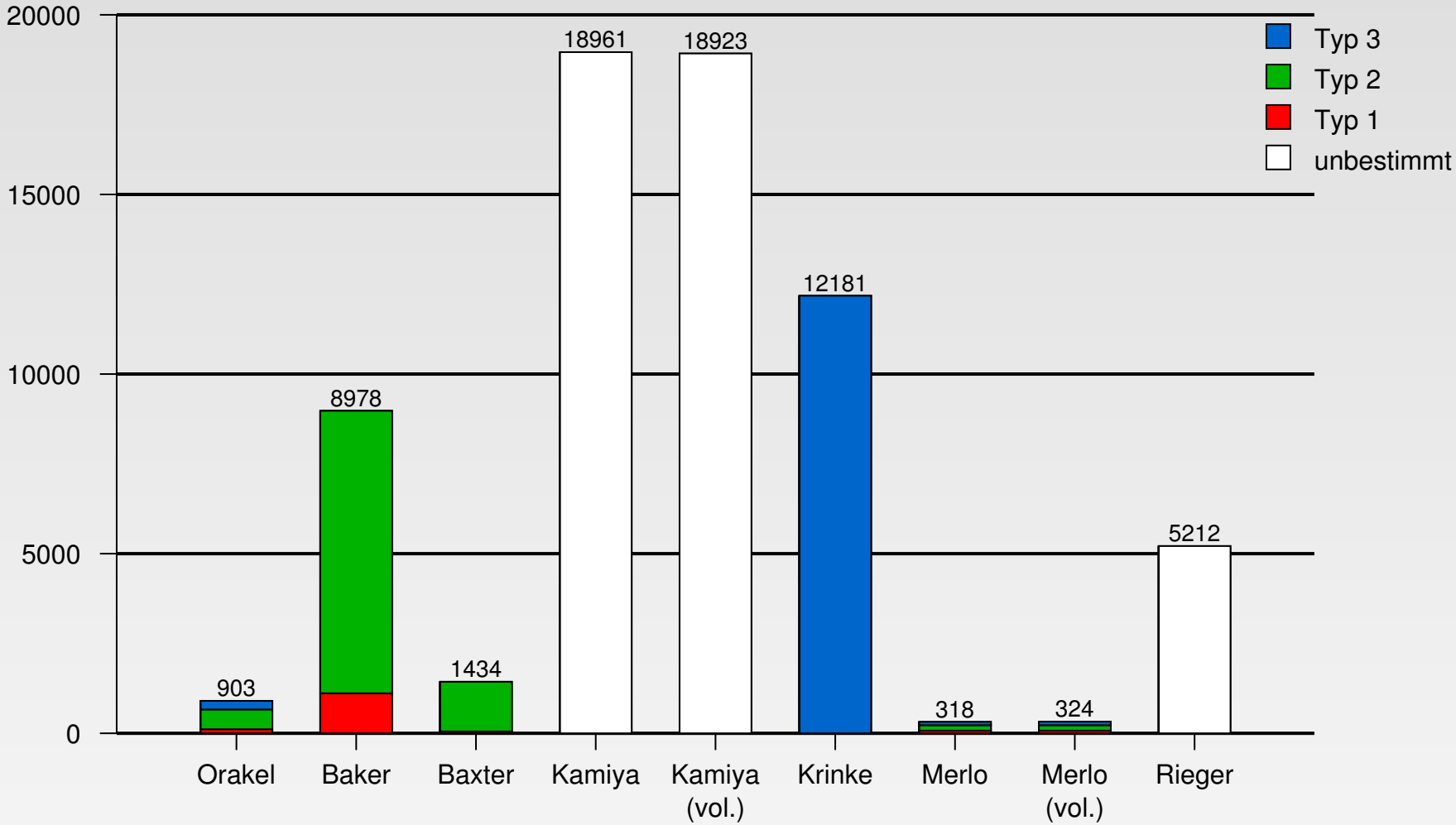
Vergleich von Klonerkennungstechniken (Bellon 2003)

