Verteidigung Masterarbeit

Evaluating the Use of a Web Browser to Unify GUI Development for IDE Plug-ins

Christian Cikryt
Freie Universität Berlin

13.08.2015

Überblick

Freie Universität Berlin

Motivation und Ziele

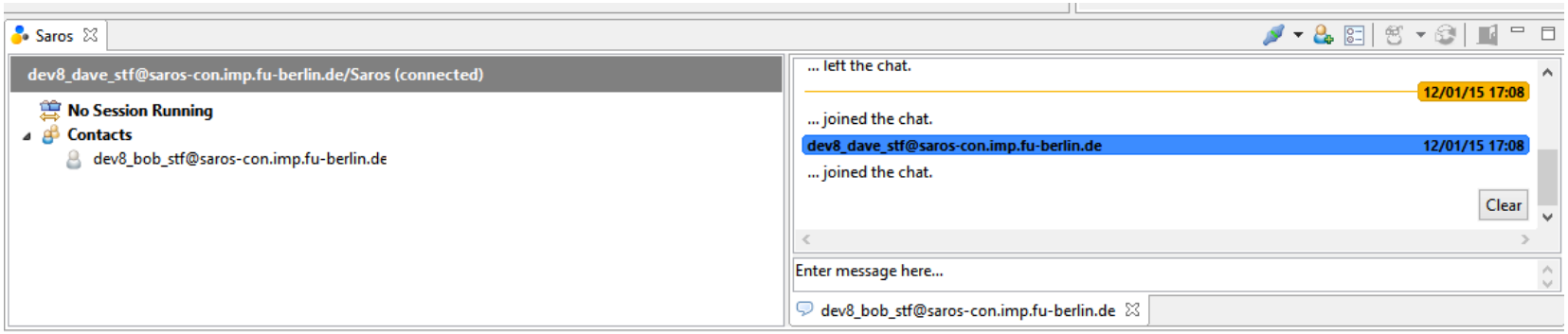
Evaluation des Browser-basierten Ansatzes

Auswertung der Prototyp-Implementierung

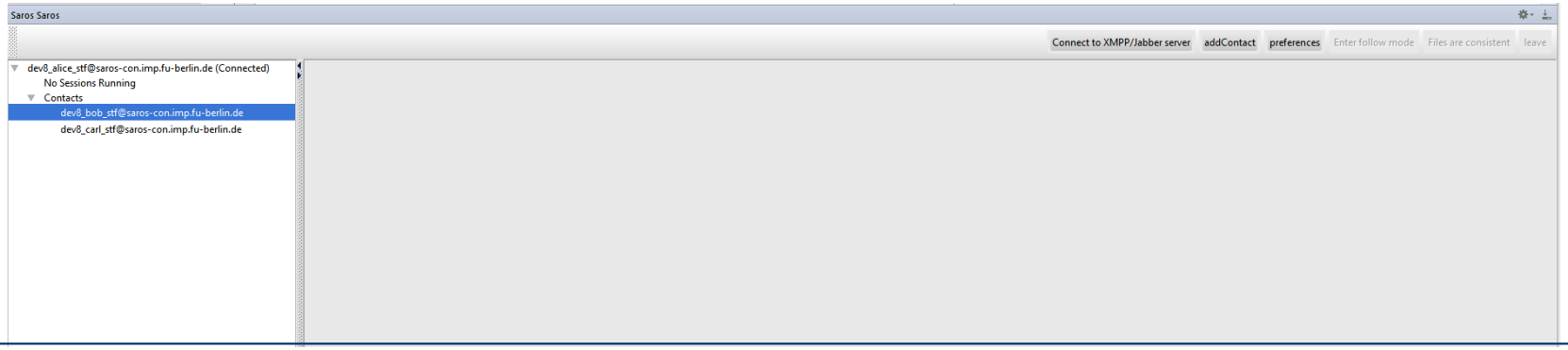
Zusammenfassung und Ausblick

Portierung von Saros nach IntelliJ

SWT



Swing



Motivation einer HTML-basierten GUI

- Vermeidung von doppeltem GUI-Code
- Vereinheitlichung der Oberfläche
- Erfahrung mit SWT Browser in Eclipse
- modernere Anzeigetechnologie
- möglicherweise Performanzvorteile

Ziele dieser Arbeit

Bewertung des HTML-basierten Ansatzes:

- Gegenüberstellung mit getrennter SWT-Swing-Entwicklung
- Realisierbarkeit durch Prototyp beantworten
  - alle grundlegenden Mechanismen auf technischer Ebene
  - Probleme aufdecken (Stabilität und Kompatibilität)
- Gerüst für spätere Implementierung vorgeben
- Testbarkeit der GUI untersuchen

