Gesammelte Inhalte

**1. Produktübersicht:**

Name: De(r)bugger / Dibagger / Derbagger ???  
Aufgabenbeschreibung aus PSE Projekt benutzten, abwandeln

**2. Zielbestimmung:**

Muss:

Definition Step

Definition Breakpoints

Interpreter für festgelegte Sprache

variable Stepsize

aktuelle Variablenwerte zu jedem Step oder Breakpoint

Breakpoint in beliebige Zeile setzten (zusammenhängend, Mausklick einzelnen Breakpoint setzen, doppelklick weiteren Breakpoint setzen, Rechtsklick löschen)

Bedingte Breakpoints

Continue: jedes Programm läuft zum nächsten Breakpoint

allg. Debugger Fähigkeiten: Werte des Variablen zu jedem Zeitpunkt anzeigen,  
davon abhängige Breakpoints (zb. wenn a = b stop)

Tooltips

Programm in versch. Sprachen (zumindest Englisch)

Konfigurationsdatei für einen Lauf / Lauf speichern

Zu debuggendes Programm einbinden durch reinkopieren und öffnen (Textdatei)

Programm reinschreiben in Textboxt

Manueller Abbruch durch Button 🡪 Editiermodus (dazu Debuggingmodus unterbrechen)

Watch-Expressions neben Programmen

hier fehlt: was kann die Sprache

Quellcode markieren

Variableninspektor

Soll:

Watch-Expressions die nur in Markierung angeschaut werden

Ausblendungsmöglichkeit für Variablen im Variableninspektor

bei gleichen Programmen versch. Eingaben vorschlagen (Vorschlag-Button, zufällig, zufällig verschieden, Intervall wählen)

Möglichkeit, bestimmte Variablen zu speichern (jeder angenommene Wert)

automatischer Abbruch bei endlosschleifen (maximum vom Nutzer einstellbar, allerdings beschränkt)

mehr als 2 Programme debuggen

Kann:

zu debuggendes Programm einbinden durch reinziehen

Programm schlägt Stepsize abhängig von Zeilen der Programme vor

Automatische Beschreibung des Unterschieds zweier Programme (wenn Benutzer vorher die Variablen angibt)

Rückschritte möglich

Vorschläge für Watch-Expressions anhand von Programmen (schlage Watch-Expressions vor(Menüpunkt))

Breakpoints verschieben sich mit, wenn Code geändert wird

Abgrenzung:

100 Zeilen Codegrenzen

Obergrenze an Iterationen

Nur einen Durchlauf

**3. Produkteinsatz:**

Institut, Hilfe bei beweisen

**4. Produktumgebung:**

Java- Programm, keine besondere Hardware benötigt

zu beschreibende Schnittstellen: GUI, Interpreter (Definition der Sprache), Format

**5. Funktionale Anforderungen:**

Siehe 2

**6. Produktdaten:**

zu speichern: Programmläufe, Konfigurationsdaten

**7. Produktleistungen:**

Zeit und Genauigkeit?

**8. Nichtfunktionale Anforderungen:**

Gesetze? Normen? Sicherheit? Urheberrecht?

keine Plattformabhängigkeit durch Java

Erweiterbarkeit: austauschbarer Interpreter für andere Sprachen

**9. Qualitätsanforderungen:**

**10. Globale Testfälle und Szenarien:**

Noninterference szenario mit zwei gleichen Programmen (zB. Voting-Fall)

Programmäquivalenz

Form: Eingabe - erwartete Ausgabe

GUI-Testfälle

Ackerman (nicht laufen)

Quicksort

**11. Systemmodelle:**

**12. Benutzungsoberfläche:**

Skizze

Menüs? mehrere Programme?

variable Fenster (rausziehen etc)

**13. Spezielle Anforderungen an Entwicklungsumgebung:**

Java-IDE

**14. Zeit und Ressourcenplanung:**

**15. Ergänzungen:**

Evtl. Sprache

**16. Glossar**

**Was soll die Sprache können?**

Java Teilmenge

Kommentare

Methoden /Funktionen

Arrays, Operationen für Arrays

primitive Datentypen

while

if

+-\*:

Rückgabewert

Kann: Strings for max()/min() System.out. (Konsole benötigt)