Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Janosch Hirsch, Robert Keßler,
Tim Krämer, Marko Schnecke
[0hirsch,8kessler,7kraemer,3schneck]
@informatik.uni-hamburg.de

Performance, Security, Testability, Usability, Other Q. Attributes

02. Dezember 2013

meta

Fragen gerne direkt stellen!

Folien im MIN-Commsy-Raum.

outline

Performance

Security

Testability

Usability

Other Quality Attributes

Was ist Performance?

- Jedes Softwaresystem soll seine Aufgaben innerhalb fest definierter Zeitintervalle vollenden.
- Aufgaben erreichen das System in verschiedenen Formen
  - Interrupts, Nachrichten, Anfragen, Timing Events
- Die Beurteilung der Performance eines Systems variiert
  - Durchsatz vs. Güte
- Performance ist nicht mehr der alleinige treibende Faktor in der Softwarearchitektur
- Performance und Skalierbarkeit sind verschiedene Qualitätsattribute

Performancemaße

- Latenz
  - Zeit zwischen Ankunft und Vollendung einer Aufgabe
- Einhaltung von Deadlines
  - Werden alle Deadlines eingehalten?
- Durchsatz
  - Transaktionen pro Zeiteinheit
- Jitter
  - Die Varianz der Latenz
- Miss-rate
  - Verhältnis der verworfenen Aufgaben zu allen Aufgaben

Ankunftsmuster

Aufgaben können in vorhersehbaren Mustern oder unvorhersehbar auftreten.

- Periodisch
  - Auftreten in festen Zeitintervallen
- Stochastisch
  - Auftreten in Wahrscheinlichkeitsverteilungen
- Sporadisch
  - Unvorhersehbares Auftreten

Programmausführungszustände

Sowie eine Aufgabe eintrifft, ist das System in einem von zwei möglichen Zuständen, um die Aufgabe zu vollenden

- Rechnend
  - Die Berechnung benötigt Ressourcen und Zeit
  - Performance von Ressourcen kann sich verschieden verhalten
- Blockiert
  - Wartend auf Ressourcen, andere Berechnungen oder zur Synchronisation
  - Umso mehr Ressourcen benötigt werden umso wahrscheinlicher ist eine größere Latenz

Taktiken zur Optimierung

Unterteilung in 2 Taktikkategorien:

- Ressourcenbedarf regeln
  - Taktiken zur Minimierung des Ressourcenbedarfs
- Ressourcen verwalten
  - Taktiken zur Effizienzsteigerung der Ressourcen

Kategorie 1: Ressourcenbedarf regeln

- Samplingrate anpassen
- Genauigkeit verringern
  - Die Latenz bleibt vorhersehbar
  - Verringerte Auflösung / Qualität
- Aufgabenpriorisierung
  - Bessere Latenz und Durchsatz für hochpriore Aufgaben
  - Niederpriore Aufgaben warten beliebig lange oder werden ignoriert
- Overhead verringern
  - *„Any performance problem can be solved by removing a layer of indirection“*
  - Co-Location
- Programmeffizienz steigern
  - z.B. durch bessere Algorithmen

Kategorie 2: Ressourcenverwaltung

- Vertikale Skalierung
  - Schnellere oder mehr CPUs, Speicher, Netzwerke, ...
- Horizontale Skalierung
  - Mehrere Server und load-balancing Taktiken
- Cache
  - um mehrfache Zugriffe auf langsame Ressourcen zu vermeiden
- Nebenläufigkeit einführen
- Scheduling der Ressourcen anpassen
  - z.B. Durchsatzsteigerung auf Kosten längerer Transaktionen

Scheduling

Besteht aus

- Prioritätszuweisung
- Ressourcenzuweisung

Ziele

- Optimale Ressourcennutzung
- Task Priorisierung
- Minimierung der Ressourcennutzung
- Minimierung der Latenz
- Maximaler Durchsatz
- Fairness

Preemption

Taskausführung unterbrechen, um einen anderen auszuführen

Möglichkeiten:

- Tasks dürfen jederzeit unterbrochen werden
- Tasks dürfen nur an spezifischen Stellen unterbrochen werden
- Tasks dürfen oder können nicht unterbrochen werden