Teilnehmer	Referenz	Ansatz
Brenda S. Baker	(Baker 1995)	Suffix-Tree, tokenbasiert
Ira D. Baxter	(Baxter u. a. 1998)	Subtree-Matching im AST
Toshihiro Kamiya	(Kamiya u. a. 2002)	Eingabetransformation, Suffix-Tree, tokenbasiert
Jens Krinke	(Krinke 2001)	Program Dependence Graph
Ettore Merlo	(Mayrand u. a. 1996)	Funktions-Metriken + Token-Vergleich
Matthias Rieger	(Ducasse u. a. 1999)	Pattern Matching auf Dotplots (textuell)

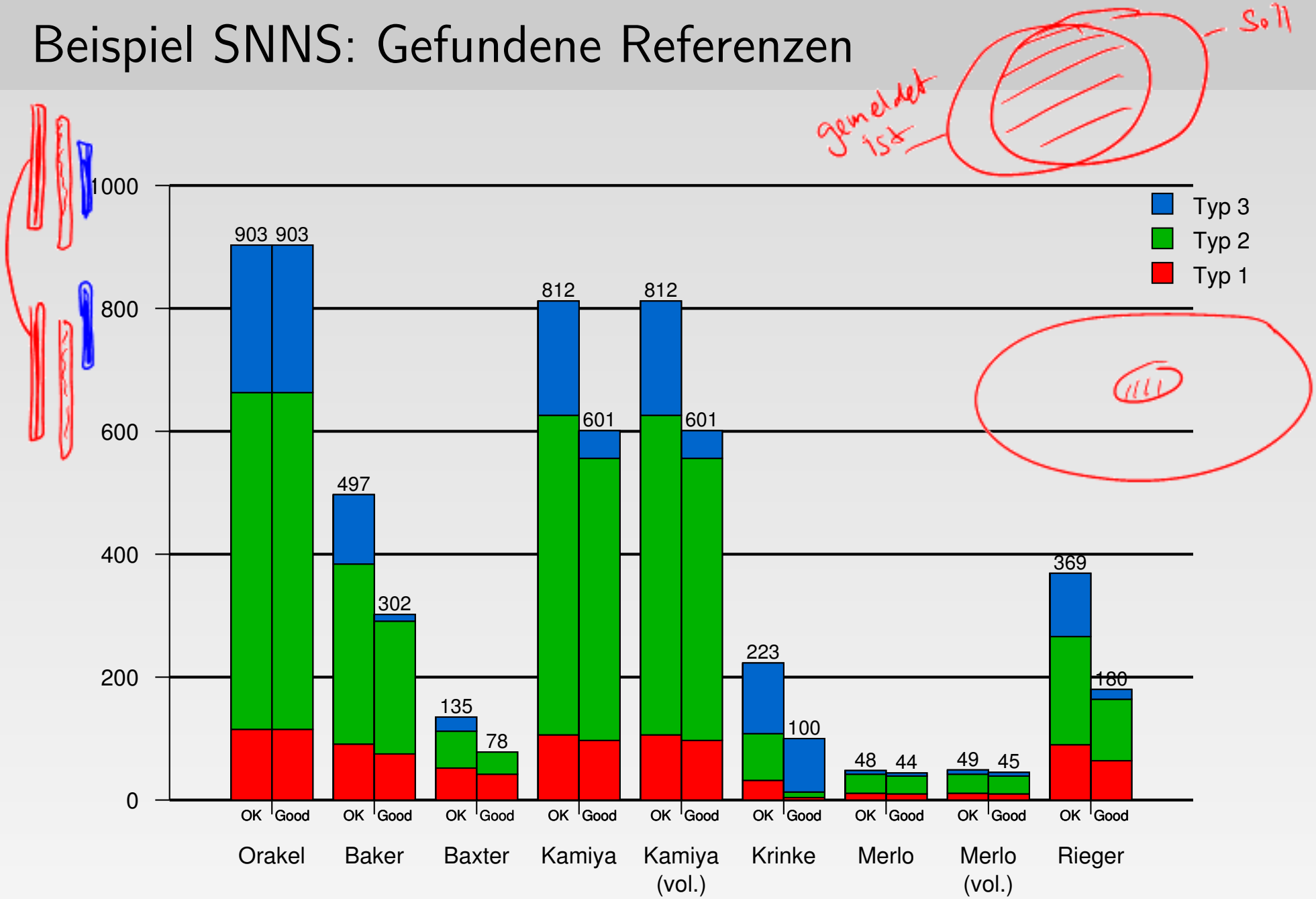
Untersuchte Software-Systeme

Projekt	Sprache	Größe (SLOC)	Kandidaten	Referenzen
weltab	C	11K	13901	252
cook	C	80K	27122	402
snns	C	115K	66331	903
postgresql	C	235K	59114	555
netbeans-javadoc	Java	19K	7860	55
eclipse-ant	Java	35K	2440	30
eclipse-jdtcore	Java	148K	92905	1345
j2sdk1.4.0-javax-swing	Java	204K	56262	777

