

Erfahrungen bei der Entwicklung großer STFL-Applikationen

Andreas Krennmair <ak@synflood.at>

Agenda

- Einleitung
- STFL in < 5 Minuten
- Einsatz mit C++
- Best Practices UI und Usability
- Testbarkeit

Einleitung (I)

- Terminal-
Applikationen: text-
basiert, Farben,
einfache Effekte
- Ursprung: interaktive
Multiuser-Systeme
- Ansteuerung via
Escape-Codes

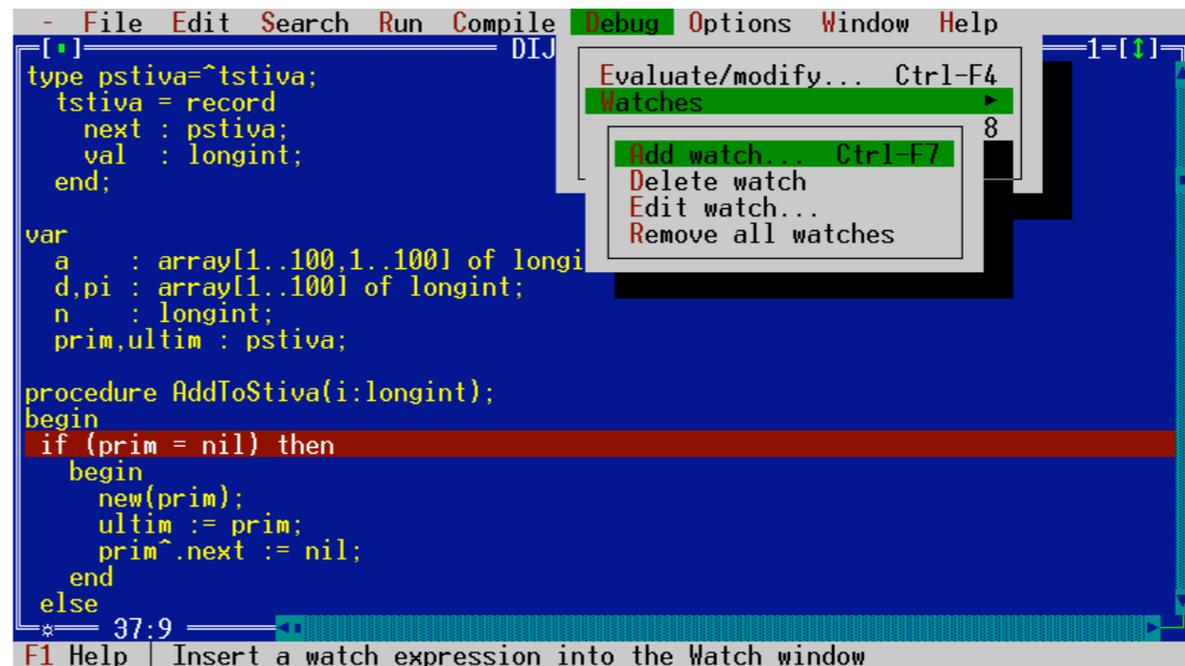


Einleitung (2)

- Library zum generischen Ansteuern von Terminals: (N)Curses
- Nachteile:
 - Manuelles Resizing & Layout Management
 - Sehr low-level
 - Widgetset “Cdk” ein Flop

Einleitung (3)

- Kommerzielle Entwicklung: Turbo Vision
- Komplexe Widget-Library, bekannt durch Turbo C++ und Turbo Pascal
- Mittlerweile frei



The screenshot displays the Turbo Vision debugger interface. The main window shows Pascal code with the following content:

```
- File Edit Search Run Compile Debug Options Window Help
[.] DIJ
type pstiva = ^tstiva;
tstiva = record
  next : pstiva;
  val : longint;
end;

var
  a : array[1..100,1..100] of longint;
d,pi : array[1..100] of longint;
n : longint;
prim,ultim : pstiva;

procedure AddToStiva(i:longint);
begin
  if (prim = nil) then
  begin
    new(prim);
    ultim := prim;
    prim^.next := nil;
  end
  else
  * 37:9
```