Preemption

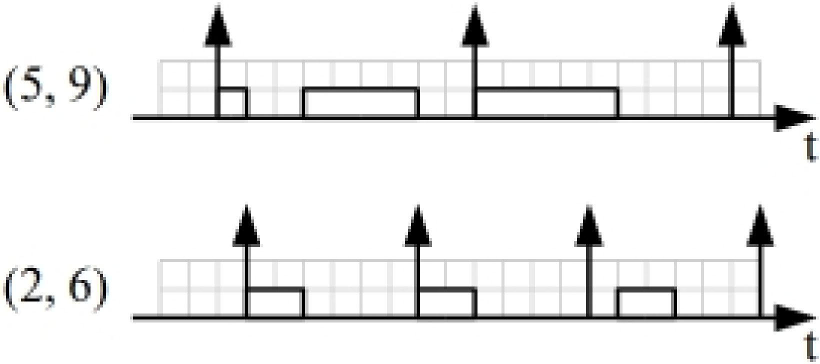


Abbildung : Schedule mit Preemption

Scheduling

- FIFO (First In First Out)
- Fixed-priority scheduling
  - □ Anhand von semantischer Wichtigkeit
  - □ Deadline Monotonic
  - □ Rate Monotonic
- Dynamic-priority scheduling
  - □ Round-Robin
  - □ EDF (Earliest Deadline First)
  - □ Least-slack-first
- Static Scheduling

Zusammenfassung

- Bei Performance geht es um das Management der Ressourcen, um akzeptables Timing des Systems zu erreichen
- Performance kann durch Latenz und Durchsatz bemessen werden
- Verringerung des Ressourcenbedarfs kann negative Auswirkungen auf Qualitäten haben
- Replikation oder bessere Hardware sind gute Ansätze zur Performancesteigerung

Questions?

Next up: Security

Motivation

my systems are secure, right?

Security as a quality attribute in s. w. a.

„*Security is a measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized.*"

- Confidentiality - Vertraulichkeit
  - □ Is my data/system protected from unauthorized **access**?
- Integrity - Unversehrtheit
  - □ Is my data/system protected from unauthorized **manipulation**?
- Availability - Verfügbarkeit
  - □ Will my data/system be available for **legitimate use**?

characteristics to support CIA

- Authentication - Authentisierung
  - Is it really you, I'm communicating with?
- Nonrepudiation - Unleugbarkeit
  - Can I be sure, you won't later deny having received my message?
- Authorization - Berechtigung
  - Will a user be able to edit his own account?

Attacks

> *„An attack is an attempt to break CIA [...]“*

Threat modeling as a technique to determine possible threats.

- attack trees
- security scenarios

Tactics for Security

Think about physical security.

- detect
  - Detect intrusion (traffic within the system)
  - Detect service denial (traffic coming in)
  - ...
- resist
  - Authenticate actors (password, certifacte, ...)
  - Authorize actors (access control, right management)
  - Encrypt data (VPN, SSL, Public-/Private Keys)
  - ...

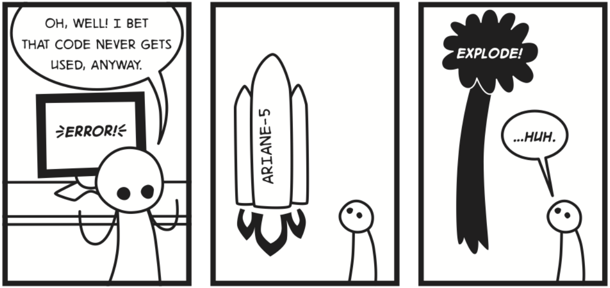
Tactics for Security

- react
  - Revoke access (limit access to minimum)
  - Lock computer (e.g.: don't allow more than 5 login attempts)
  - Inform actors (notify personnel, other systems)
- recover
  - Maintain Audit Trail (log user/system actions)
  - Restore (see Availability)

Questions?

Next up: Testability

Why test systems?

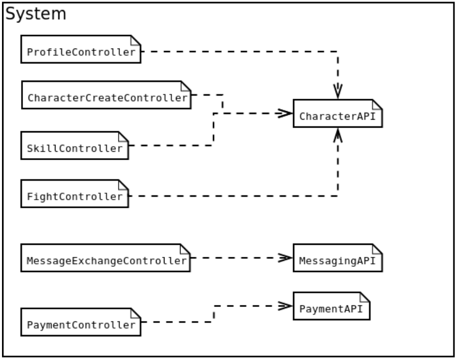


What do we need to do?

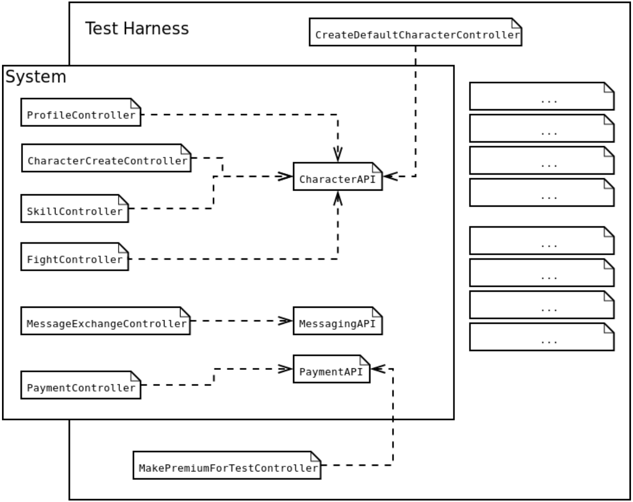
- Control Input and Observe Outputs
- Change internal states and observe changes

