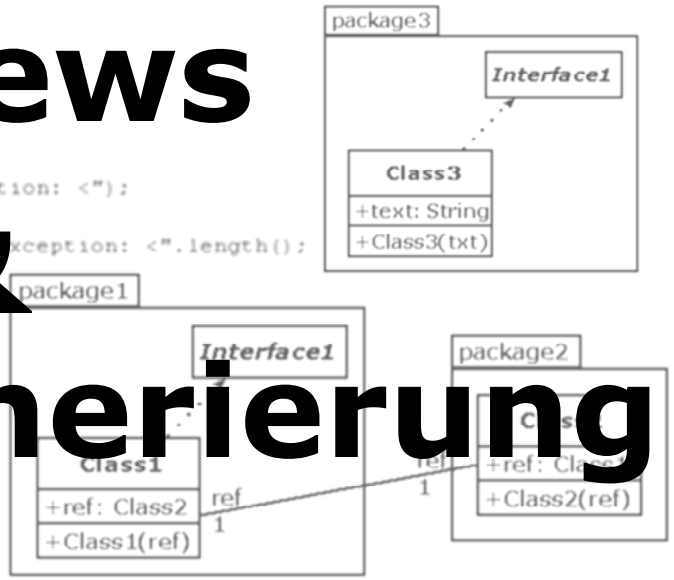
Datenbanken-Seminar: **EPTA**
Vortrag am 10. Januar 2003

Code-Reviews & Code-Generierung

Code-Reviews & Code-Generierung



als Bestandteile des Entwicklungsprozesses

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

- Kontrolle des Entwicklungsprozesses

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

- Kontrolle des Entwicklungsprozesses
- Finden von Fehlern

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

- Kontrolle des Entwicklungsprozesses
- Finden von Fehlern
- Vermeiden von Fehlern

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

- Kontrolle des Entwicklungsprozesses
- Finden von Fehlern
- Vermeiden von Fehlern
- Beschleunigung des Entwicklungsprozesses

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Reviews und Code-Generierung: Gemeinsamkeiten?

Code-Reviews & Code-Generierung

Gemeinsame Ziele

- Kontrolle des Entwicklungsprozesses
- Finden von Fehlern
- Vermeiden von Fehlern
- Beschleunigung des Entwicklungsprozesses
- **Geld sparen**

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Validierung und Verifikation

V&V

Code-Reviews & Code-Generierung

Was ist das?

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Validierung und Verifikation

V&V

Code-Reviews & Code-Generierung

Was ist das?

Validierung:
Erstellen wir das richtige Produkt?

Verifikation:
Erstellen wir das Produkt richtig?

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Validierung und Verifikation

V&V

Was ist das?

V&V

- finden im Optimalfall während des gesamten Lebenszyklus eines Softwareprodukts statt
- stellen sicher, dass die Software mit ihrer Spezifikation übereinstimmt und die Bedürfnisse der Kunden erfüllt werden

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Verifikation / Systemprüfung

Erstellen wir das Produkt richtig?

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Verifikation / Systemprüfung

Erstellen wir das Produkt richtig?

Verifikation besteht u.a. aus

- Korrektheitsbeweisen

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Verifikation / Systemprüfung

Erstellen wir das Produkt richtig?

Verifikation besteht u.a. aus

- Korrektheitsbeweisen
- formalen Tests, nicht-formalen Tests

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Verifikation / Systemprüfung

Erstellen wir das Produkt richtig?

Verifikation besteht u.a. aus

- Korrektheitsbeweisen
- formalen Tests, nicht-formalen Tests
- Software-Inspektionen
  (z.B. **Code-Reviews**)

Stefan Büttcher
<stefan@buettcher.org>

Systemprüfung I: Software-Inspektionen

Prüfung des statischen Systemverhaltens.

Analyse der Erscheinungsform des Systems:

- Entwurfsdiagramme
- Quelltext des Systems

Stefan Büttcher
<stefan@buettcher.org>

Systemprüfung II: Software-Tests

Prüfung des dynamischen Systemverhaltens.

Ausführen der Implementierung mit Testdaten.

Prüfen, ob sich das System gemäß der Spezifikation verhält.

