ulm university universität ulm

Multimedia Design Pattern

Erweiterungen und Umsetzungen des „Pipes and Filters“-Patterns für Multimedia-Komponentensysteme

Philipp Merkel | 24.11.2008 |
Seminar Forschungstrends im Bereich Verteilte Systeme

Gliederung

1 | Multimedia-Komponentensysteme

Gliederung

1 | Multimedia-Komponentensysteme

2 | Anwendungsbeispiel

Gliederung

1 | Multimedia-Komponentensysteme

2 | Anwendungsbeispiel

3 | Pipes and Filters

Gliederung

Gliederung

Gliederung

Multimedia-Komponentensysteme

Komponentensysteme

- Aufbau von Software aus wiederverwendbaren Komponenten
- Ziel: Entwicklung und Wartung komplexer Anwendungen erleichtern

Multimediaverarbeitung

- Codierung, Decodierung, Übertragung und Weiterverarbeitung
- Mächtige, aber inkompatible Bibliotheken

Ziel: Multimedia-Komponentensysteme

- Aufbau aus leichtgewichtigen, wiederverwendbaren Komponenten
- Komponentenmarkt, verteilte Komponenten im Netzwerk

Gliederung

Anwendungsbeispiel

Voice-over-IP-Telefonsystem

- Komponenten für Rufaufbau, Rufzeichen, Übertragung, Codecs...
- Austausch von Komponenten: Rufzeichen ⟷ Rufzeichenmusik
- Erweiterungen: Mailbox, Übersetzungsdienst
- Übersetzungsdienst bei externem Dienstleister

