Usability-Test für mobile Java-Anwendungen

- Problemstellung / Abgrenzung
- Usability Engineering / Patterns
- Usability Test
- Tool-Kette
- Fazit

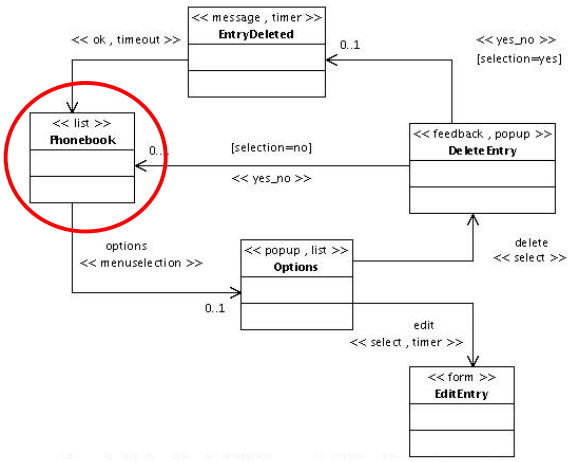
Roland Petrasch

Usability-Test für mobile Java-Anwendungen

Problemstellung / Abgrenzung: Usability for mobile apps

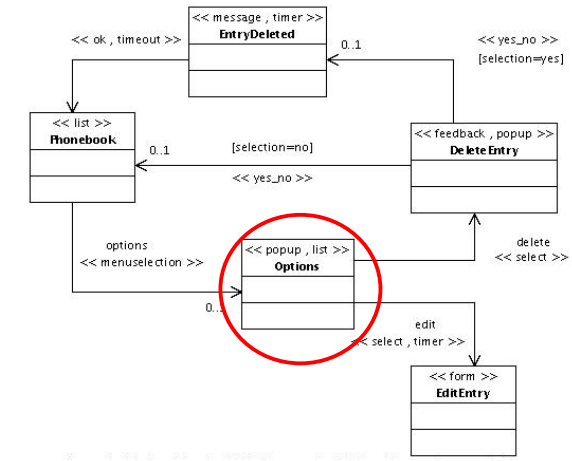
- Spezielle Anforderungen an die Ergonomie, z.B. Navigation, HW/SW-Verbindung (WIMP)
- Weniger Probleme: Screen-Layout, Nebenläufigkeit
- Projektumfeld nicht im consumer-Bereich, sondern Behörden und Organisationen mit Sicherheitsaufgaben (BOS)
- Kompromiss:
  - formale Ansätze (Markov models, GOMS ...): Nutzbarkeit ?
  - Standards / Produkte: UML, Eclipse, Java (Open Source)
- Ziele: einfacher Einstieg, sofortige Umsetzung möglich, keine Kosten, Erlernbarkeit, Produktivität ...
- Beispiel: Navigation auf mobile Device (Löschen einer Adresse)

Usability Engineering: Navigation Model



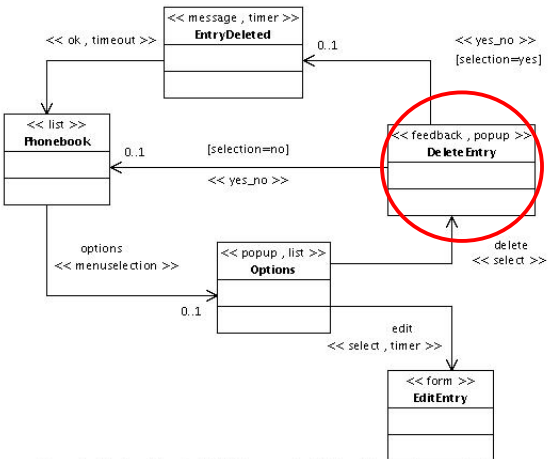
Usability-Regel (?):
„Nav-Action: rechts“

