

Vorlesung Fortgeschrittene Softwaretechnik

Wintersemester 2024/25

Prof. Dr. Stephan Diehl


Informatik

Universität Trier



Plan für die nächsten Wochen

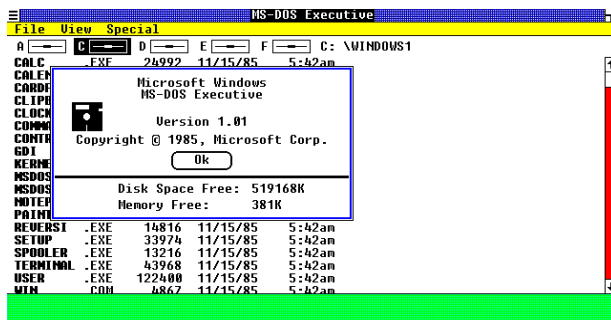
Themenblöcke bisher: Testen, CI, Patterns, VR/AR+SE

	Datum	Thema/Inhalt	Übung			Dozent	Block
DO	12.12.2024	Empirische SE + Software Evolution				SD	Quantitative Studien
DI	17.12.2024	MSR, Empfehlungsdienste	Praxis BOA		Ausgabe LIT	SD	
DO	19.12.2024	???					
	FREI						
DI	07.01.2025	RG MSR/BOA	Übung BOA			SD	
DO	09.01.2025	Research Design + Quantitative Analyse				SD	
DI	14.01.2025	Qualitative Analyse	Praxis: QA 1		Ausgabe LIT	SD	Qualitative Studien
DO	16.01.2025	Praxis: QA 2 (gemeinsames Kodieren)				SD	
DI	21.01.2025	RG QA+SE	Übung QA			SD	
DO	23.01.2025	Info zu Portfolio				SD	
DI	28.01.2025	LLM + SE	Übung LLMSE		Ausgabe LIT	SD	LLM+SE
DO	30.01.2025	LLM + SE				SD	
DI	04.02.2025	RG: LLM+SE	Übung LLMSE			SD	
DO	06.02.2025	-----				SD	
DI	11.02.2025	Abgabe Expose				SD	
DO	13.02.2025	-----				SD	

Was ist Software-Evolution ?

- Synonym für Softwareentwicklungsprozess, das hervorhebt, dass sich ein Softwareprodukt im Lauf seines „Lebens“ ändert.

Windows 1.01



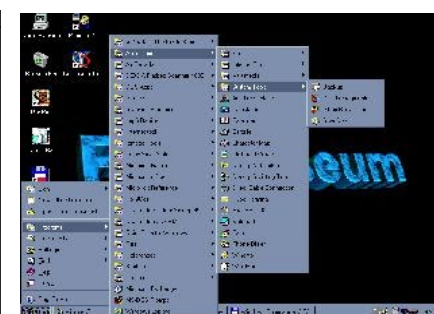
Windows 2.0



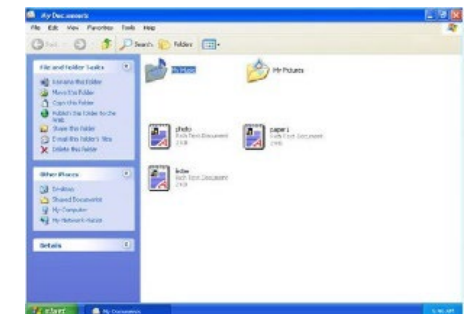
Windows 3.0



Windows 95



Windows XP



Software Aging

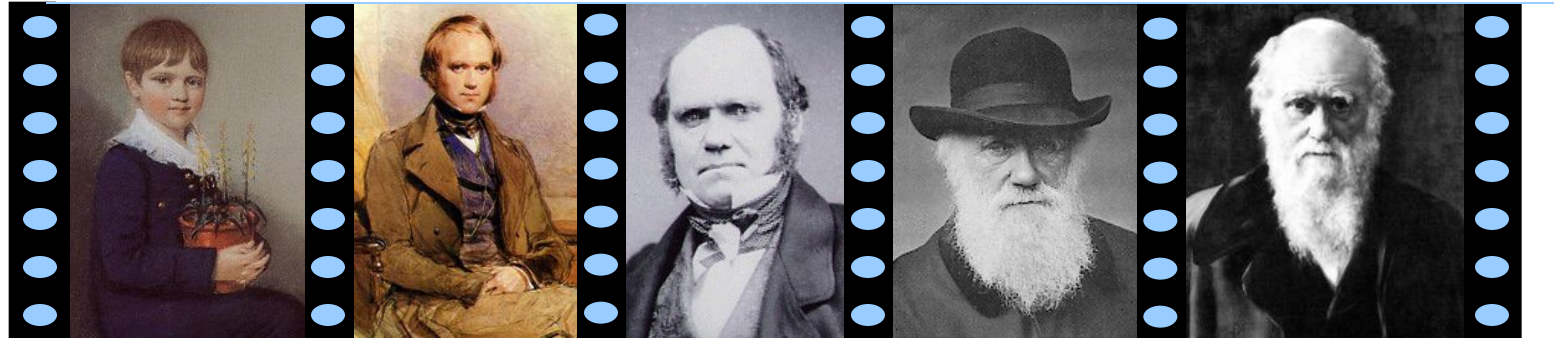
"All successful software gets changed. Two processes are at work. **First, as a software product is found to be useful, people try it in new cases at the edge of or beyond the original domain.** The pressures for extended function come chiefly from users who like the basic function and invent new uses for it. **Second, successful software survives beyond the normal life of the machine vehicle for which it is first written.** If not new computers, then at least new disks, new displays, new printers come along; and the software must be conformed to its new vehicles of opportunity. In short, the software product is embedded **in a cultural matrix of applications, users, laws, and machine vehicles.** These all change continually, and their changes inexorably force change upon the software product."

[Frederick Brooks, "No Silver Bullet", 1987]

"Software aging will occur in all successful products."

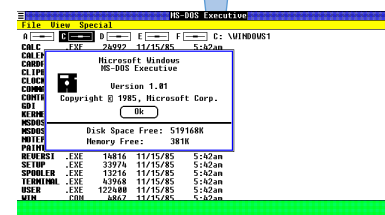
[David L. Parnas, 1994]

Altern



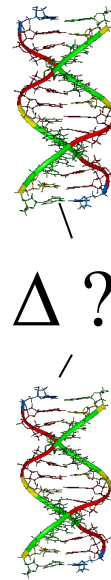
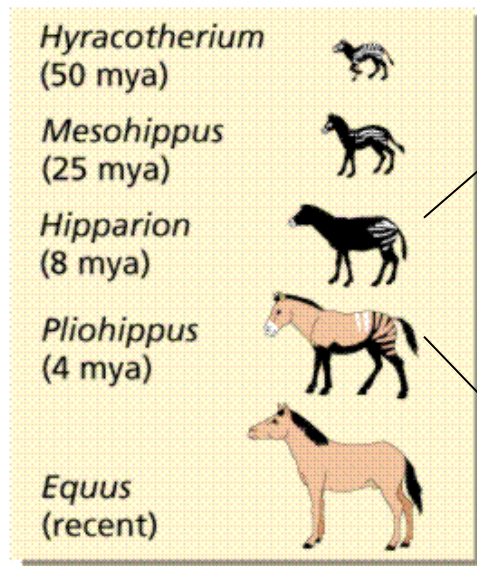
Evolution





Evolution

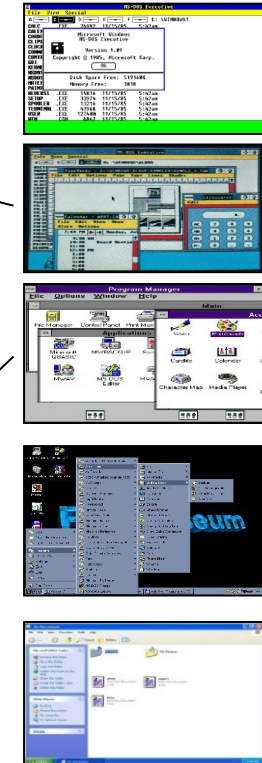
Analogie zum Evolutionsbegriff in der Biologie



```
int nonsense(int x)
{
    x=x+1;
    for (i=1;i<100;i++)
    { x=i*x; }
    return x;
}
```

$\Delta ?$

```
int nonsense(int x)
{
    x=x+1;
    for (i=1;i<100;i++)
    { x=i*x; }
    return x;
}
```



The Genotype/Phenotype Distinction

Quelle: The Stanford Encyclopedia of Philosophy, <http://plato.stanford.edu/entries/genotype-phenotype/>

- The ***genotype*** of an organism is the class to which that organism belongs as **determined by the description of the actual physical material made up of DNA** that was passed to the organism by its parents at the organism's conception.
- The ***phenotype*** of an organism is the class to which that organism belongs as **determined by the description of the physical and behavioral characteristics** of the organism, for example its size and shape, its metabolic activities and its pattern of movement.

Analogie bei Software ???