Evaluation

Überblick über anstehende Evaluationsentscheidungen

Freie Universität Berlin

1. Browser vs.

   getrennte Swing und SWT Entwicklung

2. Auswahl der Browserkomponente

Vergleich Browser mit konventionellem Ansatz

Kriterium	Browser	SWT und Swing
Aufwand (initial)	–	0
Evolvierbarkeit	++	–
Codeduplikation	++	–
Kompatibilität	–	++
GUI-Tests	+	–

Entscheidung für Browser-basierten Ansatz:

- ⇒ Aufwand vertretbar und interessantes Wissen
- ⇒ Wartbarkeit und Vermeidung von Duplikation
- ⇒ Kompatibilität als Herausforderung für den Prototypen

Mögliche Java-Browser

- Zwei Java-GUI-Technologien: Swing und SWT
- Browser muss in beide eingebettet werden
- Auswahl an Java-Browsern begrenzt:
  - SWT Browser
  - JxBrowser (Swing)
  - JavaFX Webview
  - Native Swing (SWT in Swing)

Motivation: SWT Browser erkennt vollständiges Laden einer Webseite nicht zuverlässig

⇒ z. B. kritisch bei Benutzung von Javascript-Bibliotheken

SWT-Browser-Erweiterung:

⇒ Erkennung und Verzögerung von Funktionsaufrufen bis definierter Zustand erreicht ist

⇒ Zusätzlich: verbesserte Log- und Debuggingausgabe, nützliche Methoden (z. B. Look-and-Feel) ...

SWT Browser

- verbreitetster Java-Browser
- Erfahrung durch Björn Kahlerts Widgets
- in Swing einbettbar
  - ⇒ identischer Browser-Code für beide IDEs
- Zugriff auf installierten Browser
  - ⇒ abhängig vom Betriebssystem
  - ⇒ Cross-Browser-Probleme

JxBrowser

- nativer Swing-Browser
- basiert auf Googles Chrome
- keine Cross-Browser Probleme
- proprietär, aber kostenlos für Saros
- 200 MB Footprint
- Browser-Erweiterung wahrscheinlich nötig

JavaFX Browser

- Teil von Oracles Java Plattform
- in SWT und Swing einbettbar
- keine Cross-Browser Probleme
- ab Java 7 auf allen Plattformen
- Nicht-Lauffähigkeit unter Java 6 bestätigt

Native Swing

- Bibliothek rund um die SWT-AWT-Verbindung
- Browser nicht isoliert
- Björn Kahlerts Erweiterung nicht verfügbar

⇒ nicht als separate Alternative betrachtet

⇒ gegebenenfalls eine Sammlung von Wissen

Vergleich

Kriterium	SWT browser	JxBrowser (nur IntelliJ)
Aufwand	machbar	Einbindung minimal, Erweiterung nötig
Codeduplikation	gleicher Code	Duplikation
Stabilität	Best-Effort, evtl. ausreichend	stabil
Cross-browser	identischer Aufwand	identischer Aufwand
Größe	ca. 20 MB	ca. 200 MB
Lizenz	Open-Source	proprietär, kostenlos

Entscheidung für SWT-Browser:

- ⇒ SWT-Browser naheliegendste Option in IntelliJ
- ⇒ JxBrowser als Fallback
  - ⇒ eventuell auch nur für ein Betriebssystem

Entscheidung über Benutzung der SWT-Erweiterung

Freie Universität Berlin

Ausgangslage:

- Nicht 1-zu-1 benutzbar (Mac OS spezifisch)
- beachtliche Menge an diversen Funktionen
- Code erfordert Einarbeitung

1. Frage: Neuentwicklung oder Erweiterung?

- Wie viel Code müsste neugeschrieben werden?
  - ⇒ Skizze einer Neuimplementierung
  - ⇒ grundlegende Komplexität ähnlich