A 'Watches' window is open, showing the following options:

- Evaluate/modify... Ctrl-F4
- Watches
- Add watch... Ctrl-F7
- Delete watch
- Edit watch...
- Remove all watches

The status bar at the bottom reads: F1 Help | Insert a watch expression into the Watch window

Einleitung (4)

- Meine präferierte Widget-Library: STFL
- User Interface durch Layout-Sprache beschrieben
- Automatisches Layout & Resize Management
- Einfache API, schnell erlernbar
- Für viele Programmiersprachen verfügbar

STFL (I)

- Layout-Beschreibungssprache

vbox

```
@style_normal:bg=black,fg=white
@info#style_normal:bg=red,fg=white,attr=bold
label#info
    text[firstline]:"A Test UI"
    .expand:0
textview
    .expand:vh
    listitem text:"some text"
    listitem text:"even more text"
    listitem text:"-----"
    listitem text:"wow, so much text!"
label#info
    text:"Press ESC to quit"
    .expand:0
```

A Test UI

some text
even more text

wow, so much text!

Press ESC to quit

STFL (2)

- Einfache Programmierbarkeit

```
#!/usr/bin/env ruby
require 'stfl'

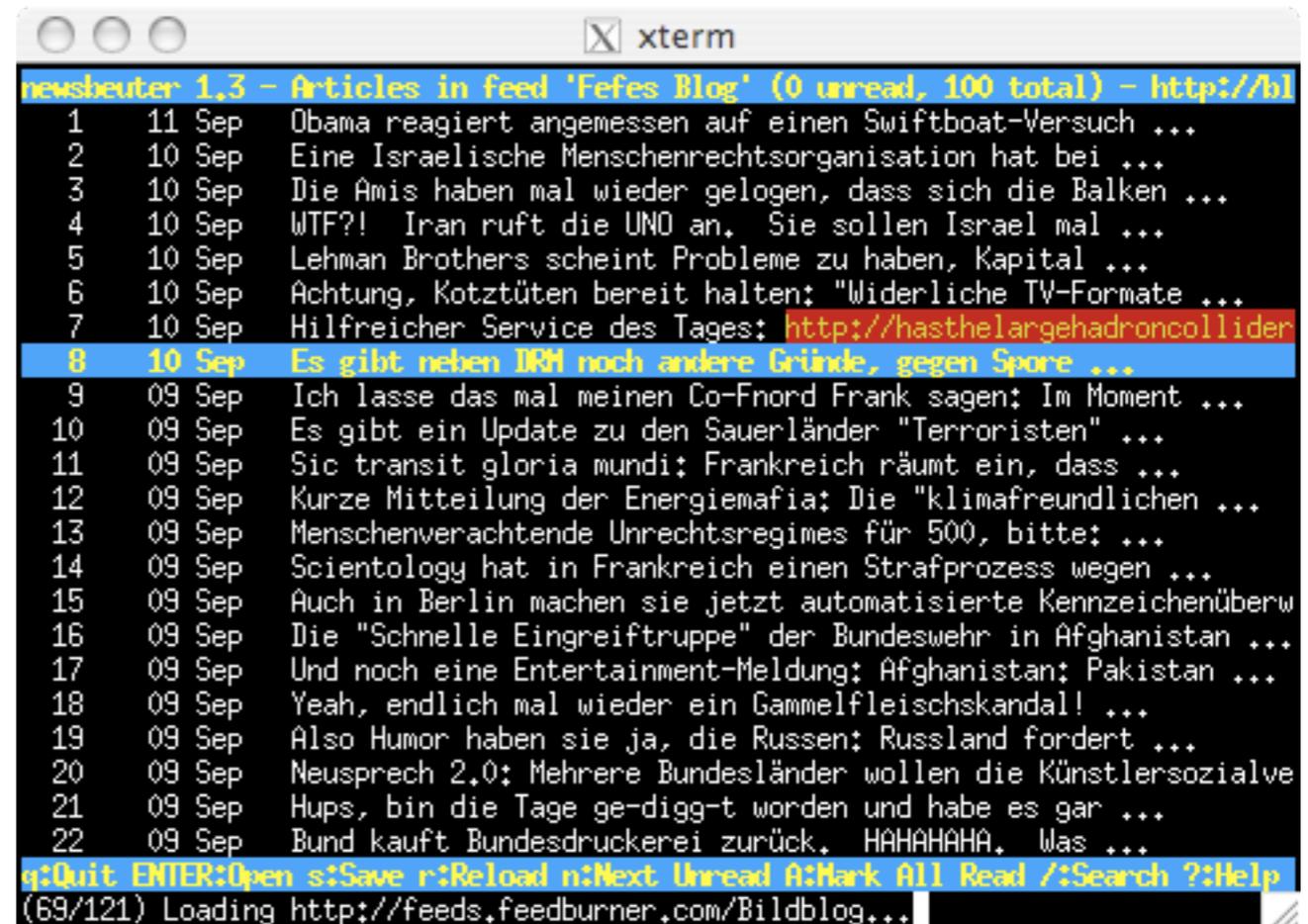
form = Stfl.create('<form.stfl>')
a = true
loop do
  event = form.run(0)
  case event
    when "a"
      form.set("firstline", a ? "Hello,
World!" : "Goodbye, World!")
      a = !a
    when "ESC"
      break
  end
end
end
```