Evolution: Kode vs. Konzepte

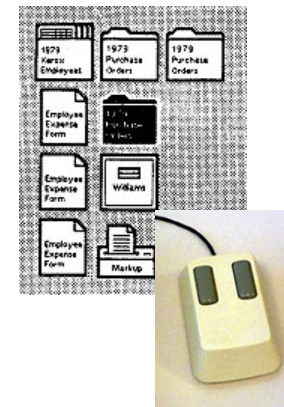
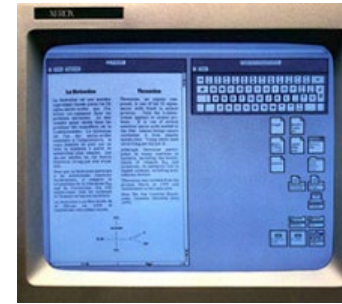
„It is many different **sections of code** ranging from five to 10 to 15 lines of code in multiple places that are of issue, up to large blocks of code that have been inappropriately **copied into Linux** in violation of our source-code licensing contract.“

[Chris Sontag, SCO]

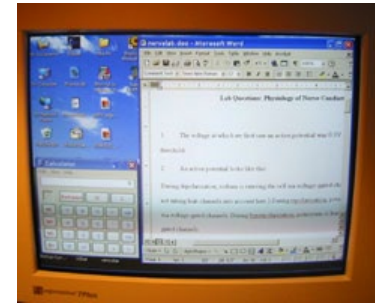


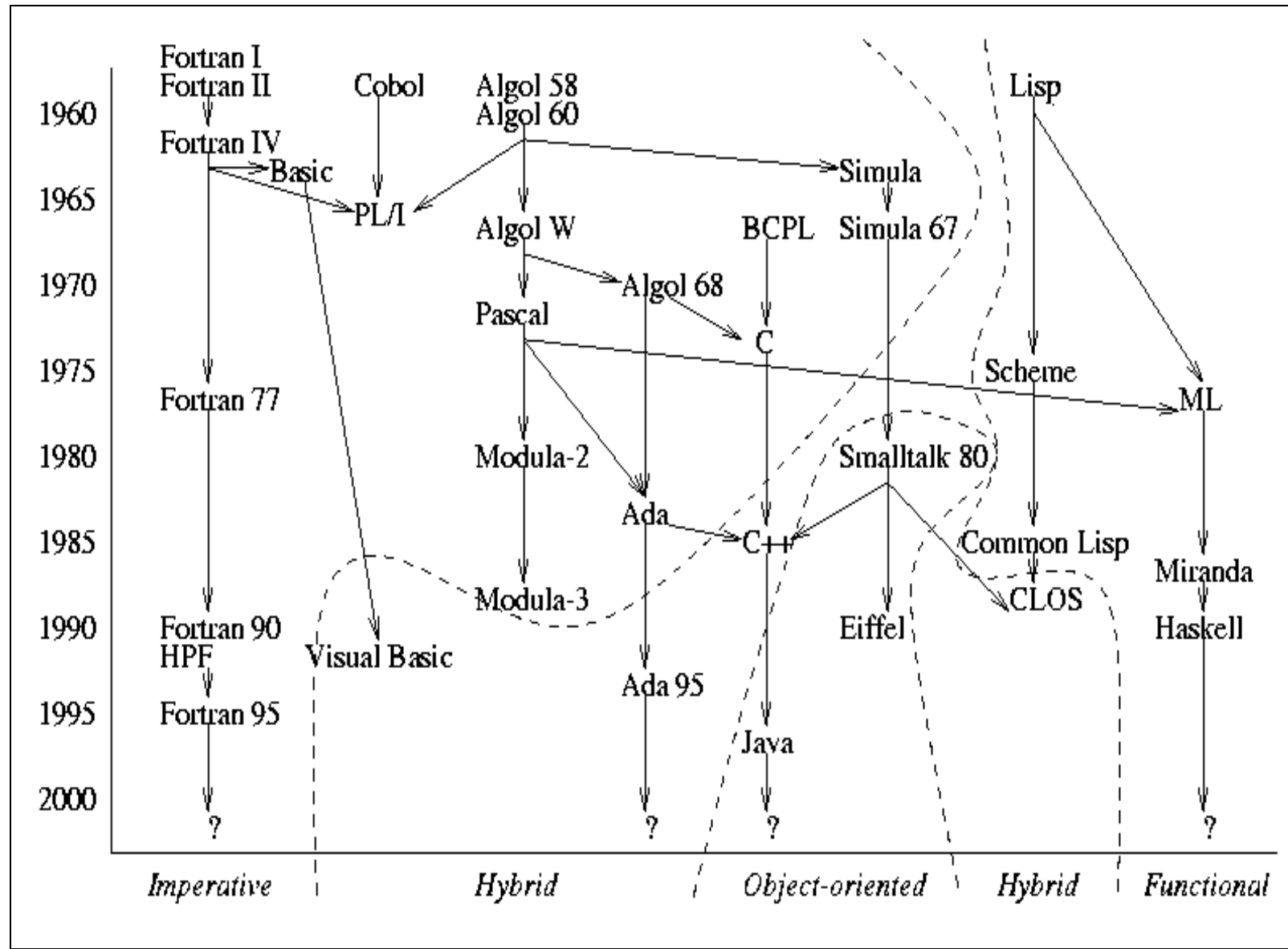
Streitwert: 5 Milliarden US\$

Xerox Star 8010
(1981)



MS Windows PC
(2003)





Lehman's Laws



Meir "Manny" Lehman
1925 – 2010

- entwickelt von Meir M. Lehman (und ursprünglich auch Laszlo A. Belady)
- Gesetze durchliefen selbst von 1968 bis 1997 eine Evolution
- zumindest für die folgenden beiden System empirisch belegt:
 - OS/360 (IBM Mainframe OS der 60er Jahre)
 - Logica FW (System für Finanztransaktionen)

Lehmans Softwaretypen

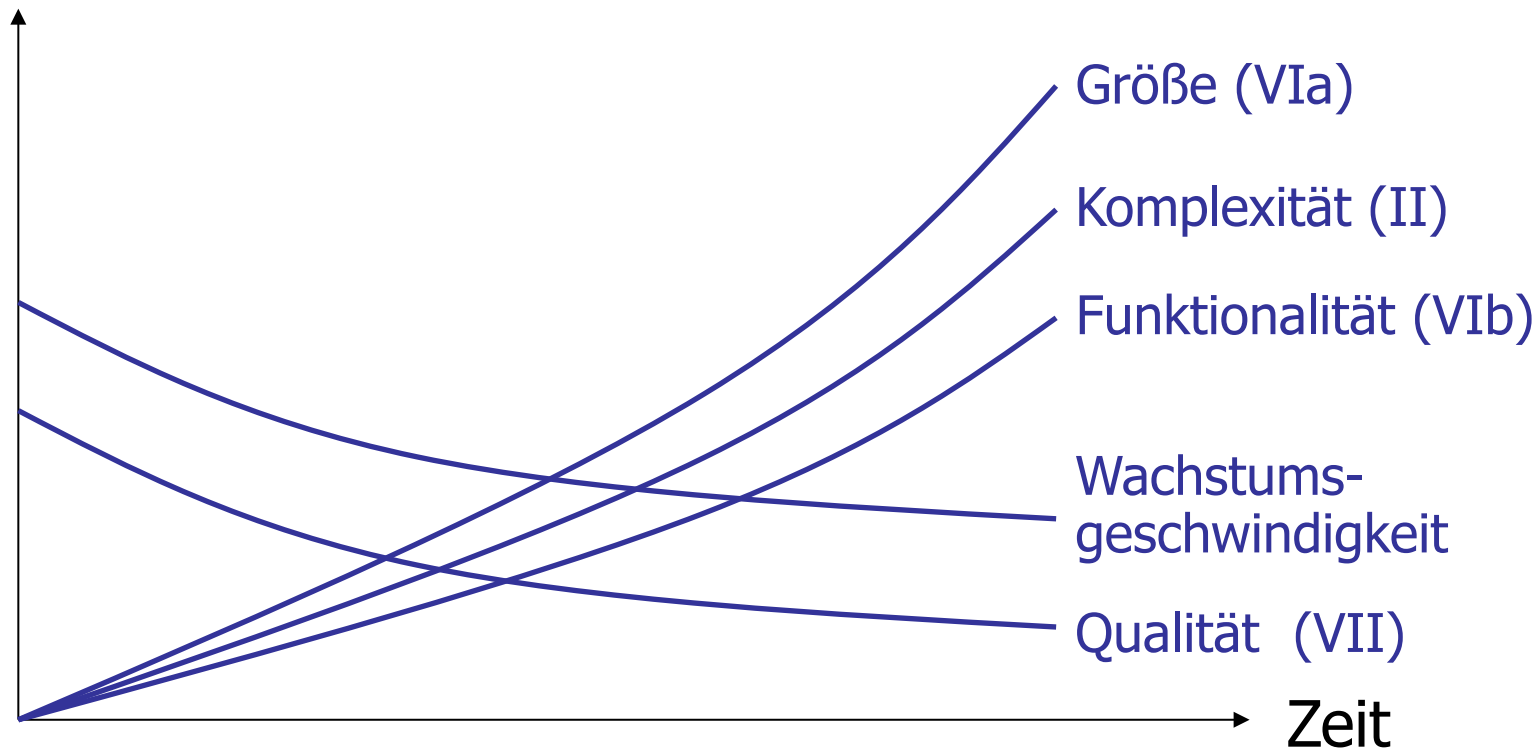
- Reale Software Systeme durchlaufen eine Evolution
- Lehman und Belady unterscheiden 3 Typen von Software:
 - **S-Typ:**
 - vollständig durch eine **formale Spezifikation** beschrieben
 - Entwicklung erfolgreich und abgeschlossen, wenn Spezifikationen erfüllt
 - Beispiel: Programme für numerische Berechnungen, z.B. IEEE Floating Point Standard
 - **E-Typ:**
 - eine **in der realen Welt eingebettete Anwendung**
 - erfolgreich, wenn die Anwender mit der Software zufrieden sind
 - **P-Typ:**
 - löst ein **spezifisches, abgegrenztes Problem**
 - Beispiel: Programme zur Berechnung gegebener Modelle, wie etwa Festigkeitsmodelle in der Statik
 - heute an Bedeutung verloren, da meistens einem der beiden anderen Typen zugeordnet

Lehmans Gesetze der Software-Evolution

- **Gesetz der kontinuierlichen Veränderung (I).** Entweder verändert sich ein Programm während seiner gesamten Lebenszeit kontinuierlich, oder es verliert nach und nach seinen Nutzen.
- **Gesetz der zunehmenden Komplexität (II).** Die Struktur eines Programms wird immer schlechter, was zu einer kontinuierlichen Zunahme der Komplexität führt.
- **Selbstregulierung (III).** Die groben Trends bezüglich Wartbarkeit werden zu einem frühen Entwicklungszeitpunkt festgelegt und können nur in beschränktem Rahmen beeinflusst werden. (vgl: Massenträgheit)
- **Gesetz der Bewahrung der organisatorischen Stabilität (IV).** Der durchschnittliche Arbeitsaufwand, der in die Pflege eines Programms gesteckt wird, ist während der gesamten Lebenszeit des Programms nahezu konstant und unabhängig von den für die Entwicklung eingesetzten Ressourcen.
- **Gesetz der Erhaltung des bekannten Zustandes (V).** Zusätzliches Wachstum wird durch die Notwendigkeit begrenzt, die Vertrautheit in einem System zu erhalten. Dadurch sind die inkrementellen Änderungen in jedem Release während der gesamten Lebenszeit des Systems annähernd konstant.
- **Gesetz des kontinuierlichen Wachstums (VI).** Kontinuierliches Wachstum während der gesamten Lebensdauer ist unerlässlich, wenn das System gebrauchstauglich bleiben soll.
- **Gesetz der abnehmenden Qualität (VII).** Die Qualität eines E-Typ Systems sinkt mit zunehmender Entwicklungsdauer.