Stefan Büttcher
<stefan@buettcher.org>

Vorteil von Software-Inspektionen gegenüber Software-Tests

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Vorteil von Software-Inspektionen gegenüber Software-Tests

- Software-Inspektionen können bereits zu einem sehr frühen Zeitpunkt des Entwicklungsprozesses beginnen.

Stefan Büttcher
<stefan@buettcher.org>

Vorteil von Software-Inspektionen gegenüber Software-Tests

- Software-Inspektionen können bereits zu einem sehr frühen Zeitpunkt des Entwicklungsprozesses beginnen.
- Software-Tests kosten mehr Zeit und können nur bedingt parallelisiert werden (Error Hiding).

Stefan Büttcher
<stefan@buettcher.org>

Vorteil von Software-Inspektionen gegenüber Software-Tests

- Software-Inspektionen können bereits zu einem sehr frühen Zeitpunkt des Entwicklungsprozesses beginnen.
- Software-Tests kosten mehr Zeit und können nur bedingt parallelisiert werden (Error Hiding).
- Software-Inspektionen sind billiger!

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (1) Warum Code-Review?

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (1) Warum Code-Review?

Code-Reviews & Code-Generierung

"Viele Programmierer wechseln nach zwei Jahren den Arbeitgeber, da sie ihren eigenen Code nicht mehr lesen können. Dafür übernehmen sie ein Projekt, von einem anderen Programmierer, der gerade gekündigt hat, da er ebenfalls den Code nicht mehr lesen kann. Nach einigen Wochen oder Monaten überzeugt der Programmierer in seiner neuen Stelle seinen Chef, den Code seines Vorgängers zu verwerfen und neu zu beginnen. Nach zwei Jahren wiederholt sich die Geschichte."

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (1) Warum Code-Review?

Code-Reviews & Code-Generierung

"Viele Programmierer wechseln nach zwei Jahren den Arbeitgeber, da sie ihren eigenen Code nicht mehr lesen können. Dafür übernehmen sie ein Projekt, von einem anderen Programmierer, der gerade gekündigt hat, da er ebenfalls den Code nicht mehr lesen kann. Nach einigen Wochen oder Monaten überzeugt der Programmierer in seiner neuen Stelle seinen Chef, den Code seines Vorgängers zu verwerfen und neu zu beginnen. Nach zwei Jahren wiederholt sich die Geschichte."

Code-Reviews können Abhilfe schaffen, indem Sie höherwertigen Code garantieren.

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (2) Problematiken

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (2) Problematiken

- Ergebnisse von Code-Reviews sieht man nicht unmittelbar.

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (2) Problematiken

- Ergebnisse von Code-Reviews sieht man nicht unmittelbar.
- Code-Reviews kosten zu Beginn des Entwicklungsprozesses Zeit, den Sie erst viel später wieder einspielen.

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (2) Problematiken

- Ergebnisse von Code-Reviews sieht man nicht unmittelbar.
- Code-Reviews kosten zu Beginn des Entwicklungsprozesses Zeit, den Sie erst viel später wieder einspielen.
- Programmierer mögen keine Kritik an ihrem Code.

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (3) Code-Review-Terminologie

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (3) Code-Review-Terminologie

Code-Reviews & Code-Generierung

"In the shower this morning, it occurred to me it would be useful to write a guide listing certain words and phrases that arise during code reviews and their actual, candid meanings. Hopefully this will help the communication process at technical reviews."

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (3) Code-Review-Terminologie

"In the shower this morning, it occurred to me it would be useful to write a guide listing certain words and phrases that arise during code reviews and their actual, candid meanings. Hopefully this will help the communication process at technical reviews."

"I think that's against the coding standards."

meaning: "That's not the way I arrange my whitespaces, ifs, etc."

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (3) Code-Review-Terminologie

Code-Reviews & Code-Generierung

"In the shower this morning, it occurred to me it would be useful to write a guide listing certain words and phrases that arise during code reviews and their actual, candid meanings. Hopefully this will help the communication process at technical reviews."

"I think that's against the coding standards."

meaning: "That's not the way I arrange my whitespaces, ifs, etc."

"Let's take this off line."

meaning: "If you carry on this pedantic discussion another minute, I'll scream."

