

# **Systementwicklungsprojekt**

## **„Entwurf und Implementierung einer Plattform zur Auslieferung mobiler Inhalte“**

Referent: Prof. Dr. Dr. h.c. Manfred Broy  
Fakultät für Informatik  
Lehrstuhl IV: Software & Systems Engineering  
Technische Universität München

Betreuer: Peter Dornbusch  
CDTM – Center for Digital Technology &  
Management

Studiengang: Informatik mit Nebenfach  
Wirtschaftswissenschaften

Eingereicht von: Alexander Scherer

Eingereicht am: München, den 26. Januar 2006

## INHALTSVERZEICHNIS

Motivation.....	8
Videocodierung mobiler Endgeräte.....	8
Nativer Ansatz.....	14
Javas native Schnittstelle.....	15
MPlayer und MEncoder.....	16
JFFmpeg.....	18
FOBS.....	18
FFmpeg.....	19
Installation von FFmpeg.....	20
Java Process Builder.....	20
Plattformintegration.....	22
Erweiterungsmöglichkeiten.....	28
Webseitenverzeichnis.....	29
Literaturverzeichnis.....	31
Anhangs CD.....	32

## TABELLENVERZEICHNIS

Tabelle 1: Originalvideo transcodiert in 3GP in verschiedenen Bitraten (QCIF Auflösung)...	12
Tabelle 2: Empfohlene Parameter zur Videocodierung für Mobiltelefone.....	14
Tabelle 3: Auszug der 183 unterstützten Video-Codecs.....	15
Tabelle 4: Parameter des MEncoders.....	17
Tabelle 5: Auszug der von FFmpeg unterstützen 38 Audio und 63 Video Codecs.....	19
Tabelle 6: FFmpeg Parameter zur Erzeugung von Vorschaubildern.....	23
Tabelle 7: FFmpeg Parameter zur Erzeugung von Flash Videos.....	24
Tabelle 8: Encodierungsspezifische Parameter des Mozean Projekts.....	25

## ABBILDUNGSVERZEICHNIS

Abbildung 1: Dateigröße in Abhängigkeit zur Audio-Bitrate am Beispiel; Wahl der Intervalle in Anlehnung an Nokia (2003).....	11
Abbildung 2: Dauer in Abhängigkeit zur Audio-Bitrate am Beispiel bei fester Dateigröße....	11
Abbildung 3: Dateigröße in Abhängigkeit von Datendurchsatz.....	13
Abbildung 4: Dauer in Abhängigkeit der Bitrate bei fester Dateigröße.....	13
Abbildung 5: Die ProcessBuilder Klasse von Java Version 5.....	21
Abbildung 6: Die Process Klasse.....	22
Abbildung 7: Video-Vorschau im Flash Video Player.....	23
Abbildung 8: Automatisch generiertes Vorschaubild.....	23
Abbildung 9: Einbindung der Encodierung in die Mozean Architektur.....	27
Abbildung 10: Darstellung des Encodierungs-Prozesses als UML Aktivitätendiagramm.....	27

## GLOSSAR

- 3GP** ist ein von 3GPP entwickeltes Dateiformat, welches insbesondere Video- und Audiokomprimierung beinhaltet und von fast allen marktüblichen Mobiltelefonen zur Darstellung von Video-Inhalten unterstützt wird.<sup>1</sup>
- 3GPP** Im Dezember 1998 schlossen sich fünf sogenannte “Organizational Partners” zum „3rd Generation Partnership Project (3GPP)“ zusammen um Standards im Mobilfunk zu schaffen<sup>2</sup>.
- AMR** Adaptive Multirate Codec (AMR)<sup>3</sup> ist ein Audiocodec. Es werden zwei Typen unterschieden: Narrowband und Wideband, welche sich in der codierten Bitrate unterscheiden. In aktuellen Mobiltelefone wird als Audio-Spur von Video-Inhalten hauptsächlich der Narrowband Codec eingesetzt.
- CIF** Das Common Intermediate Format (CIF)<sup>4</sup> entspricht einer Auflösung von  $352 \times 288$  Pixel und wurde im Standard H.323<sup>5</sup> der ITU („International Telecommunication Union“) definiert.
- Codec** „Als Codecs (englisches Akronym aus coder und decoder) bezeichnet man Verfahren bzw. Programme, die Daten oder Signale digital codieren und decodieren.“<sup>6</sup>
- Containerformat** „In der Computertechnik bezeichnet man als Container (engl. für Behälter) ein Dateiformat, dessen Inhalt mehrere andere Dateiformate erlaubt. Typischerweise definiert ein Containerformat nur die Art und Struktur, wie der Inhalt aufzubewahren ist. Container ermöglichen so zum Beispiel das synchrone Wiedergeben von Audio- und Video-Spuren.“<sup>7</sup>

---

1 3GPP (2005)

2 3GPP (03.01.2006)

3 3GPP (2004)

4 Kang / Leou (2005)

5 ITU (07.01.2006)

6 Wikipedia (06.01.2006): Codec

7 Wikipedia (06.01.2006): Containerformat

<b>CVS</b>	Concurrent Versions System (CVS) <sup>8</sup> ist ein Programm zur Versionierung von u.a. Quelltext.
<b>Decoder</b>	Ein Decoder stellt das Gegenstück zum Encoder dar; er wandelt die vom Encoder konvertierten Daten so zurück, dass diese vom Empfänger dargestellt werden können.
<b>Encoder</b>	„Ein Encoder ist ein System, das eine Datenquelle (z.B. ein digitales Audiosignal, ein Dateiformat, ein Computerbild, ein gegen Fehler empfindliches Datensignal) in ein für einen bestimmten Kanal geeignetes Format umwandeln soll. Ein 'Encoder' arbeitet nach einer fest vorgegebenen Codiervorschrift, damit der Decoder auf der Empfängerseite das Signal wieder in das ursprüngliche Format zurückkonvertieren kann.“ <sup>9</sup>
<b>GSM</b>	„Das Global System for Mobile Communications (GSM) ist ein volldigitaler Mobilfunknetz-Standard, der hauptsächlich für Telefonie aber auch für leitungsvermittelte und paketvermittelte Datenübertragung sowie Kurzmitteilungen (Short Messages) genutzt wird. Es ist der erste Standard der sogenannten zweiten Generation ("2G") als Nachfolger der analogen Systeme der ersten Generation und ist der weltweit am meisten verbreitete Mobilfunk-Standard.“ <sup>10</sup>
<b>H.263</b>	ist ein Videocodec zur Encodierung von Video-Inhalten für niedrige Datenraten, wie sie in Mobiltelefonen vorzufinden find.
<b>Header-Datei</b>	„Eine Header-Datei ist in der Programmierung [...] eine Textdatei, die Deklarationen und andere Bestandteile des Quelltextes enthält. Quelltext, der sich in einer Header-Datei befindet, ist im Allgemeinen zur Verwendung in mehreren Teilen des Programmes vorgesehen.“ <sup>11</sup>
<b>JNI</b>	Java Native Interface ist eine Java-Schnittstelle, welche die direkte Einbindung von nativem Quelltextes in Java ermöglicht.

---

8 Nongnu (06.01.2006)

9 Wikipedia (06.01.2006): Encoder

10 Wikipedia (15.01.2006): Global System for Mobile Communications

11 Wikipedia (06.01.2006): Header-Datei

<b>JSP</b>	Abkürzung für Java Server Pages – eine Java Technologie, welche die dynamische und Plattform unabhängige Webseiten-Generierung ermöglicht. <sup>12</sup>
<b>JMF</b>	Das Java Media Framework (JMF) bietet die Möglichkeit Audio- und Video-Inhalte direkt in Java Anwendungen einzubinden, wobei es neben der Wiedergabefunktion auch die Möglichkeit Inhalte zu transcodieren zu Verfügung stellt.
<b>Makefile</b>	Make <sup>13</sup> ist eine Anwendung zur automatisierten Erzeugung von ausführbaren Dateien aus Quelltext. Dabei holt es sich sein Wissen aus einem Makefile, das alle Informationen zur Generierung bereitstellt.
<b>MIME-Typ</b>	Der MIME (Multimedia Internet Message Extensions) Typ ist der Header, der den Body eines Dokuments klassifiziert, sodass der Empfänger bei einer Übertragung weiß welche Anwendung zur Darstellung verwendet werden kann.
<b>Mozean</b>	Eine „... Handelsplattform für mobile Inhalte und Dienste“ <sup>14</sup> (siehe <a href="http://www.mozean.de">http://www.mozean.de</a> )
<b>MP3</b>	In einer Kooperation mit der Universität Erlangen entwickelte das Fraunhofer Institut einen Algorithmus zur Audio-Kompression, der als ISO-MPEG Audio Layer-3 (IS 11172-3 und IS 13818-3) standardisiert ist <sup>15</sup> .
<b>MPEG4</b>	ist ein von der Moving Picture Experts Group (MPEG) entwickelter ISO Standard, der Audio- und Videocodierungsroutinen beschreibt. <sup>16</sup>
<b>Nativ</b>	Im Sinne eines Programms bedeutet nativ, dass es für eine bestimmte Umgebung compiliert wurde und deshalb nur noch in dieser eingesetzt werden kann.
<b>QCIF</b>	Quater CIF entspricht einer Auflösung von 176 × 144 Pixel.