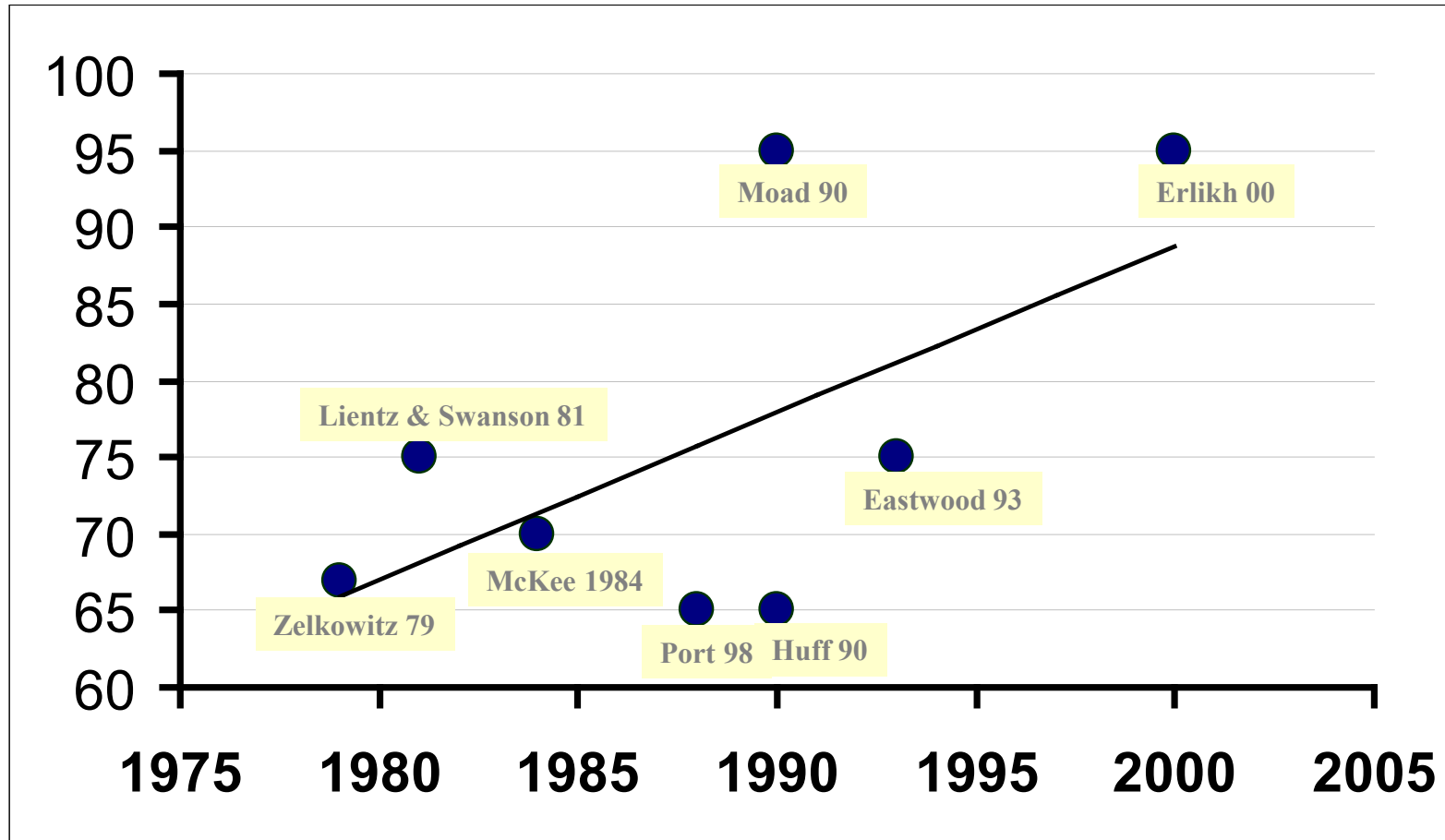
Allgemeine Regeln (E-Typ*)

Lehmans „Gesetze“ -- intuitiv



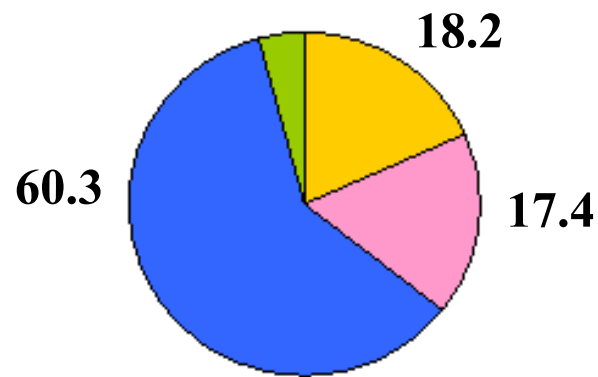
*E-Typ = in der realen Welt eingebettete Anwendung

Percentage of Project Costs Devoted to Maintenance

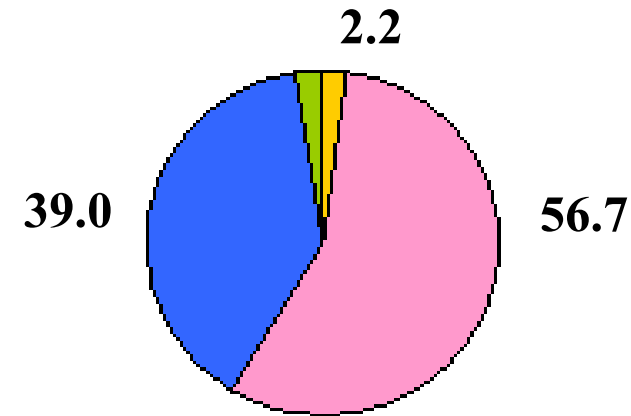


Survey of Software Maintenance Activities

- **Perfective:** add new functionality
- **Corrective:** fix faults
- **Adaptive:** new file formats, refactoring



Lientz, Swanson, Tomhkins [1978]
Nosek, Palvia [1990]
MIS Survey



Schach, Jin, Yu, Heller, Offutt [2003]
**Mining ChangeLogs
(Linux, GCC, RTP)**

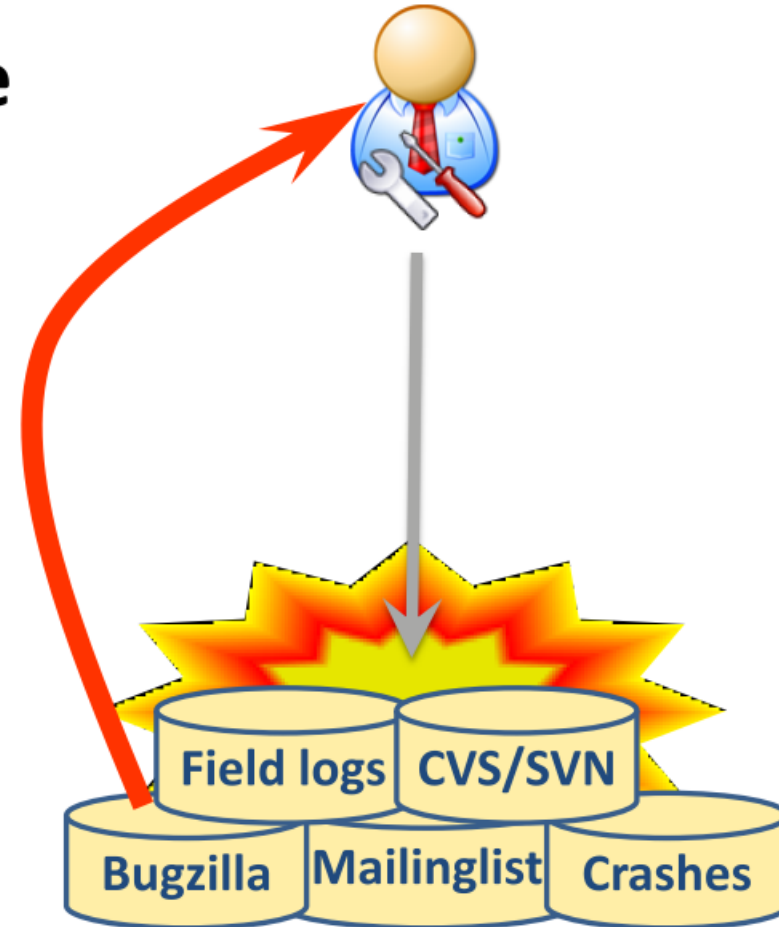
Source: Hassan A. E., Xie T. Mining Software Engineering Data, ICSE 2010, <https://doi.org/10.1145/1810295.1810451>