STFL (4)

- Klammernbasierte Alternativ-Syntax (programmatisch besser generierbar)
- Run Loop vom Programmierer selbst ausprogrammierbar
- 20 API-Funktionen (inklusive iconv-Wrapper mit Memory-Pool-Konzept)
- Bindings für Perl, Python, Ruby, SPL

Newsbeuter

- Featurereicherer RSS-Feedreader für Textkonsole
- Erste “große” STFL-Applikation
- ca. 10000 Zeilen C++-Code
- STFL-Erweiterungen zur Realisierung notwendig



```
xterm
newsbeuter 1.3 - Articles in feed 'Fefes Blog' (0 unread, 100 total) - http://bl
1 11 Sep Obama reagiert angemessen auf einen Swiftboat-Versuch ...
2 10 Sep Eine Israelische Menschenrechtsorganisation hat bei ...
3 10 Sep Die Amis haben mal wieder gelogen, dass sich die Balken ...
4 10 Sep WTF?! Iran ruft die UNO an. Sie sollen Israel mal ...
5 10 Sep Lehman Brothers scheint Probleme zu haben, Kapital ...
6 10 Sep Achtung, Kotztüten bereit halten: "Widerliche TV-Formate ...
7 10 Sep Hilfreicher Service des Tages: http://hasstheLargeHadronCollider
8 10 Sep Es gibt neben IMF noch andere Gründe, gegen Spore ...
9 09 Sep Ich lasse das mal meinen Co-Fnord Frank sagen: Im Moment ...
10 09 Sep Es gibt ein Update zu den Sauerländer "Terroristen" ...
11 09 Sep Sic transit gloria mundi: Frankreich räumt ein, dass ...
12 09 Sep Kurze Mitteilung der Energiemafia: Die "klimafreundlichen ...
13 09 Sep Menschenverachtende Unrechtsregimes für 500, bitte: ...
14 09 Sep Scientology hat in Frankreich einen Strafprozess wegen ...
15 09 Sep Auch in Berlin machen sie jetzt automatisierte Kennzeichenüberw
16 09 Sep Die "Schnelle Eingreiftruppe" der Bundeswehr in Afghanistan ...
17 09 Sep Und noch eine Entertainment-Meldung: Afghanistan: Pakistan ...
18 09 Sep Yeah, endlich mal wieder ein Gammelfleischskandal! ...
19 09 Sep Also Humor haben sie ja, die Russen: Russland fordert ...
20 09 Sep Neusprech 2.0: Mehrere Bundesländer wollen die Künstlersozialve
21 09 Sep Hups, bin die Tage ge-digg-t worden und habe es gar ...
22 09 Sep Bund kauft Bundesdruckerei zurück. HAHAHAHA. Was ...
q:Quit ENTER:Open s:Save r:Reload n:Next Unread A:Mark All Read /:Search ?:Help
(69/121) Loading http://feeds.feedburner.com/Bildblog...
```