→ Mechanisms bundled together in a so called test harness

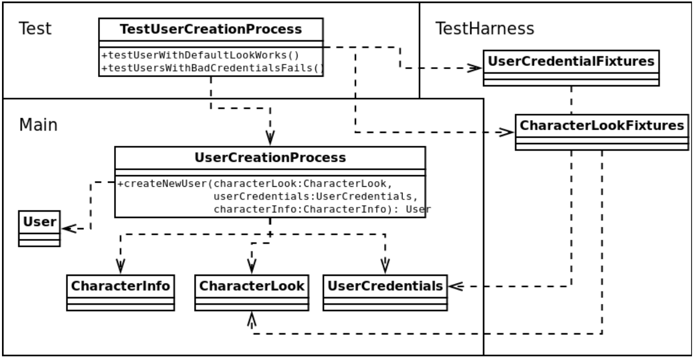
Test Harness I



Test Harness II



Test Harness III - Unit Level


Test
TestUserCreationProcess
+testUserWithDefaultLookWorks()
+testUsersWithBadCredentialsFails()
TestHarness
UserCredentialFixtures
CharacterLookFixtures
Main
UserCreationProcess
+createNewUser(characterLook:CharacterLook,
userCredentials:UserCredentials,
characterInfo:CharacterInfo): User
User
CharacterInfo
CharacterLook
UserCredentials

Tactics for Testability in Theory

Recap:

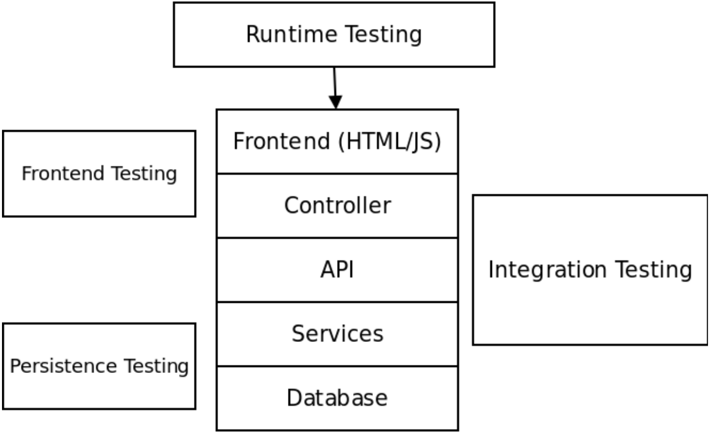
- Specialized Interfaces / Specialiced Access Mechanisms
- TestFixtures

New:

- A/B Feature Release
- Layered Testing with Mocks & Stubs [1]

[1] http://martinfowler.com/articles/mocksArentStubs.html

Testing Layered Architecture



Tactics for Testability in Practice I

"A SPELLING TEST? – SURELY THEY HAVE SOFTWARE FOR THAT SORT OF THING!"
Baloo
©2009 BALOOCARTOONS.COM

Tactics for Testability in Practice II

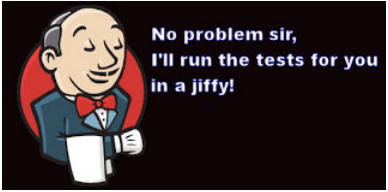
- Limit Non-determinism
- Adding Observability and Controllability
- Use executable Assertions

Design Checklist - What is still missing?

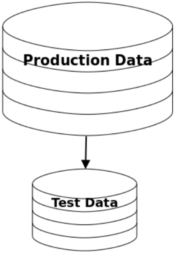
- Resource Management
  - □ sufficient resources are available
  - □ parallel execution if possible
  - □ representative test environment

- Choice of Technology
  - □ support your specific scenarios
  - □ support an appropriate level/amount of testing
  - □ sustainable choice of technology

Miscellaneous



About Test Automation



About Test Data Generation

Questions?