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (3) Code-Review-Terminologie

Code-Reviews & Code-Generierung

"In the shower this morning, it occurred to me it would be useful to write a guide listing certain words and phrases that arise during code reviews and their actual, candid meanings. Hopefully this will help the communication process at technical reviews."

"I think that's against the coding standards."

meaning: "That's not the way I arrange my whitespaces, ifs, etc."

"Let's take this off line."

meaning: "If you carry on this pedantic discussion another minute, I'll scream."

"This really needs to be documented."

meaning: "We could just leave it undocumented."

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (4) Was wird untersucht?

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (4) Was wird untersucht?

- Einhalten organisationsinterner Standards

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (4) Was wird untersucht?

Code-Reviews & Code-Generierung

- Einhalten organisationsinterner Standards
- Verständlicher Code / Verständliche Kommentierung des Codes

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (4) Was wird untersucht?

- Einhalten organisationsinterner Standards
- Verständlicher Code / Verständliche Kommentierung des Codes
- Fehlerabfragen / Initialisierung von Variablen, Array-Zugriffe

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Code-Review (4) Was wird untersucht?

- Einhalten organisationsinterner Standards
- Verständlicher Code / Verständliche Kommentierung des Codes
- Fehlerabfragen / Initialisierung von Variablen, Array-Zugriffe
- Terminierung aller Schleifen sicher-gestellt?

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (4) Was wird untersucht?

- Einhalten organisationsinterner Standards
- Verständlicher Code / Verständliche Kommentierung des Codes
- Fehlerabfragen / Initialisierung von Variablen, Array-Zugriffe
- Terminierung aller Schleifen sicher-gestellt?
- ...

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (5) Organisation der Reviews

Die klassische Code-Review

Rollen	Beschreibung
Autor	
Inspektor	
Vorleser	
Schreiber	
Moderator	
Chefmoderator	

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Rollen	Beschreibung
Autor	Der für den Programmteil verantwortliche Programmierer. Behebt die gefundenen Fehler.
Inspektor	
Vorleser	
Schreiber	
Moderator	
Chefmoderator	

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (5) Organisation der Reviews

Die klassische Code-Review

Rollen	Beschreibung
Autor	Findet Fehler, Lücken und Widersprüche in Programmen und Dokumenten. Agiert auch außerhalb der eigentlichen Inspektion.
Inspektor	
Vorleser	
Schreiber	
Moderator	
Chefmoderator	

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Rollen	Beschreibung
Autor	Stellt den Quelltext oder das Dokument bei einer Sitzung vor.
Inspektor	
Vorleser	
Schreiber	
Moderator	
Chefmoderator	

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Rollen	Beschreibung
Autor	Protokolliert die Ergeb-
Inspektor	nisse der Sitzung.
Vorleser	
Schreiber	
Moderator	
Chefmoderator	

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (5) Organisation der Reviews

Die klassische Code-Review

Rollen	Beschreibung
Autor Inspektor Vorleser Schreiber Moderator Chefmoderator	Organisiert die Inspektionen, stellt Räume bereit und berichtet dem Chefmoderator über die Ergebnisse.

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (5) Organisation der Reviews

Die klassische Code-Review

Rollen	Beschreibung
Autor	Verantwortlich für die Verbesserung des Inspektionsprozesses, etc.
Inspektor	
Vorleser	
Schreiber	
Moderator	

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Ablauf

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (6) Organisation der Reviews

Die klassische Code-Review

Ablauf	Vorbereitung durch den Moderator; Auswahl des Inspektions-Teams, Bereitstellen des Raumes, Sicherstellen, dass die Spezifikation vollständig und der Code syntaktisch korrekt ist
Planung	

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Die klassische Code-Review

Ablauf

Planung

↘

Überblick

Vorstellung des Programms durch den Autor, grobe Beschreibung der Funktionsweise

Stefan Büttcher
<stefan@buettcher.org>

Code-Review (6) Organisation der Reviews

Die klassische Code-Review

Ablauf

Planung
↘
Überblick
↘
Individuelle Vorbereitung

Individuelle Vorbereitung aller Beteiligten: Studium der Spezifikation, Fehlersuche

