Patrick Schlott
Fachbereich Informatik

Freie Universität Berlin

Automatisierte Expertenfindung

zur Fehlerbehebung in großen Softwaresystemen

Gliederung

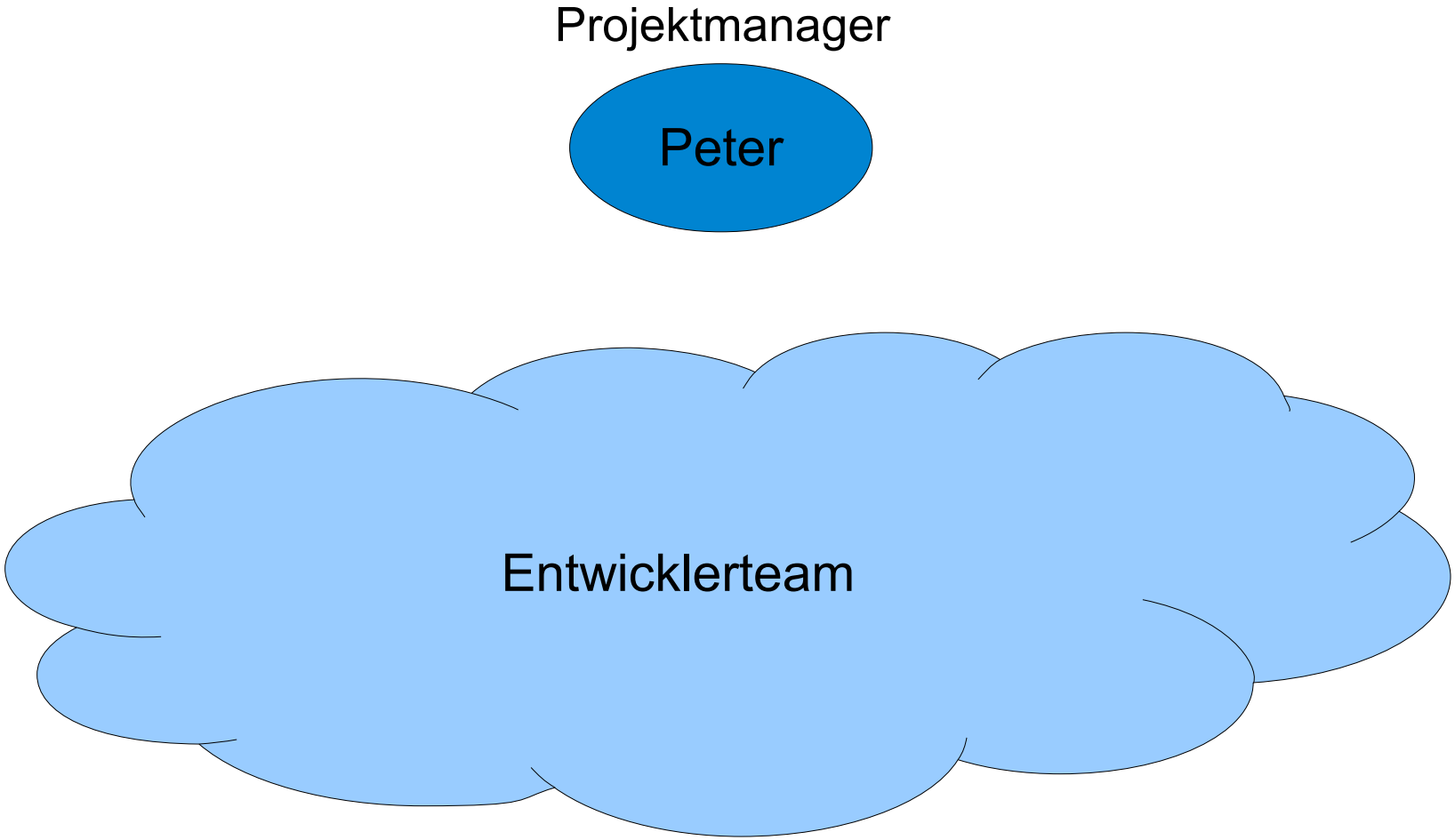
- **Einführung**
- Expertenfindung anhand von Bugreports
- Expertenfindung anhand von Testfällen
- Abschlussbetrachtung

Worum gehts hier überhaupt?

- Unterstützung von Projektmanagern zur Expertenfindung
- Ansprüche:
  - Automatisierung
  - Hohe Genauigkeit
  - Anwendbarkeit