Fragen zu Lehman's Laws

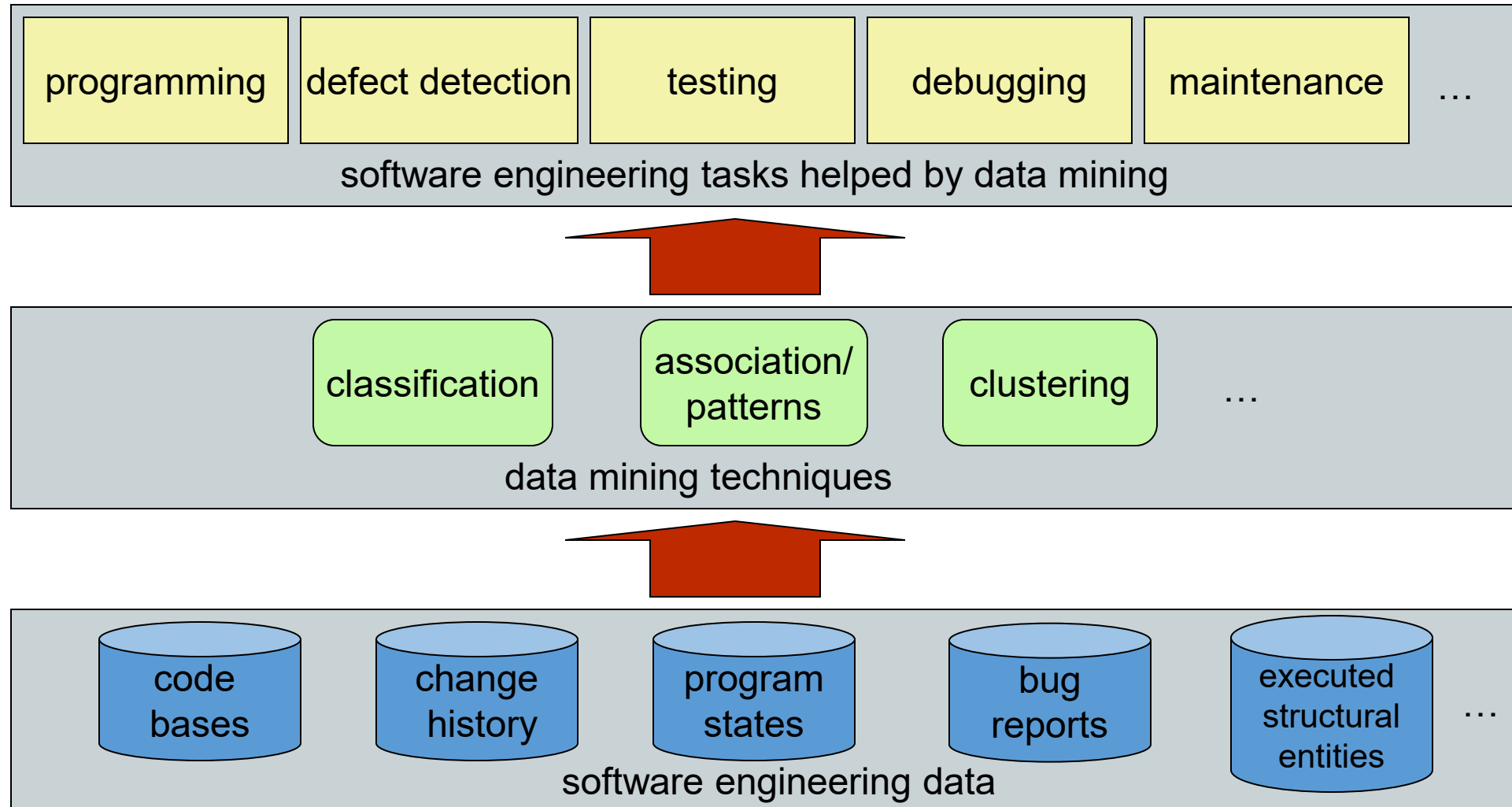
- (Wann) Sind solche "Gesetze" in der Praxis hilfreich?
- Können wir diese "Gesetze" in unseren eigenen Projekten oder in Open-Source Projekten selbst nachweisen?
- Gibt es spezifischere "Gesetze"?
- Gibt es projektspezifische "Gesetze"?

Mining Software Repositories (MSR)

- Transforms static record-keeping repositories to **active** repositories
- Makes repository data **actionable** by uncovering hidden **patterns** and **trends**



Overview of Mining SE Data



Source: Hassan A. E., Xie T. Mining Software Engineering Data, ICSE 2010, <https://doi.org/10.1145/1810295.1810451>

Recommendation Systems (Empfehlungsdienste)



Recommendation systems

- *[Recommendation] systems are software applications that aim to support users in their **decision-making** while interacting with large information spaces. **They recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly.** The ever-expanding volume and increasing complexity of information [...] has therefore made such systems essential tools for users in a variety of information seeking [...] activities. [Recommendation] systems help overcome the information overload problem by **exposing users to the most interesting items, and by offering novelty, surprise, and relevance.***

Quelle: Recommendation Systems for Software Engineering, Martin Robillard, Robert Walker, Thomas Zimmermann, IEEE Software, vol. 27, no. 4, pp. 80-86, July/Aug. 2010.

Recommendation System for Software Engineering

- An RSSE is a software application that provides information items **estimated** to be valuable for **a software engineering task** in a given **context**.

Quelle: Recommendation Systems for Software Engineering, Martin Robillard, Robert Walker, Thomas Zimmermann, IEEE Software, vol. 27, no. 4, pp. 80-86, July/Aug. 2010.

Design Space of RSSE

Informationsquellen

- Source-Code
- Bibliotheken
- Fehlerberichte
- Versionsarchive
- Dokumentation
- Emails
- etc.

Anfrage + Kontext

Empfehlungen

- Code-Fragmente
- Bibliotheken
- Entwickler
- etc.

Recommendation System: eRose

- **Gegeben:** geänderte Dateien/Methoden
- **Gesucht:** relevante andere Dateien/Methoden
- **Methode:** Häufigkeit gemeinsamer Änderungen



ICSE 10 Years Most
Influential Paper Award

Quelle: Mining Version Histories to Guide Software Changes. *Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller*. In IEEE Transactions on Software Engineering (31): June 2005, pp. 429-445.

Evolutionäre Kopplung

- Evolutionäre Kopplung $A \rightarrow B$
 - „Wenn Item A geändert wurde, dann auch Item B“
 - Gegensatz zu klassischer Kopplung
- Wie stark ist eine evolutionäre Kopplung?
 - **Support Count:** Anzahl gemeinsamer Änderungen von A und B
 - **Confidence:** Bedingte Wahrscheinlichkeit $P(B|A)$

Gleichzeitige Änderungen

	<i>S</i>	A	B	C	D	E	F
Proposal	A	8	6	0	0	0	0
Appendices	B	6	7	0	0	0	0
Third Party API	C	0	0	9	3	0	0
Data Acquisition	D	0	0	3	3	2	1
Test Cases	E	0	0	0	2	9	8
Visualization	F	0	0	0	1	8	8

**Support
Count
Matrix**

**Confidence
Matrix**

$$C_{e_1, e_2} = \frac{S_{e_1, e_2}}{S_{e_1, e_1}}$$

<i>C</i>	A	B	C	D	E	F
A	1	6/8	0	0	0	0
B	6/7	1	0	0	0	0
C	0	0	1	3/9	0	0
D	0	0	1	1	2/3	1/3
E	0	0	0	2/9	1	8/9
F	0	0	0	1/8	1	1

More to Explore

Customers who bought [The Rabbi](#) also bought:



[Shaman](#)

by Noah Gordon

Average Customer Review: ★★★★★

Our Price: **\$6.99**

Programmierer, die diese Funktion
änderten, haben auch ... geändert.



[My Big Fat Greek Wedding](#)

DVD ~ Nia Vardalos

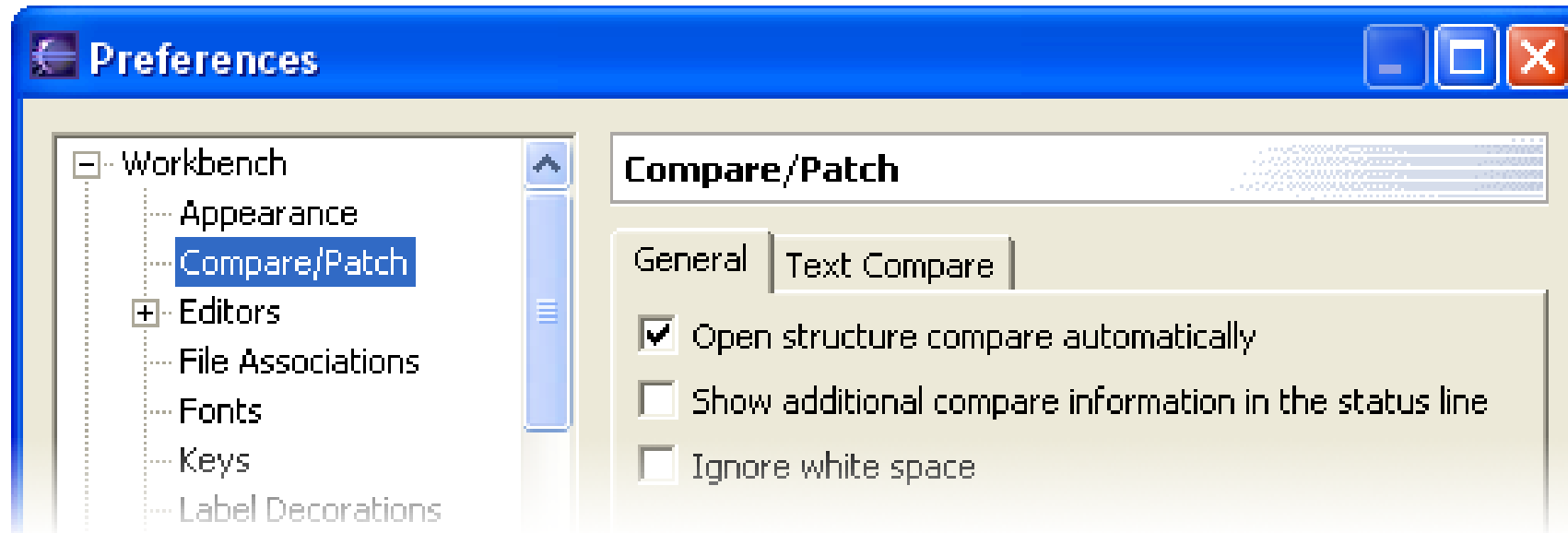
Average Customer Review: ★★★★★

Our Price: **\$20.96**

Präferenzen von Eclipse

Ihre Aufgabe:

Erweitern Sie Eclipse um eine neue Präferenz.