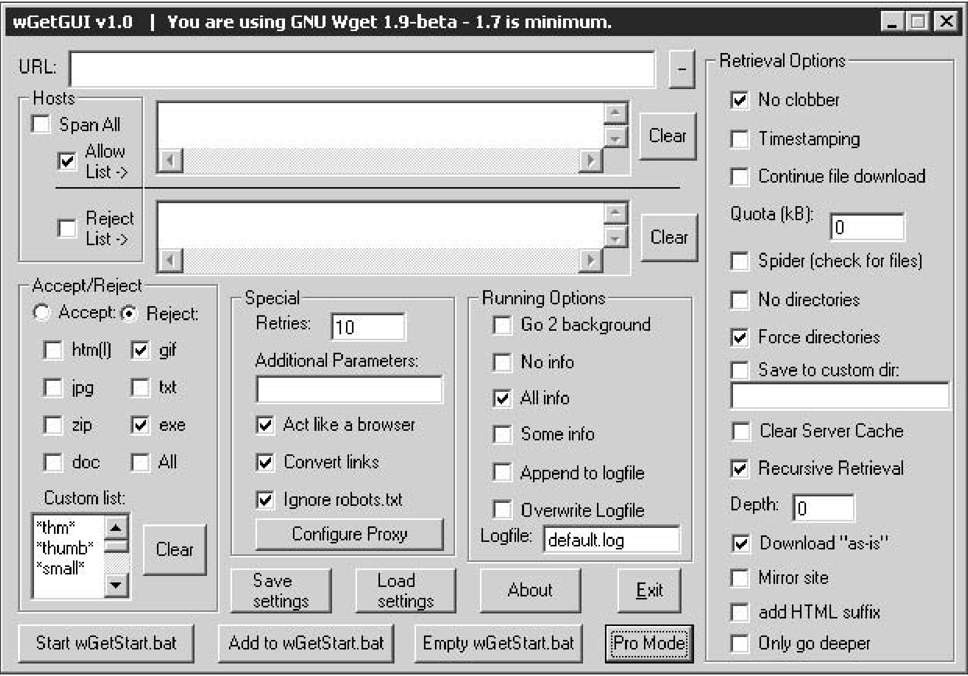
Next up: Usability

Usability

„Any darn fool can make something complex; it takes a genius to make something simple.“
Albert Einstein

What is Usability?

- Learning system features
- Using a system efficiently



wGetGUI v1.0 | You are using GNU Wget 1.9-beta - 1.7 is minimum.

URL:

Hosts

□ Span All

☑ Allow List ->

□ Reject List ->

Accept/Reject

○ Accept: ● Reject:

□ htm(l) ☑ gif

□ jpg □ txt

□ zip ☑ exe

□ doc □ All

Custom list:

thm
thumb
small

Special

Retries: 10

Additional Parameters:

☑ Act like a browser

☑ Convert links

☑ Ignore robots.txt

Running Options

□ Go 2 background

□ No info

☑ All info

□ Some info

□ Append to logfile

□ Overwrite Logfile

Logfile: default.log

Retrieval Options

☑ No clobber

□ Timestamping

□ Continue file download

Quota (kB): 0

□ Spider (check for files)

□ No directories

☑ Force directories

□ Save to custom dir:

□ Clear Server Cache

☑ Recursive Retrieval

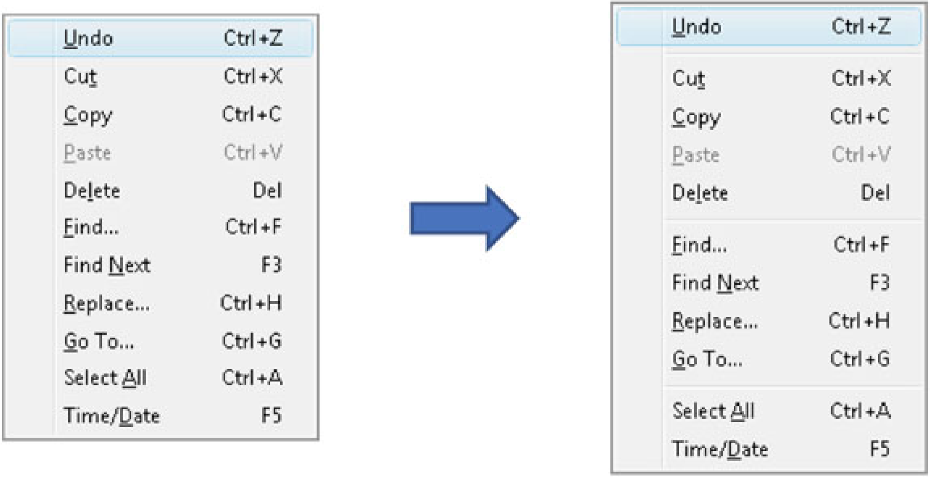
Depth: 0

☑ Download "as-is"