Gliederung

Filter

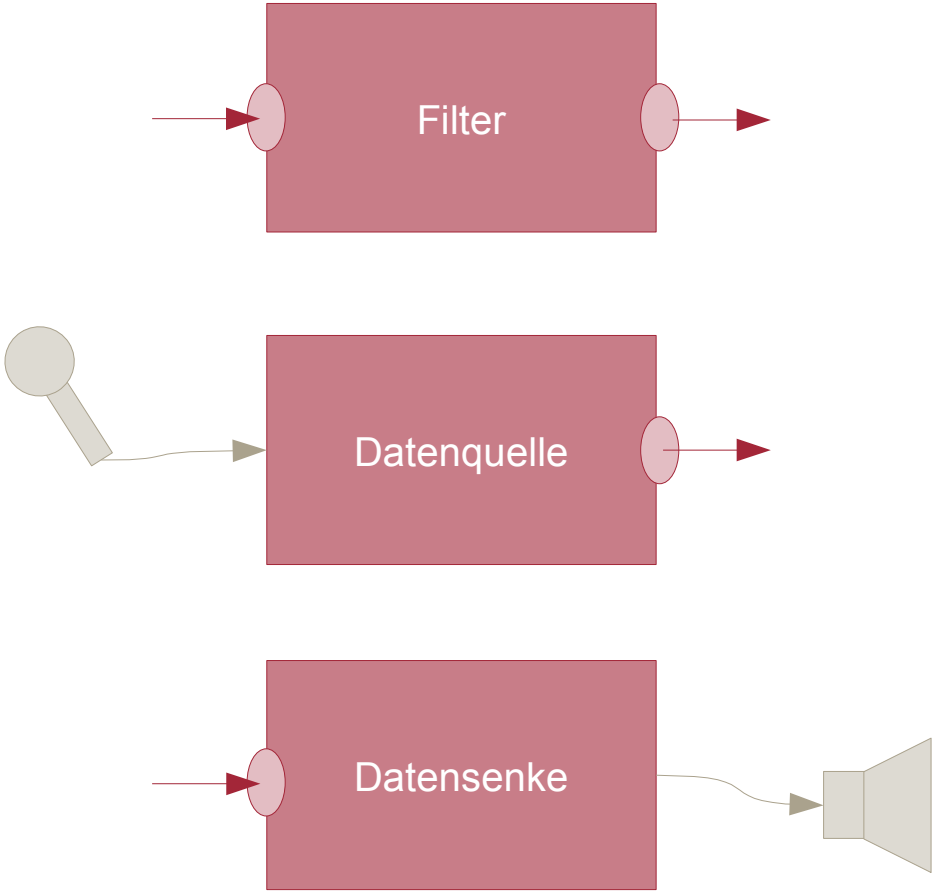
Konzeptionelle Softwarekomponente

Ein Eingang, ein Ausgang

Veränderung von Datenströmen

Datenquelle: Nur Ausgang

Datensenke: Nur Eingang



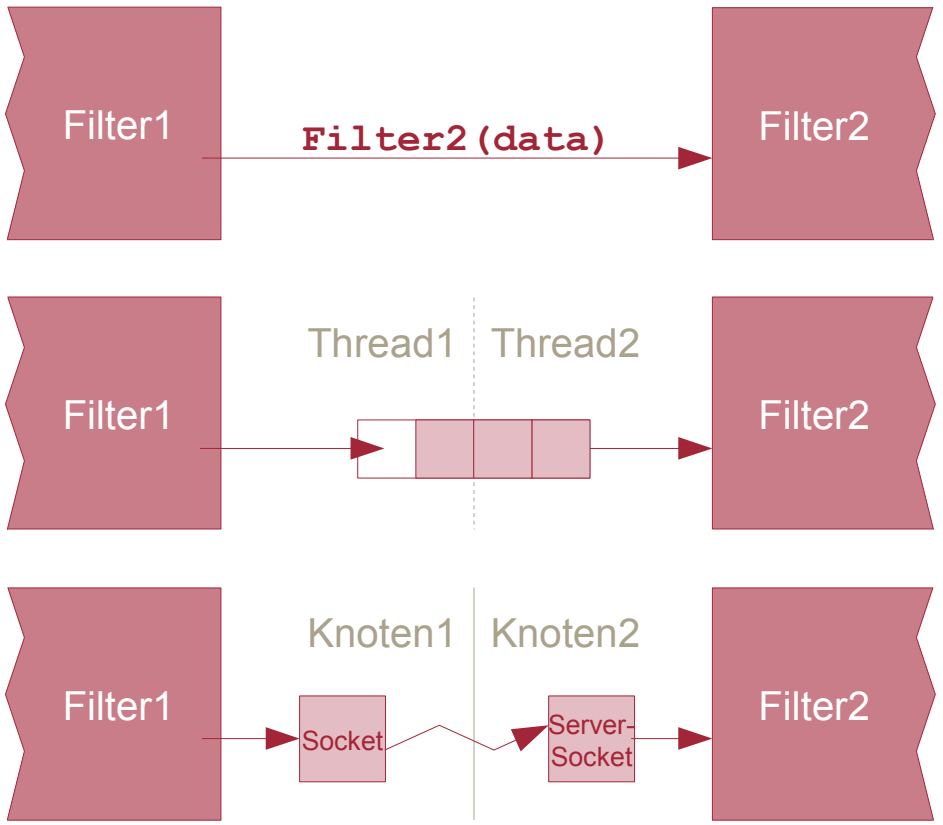
Pipe

Verbindung zwischen Filtern

Leitet Daten von Filter zu Filter

Realisierungsmöglichkeiten

- Methodenaufruf
- Puffer
- Über Netzwerk

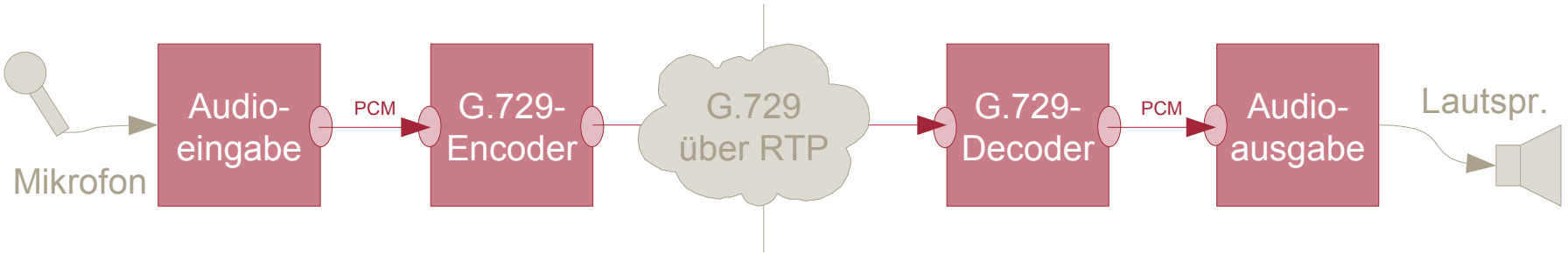


Pipeline

Inkrementelle Verarbeitung von Daten

Komplexe Verarbeitung durch Kombination kleiner Filter

→ Geeignet für den Multimediabereich



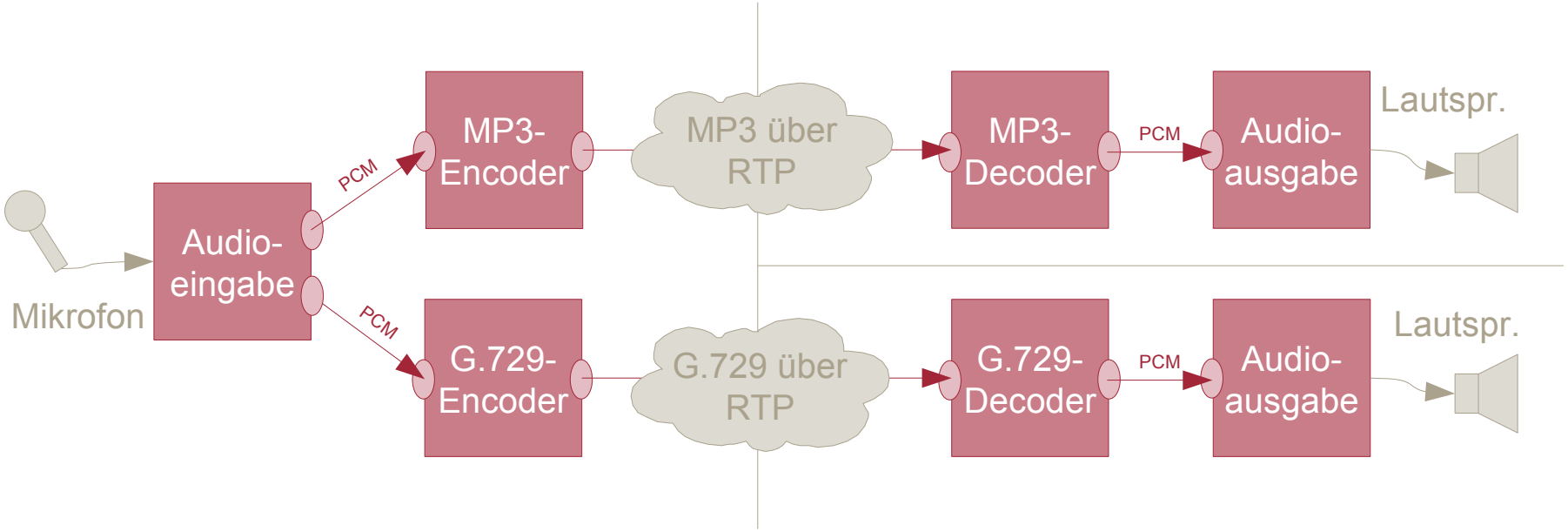
Erweiterung: Tee-and-Join-Pipeline-Systeme

Mehrere Ein- und Ausgänge pro Filter

Netze von Filtern und Pipes