---

12 Sun (20.01.2006)

13 GNU (06.01.2006)

14 Dornbusch (2005)

15 Fraunhofer (15.01.2006)

16 ISO (2002)

<b>Sample Rate</b>	„Mit Abtastrate, auch Samplingfrequenz oder Samplingrate, bezeichnet man in der Signalverarbeitung und ihren Anwendungen den Kehrwert des Abtastintervalls.“ <sup>17</sup>
<b>SQCIF</b>	Sub QCIF entspricht einer Auflösung von $128 \times 96$ Pixel.
<b>Tomcat</b>	Ist eine Servertechnologie welche die Ausführung von Java-Code, der in JSPs enthalten ist, ermöglicht und Clients eine im Browser darstellbare Webseite erzeugt. <sup>18</sup>
<b>Transcoder</b>	„Transcodierung ist das direkte Umwandeln von einem Video- oder Audioformat in ein anderes (beispielsweise von MPEG-2 nach MPEG-4 (Videokompression) oder von MP3 nach WMA (Audiokompression)).“ <sup>19</sup>
<b>UMTS</b>	Universal Mobile Telecommunications System ist ein Mobilfunkstandard, der GSM ablösen wird, und höhere Datenübertragungsraten und verschiedene multimediale Dienste umfasst. <sup>20</sup>

---

17 Wikipedia (06.01.2006): Abtastrate

18 Apache (20.01.2006)

19 Wikipedia (06.01.2006): Transcodierung

20 ETSI (1999)

## Motivation

Aufgrund des rasant wachsenden Mobilfunkmarktes und immer neuer technischer Fähigkeiten, welche mobile Endgeräte bieten, entstehen laufend neue Kundenbedürfnisse. Einige dieser Bedürfnisse werden durch Mozean, eine „... Handelsplattform für mobile Inhalte und Dienste“<sup>21</sup>, befriedigt.

An dieser Stelle besonders hervorzuheben ist die verstärkte Verbreitung von videofähigen Mobiltelefonen. Denn auf Grund dessen entstand die Idee, zusätzlich zur vorhandenen Möglichkeit Fotos zu handeln, Video-Handel in die Mozean Plattform zu integrieren. Zur Bereitstellung von Videomaterial über die Plattform sind sowohl die individuelle Aufbereitung der Inhalte für marktübliche Endgeräte als auch der automatisierte Versand im Format des Empfängergerätes nötig. Daraus entstanden zwei Systementwicklungsprojekte, wobei sich das erste mit der Auslieferung und das hier dargelegte mit der Bereitstellung des Video-Inhalts beschäftigt.

Im Folgenden werden die verschiedenen Phasen skizziert, die während der Bearbeitung durchlaufen wurden. In der ersten Phase wurden verschiedene Möglichkeiten des Video-Encodierens evaluiert, um vorliegende Videos so aufzubereiten, dass sie von marktüblichen Endgeräten dargestellt werden können. Anschließend wurde das beste Verfahren ausgearbeitet und schließlich in die Mozean Plattform integriert.

## Videocodierung mobiler Endgeräte

Zur Darstellung von Video-Inhalten wird von heutigen Mobiltelefonen hauptsächlich das von 3GPP standardisierte Dateiformat 3GP verwendet<sup>22</sup>. Im Dezember 1998 schlossen sich fünf so genannte “Organizational Partners” zusammen um Standards im Mobilfunk zu schaffen<sup>23</sup>. Aus diesem Zusammenschluss entstand unter anderem das Dateiformat 3GP, welches insbesondere Video- und Audio-Komprimierung<sup>24</sup> beinhaltet und von allen marktüblichen Mobiltelefonen unterstützt wird<sup>25</sup>.

Das 3GP Dateiformat ist ein Containerformat, das als Video-Spur MPEG4, H.263 bzw. H.264 und als Audio-Spur AAC und AMR enthalten kann. Im Gegensatz zum MPEG4 Format unterstützt 3GP AMR Audio, was den wesentlichen Unterschied beider Formate darstellt.

---

21 Dornbusch (2005)

22 Jindal/Prasad/Ramkishor (2003)

23 3GPP (03.01.2006)

24 3GPP (2005)

25 Kampmann (2004)

Als MIME-Typ wird 'video/3gpp' für audiovisuelle Inhalte und 'audio/3gpp' für reines Audio verwendet<sup>26</sup>.

Da das Ziel dieses Systementwicklungsprojekts ist, möglichst viele Video-Inhalte für marktübliche Endgeräte aufzubereiten, müssen Video- und Audiomaterial transcodiert werden. Deshalb wurde letztlich 3GP als Zielformat gewählt, da es sich um einen aktuellen Standard handelt, der auch in Zukunft Verwendung finden wird.

Entscheidend für die Darstellung von Videos auf Mobiltelefonen sind mehrere Faktoren, die im Folgenden kurz erläutert werden:

- Containerformat
- Audio- bzw. Videocodec
- Audio- bzw. Video-Übertragungsrate
- Dateigröße
- Bildrate
- Auflösung

Wie bereits angedeutet, ist das Containerformat von äußerst entscheidender Bedeutung, denn von ihm hängt es ab, ob ein Inhalt auf einem speziellen Mobiltelefon dargestellt werden kann oder nicht, weshalb das 3GP Format gewählt wurde. Damit einher geht der verwendete Video- und Audiocodec. Bei allen Handys, die das 3GP Format unterstützen, ist die Kombination aus H.263 für Video und AMR Narrowband für Audio zum Abspielen von Video-Inhalten möglich. Dem liegt zugrunde, dass H.263 ursprünglich für Video-Telefonie konzipiert wurde, welche in Zukunft durch die UMTS Technologie verstärkt in die Endgeräte integriert werden wird, und der AMR Audiocodec bereits seit langem im GSM Standard zur Kompression von Ton eingesetzt wird.

Die Bitrate, also die Frequenz der Datenübertragung, wirkt sich maßgeblich auf die Dateigröße aus. Aufgrund der hohen Übertragungskosten von Daten über das Handynetz<sup>27</sup>, wird von aktuellen Mobiltelefonen die Dateigröße eines Downloads auf unter hundert Kilobyte beschränkt. Daher müssen zum einen Videos verkürzt werden, zum anderen muss versucht werden, die Größe der Video-Dateien möglichst klein zu halten, um den Verlust an Inhalten zu minimieren. Die Dateigröße kann durch die Wahl einer niedrigen Datenübertragungsrate klein gehalten werden. Dabei leidet jedoch die Qualität der einzelnen Videos (siehe Tabelle 1). Die Spezifikation des AMR Codecs sieht für die Datenrate von Audio in AMR Codierung Frequenzen zwischen 4,75 und 12,2<sup>28</sup> Kilobits pro Sekunde vor.

---

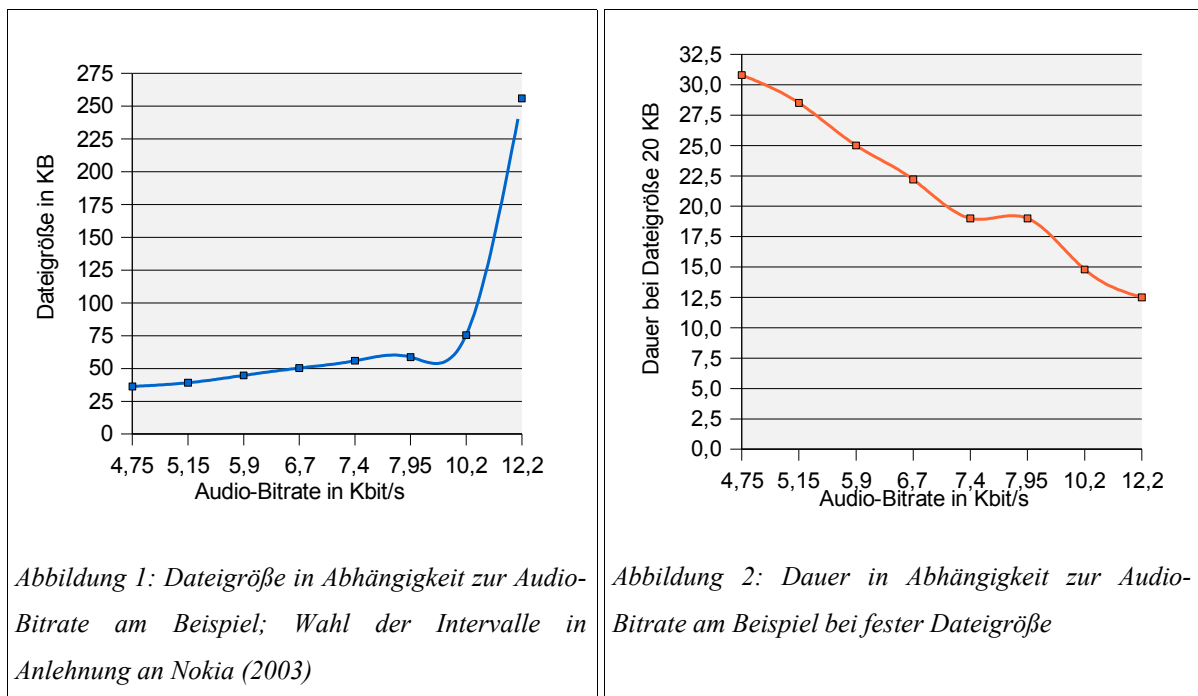
26 Vgl. The Internet Society (2004)

27 9 bzw. 19 Cent pro 10 Kilobyte im Standardtarif von T-Mobile, T-Mobile (15.01.2006)

28 Matta, J. et al. (2003), Seite 93

Da die Datenübertragungsrate zum einem entscheidend die audiovisuelle Qualität sowie die Dauer des Videos und zum anderen die Dateigröße beeinflusst, muss ein akzeptabler Kompromiss aus den einzelnen Faktoren gefunden werden. Deshalb wurden einige Tests durchgeführt, auf welche im Folgenden näher eingegangen wird.