□ Mirror site

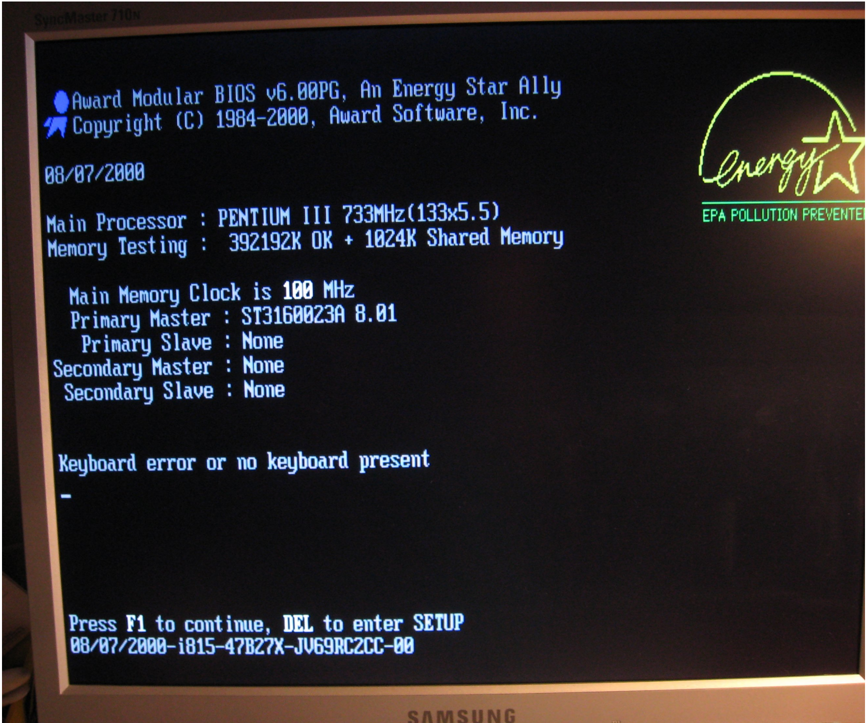
□ add HTML suffix

□ Only go deeper


Undo Ctrl+Z
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Find... Ctrl+F
Find Next F3
Replace... Ctrl+H
Go To... Ctrl+G
Select All Ctrl+A
Time/Date F5
Undo Ctrl+Z
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Find... Ctrl+F
Find Next F3
Replace... Ctrl+H
Go To... Ctrl+G
Select All Ctrl+A
Time/Date F5

What is Usability?

- Minimizing the impact of errors
- Adapting the system to user needs
- Increasing confidence and satisfaction



```
Award Modular BIOS v6.00PG, An Energy Star Ally
Copyright (C) 1984-2000, Award Software, Inc.

08/07/2000

Main Processor : PENTIUM III 733MHz(133x5.5)
Memory Testing :  392192K OK + 1024K Shared Memory

  Main Memory Clock is 100 MHz
  Primary Master : ST3160023A 8.01
   Primary Slave : None
 Secondary Master : None
  Secondary Slave : None


 Keyboard error or no keyboard present
 _



 Press F1 to continue, DEL to enter SETUP
 08/07/2000-i815-47B27X-JV69RC2CC-00
```

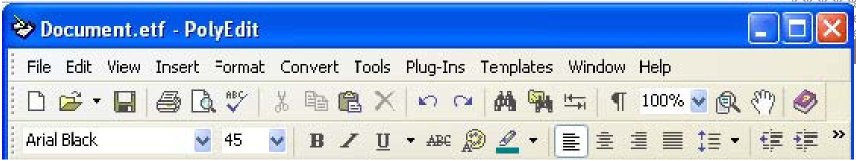


Abbildung : Old Office

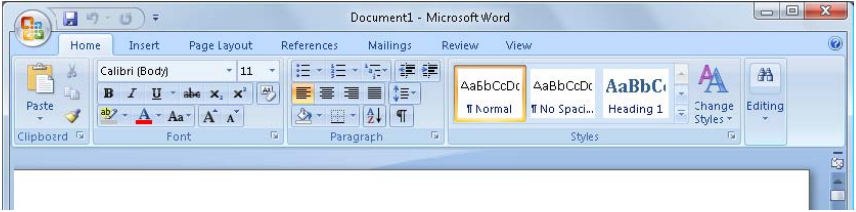


Abbildung : New Office

Usability General Scenario

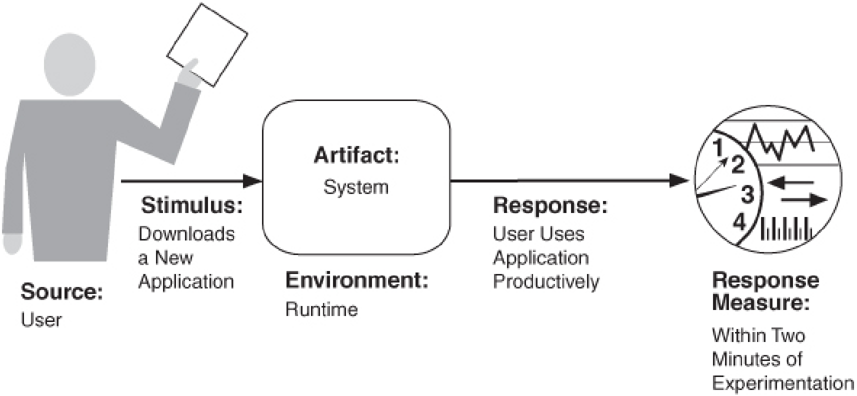

Source:
User
Stimulus:
Downloads a New Application
Artifact:
System
Environment:
Runtime
Response:
User Uses Application Productively
1
2
3
4
Response Measure:
Within Two Minutes of Experimentation

