

Universität Karlsruhe (TH)

Forschungsuniversität · gegründet 1825

Kapitel 5.4

Selbstkontrolliertes Programmieren

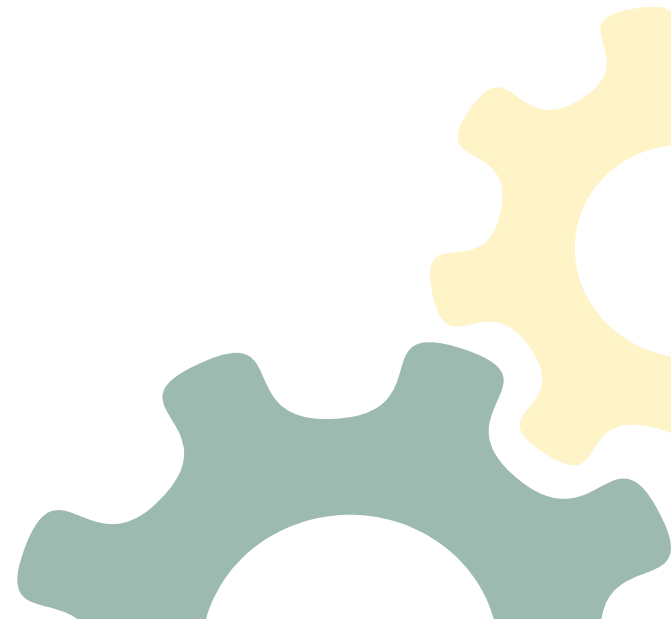
SWT I – Sommersemester 2009

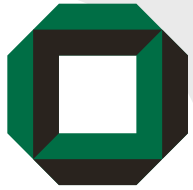
Prof. Dr. Walter F. Tichy



Fakultät für Informatik

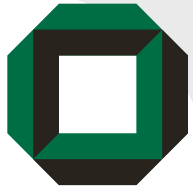
Lehrstuhl für Programmiersysteme





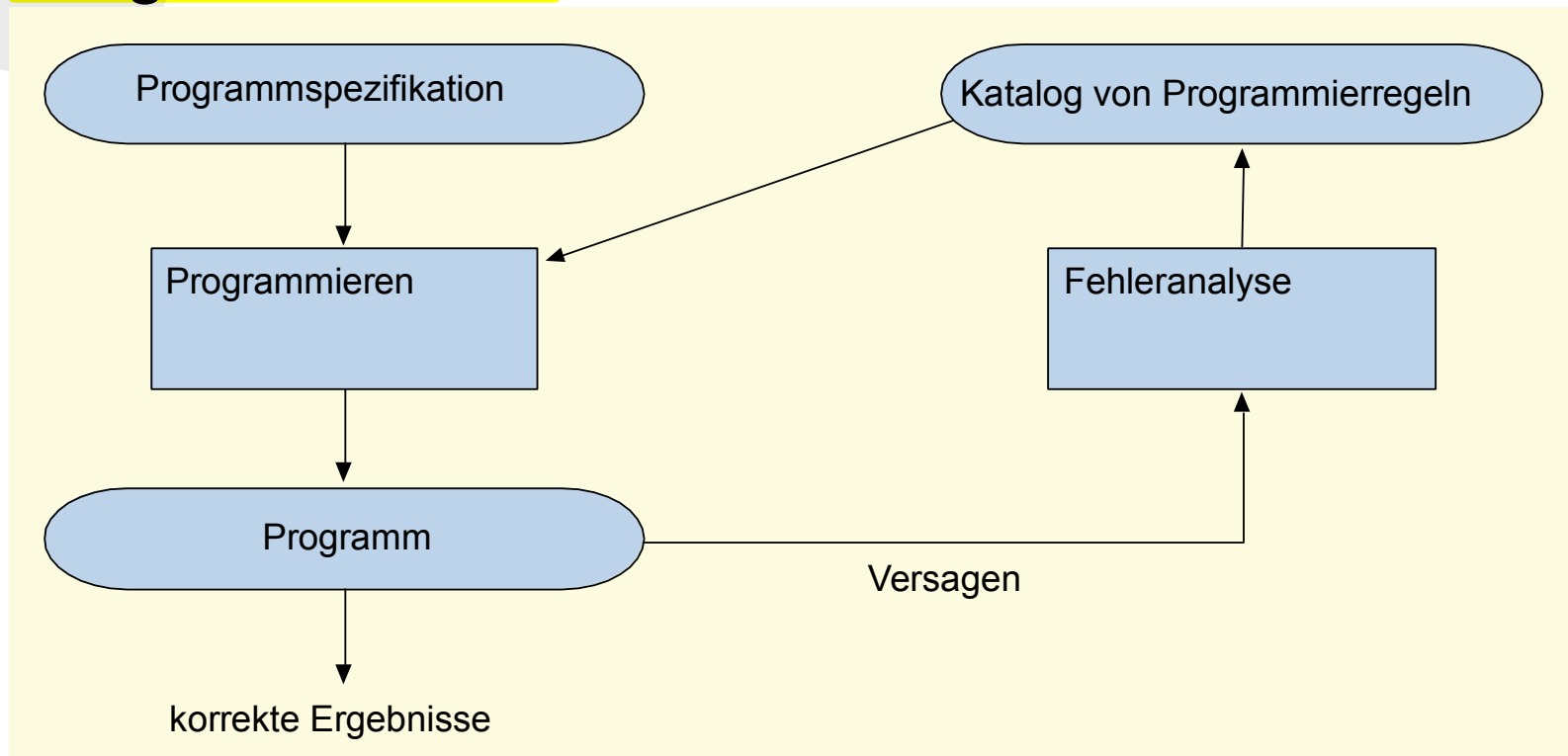
Selbstkontrolliertes Programmieren

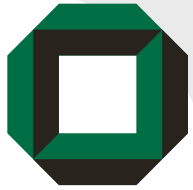
- Der erfahrene Programmierer lernt aus seinen Fehlern
 - Er hält sich an Regeln und Vorschriften, die der Vermeidung dieser Fehler dienen
 - In diesem Sinne ist das Programmieren eine Erfahrungswissenschaft
 - Jeder Programmierer sollte für sich einen Katalog von Programmierregeln aufstellen und fortlaufend verbessern
 - Es entsteht ein Regelkreis des selbstkontrollierten Programmierens.



Selbstkontrolliertes Programmieren