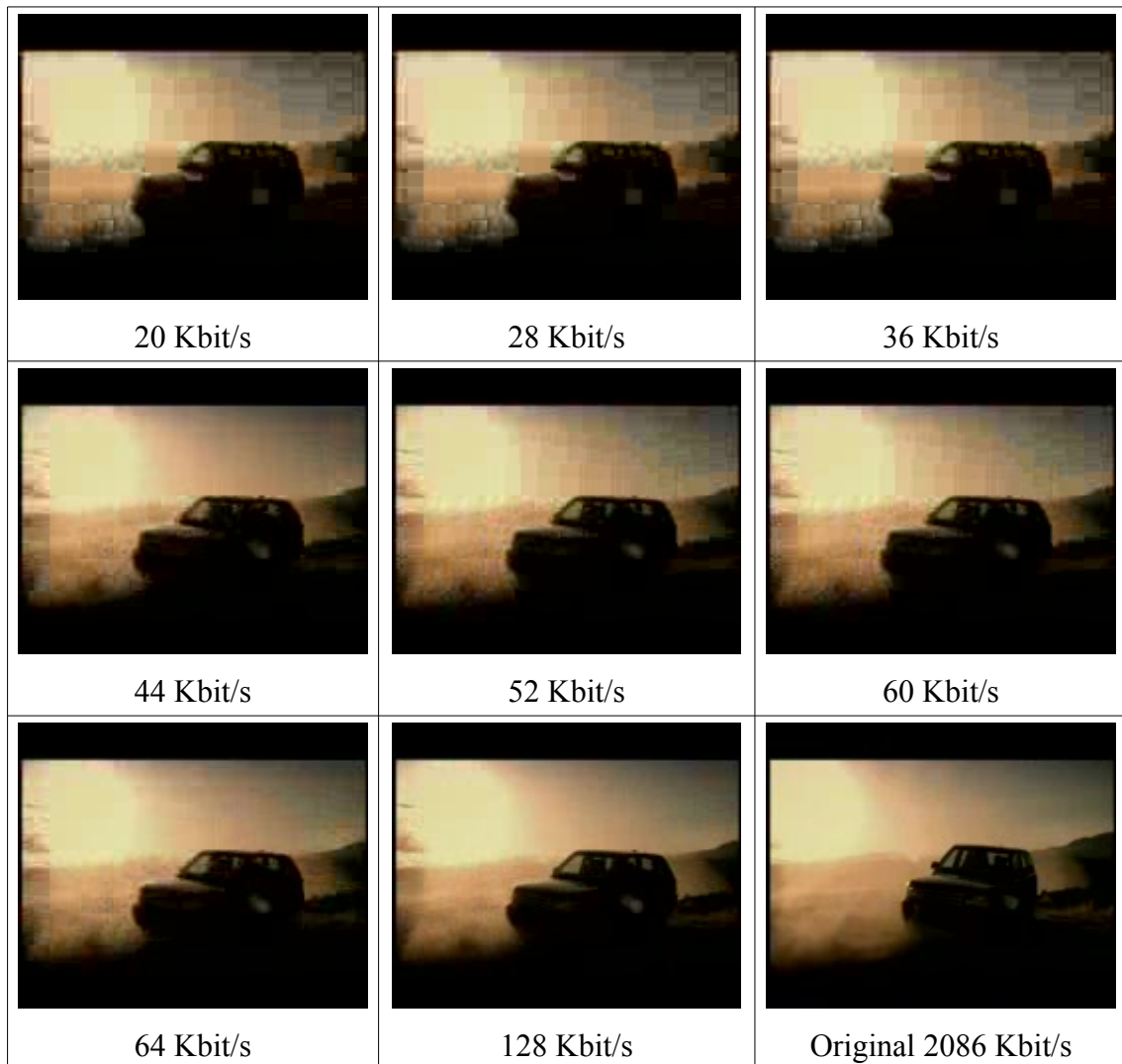
Mit Hilfe eines Beispiel-Videos<sup>29</sup> wurden verschiedene Übertragungsraten von Audio-Daten getestet. Die Ergebnisse sind Abbildungen 1 und 2 zu entnehmen. Da im niedrigen Frequenzbereich die Audio-Qualität sehr schlecht ist und die Steigung des Grafen in Abbildung 1 bei einer Übertragungsrate von 7,4 Kilobits pro Sekunde noch recht flach verläuft, wird diese Rate empfohlen.



In folgender Tabelle (Tabelle 1) sind Einzelbilder aus dem Beispiel-Video<sup>30</sup> in den jeweiligen Datenraten aufgeführt. Ab einer Frequenz der Datenübertragung von mehr als 64 Kilobits pro Sekunde ist das Bild kaum mehr vom Original zu unterscheiden:

<sup>29</sup> Jaguar Moravia (07.01.2006)

<sup>30</sup> Jaguar Moravia (07.01.2006)

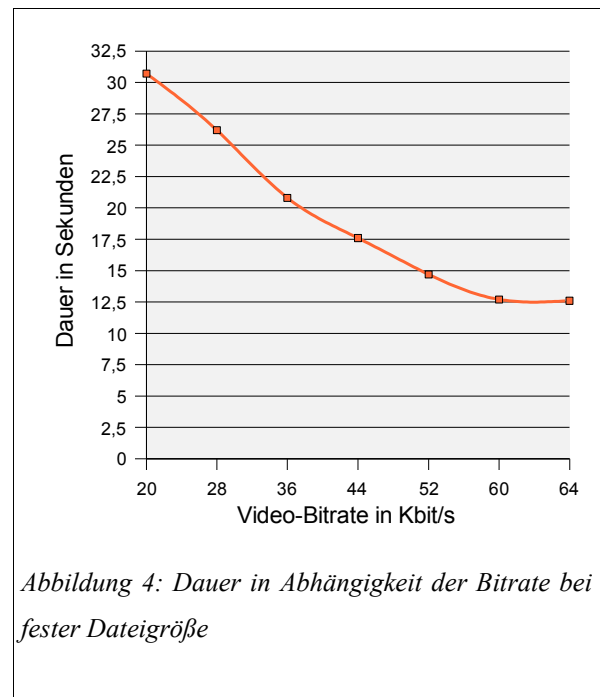
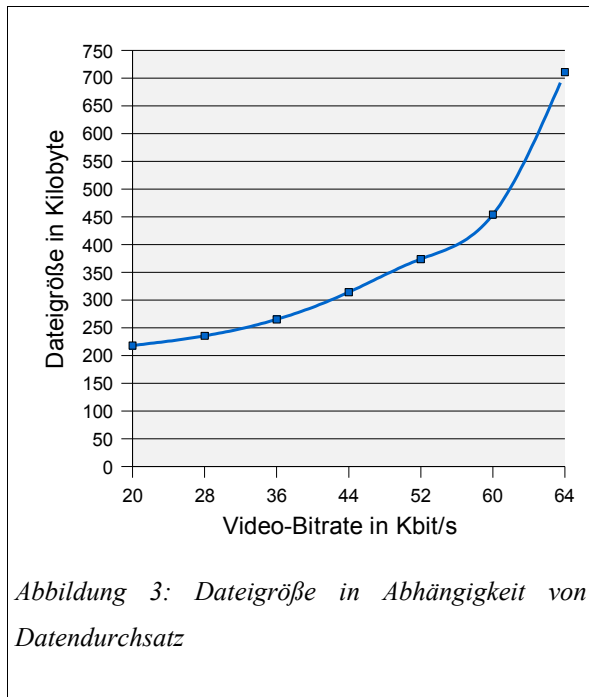


*Tabelle 1: Originalvideo transcodiert in 3GP in verschiedenen Bitraten (QCIF Auflösung)*

Neben der Audio-Spur stellt vor allem die Datenrate der Video-Spur eine Größe dar, welche die Dateigröße sowie die Dauer des transcodierten Videos stark beeinflusst.

Damit gewährleistet werden kann, dass das Videomaterial bei einer Dateigröße von weniger als hundert Kilobyte eine akzeptable Länge besitzt, muss auch hier ein Kompromiss zwischen Video-Dauer und Qualität eingegangen werden. In den Abbildungen 3 und 4 sind Dauer und Dateigröße eines Videos mit den Übertragungsraten in Relation gesetzt.

Da die Qualität bei einer Rate von 44 Kilobits pro Sekunde dem Original bereits sehr nahe kommt und die Videodatei bei ein Größe von weniger als hundert Kilobyte noch knapp zwanzig Sekunden Videomaterial umfasst, wird bei der Encodierung für Audio 7,4 Kilobits pro Sekunde und 44 Kilobits pro Sekunde für Video empfohlen. Insgesamt sollte die Datenrate also knapp 52 Kilobits pro Sekunde betragen.



Zwei weitere, die Dateigröße beeinflussende, Faktoren sind Bildwiederholungsrate und Auflösung. Da sich die Wahl der Bildrate entscheidend auf die Qualität des Videos auswirkt, und aus Beobachtungen<sup>31</sup> hervorgeht, dass die Bildqualität bei Videomaterial mit schnellen Bewegungen wichtiger ist, als der Ton, muss darauf geachtet werden, dass die Frequenz der Einzelbilder hoch genug ist. Da sich der Inhalt des Videos auf die Wahl der jeweiligen Bildrate auswirkt<sup>32</sup> und es prinzipiell möglich sein soll, beliebigen Inhalt auf die Mozean Plattform zu laden, wird eine Wiederholungsrate von zehn Bildern pro Sekunde empfohlen. Dem liegt auch zugrunde dass bei einer Datenrate von 44 Kilobit pro Sekunde manche Mobiltelefone<sup>33</sup> nicht mehr als zehn Bilder pro Sekunde darstellen können.

