Debugging

Jan Hendrik Dithmar

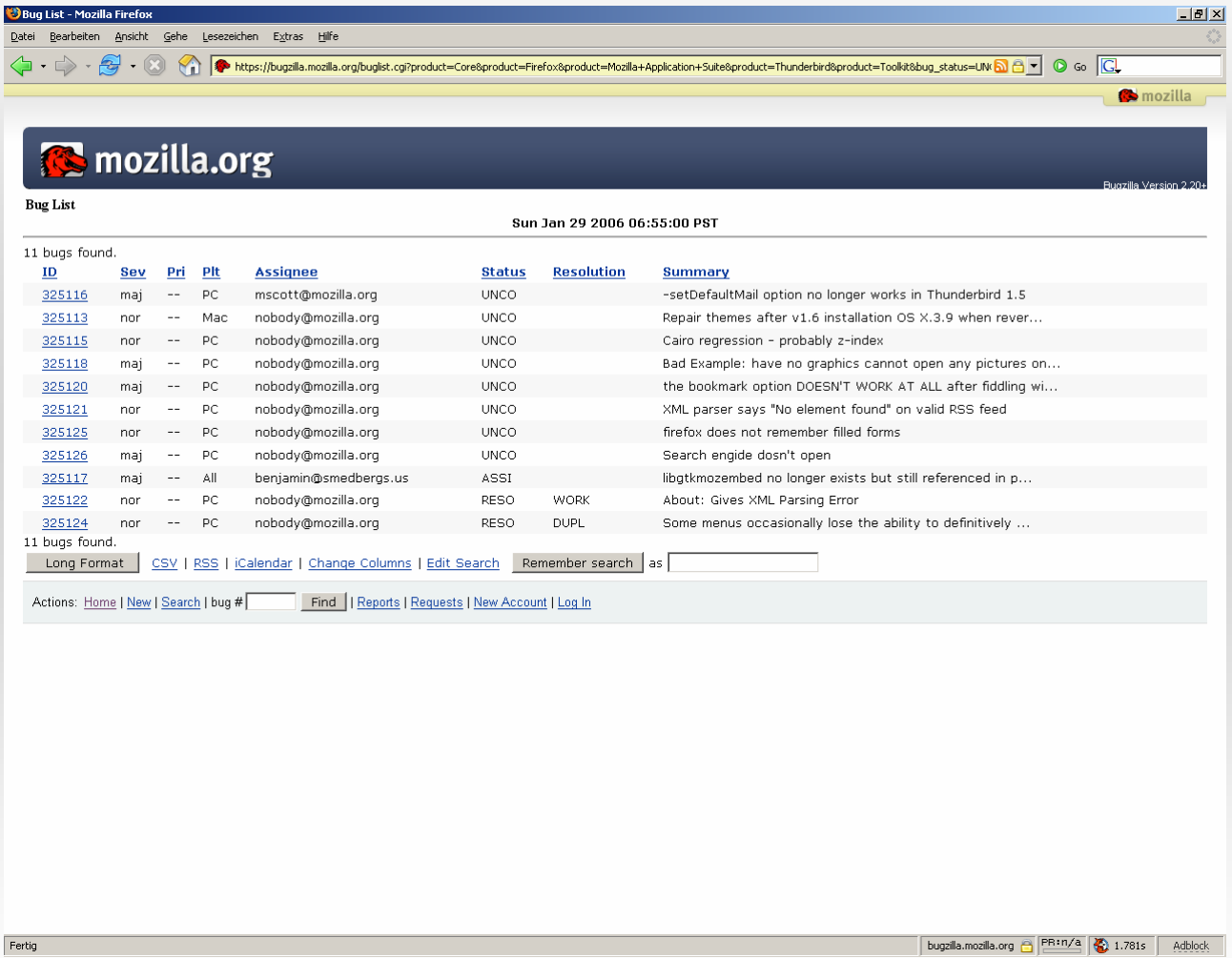
02.02.2006

Testfälle

Ein Test kann nur die Anwesenheit von Fehlern nachweisen, nicht jedoch deren Abwesenheit.

Dijkstra (1972)

Problemverwaltung



Quelle: http://bugzilla.mozilla.org

Debugging – ein Überblick

1. Ein Fehler tritt auf
2. Hypothese aufstellen
3. Eingrenzen der Fehlerursache

Das systematische Eingrenzen von Fehlerursachen nennt man auch die *wissenschaftliche Methode* der Fehlersuche.

Die wissenschaftliche Methode

1. Hypothese aufstellen
2. Hypothese prüfen
   - Der Test schlägt fehl: Hypothese bestätigt
   - Der Test ist erfolgreich: Hypothese widerlegt
   - Inkonsistenz: Test zeigt weder korrektes noch fehlerhaftes Verhalten
3. Fahre mit der neuen Hypothese fort bei Schritt 2, bis der Fehler hinreichend eingegrenzt ist.

Wie prüft man Hypothesen?

- Anreichern des Quelltextes mit Debugging-Anweisungen

```
#ifdef DEBUG
std::cout << "a=" << a << std::endl;
std::cout << "b=" << b << std::endl;
#endif
```

Typische Untersuchungstechniken

- Ausgabe des Programmzustands
  z.B. in einem Debugger nach dem Halten eines Programms
- Validierung des Programmzustands
  am besten mit Zusicherungen (assertions)
- Änderung des Programmzustands
  z.B. Ausblenden von Programmfragmenten

Beispiel

- Programm:
  Berechnung der Differenz **`a-b`**.
- Situation:
  In meinem Quelltext ist irgendwo ein Fehler.
- Ziel:
  Eingrenzen der Fehlerursache mit Hilfe der *wissenschaftlichen Methode*.