C++-Wrapper (I)

- Reine C-API für C++-Verhältnisse aufwändig zu verwenden
- C++-Wrapper, Integration mit `std::string`
- Durch Wrapper war STFL-Unicode-Umstellung relativ schmerzlos einbaubar

C++-Wrapper (2)

```
class form {
public:
    form(const std::string& text);
    ~form();

    const char * run(int timeout);

    std::string get(const std::string& name);
    void set(const std::string& name, const std::string& value);

    std::string get_focus();
    void set_focus(const std::string& name);

    std::string dump(const std::string& name, const std::string& prefix, int focus);
    void modify(const std::string& name, const std::string& mode, const std::string& text);
    std::string lookup(const std::string& path, const std::string& newname);
};
```



Dialog-Management (I)

- Dialoge: Feed List, Article List, Article View, Help, ...
- “Stapel” von Dialogen
- Jeder Dialog eine Methode der view-Klasse
- Problem: jeder Dialog enthält eigene Run-Loop
- Kommunikation zwischen Dialogen hängt schlecht zuordenbar in der View

Dialog-Management (2)

- Jeder Dialog ist ein Objekt
- Dialoge werden gestapelt
- Oberster Dialog ist jeweils aktiv
- Vorteil: eine Run-Loop
- Aufgaben eines Dialog:
 - Reaktion auf Useraktionen (“open item”) u. Kommunikation mit dem Controller
 - Update der Dialoginhalte

Article View

Article List

Feed List

Dialog-Management (3)

- Aufgaben der View:
 - Run-Loop
 - Mapping Taste → Useraktion (delegiert)
 - Messaging an aktuellen Dialog

Dialog-Management (4)

```
class formation {
public:
    formation(view *, std::string formstr);
    virtual ~formation();
    virtual void prepare() = 0;
    virtual void init() = 0;
    // [...]
    virtual keymap_hint_entry * get_keymap_hint() = 0;
    virtual std::string id() const = 0;
    // [...]
    void process_op(operation op, bool automatic = false, std::vector<std::string> * args = NULL);
    virtual void finished_qna(operation op);
    void start_cmdline();
    // [...]
    std::string get_qna_response(unsigned int i);
    void start_qna(const std::vector<qna_pair>& prompts, operation finish_op, history * h = NULL);
protected:
    virtual void process_operation(operation op, bool automatic = false,
                                   std::vector<std::string> * args = NULL) = 0;
    // [...]
};
```

Dialog-Management (5)

```
void view::run() {
    while (formation_stack.size() > 0) {
        formation * fa = *(formation_stack.begin());
        fa->prepare();

        const char * event = fa->get_form()->run(0);
        if (!event || strcmp(event, "TIMEOUT")==0)
            continue;

        operation op = keys->get_operation(event, fa->id());

        if (OP_REDRAW == op) {
            stfl::reset();
            continue;
        }

        fa->process_op(op);
    }

    stfl::reset();
}
```

Dialog-Management (6)

```
void view::push_itemlist(std::tr1::shared_ptr<rss_feed> feed) {
    if (feed->rssurl().substr(0,6) == "query:") {
        set_status(_("Updating query feed..."));
        feed->update_items(ctrl->get_all_feeds());
        set_status("");
    }
    if (feed->items().size() > 0) {
        itemlist->set_feed(feed);
        itemlist->set_show_searchresult(false);
        itemlist->init();
        formation_stack.push_front(itemlist);
    } else {
        show_error(_("Error: feed contains no items!"));
    }
}

void view::push_itemview(std::tr1::shared_ptr<rss_feed> f, const std::string& guid) {
    itemview->set_feed(f);
    itemview->set_guid(guid);
    itemview->init();
    formation_stack.push_front(itemview);
}

void view::pop_current_formation() {
    formation_stack.pop_front();
    if (formation_stack.size() > 0) {
        formation * f = (*formation_stack.begin());
        if (f) {
            f->set_redraw(true);
            f->get_form()->set("msg", "");
            f->recalculate_form();
        }
    }
}
```