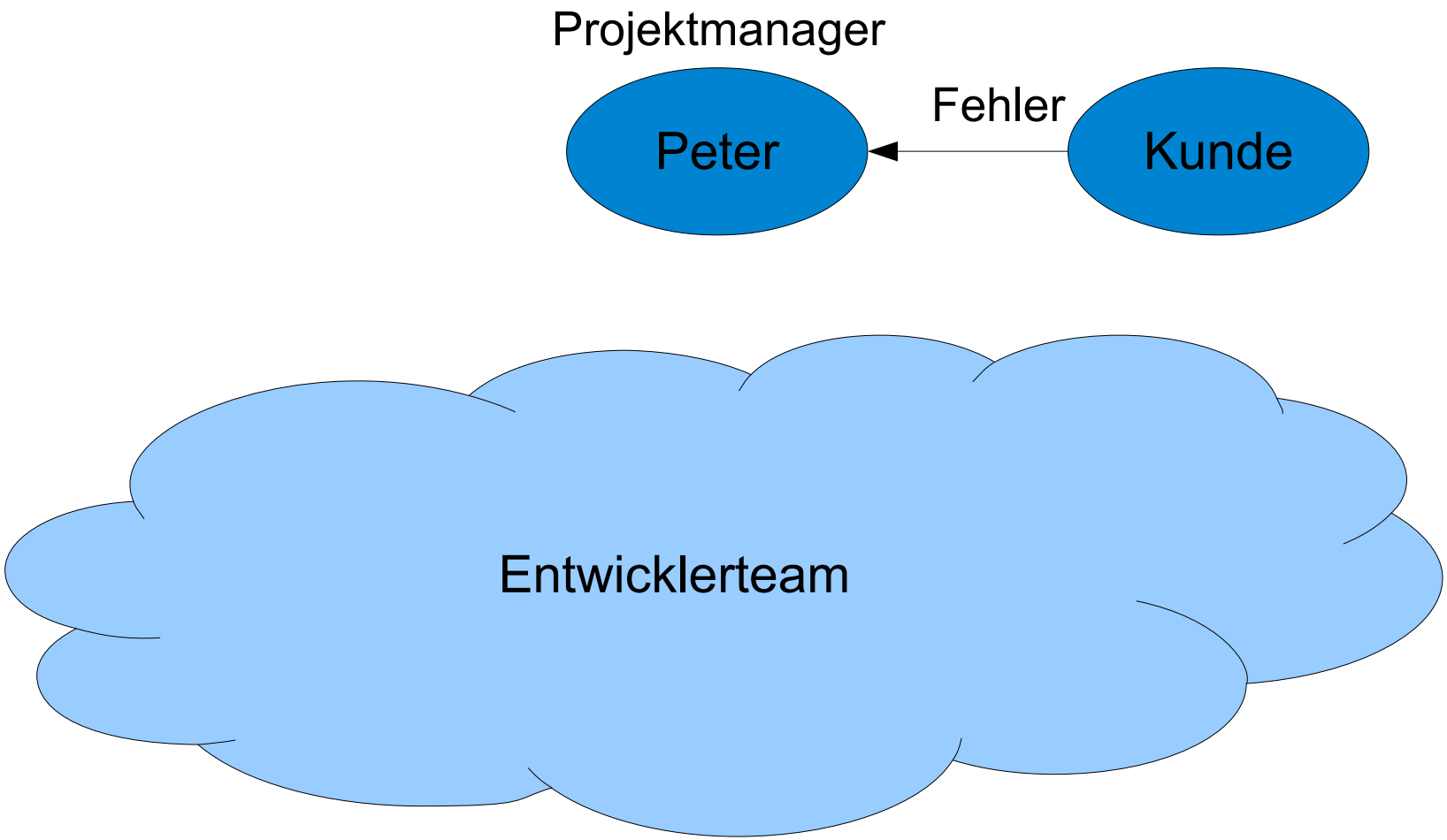
Freie Universität Berlin

Motivation



Freie Universität Berlin

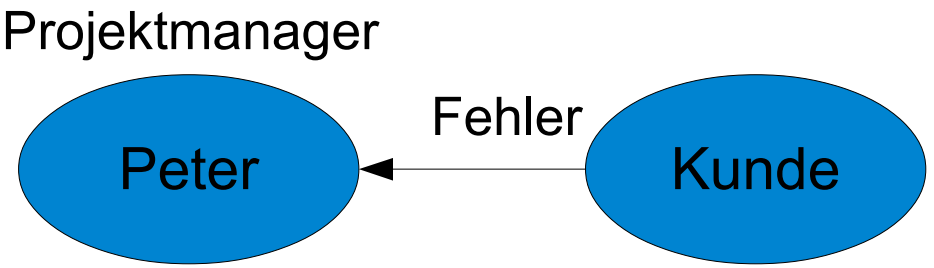
Motivation

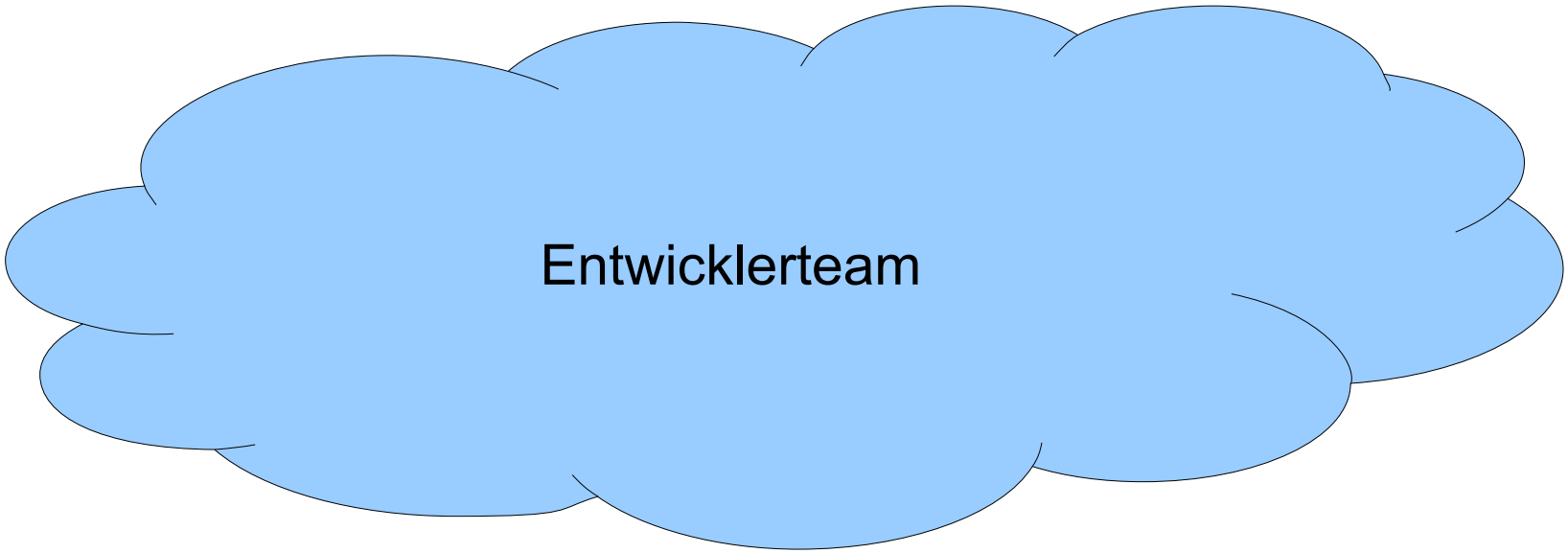


Freie Universität Berlin

Motivation

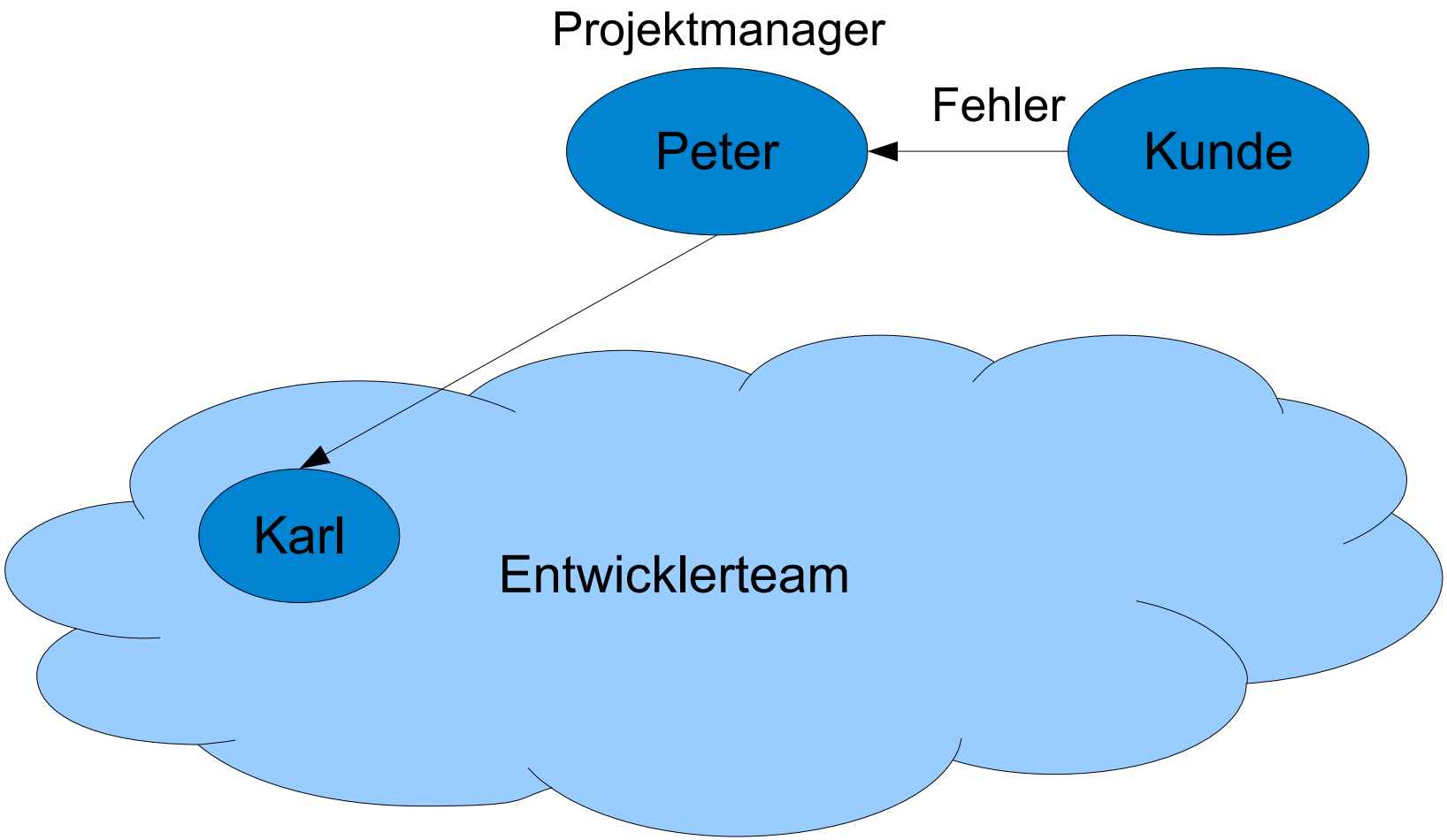
- Fehler verifizieren
- Fehlerumfeld bestimmen
- Fehler zuweisen