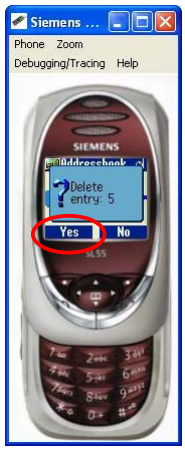
Usability Engineering: Navigation Model



Usability-Regel:
„Nav-Action: rechts“

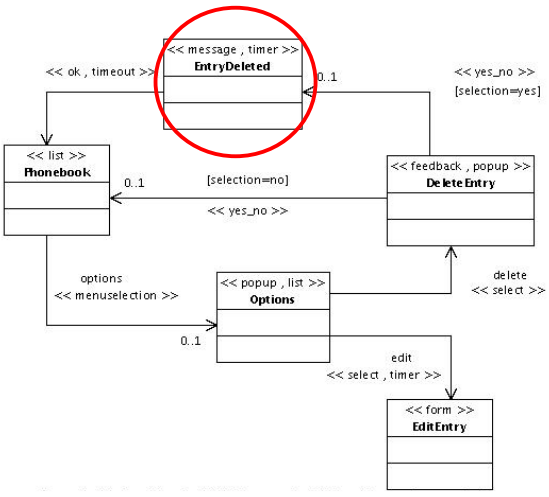
Usability Engineering: Navigation Model





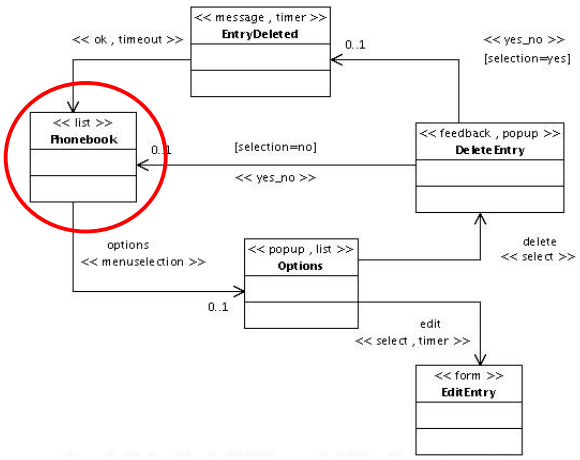
Usability-Diskussion #1:
„Yes: links oder rechts?“

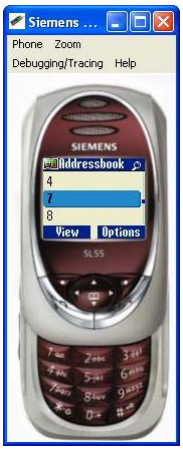
Usability Engineering: Navigation Model



Usability-Diskussion #2:
„OK: auch links?“

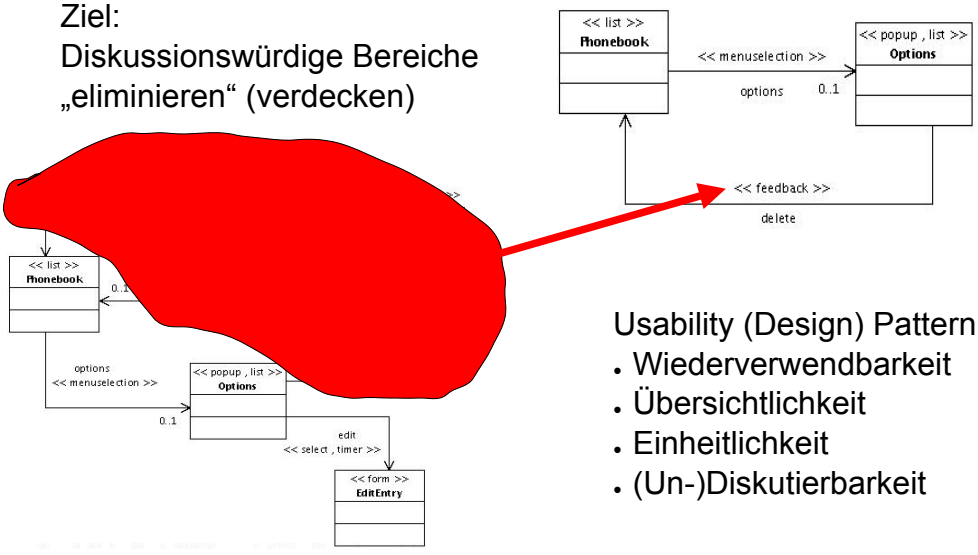
Usability Engineering: Navigation Model





Usability (Design) Patterns: Feedback Pattern

Ziel:
Diskussionswürdige Bereiche „eliminieren“ (verdecken)

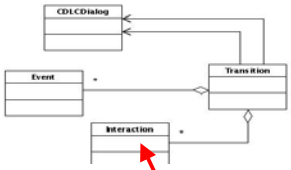


Usability (Design) Pattern:

- Wiederverwendbarkeit
- Übersichtlichkeit
- Einheitlichkeit
- (Un-)Diskutierbarkeit