Usability

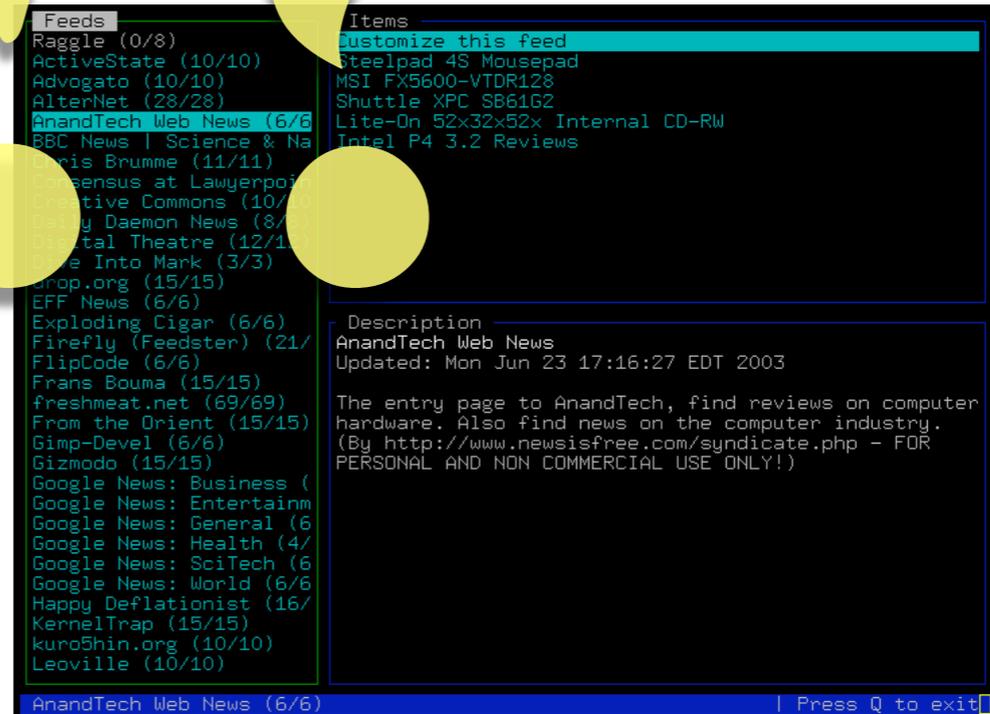
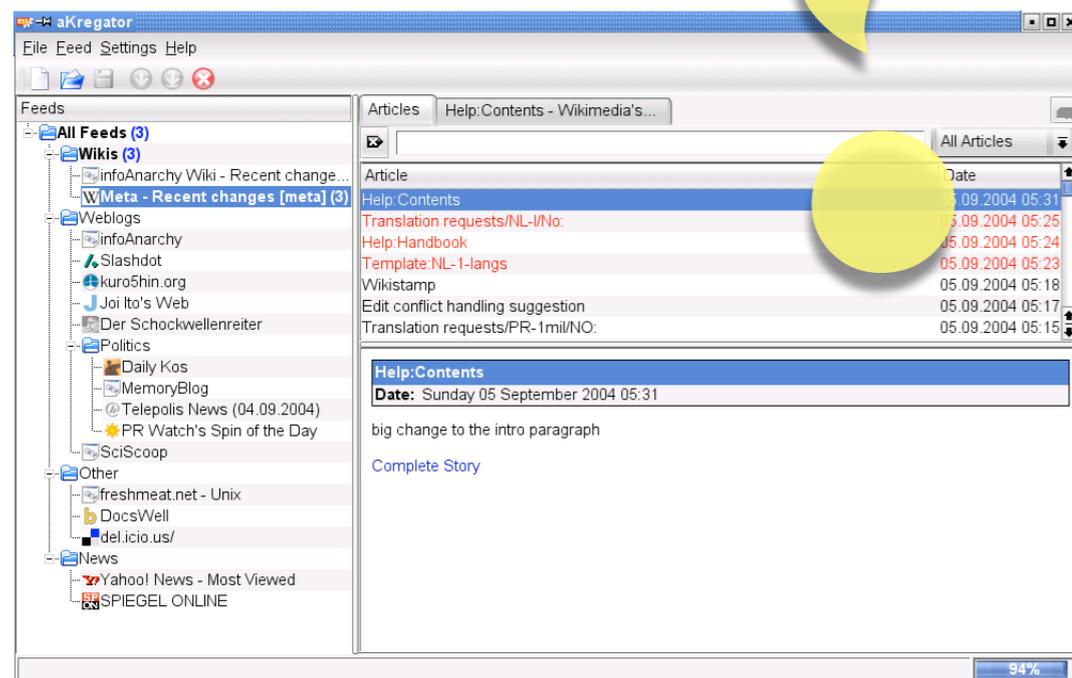
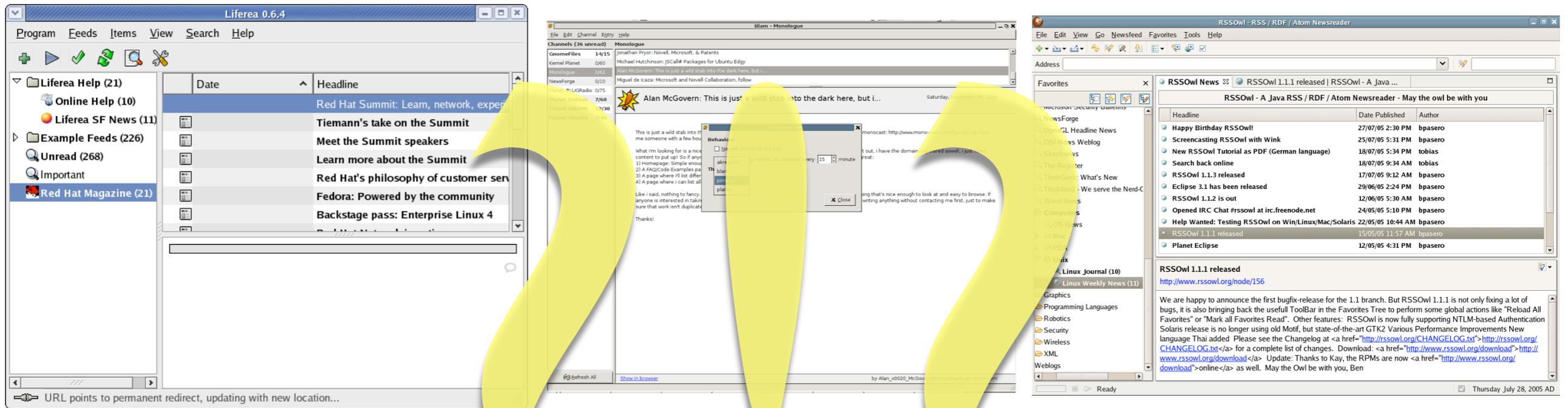
- Was macht ein gutes Text User Interface aus?
 - Übersichtlichkeit
 - Effizienz
 - Freie Konfigurierbarkeit
 - Tastenbelegungen
 - Aussehen: Farben, Formate (z.B. in Listen)
 - Commandline
 - Gut erreichbare Kurzhilfe
 - Kompatibilität mit Braille-Zeilen

Usability:

Übersichtlichkeit (I)

- “one dialog displays one thing well”
- Einheitliche Struktur
 - oberste Zeile: Basis-Infos (“wo bin ich?”)
 - Hauptteil: Anzeige des wesentlichen Inhalts
 - zweitunterste Zeile: die wichtigsten Tasten
 - unterste Zeile: Status-/Fehlermeldungen
- Anti-Pattern: “Left List-Top List-Bottom Content” (LL-TL-BC; Outlook-Layout)

Usability: Übersichtlichkeit (2)



Usability: Effizienz

- Schnelle Bedienbarkeit über Tasten (kombinationen)
- Einträge in Listen direkt adressierbar/anspringbar
- Gute und schnelle Suchfunktion
- Anti-Pattern: Navigation ausschließlich über Cursortasten und PGUP/PGDN.