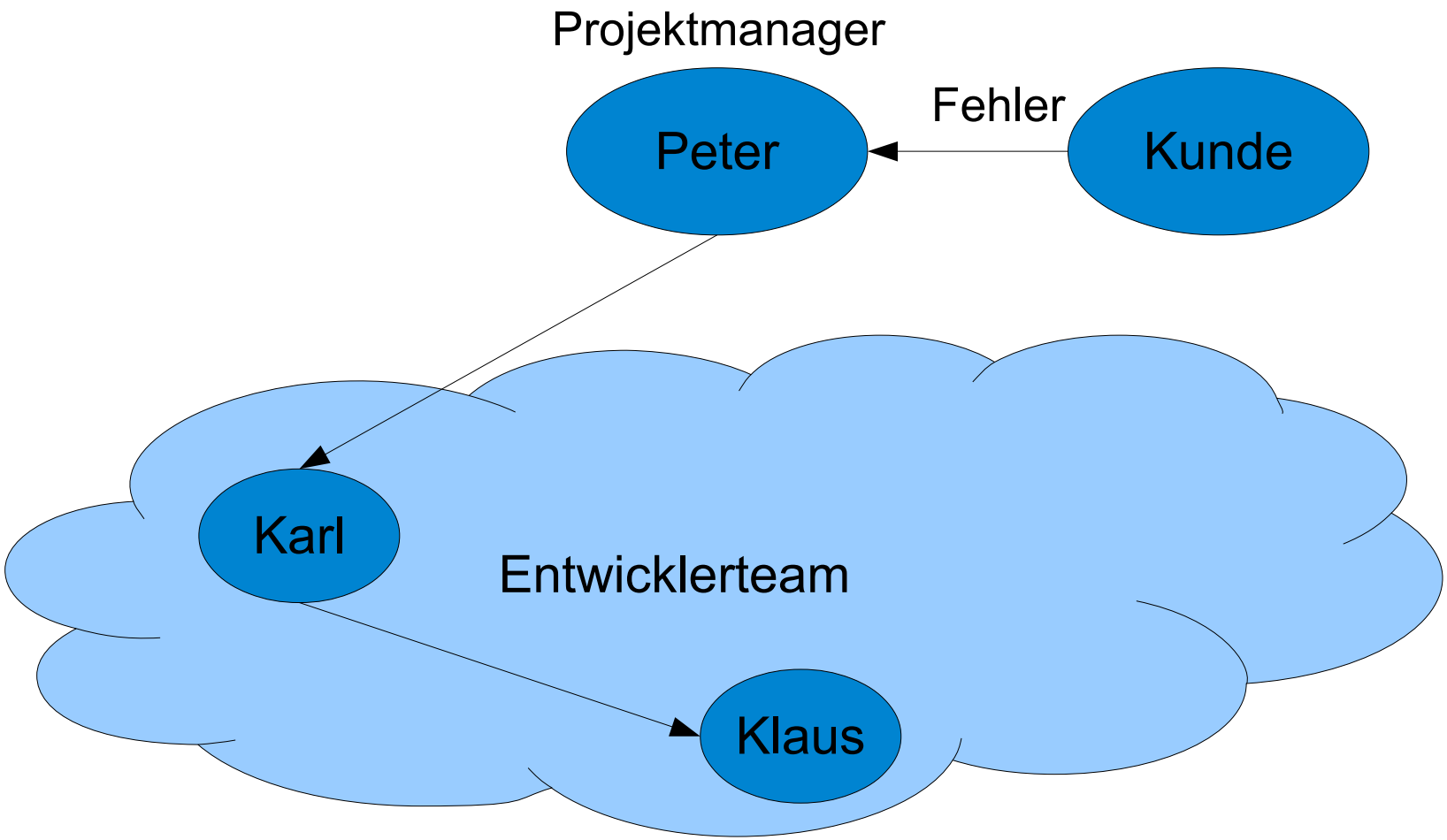
Freie Universität Berlin

Motivation



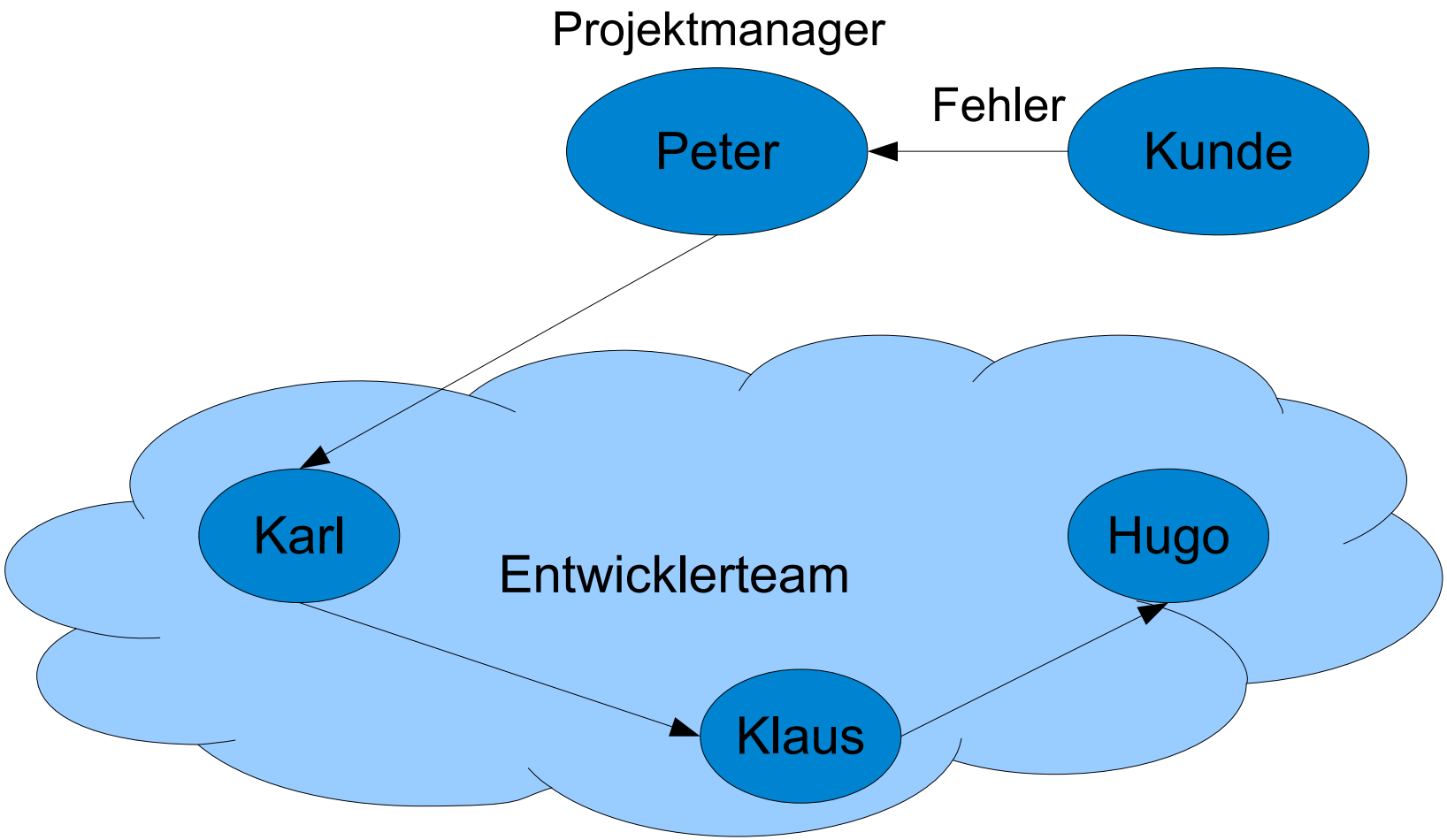
Freie Universität Berlin

Motivation



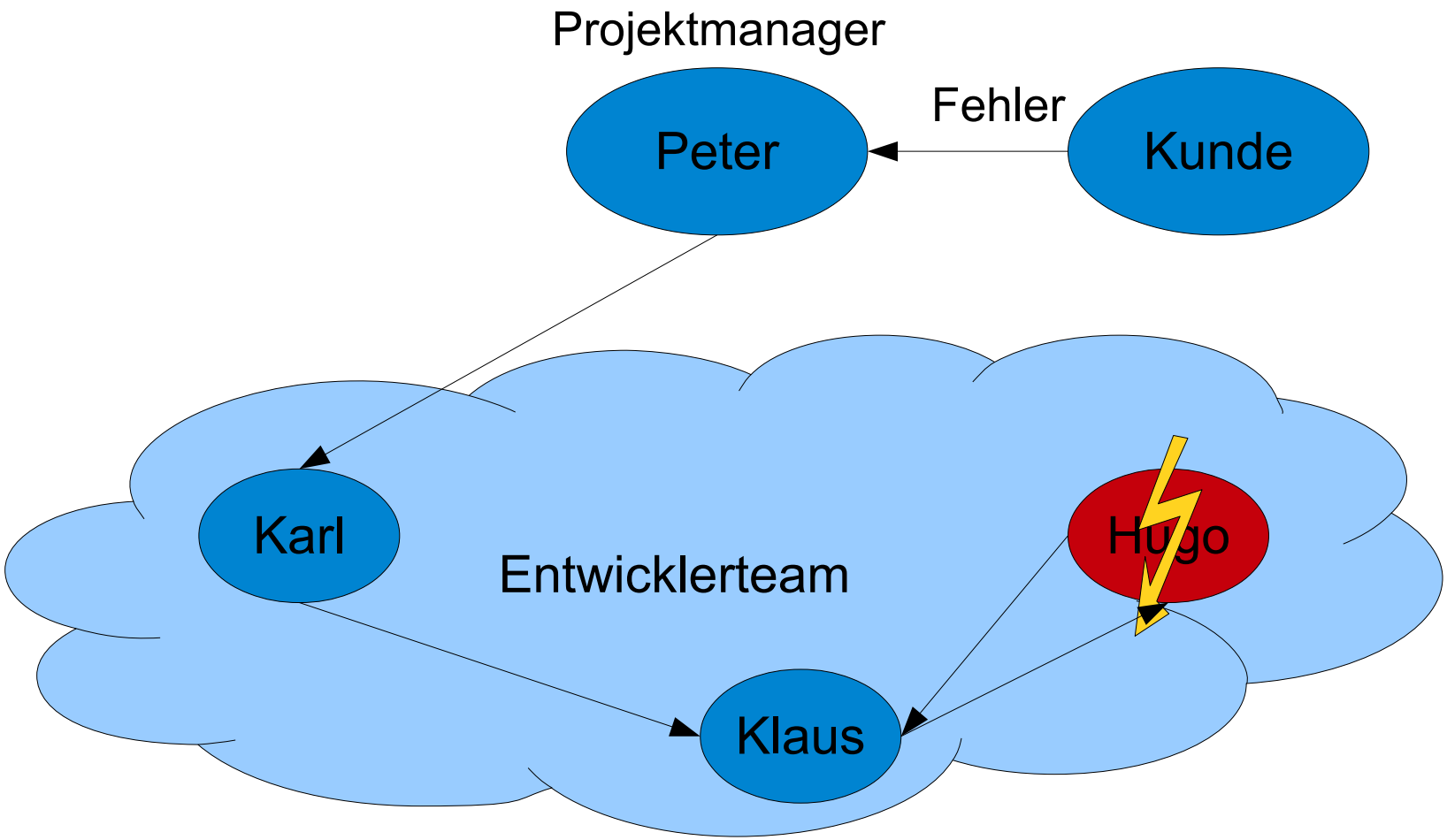
Freie Universität Berlin

Motivation



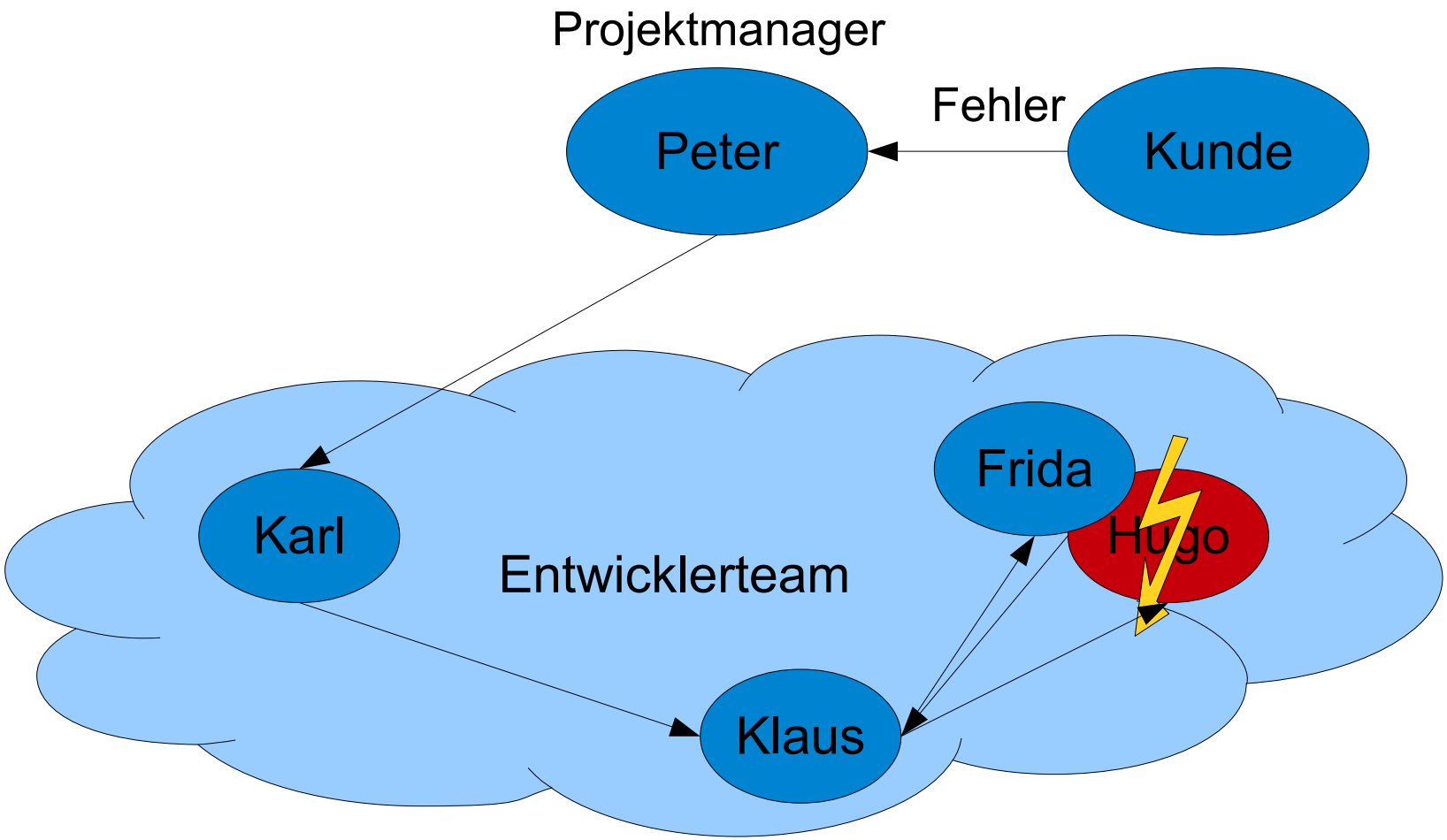
Freie Universität Berlin

Motivation