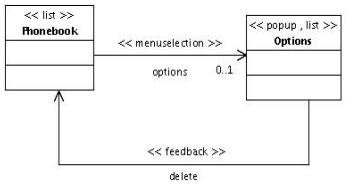
Usability Design Patterns

Ziel:
Generative SW-Entwickung betreiben



MDA: Model driven architecture

Architectural Patterns | Templates | Generator | Design-Entscheidungen | Metamodel



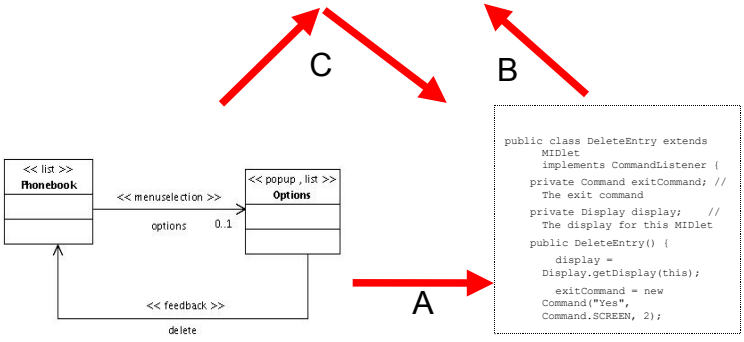

```
public class DeleteEntry extends MIDlet
    implements CommandListener {
  private Command exitCommand; // The exit command
  private Display display;    // The display for this
   MIDlet
  public DeleteEntry() {
      display = Display.getDisplay(this);
      exitCommand = new Command("Yes", Command.SCREEN, 2);
  }
  public void init() {
      TextBox t = new TextBox(" ...", 256, 0);
      t.addCommand(exitCommand);
      t.setCommandListener(this);
      display.setCurrent(t);
  }
```

Usability Test

Test #1: Referenzimplementierung als Test-Orakel

MDA: Model driven architecture

Architectural Patterns | Generator | Templates | Design-Entscheidungen | Metamodel



A: Referenz-implementierung (manuelle Progr.)

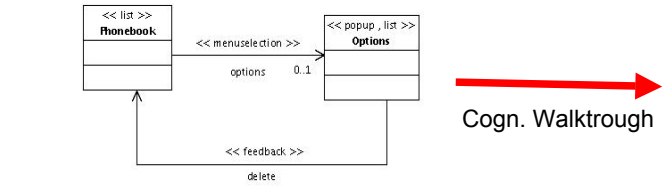
B: Templates erstellen

C: Generierung (Test gegen die Ref.-Impl.)

Usability Test

Test #2: „herkömmliche“ Usability Tests / Evaluationen

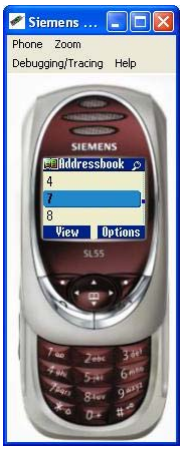
Software-ergonomische Anforderungen
z.B. ISO 9241-10:
Steuerbarkeit, Fehlertoleranz ...