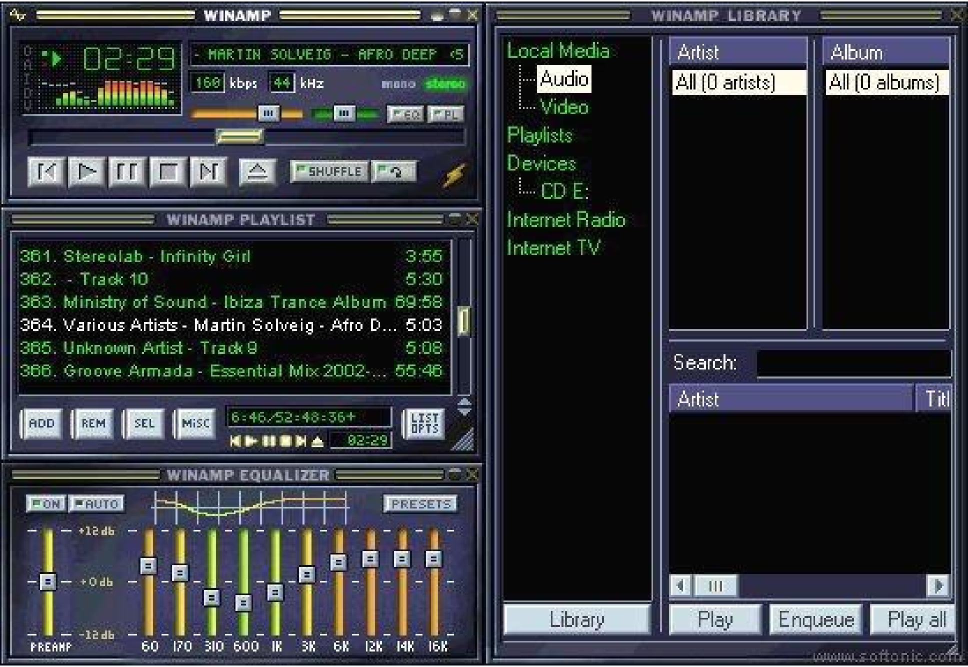
Abbildung : Beispiel Szenario

Usability and other QAs

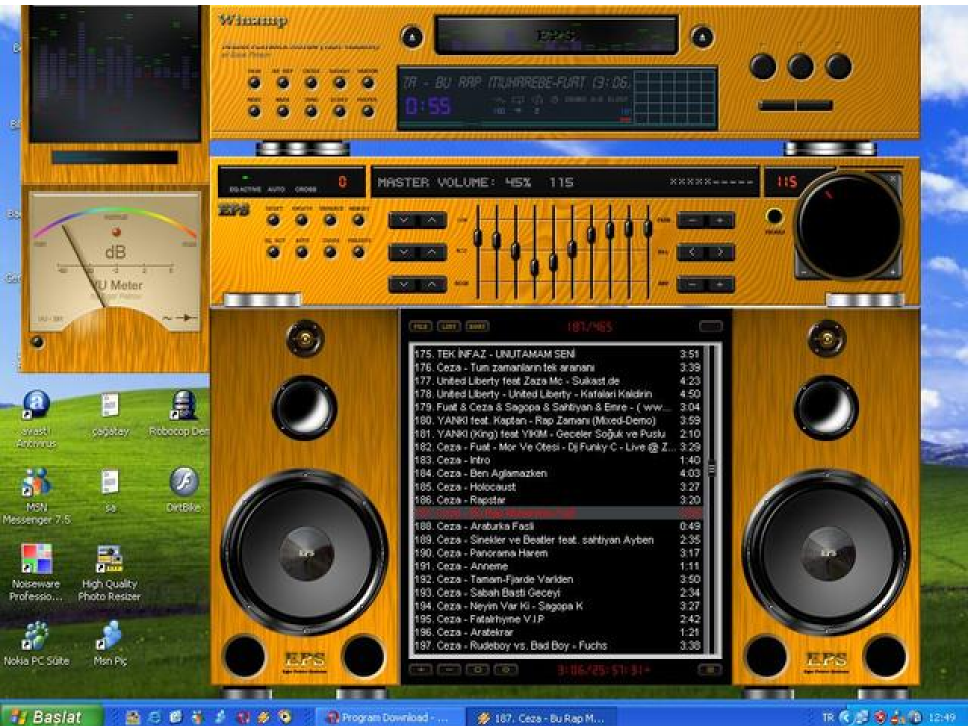
- Often in conflict with other Quality Attributes
- Examples:
  - □ Security: Webshop logout after 5 minutes
  - □ Performance: Loading parts of the UI from a server