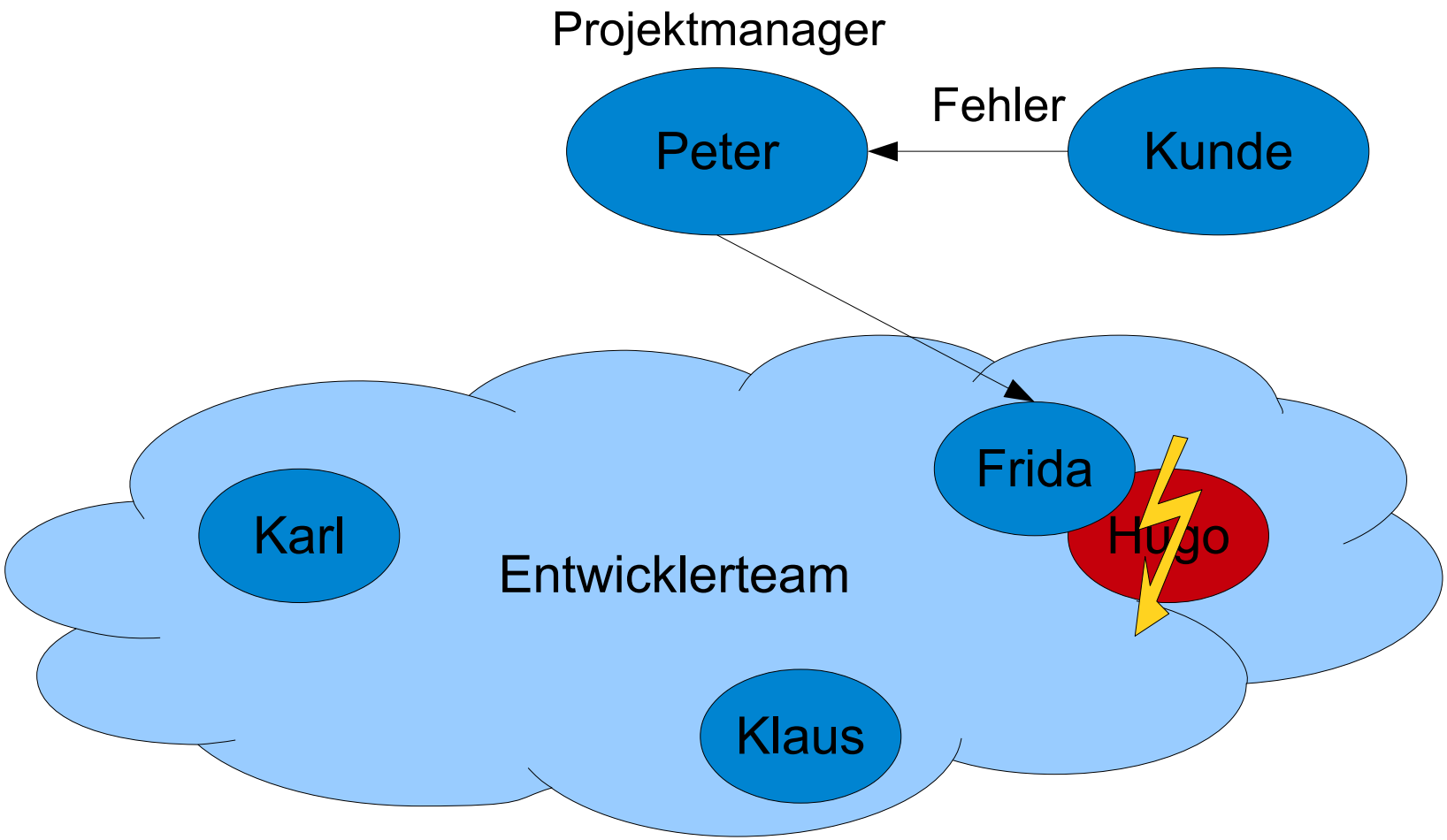
Freie Universität Berlin

Motivation



Freie Universität Berlin

Motivation



Gliederung

- Einführung
- **Expertenfindung anhand von Bugreports**
  - Who Should Fix This Bug?
  - Evaluation
  - Probleme
  - Neuzuweisungsgraphen
- Expertenfindung anhand von Testfällen
- Abschlussbetrachtung