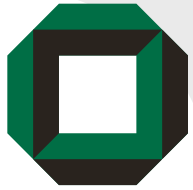
- Der Regelkreis des selbstkontrollierten Programmierens





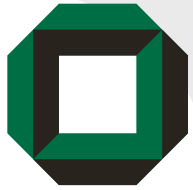
Typische Programmierfehler (1)

- Ausnahme- und Grenzfälle
 - Vorzugsweise werden nur die Normalfälle behandelt
 - Sonderfälle und Ausnahmen werden übersehen
 - Es werden nur die Fälle erfasst, die man für repräsentativ hält
- Beispiele:
 - „Um eins daneben“-Fehler
 - Indexzählfehler
 - Null-Zeiger-Zugriff



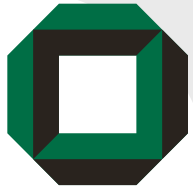
Typische Programmierfehler (2)

- Falsche Hypothesen
 - Erfahrene Programmierer haben Faustregeln entwickelt, sich einfache Modelle über die Arbeitsweise eines Rechners zurechtgelegt
 - Aber: Dieses Wissen veraltet, mit einer Änderung der Umwelt werden diese Regeln falsch.
 - Beispiele für überholte Regeln:
 - Multiplikationen dauern wesentlich länger als Additionen
 - Potenzieren ist aufwendiger als Multiplizieren
 - Cache Effekte sind unwichtig.