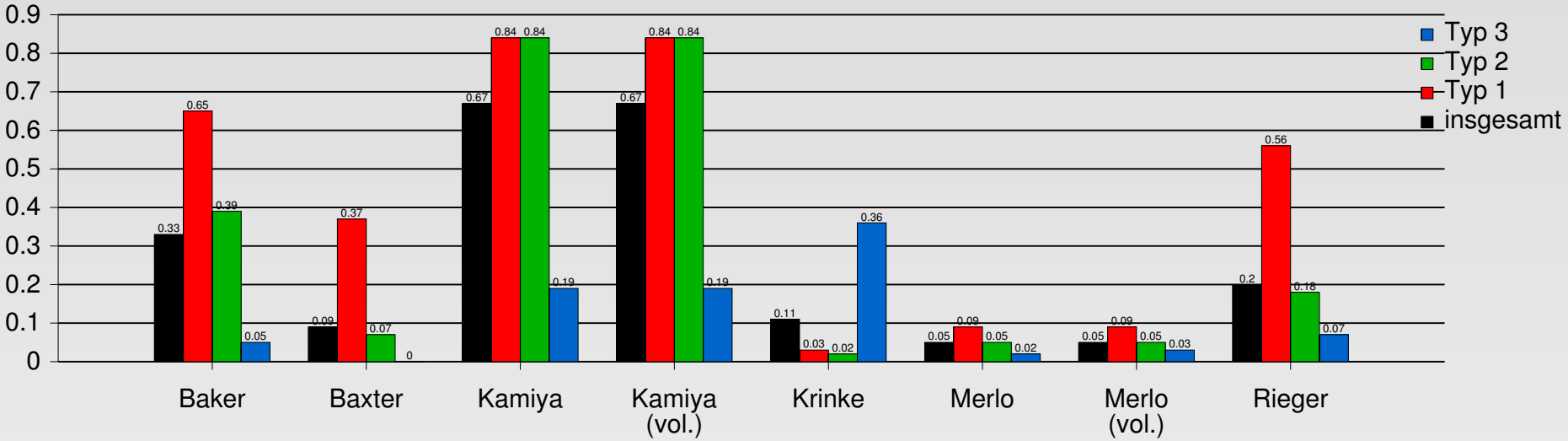
Beispiel SNNS: Kandidaten

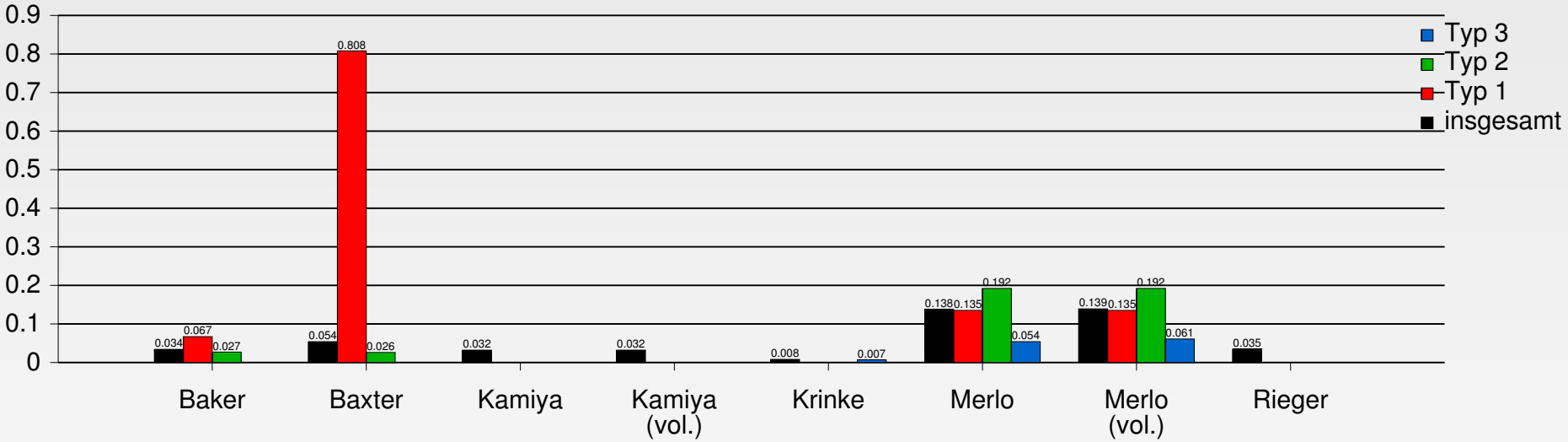


Beispiel SNNS: Gefundene Referenzen



Beispiel SNNS: Recall (oben) und Precision (unten)





Zusammenfassung

	Baker	Baxter	Kamiya	Krinke	Merlo	Rieger
Ansatz	Token	AST	Token	PDG	Metrik	Text ~~Token~~
Erkennt Typ	1, 2	1, 2	1, 2, 3	3	1, 2, 3	1, 2, 3
Klassifiziert	1, 2	1, 2	–	3	1, 2, 3	–
Geschwind.	+ +	-	+	- -	+ +	?
Speicher	+	-	+	+	+ +	?
Recall	+	-	+	-	-	+
Precision	-	+	-	-	+	-
Versteckte	42 %	28 %	46 %	4 %	24 %	31 %

Software Visualisierung

2. Software Visualisierung
   - Lernziele
   - Reengineering Kontext
   - Programm-Visualisierung
   - Statische Programm-Visualisierung
   - Dynamische Programm-Visualisierung
   - Visualisierung von Metriken
   - Klassenblaupause
   - Evolutionäre Aspekte
   - Zusammenfassung

Über diese Folien

Diese Folien entstammen einer Präsentation von Michele Lanza (Universität Bern), mit dessen freundlicher Genehmigung.
Die Unterschiede zum Original sind:

- Übersetzung ins Deutsche
- Kleinere Restrukturierungen und Verkürzungen
- Zusatz Spektrograph (Courtesy Jingwei Wu, Richard C. Holt, Ahmed Hassan, University of Waterloo, Canada)

Software-Visualisierung I

- Lernziele
  - Software-Visualisierung in einem Reengineering-Kontext
  - Statische Code-Visualisierung
  - Dynamische Code-Visualisierung
  - Visualisierung von Metriken
  - Visualisierung der Evolution
  - Leichtgewichtige Ansätze
- Kontext
  - Reengineering ist meist interaktiv
  - Große Datenmengen müssen verstanden werden

Software-Visualisierung II