Bugreports

- Viele Projekte verwenden Bugtrackersysteme
- Meldung von Fehlern durch User
- Bestandteile:
  - Freitextfeld
  - Kurzbeschreibung
  - Metadaten
  - Zugewiesener Bearbeiter (Assigned-To)
  - Statusfeld

Beispielprojekt Eclipse

- Teilweise mehrere Hundert Bugreports am Tag
  - Besonders nach Releases
- Manuelle Durchsicht
  - Statusänderung nach etwa 17 Tagen
- Manuelle Zuweisung
  - Nach etwa 24 weiteren Tagen

Paper I

- Who Should fix this Bug?
- John Anvik, Lyndon Hiew, Gail C. Murphy
- University of British Columbia
- ICSE 2006

Grundlegender Ansatz

- Verwendung der textuellen Beschreibung
- Maschinelles Lernen
- Verwenden von Trainingsdaten
- Anwenden auf neue Bugreports

Überwachte Lernende Maschine

- Endliche Anzahl an Trainingsdaten
  - Bestehend aus Merkmalen
- Endliche Anzahl an Kategorien
  - Hier Entwickler
- Einordnung von Daten in Kategorien
- Überwacht: Zuordnung bekannt und fest

Schritt 1: Trainingsdaten aufbereiten

- Textuelle Beschreibung verarbeitbar machen
  - Bug Report → Liste von Merkmalen
  - Ordnen nach relativer Häufigkeit

Schritt 2: Trainingsdaten kategorisieren

- Experte für Bugreports finden
- Projekt analysieren und Regeln aufstellen
  - Assigned-To-Feld verwenden

    oder
  - Bug Logs verwenden

Schritt 2: Mögliche Regeln

- Regel 1: FIXED
  - Bearbeiter als Experte übernehmen
- Regel 2: DUPLICATE
  - Bearbeiter des Originals als Experte übernehmen
- Regel 3: INVALID oder WONTFIX
  - Nicht Klassifizierbar

Schritt 3: Filterung

- Entfernen nicht klassifizierbarer Reports
- Nicht mehr verfügbare Entwickler ausschließen
- Entwickler mit zu wenig Trainingsdaten ausschließen

- Anschließend training der Maschine

Evaluation

- Ansatz wurde auf 3 Open Source Projekte angewandt
  - Eclipse
  - Firefox
  - GCC

Vorgehensweise

- Große Anzahl an Trainingsdaten
  - Kategorisierung durch Bug Logs
- Ansatz auf neuere Prüfdaten anwenden
- Liste empfohlener Experten mit wirklichen Experten vergleichen

Ergebnisse

	Eclipse	Firefox	GCC
Trainingszeitraum	September 2004 – April 2005		
Trainingsdaten	8655	9752	2629
Prüfdaten	170	22	194
Genauigkeit (eine Empfehlung)	58%	64%	6%
Genauigkeit (drei Empfehlungen)	47%	57%	18%

Problem I: Experten

- Keine Projektmanager mit Überblick oder Expertenlisten verfügbar
- Berechnung anhand von Modulaktivitäten
- Entwickler verwenden häufig andere Kennung als im Bugtrackersystem

Problem II: Schlechte GCC-Werte

- Genauigkeit deutlich zu niedrig
- Ansatz vermutlich nicht geeignet für Projekt
- Mögliche Gründe:
  - Eine extrem dominater Entwickler (1394 zu 160)
  - Viele Entwickler mit sehr wenig Beiträgen

    → Nur 29 von 92 Entwicklern betrachtet
  - 32 von 84 Benutzernamen konnten nicht zugeordnet werden

Problem III: Praktischer Nutzen

- Nur externe Genauigkeitsbestimmung
- Keine Studie zum praktischen Nutzen für Projektmanager

  → Keine Prozessverbesserung bestimmbar

Problem IV: Qualität von Bugreports

- Bugreports werden von Usern geschrieben
  - Häufig kein Fachwissen
  - Teilweise schlechte Beschreibung
  - Extrem unterschiedliche Formulierungen
    - Mögliche Verbesserung durch Wörterbücher

Gliederung

- Einführung
- Expertenfindung anhand von Bugreports
  - Who Should Fix This Bug?
  - Evaluation
  - Probleme
  - **Neuzuweisungsgraphen**
- Expertenfindung anhand von Testfällen
- Abschlussbetrachtung

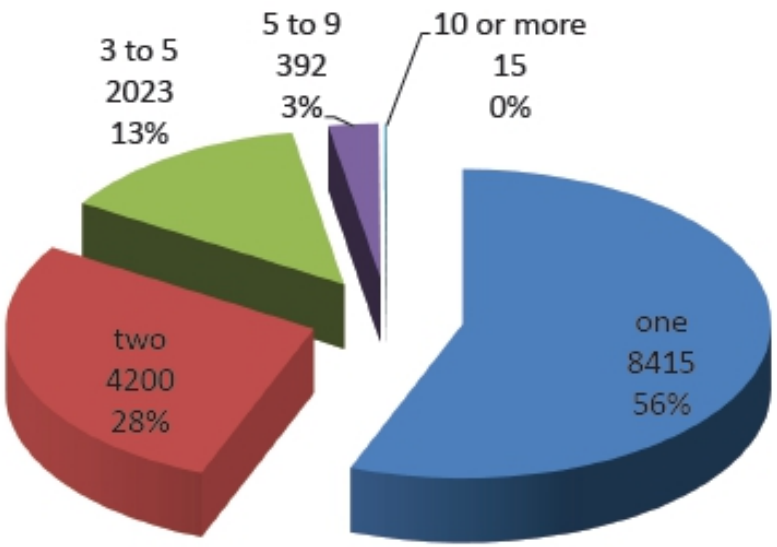
Motivation

- Bisherige Annahme: Bugreport wird genau einer Person vorgelegt.
- Realität: Häufige Neuzuweisung von Bugreports
- Gründe:
  - Falsche Zuweisung
  - Zusätzliches Wissen benötigt
  - Entwickler nicht verfügbar

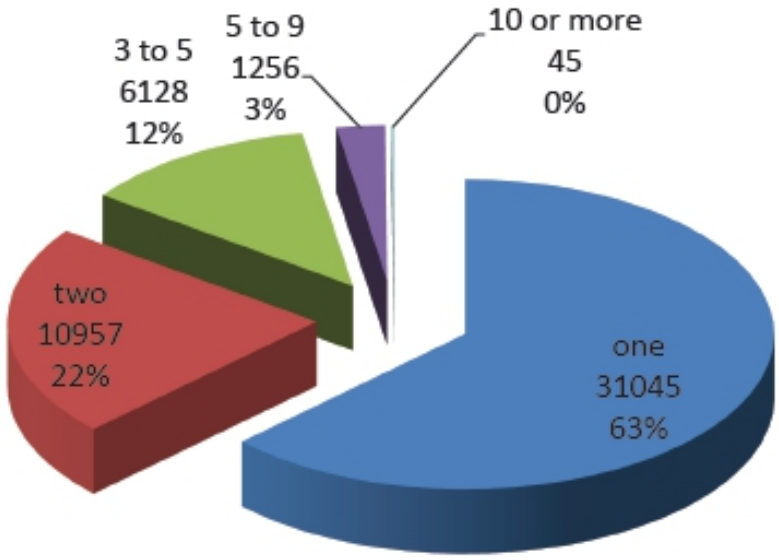
Freie Universität Berlin

Anzahl Neuzuweisungen

Eclipse



Mozilla



Neuzuweisungsgraph

- Umwandeln von vergangenen Neuzuweisungen in Pfade
- Markov-Modell aus Pfaden erstellen
  - Zustände = Entwickler
  - Zustandsübergänge = Neuzuweisungen
  - Übergangswahrscheinlichkeiten aus Pfaden
    - Nur Abhängig von aktuellem Zustand

Verbesserungspotential

- Aufzeigen von Teamstrukturen
  - Besonders wichtig für Open Source Projekte
- Kombination mit automatisierter Expertenfindung
  - Verbesserungen von 0% -16% je nach Projekt und Algorithmus