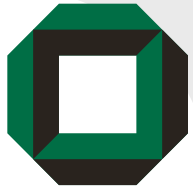
Typische Programmierfehler (3)

- Tücken der Maschinendarithmetik
 - Sonderfall der falschen Regeln
 - Der Computer hält sich nicht an die Regeln der Algebra und Analysis
 - Reelle Zahlen werden nur mit begrenzter Genauigkeit dargestellt
 - Beispiele:
 - Gleichheit reeller Zahlen so richtig:
 - $\text{abs}(a - b) < \text{epsilon}$;
 - Aufsummierung von Reihen: Summanden werden so klein, dass ihre Beträge bei der Rundung verlorengehen. Negative Summanden führen zu „Auslöschung“, d.h. Verlust von Genauigkeit.



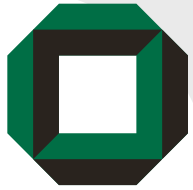
Typische Programmierfehler (4)

- Irreführende Namen
 - Wahl eines Namens, der eine falsche Semantik vortäuscht
 - Daraus kann sich eine fehlerhafte Anwendung ergeben
- Unvollständige Bedingungen
 - Das konsequente Aufstellen komplexer logischer Bedingungen fällt schwer



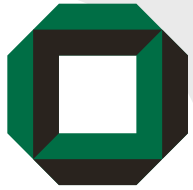
Typische Programmierfehler (5)

- Unverhoffte Variablenwerte
 - Die Komplexität von Zusammenhängen wird nicht erfasst
 - Beispiele:
 - Verwechslung global wirksamer Größen mit Hilfsgrößen
 - Falsche oder vergessene Initialisierung von Variablen.



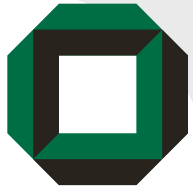
Typische Programmierfehler (6)

- Wichtige Nebensachen
 - Die Unterteilung von Programmen in sicherheitskritische und sicherheitsunkritische Teile führt oft zur Vernachlässigung der »Nebensachen«
 - Dadurch entstehen Programme mit »abgestufter Qualität«, wobei Fehler in den »Nebensachen« oft übersehen werden



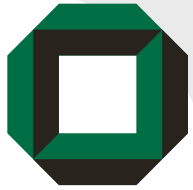
Typische Programmierfehler (7)

- Trügerische Redundanz
 - Durch achtloses Kopieren werden Strukturen geschaffen, die den menschlichen Denkapparat überfordern
 - Übertriebene Kommentierung von Programmen erhöht die Redundanz
 - Beispiele:
 - Weiterentwicklung eines Programms geschieht nicht an der aktuellen Version, sondern an einem Vorläufer
 - Bei Programmänderungen wird vergessen, den Kommentar nachzuführen.



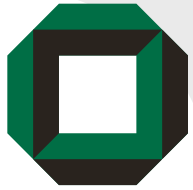
Selbstkontrolliertes Programmieren

- Das Lernen aus Fehlern ist besonders gut geeignet, um Denkfallen zu vermeiden
 - Im Laufe des Anpassungsprozesses wird der „Scheinwerfer der Aufmerksamkeit“ auf die kritischen Punkte gerichtet
 - Jeder Programmierer sollte einen Regelkatalog in einer Datei bereithalten und an seine Bedürfnisse anpassen
 - Das Lernen aus Fehlern wird außerdem dadurch gefördert, dass man die Fehler dokumentiert
 - Es sollte ein Fehlerbuch angelegt werden, das eine Sammlung typischer Fehler enthält.



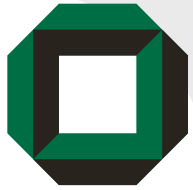
Selbstkontrolliertes Programmieren

- Ein Fehler sollte in ein Fehlerbuch eingetragen werden, wenn...
 - die Fehlersuche lange gedauert hat
 - die durch den Fehler verursachten Kosten hoch waren oder
 - der Fehler lange unentdeckt geblieben ist.