Präferenzen sind im Feld **fkeys[]** gespeichert:

```
*ComparePreferencePage.java x
public final OverlayPreferenceStore.OverlayKey[] fKeys= new OverlayPreferenceStore.OverlayKey[
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, OPEN_STRUCTURE_COM
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SYNCHRONIZE_SCROLL
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_PSEUDO_CONFLI
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, INITIALLY_SHOW_ANC
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_MORE_INFO),
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, IGNORE_WHITESPACE)
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, PREF_SAVE_ALL_EDIT
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, NEW_PREFERENCE),
```

Was müssen Sie noch ändern ?

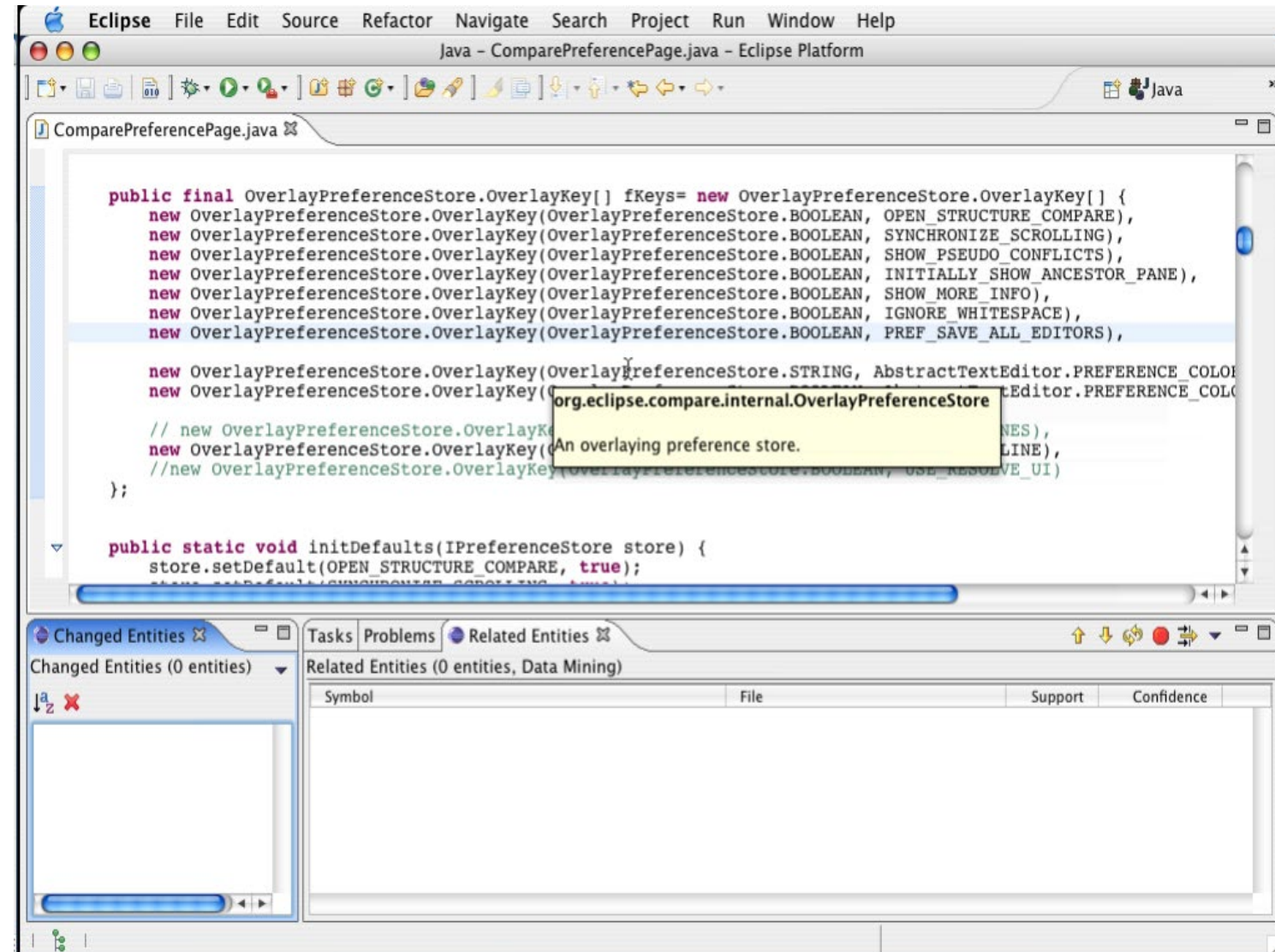
Quellcode von Eclipse

<p>27.000</p> <p>Dateien</p> <p>12,000 kein Java</p>	<p>20.000</p> <p>Klassen</p>	<p>200.000</p> <p>Methoden</p>
--	------------------------------	--------------------------------

Was müssen Sie noch ändern ?

Recommender System: eRose

- [eRose Demo](#)



Klassische Programmanalyse

- liefert Zusammenhänge zwischen Programmstellen, die durch gegenseitige **Benutzung** verbunden sind:
 - **initDefaults()** verwendet die Variable **fkeys[]**
 - Benutzung impliziert aber nicht gemeinsame Änderung
- funktioniert nur mit Programmcode
 - aber 12.000 der 27.000 Eclipse Dateien sind keine Java-Dateien!

Mining on Demand

- Klassisches Data Mining findet **alle Regeln**.
 - Nützlich, um allgemeine Muster zu erkennen.
 - Benötigt hohe Support-Schranke und braucht dann immer noch viel Zeit (3 Tage und länger).
- Alternative: Bedarfsgetriebenes Mining sucht nur **passende Regeln** $X \rightarrow Y$,
 - deren linke Seite eine Situation Σ enthält: $\Sigma \subseteq X$
 - und deren rechte Seite einelementig ist: $Y = \{ a \}$
 - Durchschnittliche Zeit pro Anfrage: 0.5 Sekunden

Recommendation System: Hipikat

- **Gegeben:** Artefakte wie Dateien, Bug-Reports oder Emails
- **Gesucht:** verwandte Artefakte
- **Methode:** Textähnlichkeit
 - berechnet Worthäufigkeitsvektoren, dann Dimensionenreduktion (Latent Semantic Indexing = Worte mit ähnlichem Kontext (latente semantische Ähnlichkeit) werden zusammengefasst), anschließend Kosinus als Ähnlichkeit zwischen Vektoren.

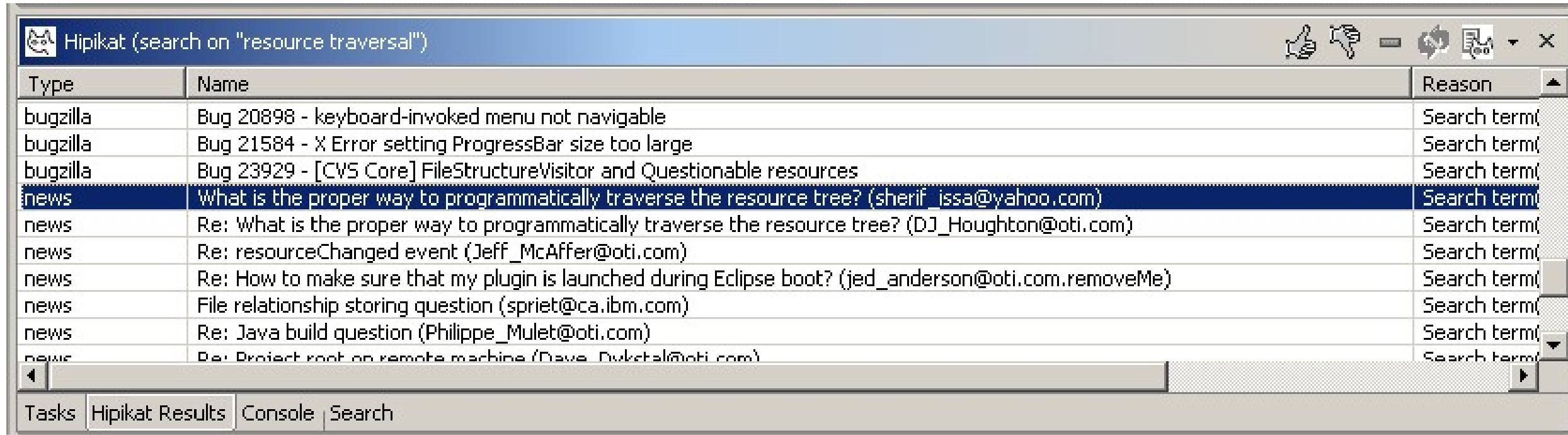
Quelle: Hipikat: recommending pertinent software development artifacts. Davor Cubranic and Gail C. Murphy. In Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, USA, 2003.

Recommendation System: Hipikat

Location in the IDE	Artifact type
Bug report open in the Bugzilla editor	change task
CVS-managed file open in the Java editor	file version
File in the CVS Repository view	file version
Revision in the CVS Resource History view	file version
CVS-managed file in the workspace Navigator	file version
Item recommended in the Hipikat Results view	<i>item's type</i>
Bugzilla search match in the Search results view	change task
Java class or method in Outline and Hierarchy views	file version

- “A user makes a query by selecting an artifact in the Eclipse project workspace and choosing ‘Query Hipikat’ from the context menu.”