Gliederung

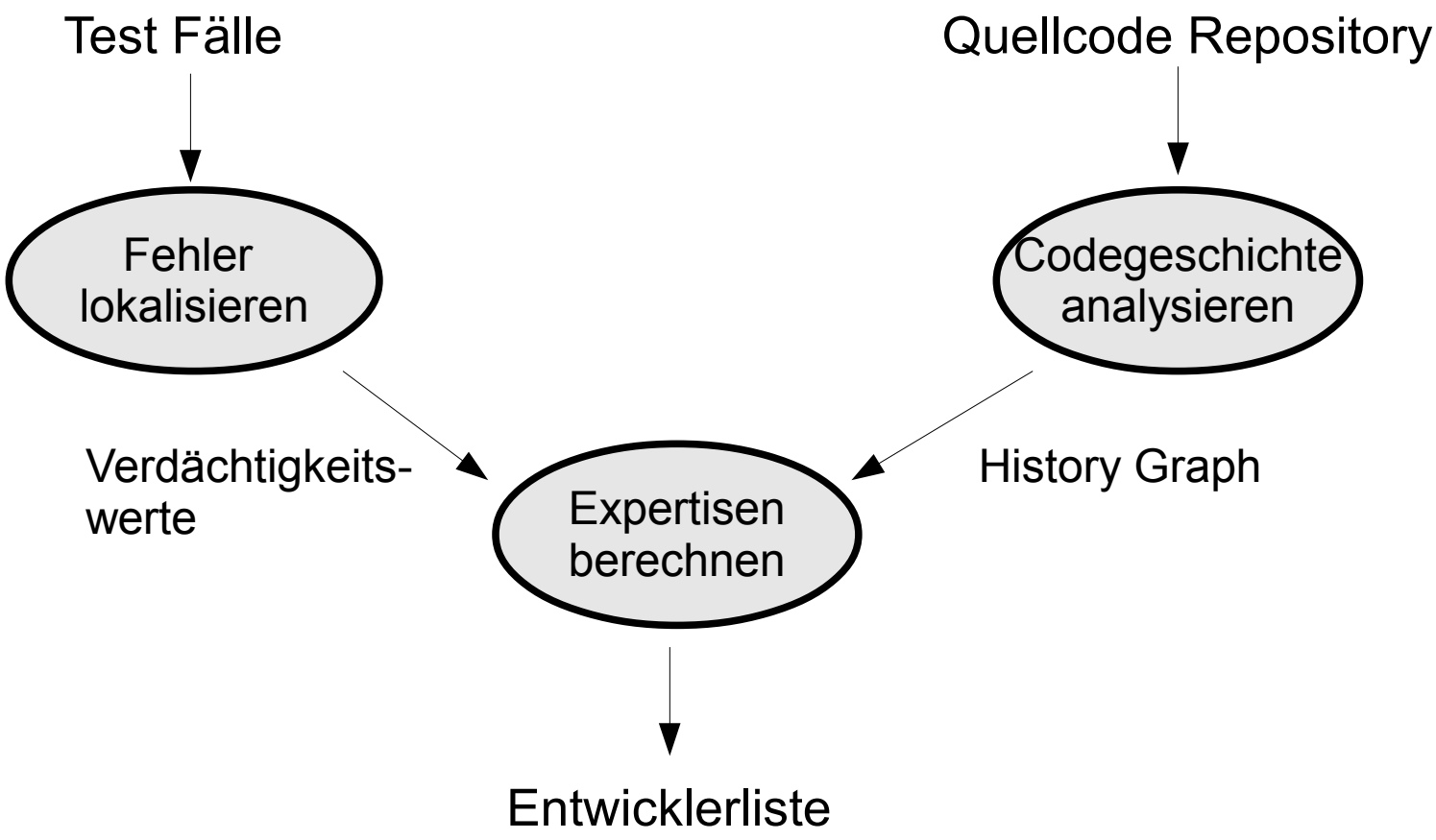
- Einführung
- Expertenfindung anhand von Bugreports
- **Expertenfindung anhand von Testfällen**
  - WhoseFault
  - Evaluation
  - Probleme
- Abschlussbetrachtung

Paper II

- WHOSEFAULT: Automatic Developer-to-Fault Assignment through Fault Localization
- Francisco Servant
- James A. Jones
- University of California, Irvine
- ICSE 2012