Schließlich stellt die Auflösung die letzte Einflussgröße auf die Video-Qualität, die Video-Dauer sowie die Dateigröße dar.

Aktuelle Mobiltelefone unterstützen zwei Auflösungen: Das QCIF und das Sub-QCIF bzw. SQCIF Format. Diesen beiden Formaten liegt das Common Intermediat Format<sup>34</sup> (CIF) zu Grunde, welches einer Auflösung von  $352 \times 288$  Pixeln entspricht. Bei QCIF handelt es sich um ein Viertel des CIF Formates, also  $176 \times 144$  Pixel, woher auch das Q („quater“) im Namen kommt. SQCIF ist ein Unterformat des QCIF („Sub QCIF“) Formats und entspricht einer Auflösung von  $128 \times 96$  Pixeln.

Da sich die Auflösung massiv auf die Qualität der Wahrnehmung auswirkt, muss

<sup>31</sup> Hands, D. (2004), S. 806

<sup>32</sup> vgl. Hands, D. (2004), S. 806ff

<sup>33</sup> Real Networks (2002), S. 3

<sup>34</sup> Kang / Leou (2005)

gewährleistet sein, dass das Video stets in der größtmöglichen Auflösung an das Mobiltelefon ausgeliefert wird. Das Systementwicklungsprojekt, welches sich mit der Auslieferung des Videomaterials beschäftigt, stellt eine Methode bereit, die Fähigkeiten des Empfänger-Gerätes zu analysieren und dementsprechend das Video in der bestmöglichen Auflösung auszuliefern. Zusammenfassend sind die Einflussgrößen sowie die Empfehlungen für die Parameterwahl in Tabelle 2 aufgeführt. Bei der Wahl der Anwendung, welche die Videos transcodiert, muss beachtet werden, dass es möglich ist, Einstellungen entsprechend Tabelle 2 vorzunehmen.

<i>Parameter der Encodierung</i>	<i>Empfohlener Wert</i>
Containerformat	3GP
Audio- bzw. Videocodec	AMR Narrowband bzw. H.263
Audio- bzw. Video-Übertragungsrate	7,4 bzw. 44 Kilobits pro Sekunde
Dateigröße	unter 100 Kilobyte
Bild Rate	10 Bilder pro Sekunde
Auflösung	QCIF bzw. SQCIF

*Tabelle 2: Empfohlene Parameter zur Videocodierung für Mobiltelefone*

Nachdem die Anforderungen an die Software formuliert worden sind, werden nun verschiedene Methoden des Transcodierens betrachtet und deren Integration-Möglichkeiten in die Mozean Plattform evaluiert.

## **Nativer Ansatz**

Zuerst wurde die Möglichkeit der nativen Einbindung von Transcodier-Methoden aus vorhandenen Projekten geprüft.

Es gibt zwei ausgereifte Projekte, die sich mit der Codierung von Video-Inhalten beschäftigen und deren Quelltext frei verfügbar ist:

- MPlayer bzw. MEncoder<sup>35</sup>
- FFmpeg<sup>36</sup>

Zuerst fiel die Entscheidung auf das MPlayer bzw. MEncoder Projekt, welches eine Vielzahl von Codecs (vgl. Tabelle 3) unterstützt und wofür es zahlreiche Mailinglisten gibt<sup>37</sup>.

---

35 Mplayer (02.01.2006)

36 FFmpeg (02.01.2006)

37 Mplayer (02.01.2006)

MPEG-1 or 2 (libmpeg2)	Windows Media Video 8
QuickTime Apple Video	Independent JPEG Group's codec
OpenDivX API (ODIVX DIVX4 DIVX5 XVID)	Win32 RealPlayer 9 RV40 decoder
DivX ;- ) (MS MPEG-4 v3)	VSS H.264 New
Microsoft MPEG-4 v1/v2	Win32/QuickTime BeHereiVideo decoder

Tabelle 3: Auszug der 183 unterstützten Videocodecs<sup>38</sup>

Im Folgenden wird zunächst allgemein die Einbindung von nativen Methoden in Java erläutert, schließlich wird im Besonderen auf den MPlayer und MEncoder eingegangen.

### **Javas native Schnittstelle**

Zur Einbindung nativen Quelltextes in Java bestehen mehrere Möglichkeiten<sup>39</sup>. Entweder der Quelltext wird stückweise in Java Code übersetzt (vereinzelt auch automatisiert möglich<sup>40</sup>) oder er wird als Bibliothek in Java eingebunden. Aufgrund des Umfangs des MPlayer Projektes wurde schließlich die Möglichkeit der nativen Einbindung des Quelltextes als Bibliothek gewählt:

Die native Java Schnittstelle (JNI) ermöglicht durch folgenden Aufruf, Methoden einer anderen Programmiersprache (wie hier im Beispiel C bzw. C++) nativ, als Bibliothek, einzubinden<sup>41</sup>:

```
static {
    System.loadLibrary("encoder");
}
```

Schließlich wird die Methode der eingebundenen Bibliothek mit dem Schlüsselwort „native“ als native Methode deklariert. Der Java-Compiler kann anschließend eine Header-Datei für den C bzw. C++ Compiler erzeugen:

```
javah -jni -o encoder.h encoder
```

Dem C bzw. C++ Compiler muss letztlich noch mitgeteilt werden, dass er eine Bibliothek unter Zuhilfenahme des Java Native Interfaces erzeugen muss:

```
gcc -Wall -shared -I /usr/lib/j2sdk1.5-sun/include/ -I /usr/lib/j2sdk1.5-sun/include/linux/ encoder.c -o libencoder.so
```

Schließlich muss zum erfolgreichen Ausführen der Java Anwendung, die Systemvariable

<sup>38</sup> MPlayer (03.01.2006)

<sup>39</sup> vgl. Martin/Müller (2002)

<sup>40</sup> vgl. Martin (1996)

<sup>41</sup> Beispiele in Anlehnung an Liang (1999), Kapitel 1

„LD\_LIBRARY\_PATH“ auf den Pfad, in dem sich die native Bibliothek befindet, gesetzt werden.

## **MPlayer und MEncoder**

Das MPlayer-Projekt beinhaltet zwei Komponenten:

Den MPlayer, welcher das Abspielen einer Vielzahl von Video-Formaten ermöglicht und im Rahmen des Systementwicklungsprojekts dazu gedacht war, einzelne Bilder aus Videos zu extrahieren, um diese als Vorschaubild zu verwenden. Die zweite Komponente bildet der MEncoder, welcher das Encodieren, Konvertieren bzw. Transcodieren von diversen Video-Formaten unterstützt.

Der MPlayer kann zusammen mit dem MEncoder per CVS über

```
cvs -d:pserver:anonymous@mplayerhq.hu:/cvsroot/mplayer login cvs -z3 -d
:pserver:anonymous@mplayerhq.hu:/cvsroot/mplayer co -P main
```

bezogen werden.

Da zur Video- bzw. Audio-Codierung und zur Erstellung derer Container unter anderem auf die Bibliotheken 'libavcodec' und 'libavformat' des FFmpeg Projekts zurückgegriffen wird, müssen diese ebenfalls per CVS heruntergeladen werden:

```
cvs -d:pserver:anonymous@mplayerhq.hu:/cvsroot/ffmpeg login
cvs -z3 -d:pserver:anonymous@mplayerhq.hu:/cvsroot/ffmpeg co -P ffmpeg
```

Für die Unterstützung des 3GP Dateiformats muss zusätzlich der AMR Codec zur die Codierung der Audio-Spur vom FTP Bereich der 3GPP Website<sup>42</sup> bezogen werden.

Nach erfolgter Konfiguration des Compilers über './configure' aus den Stammverzeichnissen von MPlayer und FFmpeg, kann das Makefile mit dem 'make'-Befehl ausgeführt werden. Durch 'make install' wird schließlich der MPlayer sowie der MEncoder installiert.

Nach einer großen, im Umfang des Projekts begründeten, Einarbeitungszeit wurde die Hauptroutine beider Komponenten (MPlayer und MEncoder) um eine native Java-Schnittstelle erweitert<sup>43</sup>. Zudem wurde das Makefile<sup>44</sup> angepasst, sodass schließlich eine Linux Bibliothek erstellt wurde, die anschließend mit Hilfe eines so genannten Java-Wrapper bzw. einer Java-Hüllenklasse eingebunden werden konnte. Auf der Anhangs CD befindet sich im Verzeichnis 'MPlayer und MEncoder/Java Wrapper/' eine beispielhafte Implementierung. Der Aufruf der Hüllenklasse erfolgt analog zu den Parametern der Kommandozeile des MPlayers bzw. MEncoders (vgl. Tabelle 4).

42 3GPP: AMR Floating-point Speech Codec (06.01.2006) bzw. Anhangs CD Verzeichnis '01 AMR Codec'

43 Anhangs CD '05 MPlayer und MEncoder/mencoder.c' und '05 MPlayer und MEncoder/mplayer.c'

44 Anhangs CD '05 MPlayer und MEncoder/Makefile'

<i>Argument</i>	<i>Bedeutung</i>
Dateiname	Name der Video Datei, welche konvertiert werden soll
-ovc lavc	„output video option“: Videocodec mit dem das Zielvideo konvertiert werden soll.
-oac lavc	„output audio option“: Audiocodec mit dem das Zielvideo konvertiert werden soll.
-srate	„sample rate“: Setzt die Abtastrate des Tones
-ofps	„output file frames per second“:
-lavcopts vcodec=h263:vbitrate=44:acodec=amr_nb:abitrage=8	„libav codec options“: Im Beispiel wird der Videocodec auf H.263 mit 44 Kbit als Videobitrate gesetzt. Der Audiocodec wird auf amr_nb mit 8 Kbit als Audiobitrate gesetzt.
-vf scale=176:144	„video frame scale“: Skaliert das Video auf die angegebene Höhe und Breite.
-o Dateiname	„output“: Spezifiziert den Dateinamen der Ausgabe.