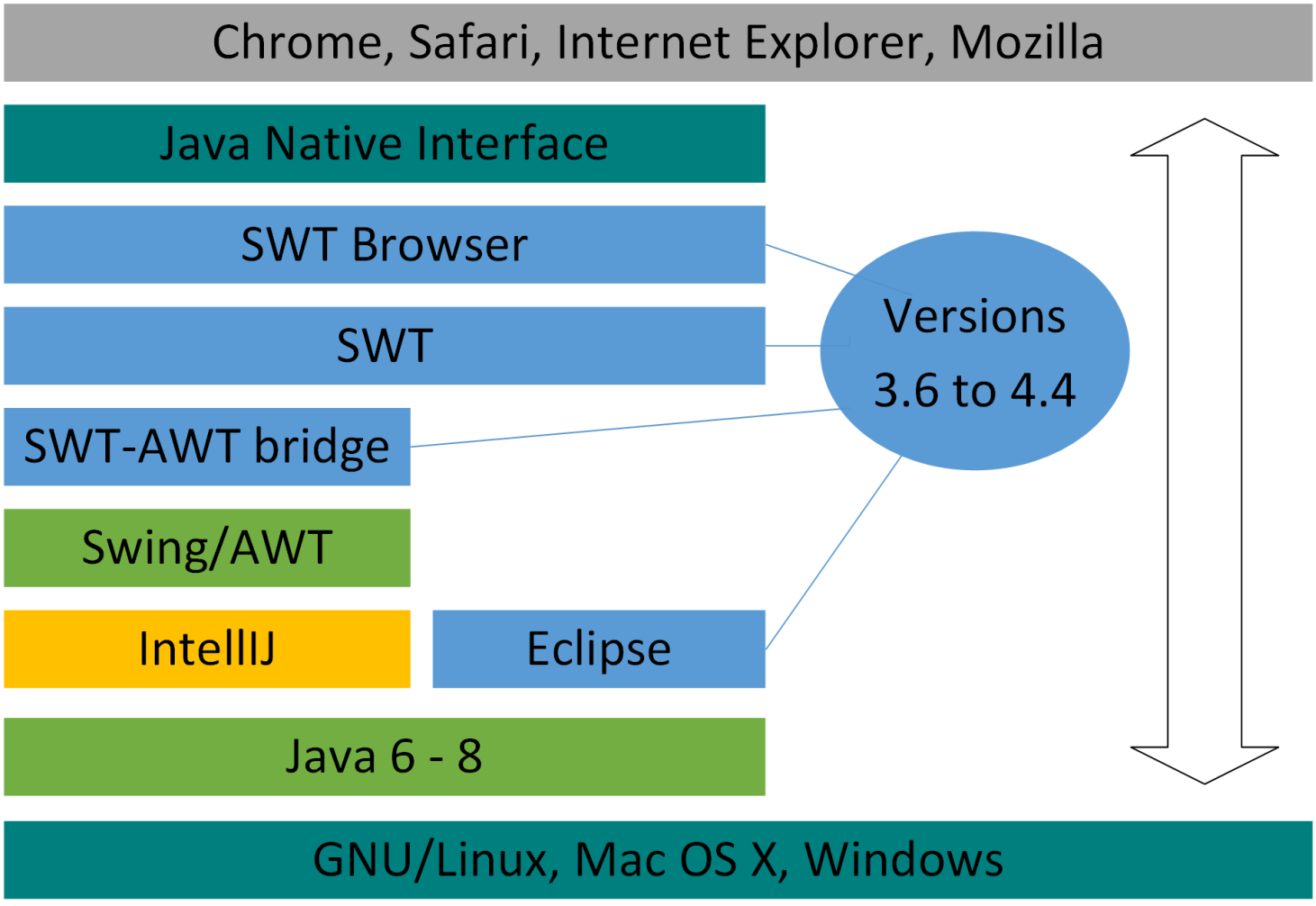
2. Frage: Änderungen zurück ins ursprüngliche Projekt?

- nicht alle Anpassungen übernehmbar (Eclipse-spezifisch)
- Entscheidung zusammen mit Björn Kahlert
  - ⇒ Eigenständiges Projekt außerhalb von Saros
  - ⇒ Vision: generische Bibliothek

Auswertung der Prototyp-Implementierung

Identifizierte Risiken & Probleme

Zusammenspiel der Technologien



Identifizierte Einschränkungen des SWT Browsers

Freie Universität Berlin

- Mac OS
  - SWT Browser in IntelliJ verlangt Java 6
  - IntelliJ unterstützt Java 7 und 8 nur experimentell
- GNU/Linux
  - Browser funktioniert nicht in Eclipse 3.7 / 3.8
  - in Ubuntu 14.10 ein Paket nachinstallieren
- Zukunftssicherheit
  - SWT_AWT Brücke wenig benutzt
  - z. B. ungefixter Bug für Java 7,8 unter Mac OS
  - JavaFX als Nachfolger von Swing

⇒ SWT wird weiterverfolgt / aktuell ist keines der Probleme gravierend

⇒ JavaFX im Auge behalten

Zusammenfassung und Ausblick

- Browseransatz erfolgsversprechend
- Entscheidung für SWT Browser (vorläufig?)
- lauffähiger Prototyp für Eclipse und IntelliJ
- Erweiterung des SWT Browsers vorangetrieben
- Grundentwurf für GUI-Tests
- Weiterentwicklung in laufenden Arbeiten von Sieker und Bohnstedt
  - Entwurf der HTML-GUI
  - Benutzbarkeit

Ende

Fragen?