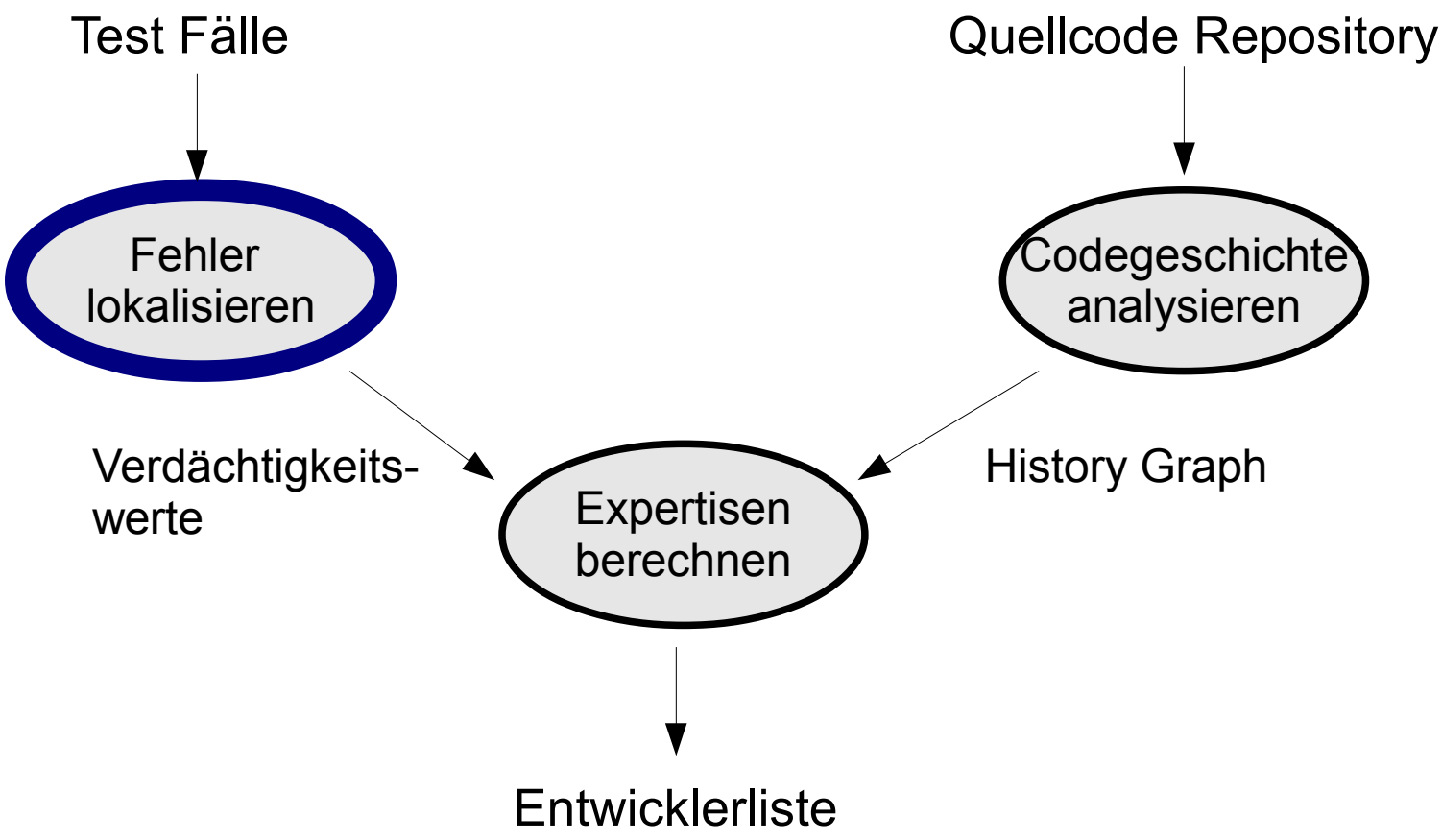
Freie Universität Berlin

Übersicht



Freie Universität Berlin

Übersicht



Tarantula

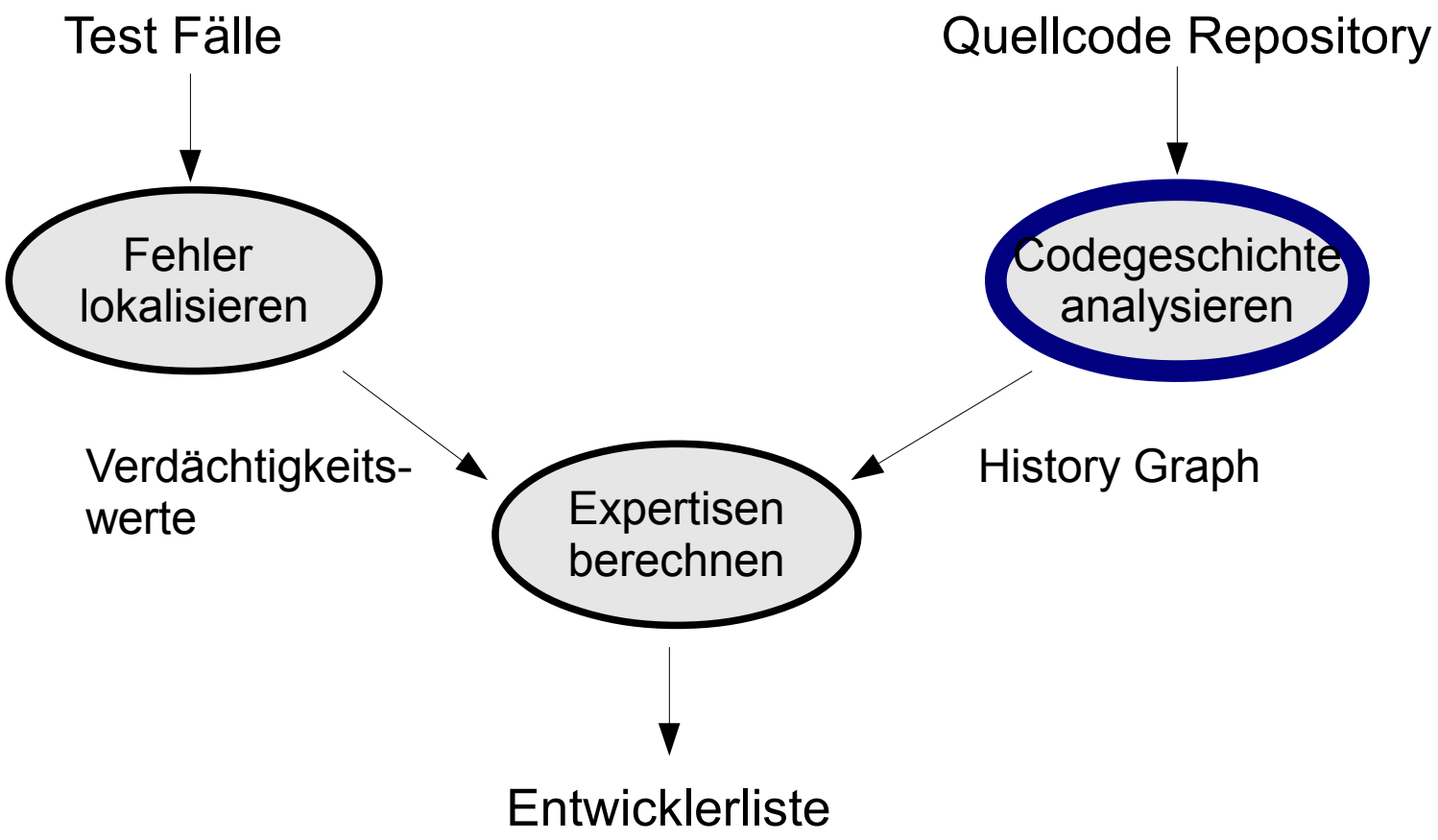
Code	Testfall 1	Testfall 2	Testfall 3
a = x[0];	x	x	x
b = x[2];	x	x	x
op = x[1];	x	x	x
if (op == 'plus')	x	x	x
return a + b;	x		
else if (op == 'minus')		x	x
return a * b;		x	
else if (op == 'mal')			x
return a * b;			x

Tarantula

Code	Testfall 1	Testfall 2	Testfall 3
a = x[0];	x	x	x
b = x[2];	x	x	x
op = x[1];	x	x	x
if (op == 'plus')	x	x	x
return a + b;	x		
else if (op == 'minus')		x	x
return a * b;		x	
else if (op == 'mal')			x
return a * b;			x

Freie Universität Berlin

Übersicht



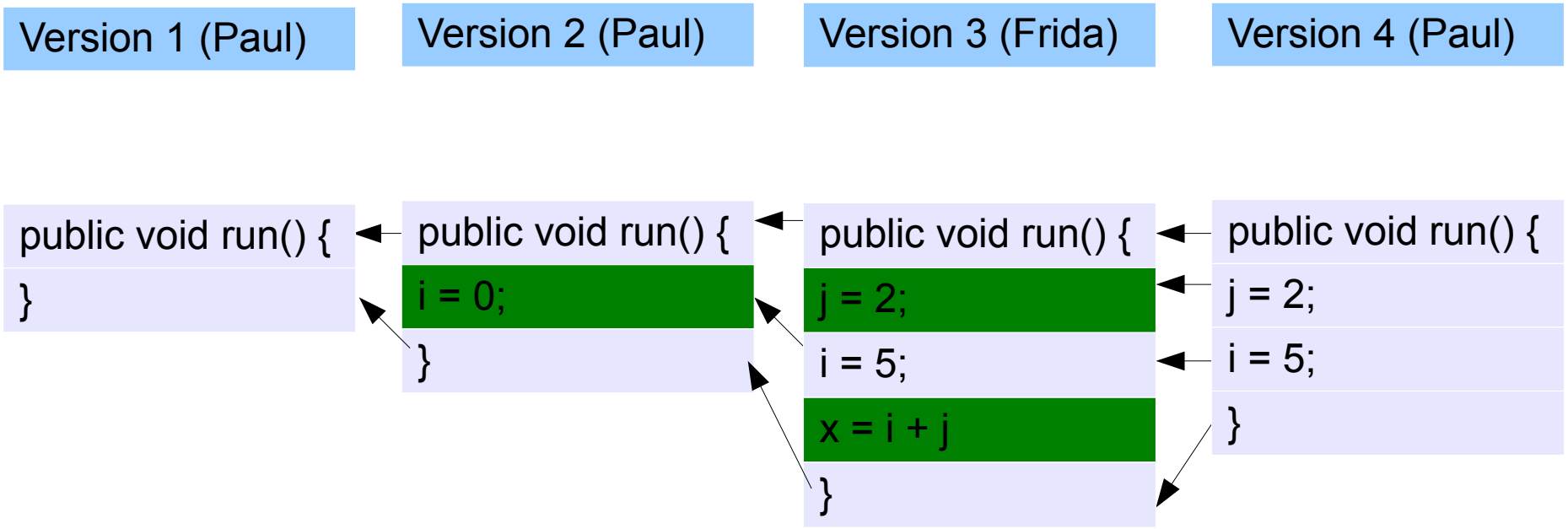
Quellcodegeschichte analysieren

- Quellcode-Repository
- History Graph aufbauen
  - Mehrteiliger Graph
  - Jede Version ein Teil
  - Teile bestehen aus Codezeilen und Autoren
  - Jede Zeile hat 0 oder 1 Vorgänger
  - Vorgänger mit Levenshtein-Distanz ermitteln

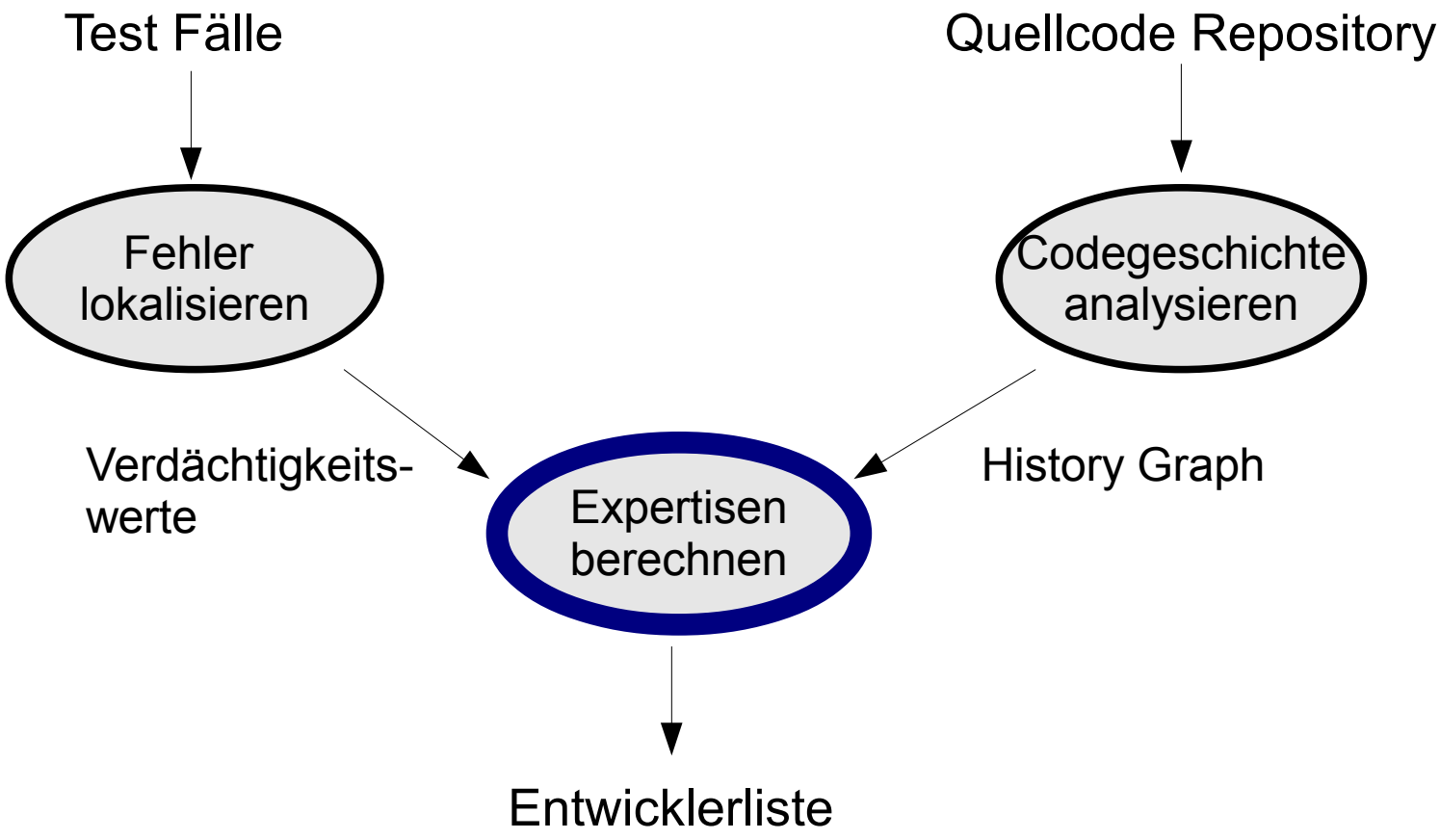
History Graph erstellen

Version 1 (Paul)	Version 2 (Paul)	Version 3 (Frida)	Version 4 (Paul)
public void run() {	public void run() {	public void run() {	public void run() {
}	i = 0;	j = 2;	j = 2;
	}	i = 5;	i = 5;
		x = i + j	}
		}	