Tabelle 4: Parameter des MEncoders<sup>45</sup>

Die Wahl dieser Möglichkeit der Video-Transcodierung brachte einige Schwierigkeiten mit sich:

Die direkte Einbindung der Hüllenklasse als Java Server Page (JSP) brachte den Tomcat Server nach erfolgreicher Encodierung des Videos mehrmals zum Absturz, da die Java-Bibliothek nach Durchlauf der Routine einen Speicherzugriffsfehler<sup>46</sup> auslöste.

Um dieses Problem zu umgehen wurde die Encodier-Routine als externe Java-Anwendung implementiert, welche separat zum Tomcat Server gestartet werden muss. In dieser Variante wird regelmäßig ein Verzeichnis nach zu bearbeitenden Video-Dateien geprüft. Sobald neue Transcodier-Aufträge vorhanden sind, wird die Java-Anwendung gestartet.

Da die durch die Java-Anwendung erzeugten Video-Dateien<sup>47</sup> lediglich auf der Linux-Plattform, aber weder auf dem Test-Mobiltelefon<sup>48</sup> noch auf dem Standard-Player von Windows abspielbar waren, und zudem vereinzelt Speicherzugriffsfehler auftraten, mussten weitere Methoden des Transcodierens evaluiert werden. Da das MPlayer Projekt auf Bibliotheken aus dem FFmpeg Projekt setzt, verlagerte sich der Fokus der Evaluation auf jenes.

<sup>45</sup> Gentoo Wiki (06.01.2006)

<sup>46</sup> Vgl. Anhangs CD '05 MPlayer und MEncoder/Java Wrapper/WEB-INF/classes/hs\_err\_pid19521.log'

<sup>47</sup> Vgl. Anhangs CD '05 MPlayer und MEncoder/Java Wrapper/content/encoded/fire.3gp'

<sup>48</sup> Sony Ericsson K750i

Durch das Internet wurden zwei Projekte gefunden, die eine Hüllklasse für FFmpeg zur Verfügung stellen, welche im Weiteren vorgestellt werden.

### **JFFmpeg<sup>49</sup>**

JFFmpeg integriert sich als Plugin in das Java Media Framework<sup>50</sup>. Jenes bietet die Möglichkeit Audio- und Video-Inhalte direkt in Java Anwendungen einzubinden, wobei es neben der Wiedergabefunktion auch die Möglichkeit Inhalte zu transcodieren zu Verfügung stellt. In der aktuellsten Version 2.1.1 bietet das JMF die Möglichkeit H.263 als Videocodec zu transcodieren, jedoch wird der AMR Audiocodec nicht unterstützt<sup>51</sup>.

JFFmpeg versucht Lücken in der Liste der unterstützten Codecs zu schließen. Dabei wird eine Java Hülle um das FFmpeg Projekt gelegt wodurch JFFmpeg als Plugin in das JMF integriert werden kann.

Für die erfolgreiche Verwendung von JFFmpeg muss mindestens das J2SE 1.4.2, JMF sowie die ffmpeg-0.4.7 Bibliothek auf dem Zielsystem installiert sein. Der Quelltext für JFFmpeg kann direkt von der JFFmpeg Website bzw. von der Anhangs CD<sup>52</sup> bezogen werden.

Zuerst sollte die Datei 'config.mak' an das Zielsystem angepasst werden. Anschließend kann JFFmpeg compiliert und installiert werden. Die Einbindung in das Java Media Framework ist mittels JMFRegistry des JMF möglich und der Website von JFFmpeg<sup>53</sup> zu entnehmen.

Dieses Projekt bietet zwar die Möglichkeit AMR Audio zu decodieren, das Encodieren und somit das Transcodieren ist jedoch nicht möglich. Aufgrund mangelnder Dokumentation und unregelmäßiger Arbeit am Projekt seitens der Entwickler, ist es im Rahmen dieses Systementwicklungsprojekts nicht möglich einen AMR Encoder für JFFmpeg zu implementieren. Aus diesem Grund scheidet auch diese Variante für die Mozean Plattform aus.

### **FOBS<sup>54</sup>**

Nachdem JFFmpeg als Möglichkeit ausgeschieden war, wurde das zweite Projekt mit Java Hülle um die FFmpeg Anwendung namens FOBS begutachtet.

Die Abkürzung FOBS bezeichnet „FFmpeg ObjectS“, welche auf die FFmpeg Bibliothek zurückgreifen um Programmierern eine einfache Schnittstelle zur Handhabung von Video-

---

49 JFFmpeg (03.01.2006)

50 Sun (06.01.2006)

51 Vgl. Sun: JMF 2.1.1 - Supported Formats (06.01.2006)

52 Anhangs CD Verzeichnis '04 JFFmpeg'

53 JFFmpeg: JFFmpeg - Java Audio and Video Codecs for JMF (06.01.2006)

54 FOBS (03.01.2006)

und Audio-Inhalten zur Verfügung zu stellen<sup>55</sup>. Das Projekt baute ursprünglich auf C++ auf und ist inzwischen auch für Java erhältlich. Dabei gliedert es sich in ähnlicher Weise wie JFFmpeg als Plugin in das Java Media Framework des Zielsystems ein.

Die aktuellste Version kann mittels CVS durch

```
cvs -d :pserver:anonymous@cvs.sf.net:/cvsroot/fobs login
cvs -d :pserver:anonymous@cvs.sf.net:/cvsroot/fobs co fobs-src
```

bezogen werden.

Anschließend kann der Quelltext mit Hilfe von './buildFobs.sh' kompiliert werden<sup>56</sup>.

In der aktuellen Version besteht die Möglichkeit Audio in AMR Codierung durch das Java Media Framework abzuspielen. Leider ist keine Schnittstelle zum AMR Encoder des FFmpeg Projekts verfügbar<sup>57</sup>.

Da keiner der verschiedenen Ansätze der nativen Implementierung den gewünschten Erfolg erzielte, und FFmpeg als Open Source Projekt sich sehr stark im Bereich des Video-Encodierens etabliert hat, wurde schließlich direkt auf das FFmpeg Projekt zurückgegriffen.

## FFmpeg<sup>58</sup>

Die letztlich umgesetzte Lösung bindet FFmpeg als externen Prozess in Java ein. Dabei wurden mehrere Schnittstellen implementiert auf die später näher eingegangen wird.

FFmpeg bietet die Möglichkeit digitales Audio- und Videomaterial aufzunehmen, zu transcodieren und zu streamen. Wie bereits angesprochen, stellt FFmpeg die Bibliotheken 'libavcodec' und 'libavformat' bereit, welche eine Vielzahl von Codecs und Containerformate unterstützen:

<i>Videocodecs</i>	<i>Audiocodecs</i>
MPEG-1 video	MPEG audio layer 2
MPEG-2 video	MPEG audio layer 1/3
MPEG-4	AC3
H.263(+)	AAC (Supported through the external library libfaac/libfaad)
RealVideo 1.0 + 2.0	AMR-NB
FLV	AMR-WB

Tabelle 5: Auszug der von FFmpeg unterstützten 38 Audio und 63 Videocodecs<sup>59</sup>

<sup>55</sup> FOBS (03.01.2006)

<sup>56</sup> Vgl. Anhangs CD '02 FOBS/README'

<sup>57</sup> Vgl. Anhangs CD '02 FOBS/src/jmf-pi/com/omnividea/media/codecs/audio'

<sup>58</sup> FFmpeg (03.01.2006)

<sup>59</sup> FFmpeg: Documentation (06.01.2006)

Wie Tabelle 5 zu entnehmen ist, wird durch die Unterstützung des H.263 Videocodecs sowie des AMR Audiocodecs das von 3GPP standardisierte Containerformat 3GP unterstützt. Dabei stellt FFmpeg sowohl Möglichkeiten des Decodierens als auch des Encodierens bereit. Somit ist eine Transcodierung von Video- und Audio-Inhalten möglich.

### **Installation von FFmpeg**

Bevor mit der Installation begonnen werden kann muss die 'lame-devel' Bibliothek installiert werden. Diese ist als Quelltext unter Sourceforge.net<sup>60</sup> verfügbar und ermöglicht FFmpeg MP3 Audio zu encodieren, was für die Audio-Spur des Flash Videos Formats benötigt wird, um Vorschauvideos zu erstellen. Diese Flash Videos werden später dazu verwendet um Vorschaufilme aus dem Videomaterial zu generieren.

Anschließend kann FFmpeg mit folgendem Kommando per CVS bezogen werden:

```
cvcs -z9 -d:pserver:anonymous@mplayerhq.hu:/cvsroot/ffmpeg co ffmpeg
```

Nach erfolgreichem Herunterladen der Quelltext-Dateien muss durch den Befehl

```
./configure --enable-amr_nb --enable-mp3lame
```

das Zielsystem konfiguriert werden. Die Optionen 'enable-amr\_nb' und 'enable-mp3lame' ermöglichen die Unterstützung von MP3 und AMR Audio-Encodierung.

Schließlich muss vor der Compilation der AMR Codec heruntergeladen werden. Dieser kann von der 3GPP Website bezogen werden<sup>61</sup>. Anschließend muss der Codec entpackt und in das Verzeichnis 'libavcodec/amr\_float' des Ffmpeg Stammverzeichnis kopiert werden.

Letztlich kann FFmpeg über das 'make' Kommando compiliert werden. Nach der Compilation kann daraufhin die Anwendung über 'make install' installiert werden<sup>62</sup>.