Recommendation System: Hipikat



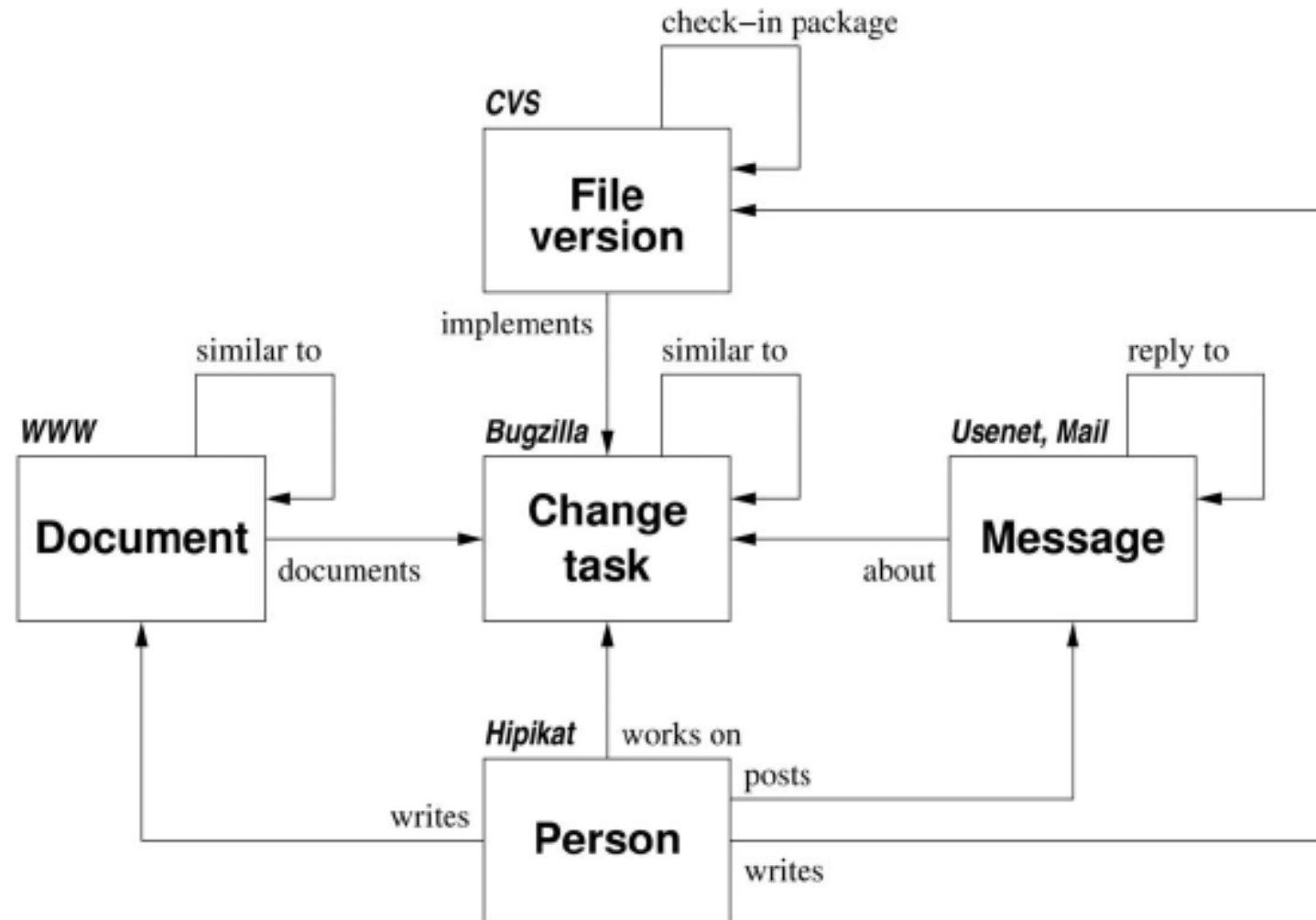
The screenshot shows a window titled "Hipikat (search on 'resource traversal')". It contains a table with three columns: "Type", "Name", and "Reason". The table lists several search results, including bug reports from Bugzilla and news items. The fourth row is highlighted in blue.

Type	Name	Reason
bugzilla	Bug 20898 - keyboard-invoked menu not navigable	Search term(
bugzilla	Bug 21584 - X Error setting ProgressBar size too large	Search term(
bugzilla	Bug 23929 - [CVS Core] FileStructureVisitor and Questionable resources	Search term(
news	What is the proper way to programmatically traverse the resource tree? (sherif_issa@yahoo.com)	Search term(
news	Re: What is the proper way to programmatically traverse the resource tree? (DJ_Houghton@oti.com)	Search term(
news	Re: resourceChanged event (Jeff_McAffer@oti.com)	Search term(
news	Re: How to make sure that my plugin is launched during Eclipse boot? (jed_anderson@oti.com.removeMe)	Search term(
news	File relationship storing question (spriet@ca.ibm.com)	Search term(
news	Re: Java build question (Philippe_Mulet@oti.com)	Search term(
news	Re: Project root on remote machine (Dave_Dykstra@oti.com)	Search term(

At the bottom of the window, there are tabs labeled "Tasks", "Hipikat Results", "Console", and "Search".

- Finden relevanter Texte (email, bug reports, news, etc.) anhand der Ähnlichkeit von Code- und Textfragmenten.

Das Datenmodell von Hipikat

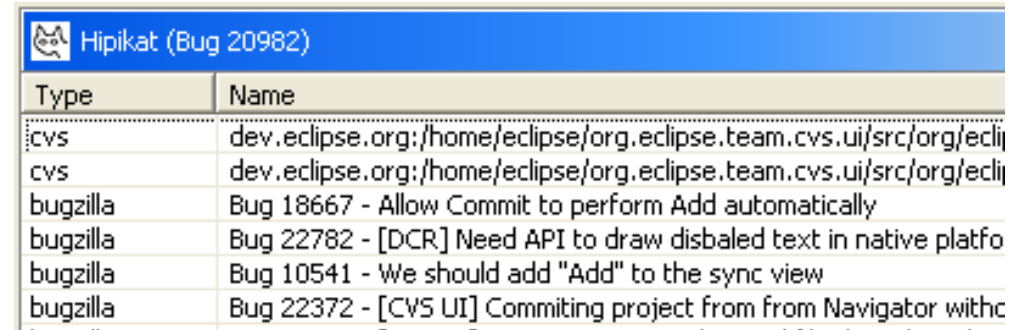


Fast jedes Artefakt im Eclipse Workspace kann als Anfrage dienen



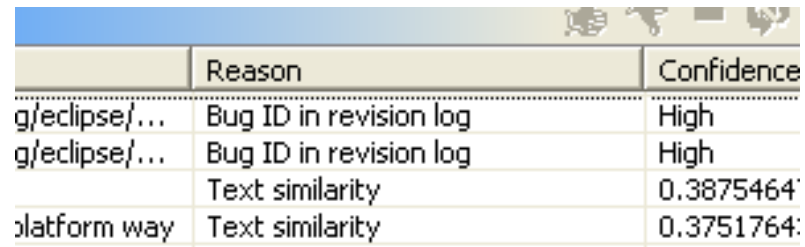
Darstellung der Ergebnisse

- Passende Artefakte werden in der Hipikat-View angezeigt



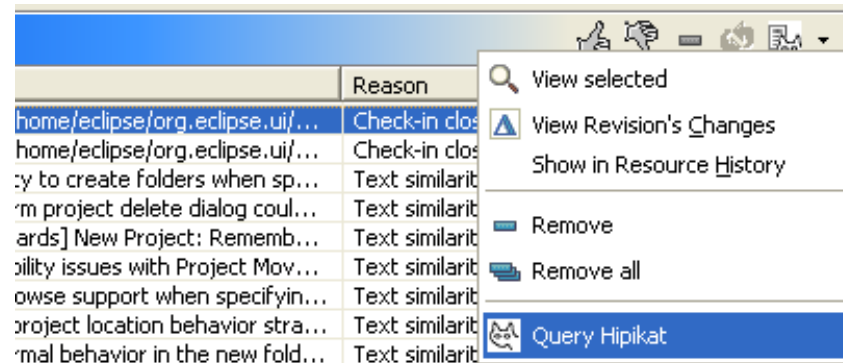
Type	Name
cvs	dev.eclipse.org:/home/eclipse/org.eclipse.team.cvs.ui/src/org/ecli
cvs	dev.eclipse.org:/home/eclipse/org.eclipse.team.cvs.ui/src/org/ecli
bugzilla	Bug 18667 - Allow Commit to perform Add automatically
bugzilla	Bug 22782 - [DCR] Need API to draw disbaled text in native platfo
bugzilla	Bug 10541 - We should add "Add" to the sync view
bugzilla	Bug 22372 - [CVS UI] Committing project from from Navigator withc

... einschließlich Begründung und Konfidenz



	Reason	Confidence
g/eclipse/...	Bug ID in revision log	High
g/eclipse/...	Bug ID in revision log	High
	Text similarity	0.3875464
platform way	Text similarity	0.3751764

... man kann auch die Empfehlungen wieder für Anfragen verwenden



	Reason
home/eclipse/org.eclipse.ui/...	Check-in clo
home/eclipse/org.eclipse.ui/...	Check-in clo
y to create folders when sp...	Text similarit
m project delete dialog coul...	Text similarit
ards] New Project: Rememb...	Text similarit
ility issues with Project Mov...	Text similarit
rowse support when specifyin...	Text similarit
project location behavior stra...	Text similarit
mal behavior in the new fold...	Text similarit

☐ View Revision's Changes
☐ Show in Resource History
☐ Remove
☐ Remove all
☐ Query Hipikat

Recommendation System: PARSEWeb

- **Gegeben:** verfügbare Objekttypen
- **Gesucht:** Code zum Erzeugen eines Objekts eines bestimmten Typs.
- **Empfehlungen:** Folge von Methodenaufrufen zur Erzeugung des gewünschten Objekts.
- **Methode:** Häufig vorkommende Aufrufmuster