History Graph erstellen



Übersicht


Test Fälle
Quellcode Repository
Fehler lokalisieren
Codegeschichte analysieren
Verdächtigkeits-werte
History Graph
Expertisen berechnen
Entwicklerliste

Expertise

- Annahmen:
  - Mehr Änderungen = Höhere Expertise
  - Zeitnahe Änderungen = Höher die Expertise
- Kombination aus Quellcodegeschichte und Fehlerlokalisation
- Nur Zeilen mit Wahrscheinlichkeit > 0 betrachten

Expertise berechnen

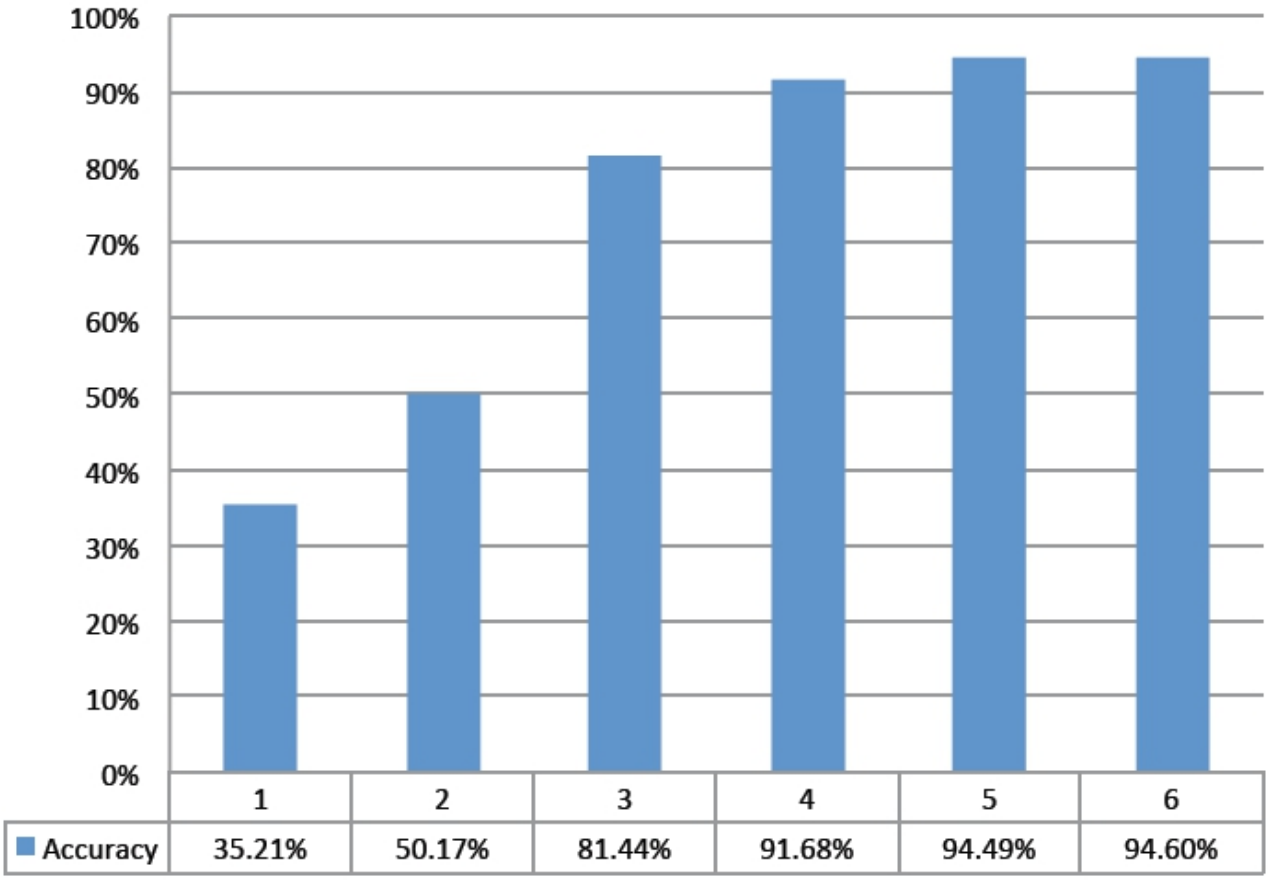
- Für jede Zeile
  - Betrachte Verdächtigkeitswert **v**
  - Wenn **v** > 0
    - Verfolge History Graph
    - Für jede ändernde Version
      - Betrachte Ersteller **e**
      - Berechne Aktualität **a** der Version (0-1)
      - Addiere (**a** * **v**) zum Expertisenwert von **e**

Evaluation

- Ansatz wurde im Tool WhoseFault implementiert
- Angewandt auf AspectJ
  - Ca 10.500 Versionen aus 8,5 Jahren
- Fehlersuche mit IBugs
- Verwendet wurden Testfälle, die in der Vorversion fehlschlagen und in der Folgeversion nicht mehr
- Prüfdaten: 889 Testfälle

Freie Universität Berlin

Evaluation



	1	2	3	4	5	6
Accuracy	35.21%	50.17%	81.44%	91.68%	94.49%	94.60%

Zusätzliche Untersuchungen

- Vergleich mit Anzahl der Commits
  - 4% besser für 1; 40% für 3 Entwickler
- Einfluss der drei Bestandteile

Änderung	Empfehlungen	1	2	3
Historie auf Dateiebene		-1%	-0%	-20%
Lokalisation ohne bestandene Testfälle		+0%	-1%	-0%
Expertise nach Commitzahl		-6%	-9%	-31%

Problem I: Genauigkeit

- 35% für eine Empfehlung sehr gering
- Andere Berechnung als erstes Ansatz
  - Nur ein Experte pro Testfall
    - Ist nur der Beheber ein Experte?
- Nur 8 Entwickler durchschnittlich

Problem II: Praktischer Nutzen

- Keine Nutzbarkeitsstudie
- Für welche Testarten anwendbar?
- Evaluation an nur einem Projekt

Gliederung

- Einführung
- Expertenfindung anhand von Bugreports
- Expertenfindung anhand von Testfällen
- **Abschlussbetrachtung**

Hohe Genauigkeit?

- **Paper I: Bugreports**
  - Zwischen 6% und 60% für Expertengruppen
  - Verbesserung bis auf 70% mit Neuzuweisungen
- **Paper II: Testfälle**
  - Ungefähr 35% für einen möglichen Experten

- Expertenfindung diskutabel in beiden Ansätzen

Automatisierung?

- Beide Ansätze nur als Unterstützung nutzbar
  - Auswahl weiterhin manuell aber mit Vorschlägen
- Vermutlich hilfreicher für unerfahrene Projektmanager und Entwickler, die Hilfe benötigen
- Vorteil Paper II: Fehlerlokalisation könnte gleich mitgeliefert werden

Anwendbarkeit?

- **Paper I:**
  - Bestimmte Vorraussetzung müssen erfüllt sein (Siehe GCC-Beispiel)
  - Testdaten benötigt,
    - kann erst nach einiger Projektlaufzeit eingeführt werden
  - Projektanaylse für Kategorisierung nötig
  - Getestet nur an Open Source Projekten

Anwendbarkeit?

- **Paper II:**
  - Gute Testabdeckung benötigt
  - Von Beginn an anwendbar
  - Nutzen variiert stark mit Teststrategie
  - Instrumentierung notwendig
  - Wenig Anpassungen notwendig
  - Auch nur an Open Source Projekten getestet

Quellen

- **Who Should Fix This Bug?**
  - J. Anvik, L. Hiew, G. C. Murphy (ICSE 2006)
- **Improving Bug Triage With Tossing Graphs**
  - G. Jeong, S. Kim, T. Zimmermann (ESEC/FSE 2009)
- **WhoseFault: Atomatic Developer-To-Fault Assignment Through Fault Localization**
  - F. Servant, J. Jones (ICSE 2012)
- **Visualization Of Test Information To Assist Localisation**
  - J. A. Jones, M. J. Harrold, J. Stasko (ICSE 2002)

Vielen Dank für Ihre Aufmerksamkeit