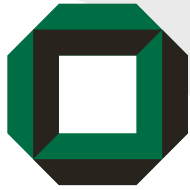
Selbstkontrolliertes Programmieren

- Folgende Daten sollten im Fehlerbuch erfasst werden:
 - Laufende Fehlernummer
 - Datum (wann entstanden, wann entdeckt)
 - Aus welcher Phase (Anforderung, Entwurf, Implementierung)
 - Fehlerkurzbeschreibung (Titel)
 - Ursache (Verhaltensmechanismus)



Selbstkontrolliertes Programmieren

- Rückverfolgung
 - Gab es schon Fehler derselben Sorte?
 - Warum war eine früher vorgeschlagene Gegenmaßnahme nicht wirksam?
 - Programmierregel, Gegenmaßnahme
 - Ausführliche Fehlerbeschreibung.
-
- Literatur: Watts Humphrey, „A Discipline for Software Engineering“, Addison Wesley, 1995.
 - Beschreibt im Detail, wie man selbst seinen „persönliche Softwareprozess“ optimiert.



Fehlerlogbuch

	A	B	C	D	E	F	G	H
1	Fehlerlogbuch							
2	Name:				Datum:			
3	Betreuer:				Projekt/Modul:			
4								
5	Entdeckung des Fehlers							
6	Lf.-Nr.	Datum	in Phase	Fehlertyp	Quellphase	Dauer (min)	Folgefehler von	Beschreibung
7	1							
8	2							
9	3							
10	4							
11	5							
12	6							
13	7							
	8							



Fehlerlogbuch

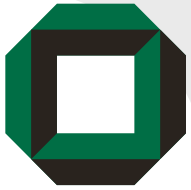
	A	B	C	D	E	F	G	H
1	Fehlerlogbuch							
2	Name: Stefan-Thorsten Udent				Datum: 15.12.2004			
3	Betreuer: Bernd Erwin Treuer				Projekt/Modul: Übungsblatt 17 Aufgabe 3			
4								
5	Entdeckung des Fehlers							
6	Lf.-Nr.	Datum	in Phase	Fehlertyp	Quellphase	Dauer (min)	Folgefehler von	Beschreibung
7	1	15.12.04	Implementierung	Dokumentation	Implementierung	3	-	Kommentar vergessen
8	2		Anforderung Entwurf Implementierung Testen Aufstellen Betrieb					
9	3							
10	4							
11	5							
12	6							
13	7							
	8							

Fehlertyp	Beschreibung
Dokumentation	Kommentare, Nachrichten
Syntax	Rechtschreibung, Punctuation, Tippfehler, Befehlsformate
Build	Änderungsverwaltung, Bibliotheken, Versionskontrolle
Zuweisung	Deklaration, Namensüberlagerung, Gültigkeitsbereich, Typbeschränkungen
Schnittstellen	Aufrufe und Referenzen, Ein-/Ausgabe, Semantik
Prüfung	Fehlermeldungen, ungeeignete Überprüfungen
Daten	Struktur, Inhalt, Codierung, Interpretation
Funktion	Logik, Zeiger, Schleifen, Rekursionen, Berechnungsmethode
System	Konfiguration, Timing, Speicherverbrauch und -zugriffsmuster
Umgebung	Design, Übersetzungsprozess, Testumgebung

Tom Gelhausen:
Phase, während der der Fehler (wahrscheinlich) implantiert wurde

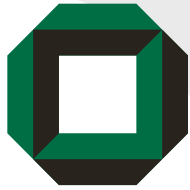
Tom Gelhausen:
Wenn dieser Fehler durch das Beheben eines anderen Fehlers eingefügt wurde, dann hier die entsprechende Fehlernummer

Tom Gelhausen:
Phase, während der der Fehler entdeckt wurde



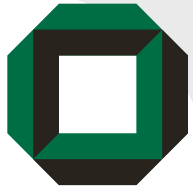
Zeitlogbuch

- Zeitlogbuch wird geführt, um die Zeit für einzelne Tätigkeiten zu erfassen.
- Man muss für viele Aufgaben eine Schätzung abgeben. Das Zeitlogbuch hilft, Schätzungen mit hinreichender Genauigkeit geben zu können (bei der Schätzung genauer zu werden).