Nachdem FFmpeg erfolgreich installiert wurde, kann der Prozess von Java gesteuert werden. Dazu wird die ab Java Version 5 vorhandene 'ProcessBuilder' Klasse aus dem 'java.lang' Paket verwendet<sup>63</sup>.

### **Java Process Builder**

Der ProcessBuilder von Java Version 5 stellt eine Plattform unabhängige Möglichkeit dar, Betriebssystem-Prozesse zu starten<sup>64</sup>. Zudem können mittels ProcessBuilder native Methoden vermieden werden, indem externe Programme, wie beispielsweise FFmpeg, eingebunden werden können, die nicht auf Java Quelltext basieren. Dabei sind einzelne Prozesse durch das

<sup>60</sup> Sourceforge.net (06.01.2006)

<sup>61</sup> 3GPP (06.01.2006): 3GPP AMR Floating-point Speech Codec

<sup>62</sup> Vgl. Verzeichnis '02 FFmpeg' der Anhangs CD

<sup>63</sup> Sun (08.01.2006)

<sup>64</sup> Esser (2004), S 188ff.

Betriebssystem voneinander geschützt, wodurch Speicherzugriffsfehler, wie sie bei den zuvor erwähnten Encodier-Methoden auftraten, nur den aufgerufenen Prozess, nicht jedoch die restliche Umgebung betreffen. Der Rückgriff auf die 'ProcessBuilder' Klasse, welche externe Programme aufruft, kann wie im implementieren Falle zum Verlust der Plattformunabhängigkeit führen. Jedoch unterstützt FFmpeg eine Vielzahl von Betriebssystemen (Linux, BSD, Windows, Mac OS X, BeOS<sup>65</sup>); zur Migration dieses Encodier-Verfahrens auf eine andere Plattform muss lediglich FFmpeg für die entsprechende Plattform kompiliert werden.

Folgendes Klassendiagramm (Abbildung 5) stellt die einzelnen von der ProcessBuilder-Klasse zur Verfügung gestellten Methoden dar:

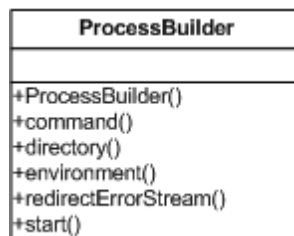


Abbildung 5: Die ProcessBuilder Klasse von Java Version 5

Um externe Programme mittels Java-Code auszuführen muss zuerst eine Instanz der ProcessBuilder-Klasse erzeugt werden:

```
ProcessBuilder thumbnailBuilder = new ProcessBuilder("ffmpeg", "-i",
"eiffelturm.mpeg", "-y", "-f", "image2", "-t", "0.0001", "-s", "sqcif",
"eiffelthumb.jpeg");
```

Dabei wird dem Konstruktor der auszuführende Befehl (im Beispiel 'ffmpeg') mit gewünschten Parametern übergeben. Zudem kann das Arbeitsverzeichnis über folgende Methode gesetzt werden:

```
thumbnailBuilder.directory( new File („pathToVideoFile“) );
```

Schließlich wird der Prozess über die 'start()' Methode der ProcessBuilder-Klasse aufgerufen, welche ein Objekt der Klasse Process zurückgibt:

```
Process thumbnailProcess = thumbnailBuilder.start();
```

Bei Eintritt eines Fehlers wird eine IOException geworfen, die zuvor mittels 'try/catch' Block abgefangen werden muss.

Das Prozess-Objekt beschreibt den Betriebssystemprozess und stellt mehrere Methoden zur Bearbeitung, wie beispielsweise Zugriff auf den Ausgabestrom, bereit (siehe Abbildung 6).

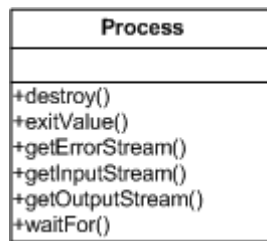


Abbildung 6: Die Process Klasse

Die 'getInputStream' Methode des Prozesses ermöglicht es den Ausgabestrom umzuleiten. Falls – wie im Falle von FFmpeg – die Anwendung die Ausgabe auf dem Fehlerstrom umleitet bzw. aus anderen Gründen der Fehlerstrom gelesen werden muss, so kann auf die 'getErrorStream' Methode zurückgegriffen werden.

Da FFmpeg zum einen für verschiedene Plattformen verfügbar ist und zudem die ProcessBuilder Klasse eine sichere und stabile Möglichkeit bietet, externe Programme in Java einzubinden, gewährleistet der gewählte Weg der Encodierung ein hohes Maß an Sicherheit und Stabilität.

## Plattformintegration

Die letzte Phase dieses Systementwicklungsprojektes bildet die Integration in die Mozean Plattform. Um den Aufruf externer Prozesse zu ermöglichen und eine sichere und stabile Einbindung nativer Prozesse durch die ProcessBuilder-Klasse zu gewährleisten, musste zuerst die Plattform von Java Version 1.4 in die aktuellste Version 5 überführt werden.

Anschließend wurden Schnittstellen zum FFmpeg Encoder geschaffen, um diesen bequem ansprechen zu können.

Die Möglichkeit des Video-Handels über die Plattform (siehe Abbildung 8) ist eine Weiterentwicklung der Funktion Fotos über Mozean zu tauschen. Daher entstanden dort ähnliche Anforderung an die Visualisierung. Von Fotos, die auf die Plattform zum Weiterversenden geladen werden, konnten bereits Vorschaubilder erzeugt werden. Daher war diese Funktion auch für den Video-Tausch nötig. Deshalb wurde schließlich eine Klasse geschrieben, welche aus einem gegebenen Video das erste Bild extrahiert:

```
createThumb(String videoFile, String pathToVideoFile, String
pathToThumbnail)66
```

Anschließend hinterlegt die Mozean Plattform das Bild mit einer Verknüpfung auf das encodierte Video.

<sup>66</sup> Die folgenden Passagen beziehen sich auf den Quelltext der Anhangs CD Verzeichnis '06 Mozean Plattform'

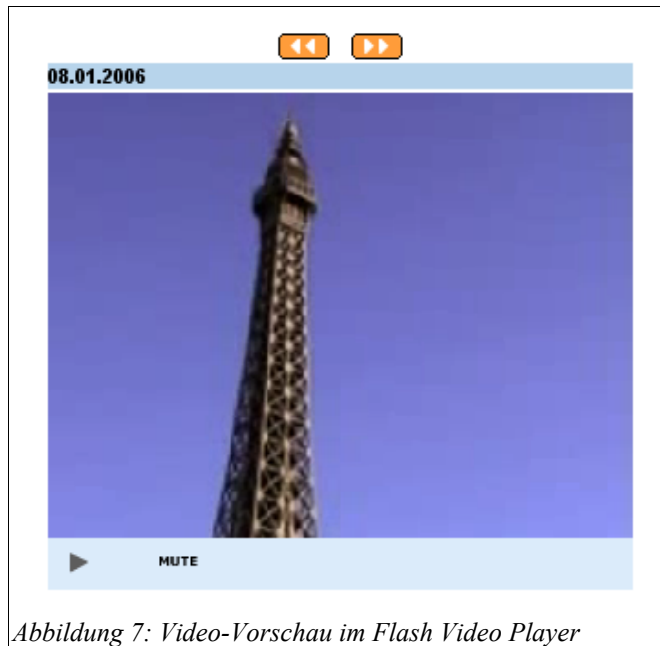


Abbildung 7: Video-Vorschau im Flash Video Player

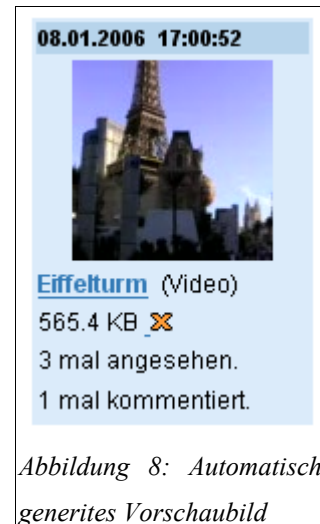


Abbildung 8: Automatisch generiertes Vorschaubild

Dabei ruft die 'createThumb' Methode den FFmpeg Encoder mit folgenden Argumenten auf:

<i>Argument</i>	<i>Bedeutung</i>
-i videoFile	Spezifiziert die Position und den Namen der Eingabedatei (-i: input)
-y	Überschreibt die Ausgabedatei (falls vorhanden)
-f image2	Erzwingt die Bilderzeugung als Ausgabeformat
-t 0.0001	Bestimmt die Dauer des Ausgabevideos
-s qcif	Setzt die Auflösung auf QCIF
thumbnailFile	Spezifiziert die Position und den Namen der Ausgabedatei

Tabelle 6: FFmpeg Parameter zur Erzeugung von Vorschaubildern

Zudem wurde eine Vorschau des Videos gewünscht (siehe Abbildung 7). Da aktuelle Browser keine Möglichkeit bieten, 3GP Inhalte darzustellen, wurde entschlossen für diesen Zweck das Flash Video Format (FLV) zu verwenden. Dabei handelt es sich beim FLV Dateiformat – wie auch beim 3GP Format – um ein Containerformat. FLV benutzt den H.263 Codec zur Video-Encodierung mit kleinen Veränderungen<sup>67</sup>. Dabei kann die Audio-Spur sowohl unkomprimiert als auch komprimiert codiert werden, wobei fünf verschiedene Codecs, unter denen sich auch der MP3 Codec befindet, zum Einsatz kommen können<sup>68</sup>. Aus diesem Grunde ist die Installation der LAME MP3 Bibliothek auf dem Zielsystem nötig.