Software Visualization is the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software.

– Price, Baecker and Small, Introduction to Software Visualization

Software is intangible, having no physical shape or size. Software visualisation tools use graphical techniques to make software visible by displaying programs, program artifacts and program behaviour.

– Thomas Ball

SV im Reengineering-Kontext

- Ziele: Reduktion der Komplexität
- Herausforderungen:
  - Skalierbarkeit
  - Aufgabenabhängigkeit der Visualisierung
  - Art der Visualisierung
  - Begrenzte Ressourcen

Programm-Visualisierung

Program visualization is the visualization of the actual program code or data structures in either static or dynamic form.

– [Price, Baecker und Small]

- Gebiete
  - Statische Programm-Visualisierung
  - Dynamische Programm-Visualisierung
- Aufgaben
  - Verschiedene Sichten generieren
  - Inferenzen ermöglichen
- Spezifische Probleme (aktives Forschungsgebiet)
  - Effiziente Ausnutzung des Platzes, Kantenüberschneidungen, Layout-Probleme, Fokus, Human-Computer-Interaction, ...
  - Keine Konventionen (Farben, Symbole, Interpretation, ...)
- Granularität?
  - Ganze System, Subsysteme, Module, Klassen, Hierarchien,...
- Wofür, wie und wann anzuwenden?

Statische Programm-Visualisierung

- Visualisierung von Information, die statisch abgeleitet ist.
- Hängt von Sprache und Sprachparadigma ab:
  - objektorientierte Sprachen: Klassen, Methoden, Attribute, Vererbung, ...
  - prozedurale Sprachen: Prozeduren, Aufrufe, ...
  - ...

Klassendiagramme