Multiplexer, Demultiplexer

Übertragung zu verschiedenen Zielen



Gliederung

Verteilte Installation

Installation von Komponenten auf verschiedenen Rechnern

Dienste anderen Komponenten im Netzwerk bereitstellen

Verschiedene Ansätze:

- Als Plugins auf den Proxy-Servern (Wijnants et al.)
  - Einklinken in Datenströme auf Netzwerkprotokollebene
- MSODA: Eine virtuelle Maschine pro Komponente (Xu et al.)
  - Services beeinflussen sich nicht gegenseitig
- Komponenten als Webservices: Werden auf Webservern installiert (Wagner und Keller, Scholz et al.)

Kombination

Dynamische Kombination von Komponenten zur Erzeugung von Diensten

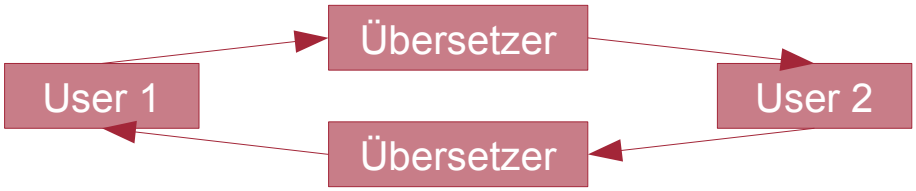
Verschiedene Ansätze:

- Manuelle Zusammenstellung im Programmcode (Gibbs, Posnak et al.)
  - Verbindung der Ein- und Ausgänge „von Hand“
  - Aber dynamisch zur Laufzeit neu kombinierbar (bei Posnak et al.)
- WS-AMUSE: Angabe als Zustandsautomat (Scholz et al.)
  - Automatische Umwandlung in BPEL
  - Ausführung durch BPEL-Engine

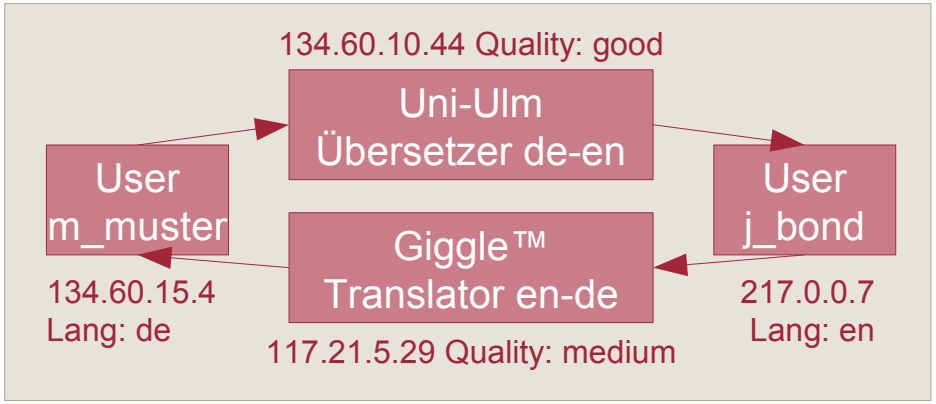
Kombination (2)

Weitere Ansätze:

- Abstrakte Beschreibung der gewünschten Funktionalität (Lohse et al., Xu und Jiang)



  - Generierung von möglichen Filtergraphen durch System und Auswahl des besten



Kombination (3)

Weitere Ansätze:

- DFC: Angabe von partieller Ordnung auf Filtern durch Administrator (Jackson u. Zave)
  - Nutzer kann verschiedene Komponenten für sich aktivieren
  - Reihenfolge durch partielle Ordnung eingeschränkt
- Semantische Beschreibung von Komponenten (Wagner und Kellerer)
  - OWL-S: Semantik, Funktionsweise und Schnittstellen
  - Automatische Auswahl anhand gewünschter Funktionalität

Kooperation

Signalisierung zwischen Komponenten

Bei lokalen Komponenten:

- Direkte Methodenaufrufe (Posnak et al., Gibbs)

Bei verteilten Komponenten:

- Entfernter Methodenaufruf über CORBA bei NMM (Lohse et al.)
  - Proxyobjekte

Kooperation (2)

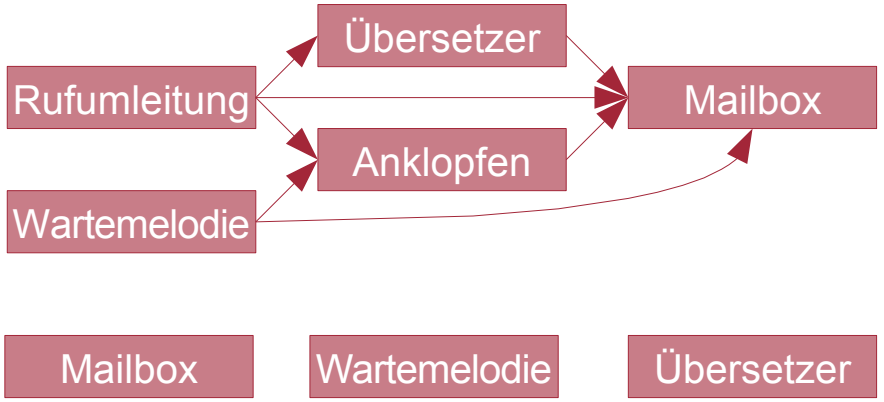
Bei verteilten Komponenten (2):

- WS-AMUSE: Web-Service-basiert (Scholz et al.)
  - Kommunikation über SOAP-Nachrichten (Request-Reply)
  - Nachrichten von Server an Client: Client-Anfrage auf Server halten
- MultiTEL: Konnektoren (Fuentes u. Troya)
  - Austauschbare Kommunikationsobjekte
  - Repräsentieren Übertragungsprotokolle
  - Fangen Ereignisse von Komponenten ab und übertragen sie