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Die klassische Code-Review

Ablauf

Planung

→ Überblick

→ Individuelle Vorbereitung → Review-Sitzung

relativ kurz (1 bis 2 Stunden), Feststellen von Fehlern und Abweichungen von Standards; Vorschläge zur Veränderung des Programms

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Ablauf

Planung

Überblick

Individuelle Vorbereitung

Review-Sitzung

Überarbeitung

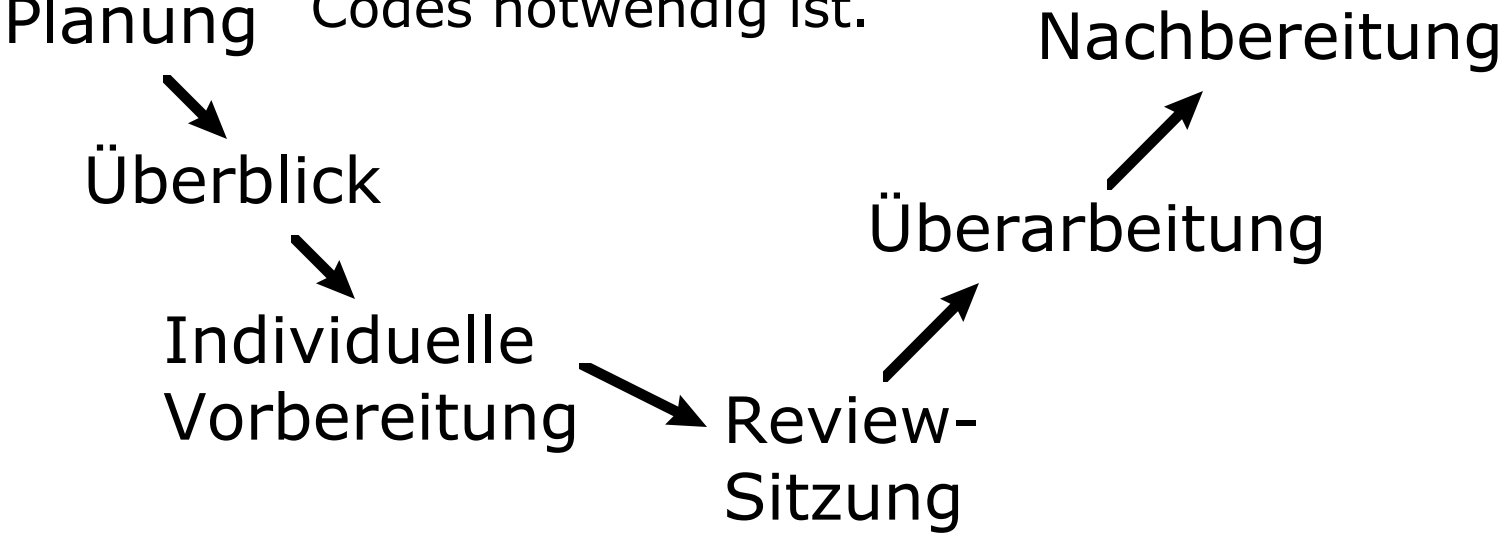
Der Programmierer behebt die festgestellten Mängel (hoffentlich!).

Stefan Büttcher
<stefan@buettcher.org>

Die klassische Code-Review

Ablauf

Der Moderator entscheidet, ob die problematischen Teile korrigiert worden sind und ob eine erneute Review des Codes notwendig ist.



Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

5 Schlüsseleigenschaften:

1. Formale Spezifikation mit Zustandsübergangsmodell

Stefan Büttcher
<stefan@buettcher.org>

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

5 Schlüsseleigenschaften:

1. Formale Spezifikation mit Zustandsübergangsmodell
2. Inkrementelle Entwicklung

Stefan Büttcher
<stefan@buettcher.org>

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

5 Schlüsseleigenschaften:

1. Formale Spezifikation mit Zustandsübergangsmodell
2. Inkrementelle Entwicklung
3. Strukturierte Programmierung. Entwicklung durch schrittweise (sic!) Verfeinerung der Spezifikation

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

5 Schlüsseleigenschaften:

1. Formale Spezifikation mit Zustandsübergangsmodell
2. Inkrementelle Entwicklung
3. Strukturierte Programmierung. Entwicklung durch schrittweise (sic!) Verfeinerung der Spezifikation
4. Statische Verifikation ohne Einzel- oder Modultests

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (1)

Vermeidung von Fehlern durch ständige Inspektionen.