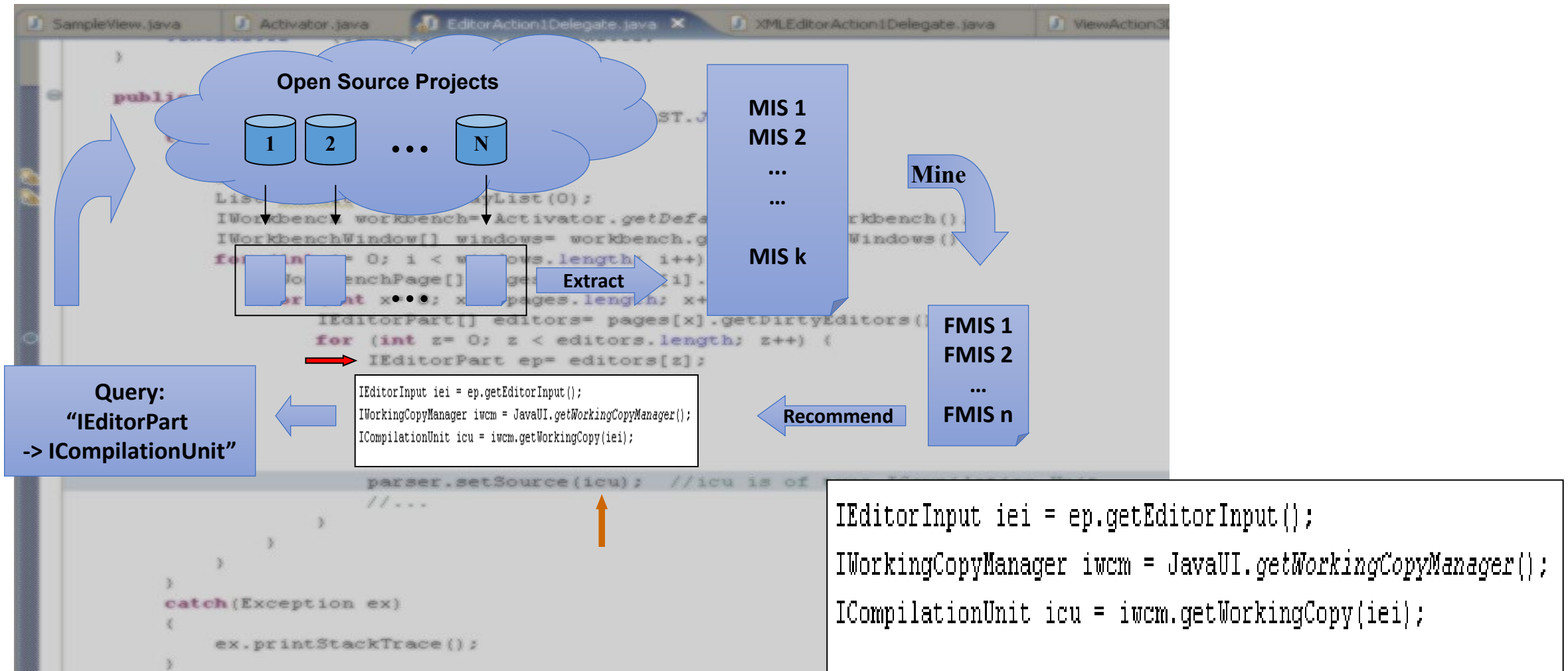
Quelle: Parseweb: a programmer assistant for reusing open source code on the web. Suresh Thummalapenta and Tao Xie. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE '07)*. ACM, New York, NY, USA, 204-213.

Recommendation System: PARSEWeb

*MIS: Method-Invocation sequence

FMIS: Frequent MIS

Aufgabe: Wie “parsed” man in einem “dirty editor” in Eclipse?

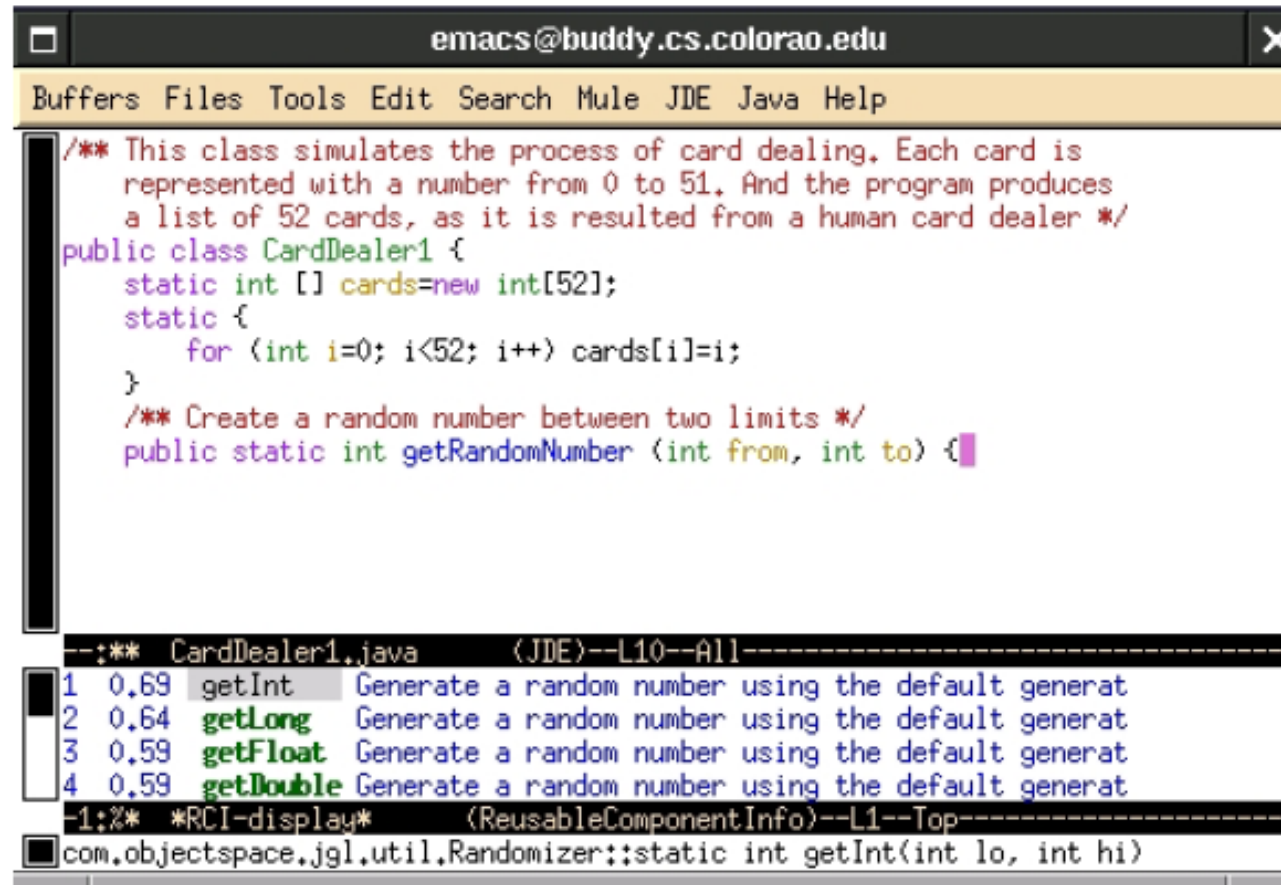


Recommendation System: CodeBroker

- **Gegeben:** Kommentar und Signatur
- **Gesucht:** ähnliche Methode
- **Empfehlungen:** Folge von ähnlichen Methoden (Name+Kommentar)
- **Methode:** Text-Ähnlichkeitsanalyse und Übereinstimmung der Typsignaturen

Quelle: Supporting reuse by delivering task-relevant and personalized information.
Yunwen Ye and Gerhard Fischer. In Proceedings of the 24th International Conference on Software Engineering (ICSE '02). ACM, New York, NY, USA, 2002.

Recommendation System: CodeBroker



The screenshot shows the Emacs IDE window titled 'emacs@buddy.cs.colorao.edu'. The menu bar includes 'Buffers', 'Files', 'Tools', 'Edit', 'Search', 'Mule', 'JDE', 'Java', and 'Help'. The main editor displays a Java class named 'CardDealer1' with a doc comment describing a card dealing simulation. The code includes a static array 'cards' and a 'getRandomNumber' method. Below the editor, a buffer titled 'CardDealer1.java (JDE)--L10--All--' displays recommendations. The first recommendation, 'getInt', is highlighted and matches the task described in the doc comment. Other recommendations include 'getLong', 'getFloat', and 'getDouble'. A second buffer titled '*RCI-display*' (ReusableComponentInfo) shows the source code for 'com.objectspace.jgl.util.Randomizer::static int getInt(int lo, int hi)'.