<sup>67</sup> Macromedia (2002), S. 165

<sup>68</sup> Macromedia (2002), S. 176

Der Aufruf des FFmpeg Encoders zur Codierung, welcher in Tabelle 7 abgebildet ist, erfolgt in Java durch die 'mozean.com.video.Encoder' Klasse:

```
encodeFlv (String videoFile, String pathToVideoFile)
```

Dabei ist anzumerken, dass die Audio-Abtastrate auf 44 Kilohertz gesetzt wurde, um die bestmögliche Audio-Qualität, welche das FLV Format vorsieht<sup>69</sup>, für die Vorschau zu gewährleisten. Bei einer Abtastrate von 8000 Hz, wie sie für Mobiltelefone verwendet wird, traten bei der Encodierung in das FLV Format erhebliche Qualitäts-Verluste ein, weshalb diese angehoben wurde.

Als Auflösung wurde das CIF Format gewählt, welches zwar mehr Speicherplatz für das encodierte Video beansprucht, jedoch einen besseren Eindruck des Inhaltes vermittelt als QCIF oder SQCIF. Zudem ist sowohl der Mehrbedarf an Speicherplatz auf dem Mozean Server als auch die größere Transfermenge für den Anwender zu vernachlässigen, da sowohl Festplatten als auch Datentransfer über das Internet sehr kostengünstig sind.

<i>Argument</i>	<i>Bedeutung</i>
-acodec mp3	Verwendet als Audiocodec MP3
-ar 44100	Setzt die Sampling Frequenz
-ac 1	Setzt die Anzahl der Audiokanäle
-ab 32	Setzt die Audiobitrate
-s cif	Bestimmt die Auflösung
FlvFile.flv	Spezifiziert die Position und den Namen und das Format (FLV) der Ausgabedatei

Tabelle 7: FFmpeg Parameter zur Erzeugung von Flash Videos

Zur Transcodierung der Videos in das 3GP Format, für die Darstellung der Videos auf Mobiltelefonen, wurden zwei Methoden geschrieben:

```
encodeQcif_3gp (String videoFile, String pathToVideoFile) und  
encodeSqcif_3gp (String videoFile, String pathToVideoFile).
```

Diese beiden Methoden arbeiten ähnlich wie die 'encodeFlv' Methode zur Erzeugung von FLV Dateien, welche mit Hilfe der ProzessBuilder-Klasse eine Instanz von FFmpeg startet.

Aufgrund der Beschränkung des Video-Downloads aktueller Mobiltelefone auf unter einhundert Kilobyte kann es vorkommen, dass durch das Transcodieren das Videomaterial verkürzt wird. Da dies eine äußerst wichtige Information für den Benutzer darstellt, wurde eine Funktion geschrieben, welche die Differenz zwischen Ausgangs- und encodiertem Video berechnet:

```
getInfo (String videoFile, String pathToVideoFile)
```

<sup>69</sup> Macromedia (2002), S. 176

Die Ausgabe der Anwendung kann durch den Fehlerstrom von FFmpeg mit der `getErrorStream()` Methode analysiert werden. Mit Hilfe der Ausgabe von FFmpeg berechnet die `'getInfo'` Methode die Differenz der Dauer des Eingabe- und Ausgabe-Videos. Schließlich wurde eine Methode zum Initialisieren der Video-Encodierung geschrieben, welche als Wert ein Feld mit der Zeit, um die das QCIF Video verkürzt wurde und der Dauer auf die das SQCIF Video beschränkt wurde, zurück gibt:

```
long[] convertVideo(String videoFile, String pathToVideoFile)
```

Dabei greift die Encoder Klasse auf `'Parameters.java'` in `'mozean.utils'`<sup>70</sup> zurück. Dort können Einstellungen getätigt werden, welche zum einen die Qualität der encodierten Videos beeinflussen und zum anderen die gewählten Codecs beinhalten (vgl. Tabelle 8).

<i>Parameter</i>	<i>Wert</i>
VIDEO_POSTFIX_QCIF	_qcif
VIDEO_POSTFIX_SQCIF	_sqci
VIDEO_POSTFIX_FLV	_flv
VIDEO_POSTFIX_AMR	_amr
VIDEO_FLV_AUDIO_CODEC	MP3
VIDEO_FLV_AUDIO_FREQUENCY	44100
VIDEO_AMR_AUDIO_FREQUENCY	8000
VIDEO_AUDIO_BITRATE	7.4
VIDEO_FILE_SIZE	89
VIDEO_THUMBNAIL_SIZE	QCIF
VIDEO_BITRATE	44
VIDEO_FPS	10
AUDIO_FILE_SIZE	89

Tabelle 8: Encodierungsspezifische Parameter des Mozean Projekts

Der Encodierungs-Prozess wird durch den Video-Upload durch einen Anwender initiiert, welcher über die Web-Schnittstelle Inhalte auf den Server lädt. Die zugrunde liegende Systemarchitektur ist in Abbildung 9 dargestellt. Der Upload-Vorgang löst den Aufruf der `'convertVideo()'` Methode aus, wodurch der Encodier-Vorgang angestoßen wird. Dadurch wird die Erzeugung von Vorschaubild und -video veranlasst und das Videomaterial in die Zielformate transcodiert. Dabei werden die zuvor vom Benutzer getätigten Meta-Informationen, wie Beschreibung und Name, sowie der Speicherort im Dateisystem des Mozean Servers in die Datenbank geschrieben. Schließlich erhält der Anwender einen Link auf das Vorschaubild, welcher mit dem ersten Bild des Videos hinterlegt ist (vgl. Abbildung 8).

<sup>70</sup> Vgl. Anhangs CD Verzeichnis '06 Mozean Plattform/utills'

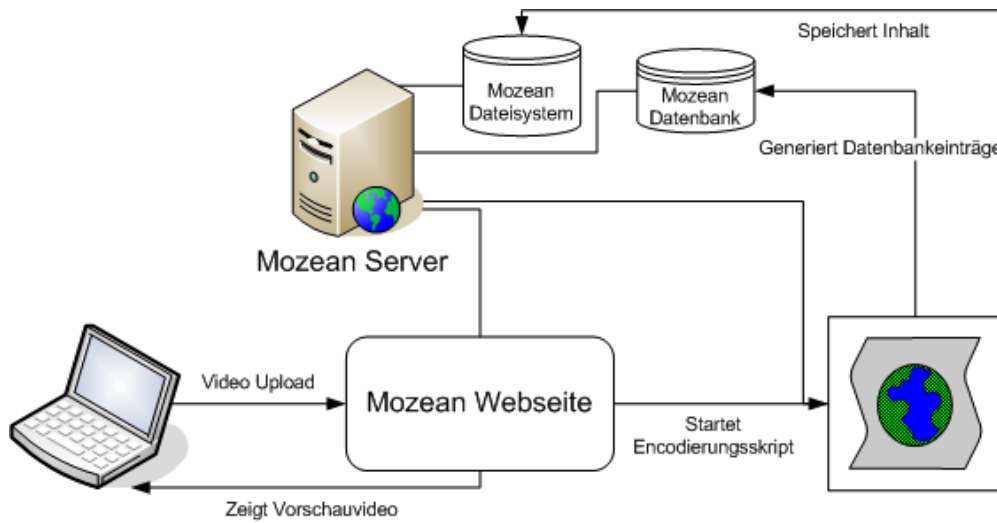


Abbildung 9: Einbindung der Encodierung in die Mozean Architektur

Der gesamte Prozess der Video-Encodierung ist Abbildung 10 zu entnehmen.

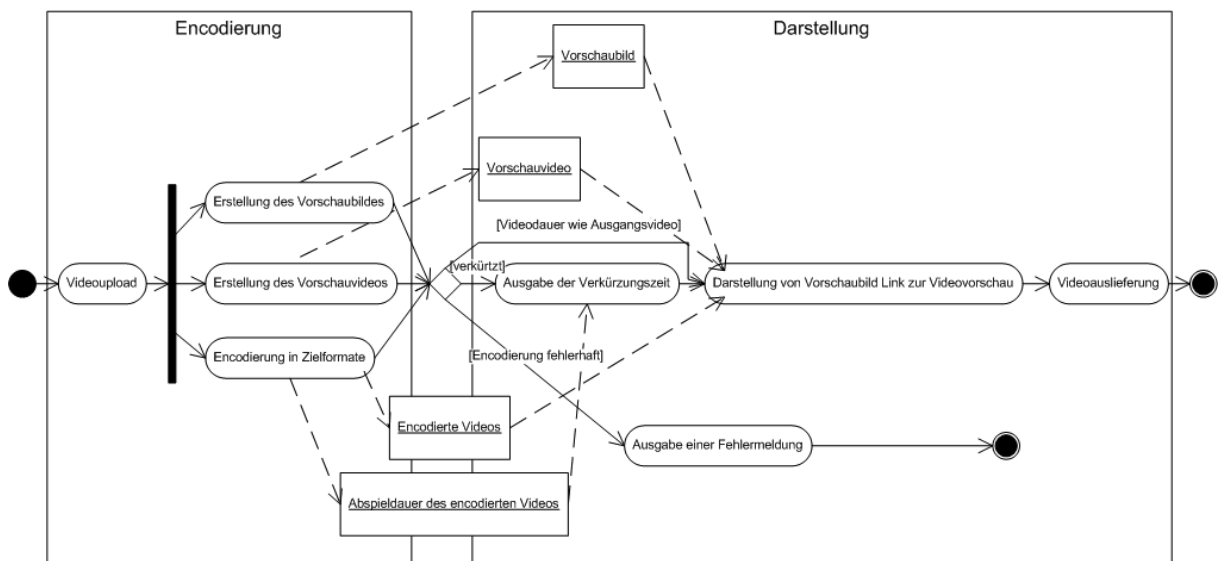


Abbildung 10: Darstellung des Encodierungs-Prozesses als UML Aktivitätendiagramm

Durch den Video-Upload wird die 'convertVideo()' Methode aufgerufen, welche zum einen die für die Vorschau nötigen Objekte erzeugt und zum anderen die Videos transcodiert. Verkürzungen der Abspieldauer der codierten Videos sowie Fehler in der Bearbeitung werden dem Benutzer mitgeteilt.