5 Schlüsseleigenschaften:

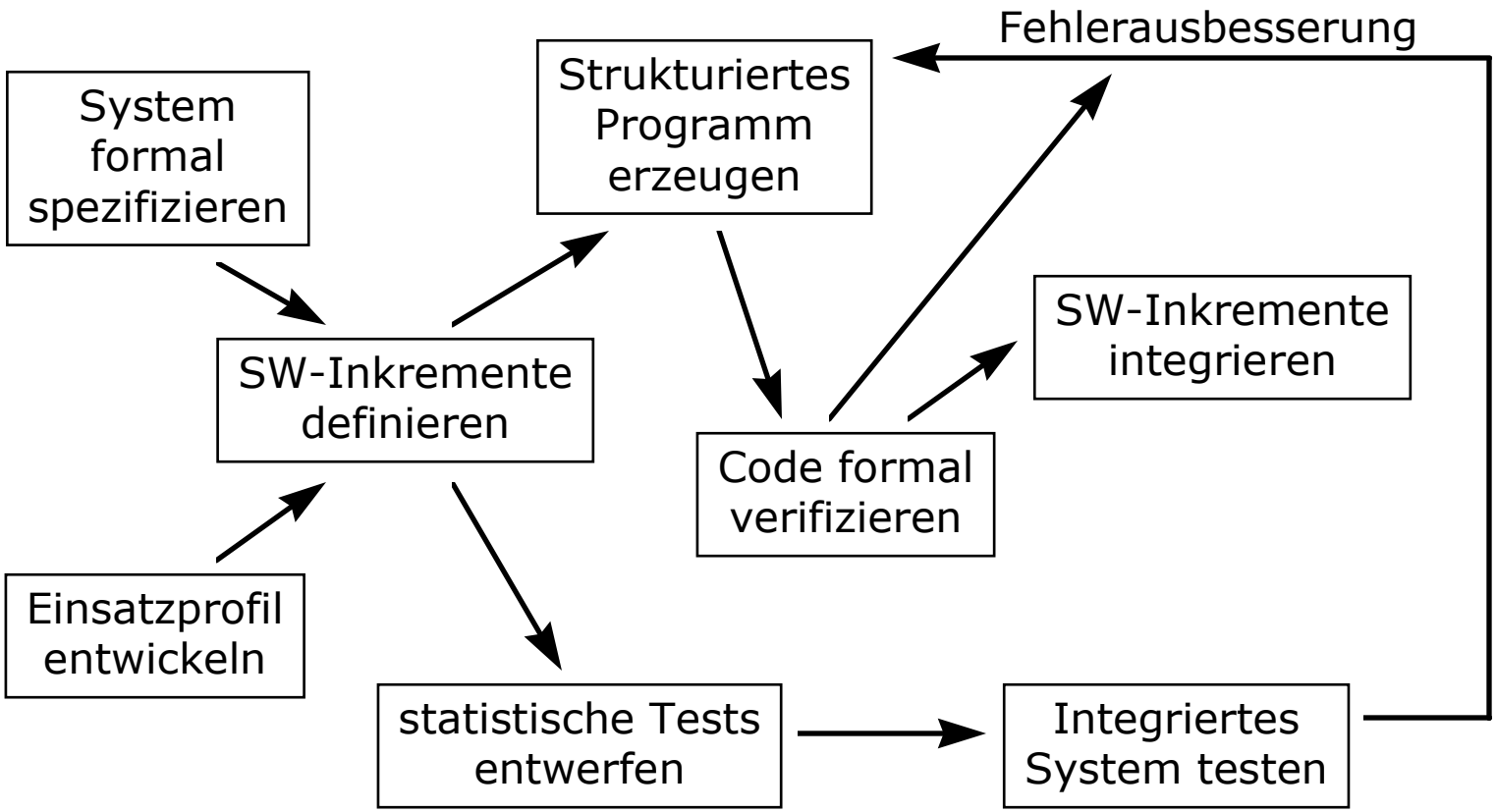
1. Formale Spezifikation mit Zustandsübergangsmodell
2. Inkrementelle Entwicklung
3. Strukturierte Programmierung. Entwicklung durch schrittweise (sic!) Verfeinerung der Spezifikation
4. Statische Verifikation ohne Einzel- oder Modultests
5. Statistische Systemtests für jedes neu integrierte Element