Tactics for Usability

- Separate the User Interface
- Reduce dependencies → better Modifiability
- Defer binding → no need to rebuild for every change
- Rapid Prototyping → testing different UIs
- Model-View-Controller Pattern



```
361. Stereolab - Infinity Girl                       3:55
362. - Track 10                                      5:30
363. Ministry of Sound - Ibiza Trance Album         69:58
364. Various Artists - Martin Solveig - Afro D...    5:03
365. Unknown Artist - Track 9                        5:08
366. Groove Armada - Essential Mix 2002-...         55:46
```



TR - BU RAP MUHAREBE-FUAT (3:06)

0:55

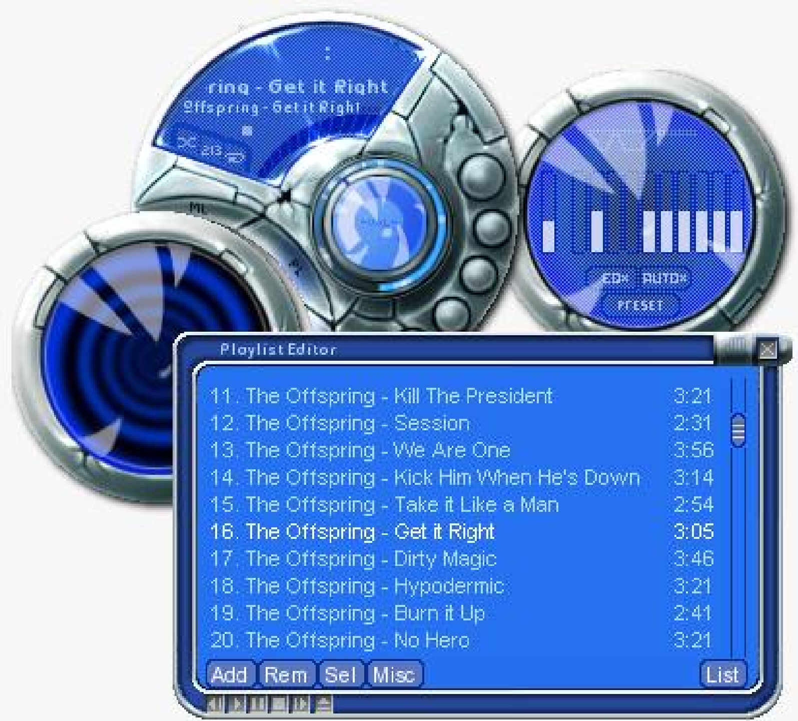
MASTER VOLUME: 45% 115

115

187/465

```
175. TEK İNFAZ - UNUTAMAM SENİ                          3:51
176. Ceza - Tum zamanlarin tek aranani                   3:39
177. United Liberty feat Zaza Mc - Suikast.de            4:23
178. United Liberty - United Liberty - Kafalari Kaldirin 4:50
179. Fuat & Ceza & Sagopa & Sahtiyan & Emre - ( ww...    3:04
180. YANKI feat. Kaptan - Rap Zamani (Mixed-Demo)        3:59
181. YANKI (King) feat YIKIM - Geceler Soğuk ve Puslu    2:10
182. Ceza - Fuat - Mor Ve Otesi - Dj Funky C - Live @ Z... 3:29
183. Ceza - Intro                                        1:40
184. Ceza - Ben Aglamazken                               4:03
185. Ceza - Holocaust                                    3:27
186. Ceza - Rapstar                                      3:20
187. Ceza - Bu Rap Muharebe-Fuat                         3:06
188. Ceza - Araburka Fasli                               0:49
189. Ceza - Sinekler ve Beatler feat. sahtiyan Ayben     2:35
190. Ceza - Panorama Harem                               3:17
191. Ceza - Anneme                                       1:11
192. Ceza - Tamam-Fjarde Varlden                         3:50
193. Ceza - Sabah Basti Geceyi                           2:34
194. Ceza - Neyim Var Ki - Sagopa K                      3:27
195. Ceza - Fatalrhyme V.I.P                             2:42
196. Ceza - Arabekrar                                    1:21
197. Ceza - Rudeboy vs. Bad Boy - Fuchs                  3:38
```

3:06/25:51:31+



11. The Offspring - Kill The President 3:21
12. The Offspring - Session 2:31
13. The Offspring - We Are One 3:56
14. The Offspring - Kick Him When He's Down 3:14
15. The Offspring - Take it Like a Man 2:54
16. The Offspring - Get it Right 3:05
17. The Offspring - Dirty Magic 3:46
18. The Offspring - Hypodermic 3:21
19. The Offspring - Burn it Up 2:41
20. The Offspring - No Hero 3:21

User Initiative

- Gives the user the opportunity to influence the system
- Common examples:
  - Cancel
  - Undo
  - Pause/resume
  - Aggregate

System Initiative

- System is initiating an action of its own
- Needs an apropriate Model:
  - □ Task model → correcting a lowercast letter at the beginning of a sentence
  - □ User model → providing an explanation of every for every function for every user only once
  - □ System model → showing a progress bar

Questions?