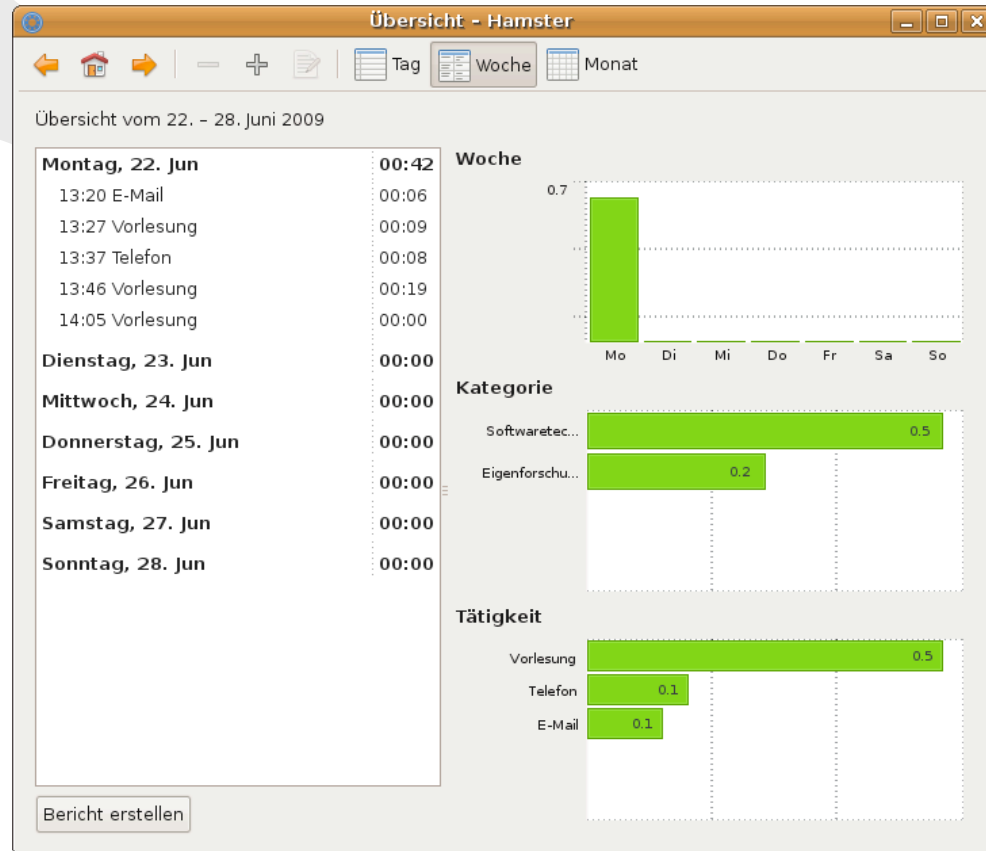


Zeitlogbuch (manuell)

	A	B	C	D	E	F	G
1	Zeitlogbuch						
2	Name: Stefan-Thorsten Udent					Datum:	15.12.2004
3	Betreuer: Bernd Erwin Treuer					Projekt/Modul:	Übungsblatt 17 Aufgabe 3
4							
5	Zeit						
6	Datum	Anfang	Ende	Unterbrechungen	verbraucht	Phase	Bemerkung
7	15.12.04	16:25	16:30	00:00,0	05:00,0	Anforderung	Nochmal gelesen
8		16:35	17:05	00:00,0	30:00,0	Entwurf	Nachbesserung
9		17:05	17:40	03:00,0	32:00,0	Implementierung	Telefonanruf
10					00:00,0		
11					00:00,0		
12					00:00,0		
					00:00,0		



Zeitlogbuch (Software-unterstützt)



Hier: „Hamster“ für Linux