Usability:

Konfigurierbarkeit

- Das Programm hat sich nach dem User zu richten, nicht der User nach dem Programm
- Frei konfigurierbare Tasten auf Dialogebene
- Farben und Formate (fast) aller Elemente einstellbar
- Farbliche Hervorhebung von Text mit Regular Expressions

Eine kleine Umfrage:

Eine kleine Umfrage:

**Wer testet seine
Software? Wie?**

Eine kleine Umfrage:

**Wer testet seine
Software? Wie?**

**Wer automatisiert
diese Tests?**

Testbarkeit (I)

- Arten von Softwaretests (Auswahl):
 - Unit Test
 - Integration Test
 - Functional Test
 - GUI Test
 - User Acceptance Test
 - Performance Test



Testbarkeit (2)

- Unit Tests quasi von anfang an verwendet
 - Boost.Test hat sich bewährt
 - Werden mit jedem CI-Build mitgebaut und ausgeführt
 - Geben Sicherheit bei der Modifikation von Basisklassen (RSS-Parser, Commandline History, Query Parser, Keymapping, Tagsoup Pull Parser, String Formatter, ...)

User Interface Testing

- Wie User Interface testen?
- Erste Idee: STFL erweitern oder Mock-Up implementieren
 - Nachteil: viele STFL-Verhaltensweisen müssten nachimplementiert werden
- Zweite Idee: Emulation eines virtuellen Terminals, automatisiertes Treiben von Benutzereingaben, Verifikation von Terminal-Ausgaben

User Interface Testing

- Ergebnis: “tuitest”
- Recorder: startet Application under Test (AUT), zeichnet Eingaben auf (optional inkl. Timing), generiert Ruby-Skript, kann Verifikationen semi-automatisch erzeugen
- Ruby-Modul: wird von den generierten Skripten benötigt, startet AUT, spielt Benutzereingaben ab, überprüft Verifikationen, loggt Testergebnis, optional auch in ein XML-File (wie xUnit-Frameworks)

User Interface Testing

- Mit automatisierten UI-Tests lassen sich Regressions (“Umfaller”) im UI-Code entdecken
- Möglichkeit, Testabdeckung von UI-Code zu erhöhen
- Vollständig automatisierbar, d.h. beliebige Testwiederholungen sind “billig”

Verifikationen

- Befindet sich nach Benutzereingabe Text “foo” an Bildschirmposition (x,y)?
- Automatische Generierung beim Recorden:
 - Snapshot von Bildschirm erstellen
 - Benutzereingabe
 - zweiten Snapshot erstellen
 - Unterschiede der zwei Snapshots werden als Verifikationen geskriptet

Continuous Integration (I)

- Bei jeder Änderung im Source-Repo wird ein “clean build” gestartet
- Ausführung der Test Suite nach jedem Build
 - Unit Tests
 - UI Tests
- Jede erkannte Regression ist einer definierten, kleinen Menge an Changesets zuordenbar

Continuous Integration (2)

- Nächtllicher Build für Code Coverage
- Ausführung aller Tests
- Generierung eines Reports
- Zeigt Defizite bei Testabdeckung auf
- Tools of the trade: GCOV + LCOV
- Brauchbares CI-System: Hudson

Fragen?



- Wichtige URLs:
 - <http://www.clifford.at/stfl/>
 - <http://www.newsbeuter.org/>
 - <http://bereshit.synflood.at/svn/noos/trunk>
 - <http://bereshit.synflood.at/svn/tuitest/trunk>