Next up: Other Quality Attributes

Other Quality Attributes

Deployability

- How does software gets to the customer?
  - Pull vs. Push
  - Update on runtime
  - Medium of transportation

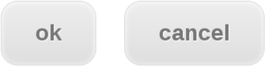
- Examples
  - Steam-Games
  - Google Play & App-Store
  - System Updates (Linux/Win/Mac/...)

Why is deployability a concern?

With the growth of the internet the transportation medium for updates generates very low costs. Internet for frequent updates or as the initial distribution way became attractive.

1. Regular Releases
2. Weekly Releases
3. Continuous Delivery

Safety



Safety

ok
cancel
destroy everything!

Safety

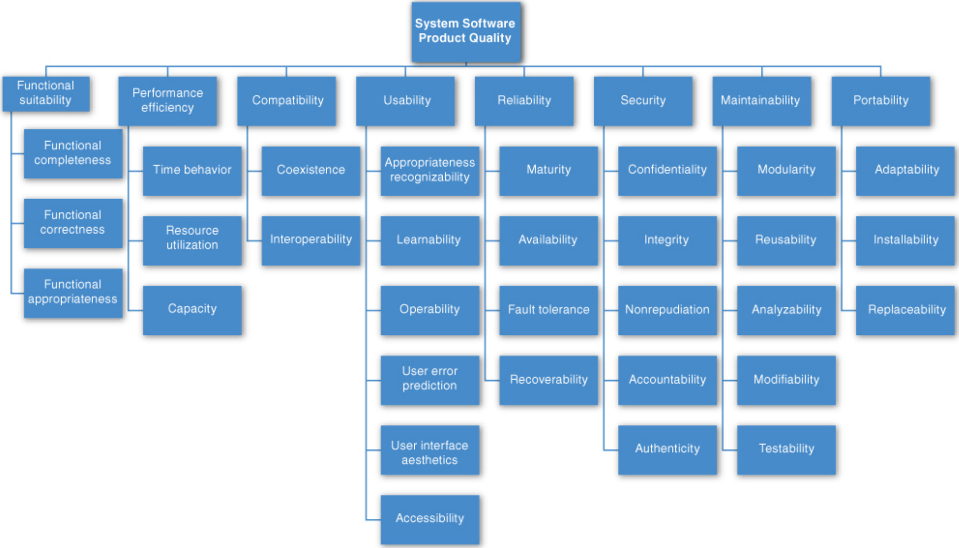
- Safety $\neq$ Security
  - □ avoid entering states that cause or lead to damage
- Safety $\neq$ Reliability
  - □ a reliable system (consistent with its specification) is unsafe, if the the specification ignores conditions leading to unsafe action
- Safety is not purely a software concern

System QAs and Software QAs

- Software choices always have an impact on the containing system
- Example:
  - □ An instant messaging app that contacts the server asking for new messages every 5 seconds → high performance but wrecked battery running time
- It's important for an software architect to have the important system quality attributes in mind

Using standard lists of quality attributes

- Standard lists are helping with non common QAs
- Normaly they are categorized so one can see immediately in wich direction the QA is heading
- Its not possible to get all quality attributes - there is always another one (example: lowability)


System Software Product Quality
Functional suitability
Functional completeness
Functional correctness
Functional appropriateness
Performance efficiency
Time behavior
Resource utilization
Capacity
Compatibility
Coexistence
Interoperability
Usability
Appropriateness recognizability
Learnability
Operability
User error prediction
User interface aesthetics
Accessibility
Reliability
Maturity
Availability
Fault tolerance
Recoverability
Security
Confidentiality
Integrity
Nonrepudiation
Accountability
Authenticity
Maintainability
Modularity
Reusability
Analyzability
Modifiability
Testability
Portability
Adaptability
Installability
Replaceability

Getting a grasp on X-abilities

- Capture Scenarios
  - Interview the stakeholders about the new QA
  - Try to generealize the szenarios
- Assemble Design Approaches
  - Find design patterns that have an impact on the new QA
  - Finding experts in this area and interview them
  - Using the general scenario to try to catalog a list of design approaches to produce the responses
- Try to create a model
  - Collect all parameters that have an impact
- Assemble a Set of Tactics
  - Interview experts
  - Use the model

Danke für die Aufmerksamkeit!

Fragen? Diskussion!