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (2)

Code-Reviews & Code-Generierung



Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (3)

Bei der entwicklung großer Systeme sind i.d.R. drei Teams beteiligt:

Stefan Büttcher
<stefan@buettcher.org>

Der CleanRoom-Entwicklungsprozess (3)

Bei der entwicklung großer Systeme sind i.d.R. drei Teams beteiligt:

- Das *Spezifikationsteam* entwickelt und wartet die Spezifikation. Pflichtenheft.

Stefan Büttcher
<stefan@buettcher.org>

Der CleanRoom-Entwicklungsprozess (3)

Bei der entwicklung großer Systeme sind i.d.R. drei Teams beteiligt:

- Das *Spezifikationsteam* entwickelt und wartet die Spezifikation. Pflichtenheft.
- Das *Entwicklerteam* ist verantwortlich für Entwicklung und Verifikation. Die Software wird nicht ausgeführt.

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Der CleanRoom-Entwicklungsprozess (3)

Bei der entwicklung großer Systeme sind i.d.R. drei Teams beteiligt:

- Das *Spezifikationsteam* entwickelt und wartet die Spezifikation. Pflichtenheft.
- Das *Entwicklerteam* ist verantwortlich für Entwicklung und Verifikation. Die Software wird nicht ausgeführt.
- Das *Zertifizierungsteam* entwickelt Testfälle und testet das System.

Stefan Büttcher
<stefan@buettcher.org>

Macadamian CodeReview-AddIn für Visual Studio .NET

Code-Reviews & Code-Generierung

CodeReview

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

System-Modellierung mit UML (1)

UML (Unified Modeling Language) kann u.a. zur Definition von

genutzt werden.

Vorteil: Formale Spezifikation bei gleichzeitiger einfacher Lesbarkeit (Diagramme) möglich!

Stefan Büttcher
<stefan@buettcher.org>

System-Modellierung mit UML (1)

UML (Unified Modeling Language) kann u.a. zur Definition von

- Klassenhierarchien und

genutzt werden.

Vorteil: Formale Spezifikation bei gleichzeitiger einfacher Lesbarkeit (Diagramme) möglich!

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

System-Modellierung mit UML (1)

UML (Unified Modeling Language) kann u.a. zur Definition von

- Klassenhierarchien und
- Use-Cases

genutzt werden.

Vorteil: Formale Spezifikation bei gleichzeitiger einfacher Lesbarkeit (Diagramme) möglich!

Stefan Büttcher
<stefan@buettcher.org>

System-Modellierung mit UML (2)

Code-Reviews & Code-Generierung

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

System-Modellierung mit UML (2)

Konsistenzprüfungen und einfache Tests lassen sich mit Hilfe von UML leicht automatisieren.

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

System-Modellierung mit UML (2)

Konsistenzprüfungen und einfache Tests lassen sich mit Hilfe von UML leicht automatisieren.

Warum *Codegenerierung*?

Aus den erstellten Diagrammen fertiger, compilierbarer Code (z.B. Java) erzeugt werden.

Stefan Büttcher
<stefan@buettcher.org>

UML Reverse Engineering

Auch die Gegenrichtung (Diagramme aus bestehendem Java-Code erzeugen, Reverse Engineering) ist möglich.

Den Prozess, aus Diagrammen Code zu erzeugen, diesen Code zu bearbeiten und daraus wieder Diagramme zu erzeugen, nennt man *Round-Trip-Engineering*.

Stefan Büttcher
<stefan@buettcher.org>

UML und Java mit Rational Rose

Code-Reviews & Code-Generierung

Rational Rose

Stefan Büttcher
<stefan@buettcher.org>

EPTA rulez!

Vielen Dank für die Aufmerksamkeit!

Stefan Büttcher
<stefan@buettcher.org>