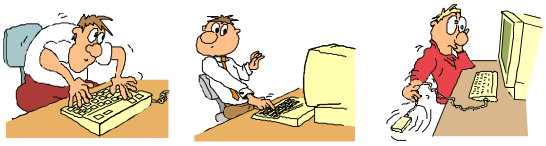


Conf. Test

Cogn. Walktrough

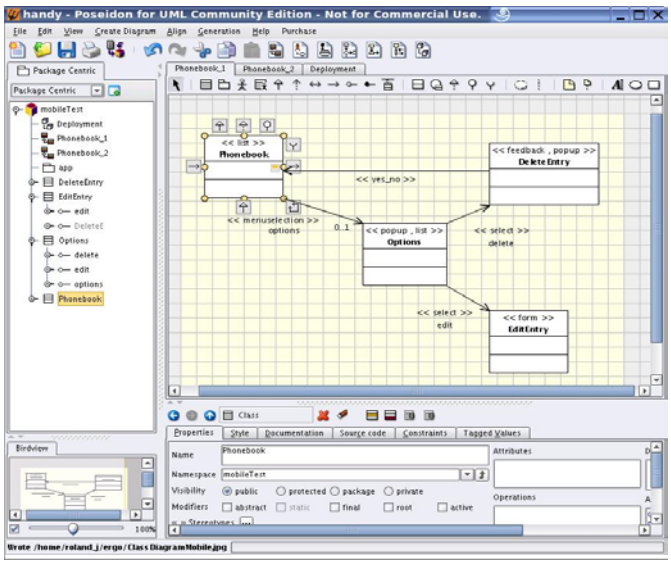
Evaluation





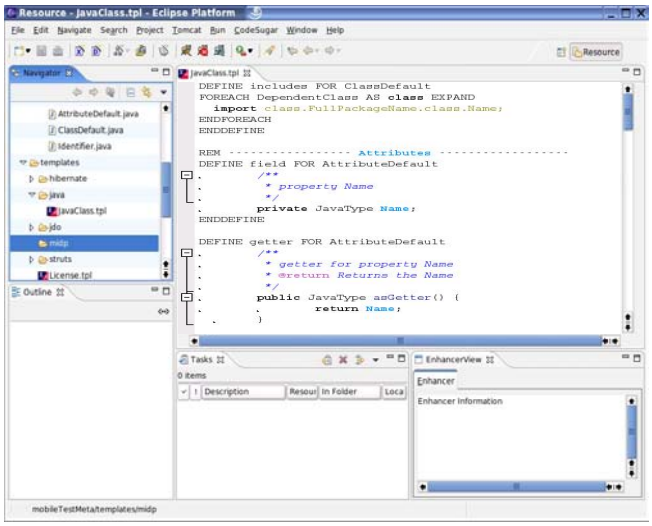
Tool-Kette:
Upper-Case, IDE, MDA, Emulator

- Upper-Case: UML-Tool
- XMI-Schnitt-stelle, z.B. Poseidon CE



Tool-Kette: Upper-Case, IDE, MDA, Emulator

- IDE: Eclipse 3.1
- OpenGenerator Framework (http://architectureware.sourceforge.net)
- diverse Java ME-Plugins
- Wireless Toolkits, z.B. SMTK



Fazit

- Rollen: MDA-Nutzer (SW-Entwickler), Metamodellierer (MDA-Spezialist + Tester)
- Metamodelle & Templates für den Generator entwickeln
- Usability Patterns „entdecken“ und anwenden
- Tool-Integration verbessern: MDA = Forward Engineering
- Java: RTSJ, Java ME -> SE, EE-Anbindung, Produkte
- Konkrete Anwendung, z.B. Berliner Polizei

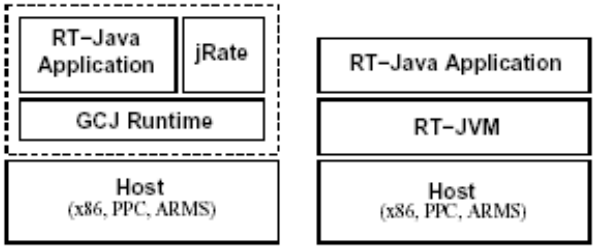
Real-time Specification for Java (RTSJ)

- Connected Device Configuration (CDC): für „mobile applications“ (auf „personal mobile devices“ mit wenig Resourcen: ~2 MB RAM, ~2.5 MB ROM, compatibility mit J2SE APIs, CDC: Foundation Profile, Personal Basis Profile, Personal Profile.
- Connected Limited Device Configuration (CLDC) für sehr geringe Resourcen, ~128 KB - 512 KB RAM, CLDC enthält das Mobile Information Device Profile (MIDP) für GUI
- MIDP + CDLC = Mobile Phone / PDA
- Java ME Wireless Toolkit: Java Technology for the Wireless Industry (JTWI) Roadmap 1 specification, support for Wireless Messaging API (WMA) 2.0 (JSR 205), Mobile Media API (MMAPI) 1.1 (JSR 135), PDA Optional Packages (JSR 75), Java API's for Bluetooth (JSR 82), Mobile 3D Graphics (JSR 184), and J2ME Web Services API (JSR 172)
- Mobile Information Device Profile (MIDP):
  User Interface APIs für „wireless devices“
  (Minimum: 96 Pixel breit * 54 Pixel hoch),
  Elemente:
  List, Alert / Text Box, Form ...
- MIDP + CDLC = Mobile Phone / PDA
- Java ME Wireless Toolkit: Java Technology

Performance: 2 Plattformen für RT Java

- UCI: jRate und GCJ
  (Gnu Compiler for Java),
  ahead-of-time-Compilation
  into native code,
  RTSJ Services:
  GC, RT Threads ...

- TimeSys: RTSJ
  Reference Implementation
  (Basis: J2ME JVM),
  interpreted execution mode
  (kein JiT-Compiler)



Quelle: Corsaro et al.: Evaluating Real-Time Java Features and Performance for Real-time Embedded Systems

Quelle: Corsaro et al.: Evaluating Real-Time Java Features and Performance for Real-tim