkeiten

Zunächst eine Tabelle...



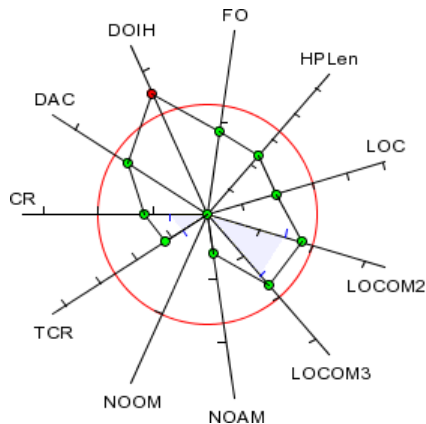
Item	CC	CR	LOC	MNOL	MNOP	MSOO	HDiff	HPLen
<default>	20	3	212	3	4	14	160	889
metrics	20	3	212	3	4	14	160	889
Main	3	3	34	1	1	2	14	152
Sorter	4	4	19	3	1	4	15	56
Sorter1	4	4	19	3	1	4	19	60
Sorter2	4	4	19	3	1	4	15	56
Sorter5	20	11	66	3	4	14	48	353
Sorter7	7	42	29	2	1	7	21	103
Sorter8	7	3	26	2	1	7	28	109

Auswertungsmöglichkeiten (Fortsetzung)

- Farbliche Kennzeichnung der Ergebnisse:
 - blau: Wert liegt unter einer gegebenen unteren Schranke
 - rot: Wert liegt über einer gegebenen oberen Schranke
 - schwarz: Wert liegt zwischen den Schranken
- Aggregation der Werte
 - Zur Klasse
 - Zum Package
- Kiviat-Diagramme

Kiviat-Diagramme

Mehrere Werte werden sternförmig aufgetragen:



Kiviat-Diagramme (Fortsetzung)

- Richtwerte bestimmen
 - Vom Tool vorgeschlagen
 - Einstellbar
 - Maximum und Minimum
- Maßzahlen normieren
- Im Graphen eintragen
 - Richtwerte als Kreise
 - Metriken als Strahlen
 - Werte des Betrachteten Moduls als Polygon

Findbugs als Codeanalyse-Werkzeug

- Zusätzlich zu herkömmlichen Metriken erlaubt das Werkzeug *Findbugs* die Suche nach Bugpattern
- Nützlich, um nach bestimmten Kriterien die Qualität des Codes zu erhöhen
- Zum Beispiel: Einhalten von Codingstandards, Streams nach dem Öffnen schließen, ...