Am Ende dieses Prozesses kann der Benutzer das Video über die Mozean-Plattform an verschiedenste Mobiltelefone versenden.

## Erweiterungsmöglichkeiten

Mit wachsender Verbreitung von UMTS und sinkender Kosten für die Datenübertragung über das Mobiltelefon werden Downloads jenseits der einhundert Kilobyte in Zukunft technisch und finanziell keine Schwierigkeiten mehr bereiten. Zudem werden sich die Möglichkeiten der Video-Darstellung, welche heutzutage von der Beschränkung der Download-Größe reguliert werden, wandeln. Da dies bereits heute vorhersehbar ist, wurden Anpassungsmöglichkeiten sensibler Einstellungen in Form von Parametern in die Parameter-Klasse des Mozean Projektes verlagert.

Sobald JFFmpeg bzw. FOBS Encodierungs-Schnittstellen zur Verfügung stellen, ist eine native Integration denkbar. Langfristig kann in Erwägung gezogen werden von der nativen zur Plattform unabhängigen Encodierung überzugehen. Dies kann beispielsweise realisiert werden, indem der Encodierungs-Vorgang vom Java Media Framework übernommen wird. Sun ist heute bereits dabei Plattform unabhängige Codecs zu entwickeln, sodass es lediglich eine Frage der Zeit ist, bis der H.263 Videocodec sowie der AMR Audiocodec als Java Quelltext zur Verfügung stehen.

Derzeit ist zum sicheren und stabilen Betrieb, die in diesem Dokument skizzierte und von Mozean eingesetzte Lösung die beste, zumal sie die Möglichkeit der Parameter-Modifikation und somit flexible Anpassungsmöglichkeiten bietet.

## WEBSEITENVERZEICHNIS

- 3GPP (03.01.2006): About 3GPP, <http://www.3gpp.org/About/about.htm>.
- 3GPP (06.01.2006): 3GPP AMR Floating-point Speech Codec, [http://www.3gpp.org/ftp/Specs/latest/Rel-5/26\\_series/26104-540.zip](http://www.3gpp.org/ftp/Specs/latest/Rel-5/26_series/26104-540.zip).
- Apache (20.01.2006): Apache Tomcat, <http://tomcat.apache.org/>.
- FFmpeg (02.01.2006): FFmpeg, <http://ffmpeg.sourceforge.net/>.
- FFmpeg (03.01.2006): FFmpeg, <http://ffmpeg.sourceforge.net/>.
- FFmpeg (06.01.2006): FFmpeg Documentation, <http://ffmpeg.sourceforge.net/ffmpeg-doc.html#SEC19>
- FOBS (03.01.2006): Omnividea FOBS - FFMpeg C++ & JMF Bindings, <http://fobs.sourceforge.net/>.
- Fraunhofer (15.01.2006): Fraunhofer IIS - Audio & Multimedia - MPEG Audio Layer-3, <http://www.iis.fraunhofer.de/amm/techinf/layer3/>.
- Gentoo Wiki (06.01.2006): HOWTO Mencoder Introduction Guide, [http://gentoo-wiki.com/HOWTO\\_Mencoder\\_Introduction\\_Guide](http://gentoo-wiki.com/HOWTO_Mencoder_Introduction_Guide).
- GNU (06.01.2006): GNU Make - GNU Project - Free Software Foundation (FSF), <http://www.gnu.org/software/make/>.
- ITU (07.01.2006): Audiovisual and multimedia systems, <http://www.itu.int/rec/recommendation.asp?type=products&parent=T-REC-h>.
- Jaguar Moravia (07.01.2006): JAGUAR MORAVIA a.s. - prodej a servis vozů Jaguar a Land Rover, záruka, jeté a skladové vozy, [http://www.jaguar-moravia.cz/landrover/downloads.php?page\\_id=3](http://www.jaguar-moravia.cz/landrover/downloads.php?page_id=3).
- JFFmpeg (03.01.2006): Jffmpeg - Java Audio and Video Codecs for JMF, <http://jffmpeg.sourceforge.net/>.
- JFFmpeg (06.01.2006): Jffmpeg - Java Audio and Video Codecs for JMF, <http://jffmpeg.sourceforge.net/download.html>
- Mplayer (02.01.2006): Mplayer – The Movie Player, <http://www.mplayerhq.hu/>.
- MPlayer (03.01.2006): Codec Status Table - MPlayer - The Movie Player, <http://www.mplayerhq.hu/DOCS/codecs-status.html>.

- Nongnu (06.01.2006): CVS - Open Source Version Control, <http://www.nongnu.org/cvs/>.
- Sourceforge.net (06.01.2006), SourceForge.net: LAME (Lame Aint an MP3 Encoder), <http://sourceforge.net/projects/lame>.
- Sun (06.01.2006): Java Media Framework API (JMF), <http://java.sun.com/products/java-media/jmf/>.
- Sun (06.01.2006): JMF 2.1.1 - Supported Formats, <http://java.sun.com/products/java-media/jmf/2.1.1/formats.html>
- Sun (08.01.2006): Java 2 Platform SE 5.0, <http://java.sun.com/j2se/1.5.0/docs/api/index.html?java/lang/ProcessBuilder.html>.
- Sun (20.01.2006): JavaServerPages Technology, <http://java.sun.com/products/jsp/>.
- T-Mobile (15.01.2006): Daten | Optionen | Voreingestellt | GPRS | UMTS | Volumen | Data | Tarif | Handy | T-Mobile - - Daten, Optionen, Voreingestellt, GPRS, UMTS, Volumen, Data , Tarif, Handy , T-Mobile, [http://www.t-mobile.de/datenoptionen/1,9708,13507-\\_,00.html](http://www.t-mobile.de/datenoptionen/1,9708,13507-_,00.html).
- Wikipedia (06.01.2006): Abtastrate, [http://de.wikipedia.org/wiki/Sample\\_rate](http://de.wikipedia.org/wiki/Sample_rate).
- Wikipedia (06.01.2006): Codec, <http://de.wikipedia.org/wiki/Codec>.
- Wikipedia (06.01.2006): Containerformat, <http://de.wikipedia.org/wiki/Containerformat>.
- Wikipedia (06.01.2006): Encoder, <http://de.wikipedia.org/wiki/Encoder>.
- Wikipedia (06.01.2006): Header-Datei, <http://de.wikipedia.org/wiki/Header-Datei>.
- Wikipedia (06.01.2006): Transcodierung, <http://de.wikipedia.org/wiki/Transcodieren>.
- Wikipedia (15.01.2006): Global System for Mobile Communications, <http://de.wikipedia.org/wiki/GSM>.

## LITERATURVERZEICHNIS

- 3GPP (2005): Technical Specification Group Services and System Aspects  
Transparent end-to-end packet switched streaming service (PSS); 3GPP file format  
(3GP), 6. Aufl., Valbonne 2005.
- 3GPP(2004): Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; General  
description, 6. Aufl., Valbonne 2004.
- Dornbusch, P (2005): Herausforderungen bei der Distribution von mobilen Inhalten,  
München 2005.
- Esser, F. (2004): Das Tiger Release: Java 5 im Einsatz, Galileo Press GmbH, Bonn  
2004.
- ETSI (1999): Overview of 3GPP Release 99, Version xx/07/04, Sophia-Antipolis  
1999.
- Hands, D. (2004): A Basic Multimedia Quality Model, IEEE transactions on  
multimedia, 6. Aufl, Ipswich, 2004.
- ISO (2002): ISO/IEC JTC1/SC29/WG11 CODING OF MOVING PICTURES AND  
AUDIO, Chicago 2002
- Jindal, M / Prasad, R.S.V / Ramkishor, K. (2003): Fast video coding at low bit-rates  
for mobile devices, Emuzed India, Singapore 2003.
- Kampmann, M. (2004): Standardization of mobile multimedia services, Kbolenz,  
2004.
- Kang, L. / Leou, J. (2005): An error resilient coding scheme for H.263 video  
transmission based on data embedding, Taiwan, 2005.
- Liang, S. (1999): The Java™ Native Interface, 1. Aufl., Menlo Park 1999.
- Macromedia (2002): Macromedia Flash (SWF) File Format Specification, Version 7,  
2002.
- Martin, J. (1996): Ephedra: A C to Java Migration Environment, Northern Illinois  
1996.
- Martin, J. /Müller, H. (2002): C to Java Migration Experiences, Victoria 2002.
- Matta, J. / Pépin, C. / Lashkari, K. / Jain R. (2003): A Source and Channel Rate  
Adaptation Algorithmfor AMR in VoIP Using the Emodel, San Jose, 2003.

- Nokia (2003): Multimedia Content Adaption Using Nokia Multimedia Converter 2.0, Version 1.3, 2003.
- Real Networks (2002): Encoding Recommendations for Mobile Devices, Seattle, Version 2.02, 2002.
- The Internet Society (2004): 3rd Generation Partnership Project (3GPP) Multimedia files, 2004.

## **ANHANGS CD**

01 AMR Codec

02 FFmpeg

03 FOBS

04 JFFmpeg

05 MPlayer und MEncoder

06 Mozean Plattformen