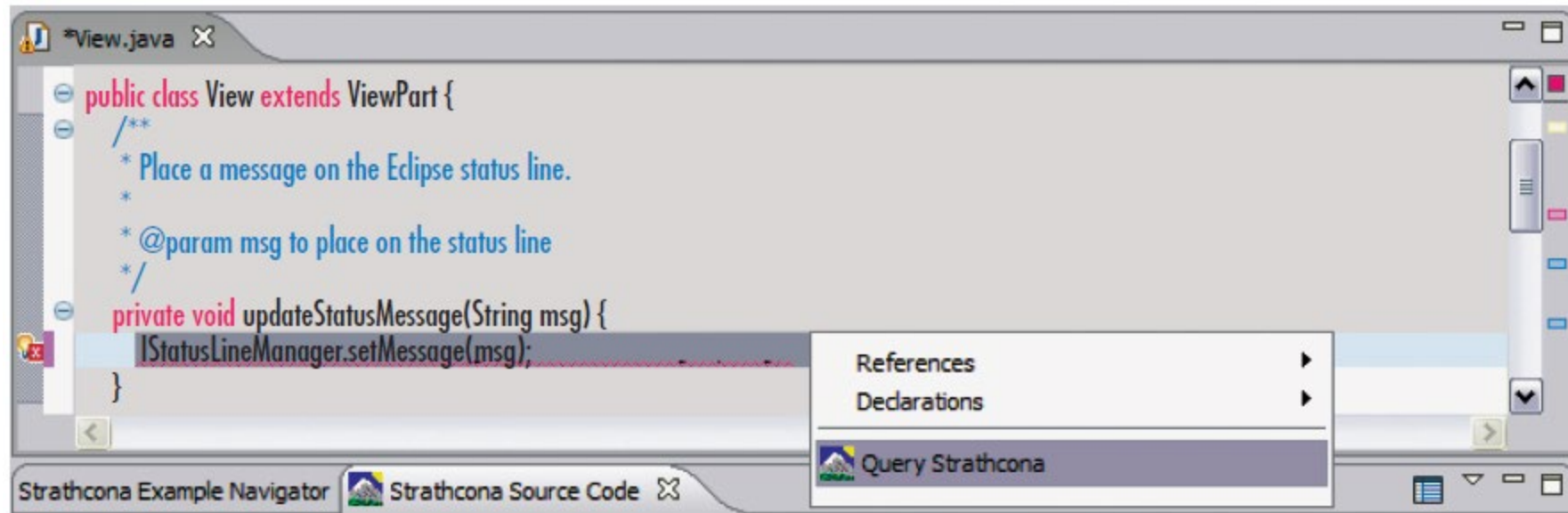
```
/** This class simulates the process of card dealing. Each card is
    represented with a number from 0 to 51. And the program produces
    a list of 52 cards, as it is resulted from a human card dealer */
public class CardDealer1 {
    static int [] cards=new int[52];
    static {
        for (int i=0; i<52; i++) cards[i]=i;
    }
    /** Create a random number between two limits */
    public static int getRandomNumber (int from, int to) {
```

```
--:** CardDealer1.java (JDE)--L10--All-----
1 0.69 getInt Generate a random number using the default generat
2 0.64 getLong Generate a random number using the default generat
3 0.59 getFloat Generate a random number using the default generat
4 0.59 getDouble Generate a random number using the default generat
-1:** *RCI-display* (ReusableComponentInfo)--L1--Top-----
com.objectspace.jgl.util.Randomizer::static int getInt(int lo, int hi)
```

Figure 2: An example of the use of *CodeBroker*

This screen image shows what a developer using *CodeBroker* sees. The developer wants to write a method that creates a random number between two integers, and describes the task in the doc comment and signature before the cursor, based on which several components are delivered in the *RCI-display* (the lower buffer). The first of these delivered components, `getInt`, is a perfect match and can be reused immediately.

Recommendation System: Strathcona

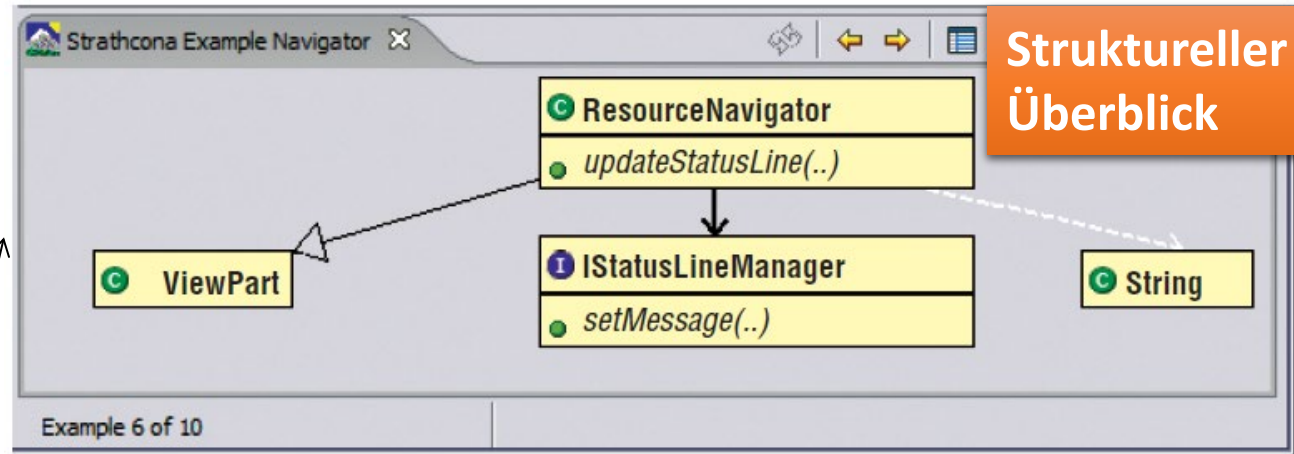


- **Gegeben:** Markiertes Kode-Fragment
- **Gesucht:** relevante Kode-Beispiele
- **Methode:** strukturelle Eigenschaften des Fragments wie referenzierte Typen oder Signaturen aufgerufener Methoden (Fan-Out).

Recommendation System: Strathcona

Anfrage

```
public class View extends ViewPart {  
    /**  
     * Place a message on the Eclipse status line.  
     *  
     * @param msg to place on the status line  
     */  
    private void updateStatusMessage(String msg) {  
        IStatusLineManager.setMessage(msg);  
    }  
}
```



Hervorgehobener
Quelltext

Strathcona Example Navigator

Strathcona Source Code

```
/**  
 * Updates the message shown in the status line.  
 *  
 * @param selection the current selection  
 */  
protected void updateStatusLine(IStructuredSelection selection) {  
    String msg = getStatusLineMessage(selection);  
    getViewSite().getActionBars().getStatusLineManager().setMessage(msg);  
}
```

Begründung

Rationale	Artifact
Class Inherits From	org.eclipse.ui.part.ViewPart
Method Calls Method	org.eclipse.jface.action.IStatusLineManager.setMessage(Ljava.lang.String;)
Method Uses Type	org.eclipse.jface.action.IStatusLineManager
Method Uses Type (S)	java.lang.String

OK Cancel

Weitere Empfehlungsdienste

- **Dhruv** empfiehlt Personen und Artefakte die relevant für einen Bug-Report sind [Ankolekar et al., WWW 2006]
- **Expertise Browser** empfiehlt Personen aufgrund vergangener Änderungen am Quellcode oder der Dokumentation [Mockus&Herbsleb, ICSE 2002].
- ...

RSSE design dimensions

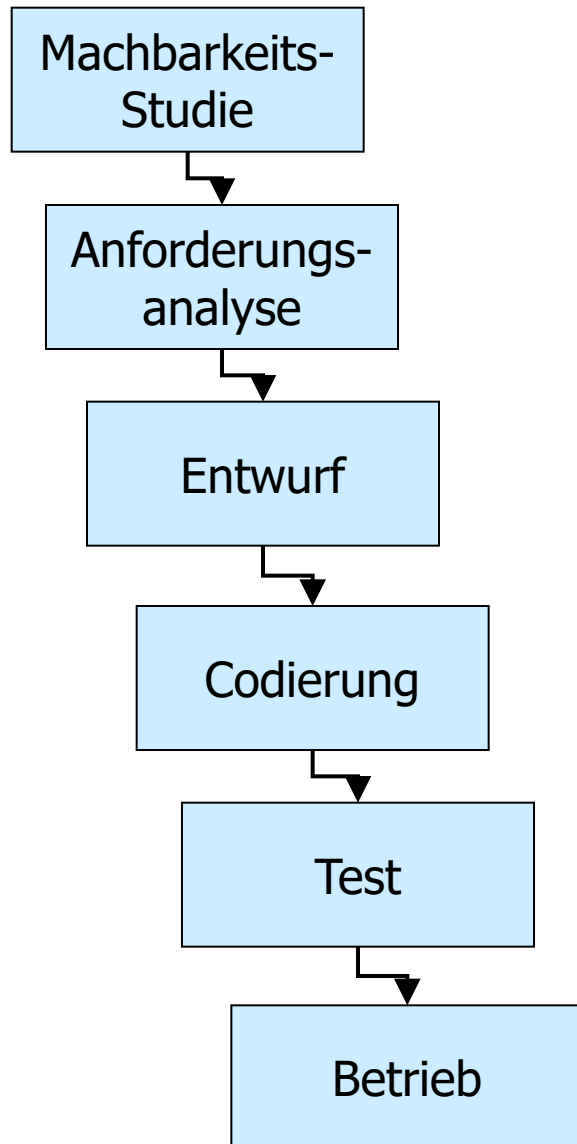
Nature of the context	Recommendation engine	Output mode
Input: explicit implicit hybrid	Data: source change bug reports mailing lists interaction history peers' actions	Mode: push pull
	Ranking: yes no	Presentation: batch inline
	Explanations: from none to detailed	
User feedback: none locally adjustable individually adaptive globally adaptive		

Quelle: Recommendation Systems for Software Engineering, Martin Robillard, Robert Walker, Thomas Zimmermann, IEEE Software, vol. 27, no. 4, pp. 80-86, July/Aug. 2010.

Erläuterungen zur Tabelle

- **Push:** Benutzer stellt explizite Anfrage
- **Pull:** System macht ständig Vorschläge
- **Batch:** Alle Empfehlungen in einer separaten Darstellung (z.B. Liste)
- **Inline:** Vorschläge werden bei den betroffenen Artefakten angezeigt
- **Locally adjustable:** Der Benutzer kann das System für seinen Kontext anpassen
- **Individually adaptive:** Das System passt sich automatisch aufgrund expliziten oder impliziten Feedbacks des Benutzers an
- **Globally adaptive:** wie oben, aber das Feedback anderer Benutzer wird auch berücksichtigt.

Einfaches Wasserfall-Modell



Aufgabe:

Welche Daten fallen an?

Welche Empfehlungen könnten berechnet werden?

- Compiler
- Entwicklungsumgebungen
- Debugger
- Profiling
- Build-Prozess
- Konfigurationsmanagement
- Issue-Tracking
- Kommunikation
- Dokumentationswerkzeuge
- Projektmanagement
- Deployment (InstallShield, PackageManagement, AutoUpdate)
- Entwurfswerkzeuge