Auf Debugging-Anweisungen wird aus Platzgründen verzichtet.

Beispiel

```
#include <iostream>
int
   re
}

int
   in
   in

   st
   st

   return 0;
}
```

```
int differenz(int a, int b) {
        #ifdef DEBUG
                std::cout << "a=" << a << std::endl;
                std::cout << "b=" << b << std::endl;
        #endif

        return (b-a);

}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (b-a);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Hypothese prüfen

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (b-a);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (b-a);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Fehler kann hier behoben werden!

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Hypothese

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

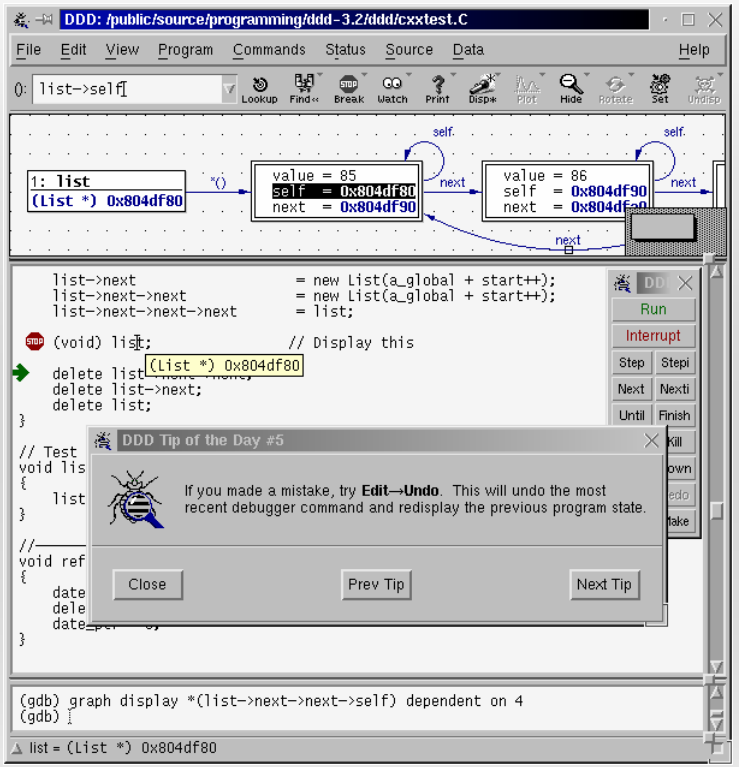
Hypothese

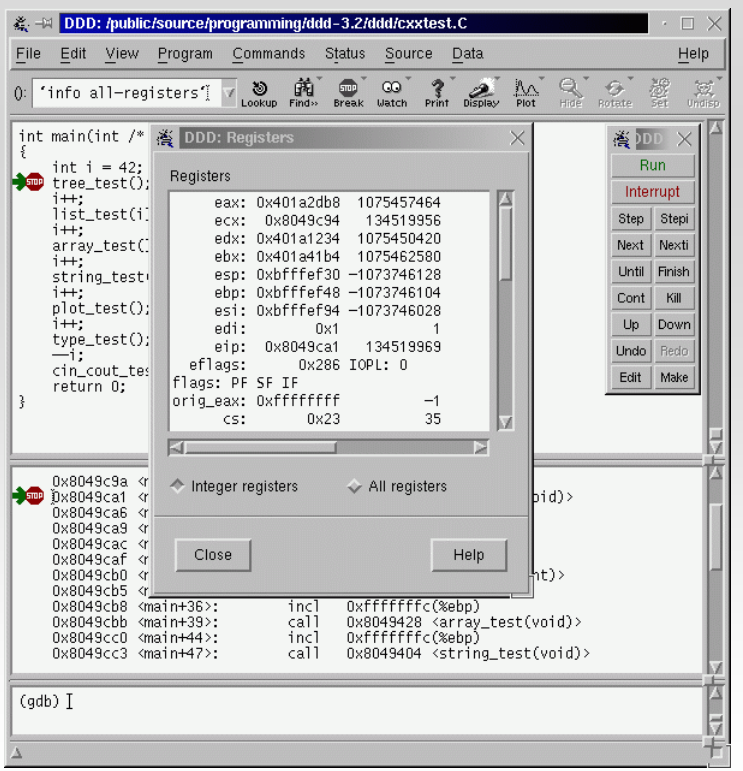
Beispiel

```
#include <iostream>
int differenz(int a, int b) {
   return (a-b);
}

int main() {
   int a = 0;
   int b = 0;

   std::cout << "a: "; std::cin >> a;
   std::cout << "b:"; std::cin >> b;

   std::cout << "a - b = " << (differenz(a,b)) << std::endl;

   return 0;
}
```

Fehler erfolgreich beseitigt!

Arbeiten mit einem Debugger





Quelle: http://www.gnu.org/software/ddd/

Arbeiten mit einem Debugger

Ein Debugger ist kein Ersatz für gutes Nachdenken. In einigen Fällen ist aber Nachdenken auch kein Ersatz für einen guten Debugger. Die effektivste Kombination ist gutes Nachdenken und ein guter Debugger.

McConnell (1993)

Debugging-Prozess

Wie kann der Debugging-Prozess effektiver gestaltet werden?

Durch:

- Testfälle und Verwendung von Assertions
- Problemverwaltung
- Debugger
- scharfes Nachdenken