Kooperation (3)

Bei verteilten Komponenten (3):

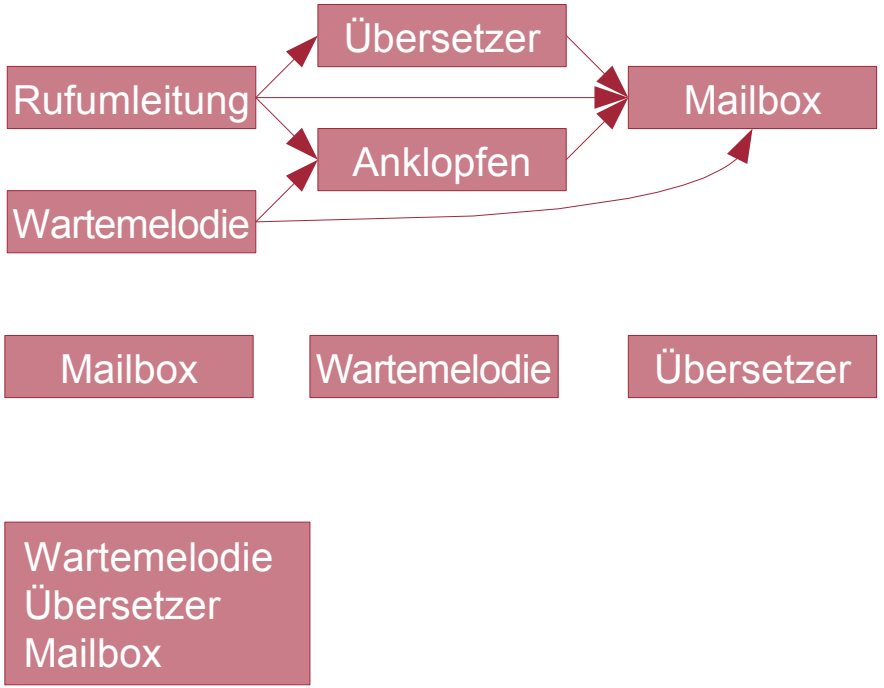
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features



Kooperation (3)

Bei verteilten Komponenten (3):

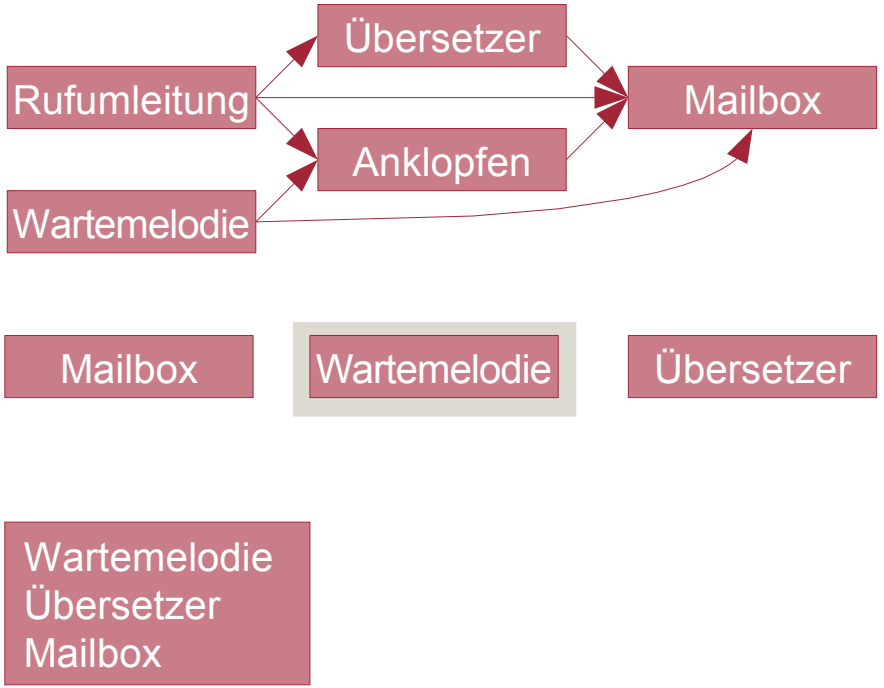
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste



Kooperation (3)

Bei verteilten Komponenten (3):

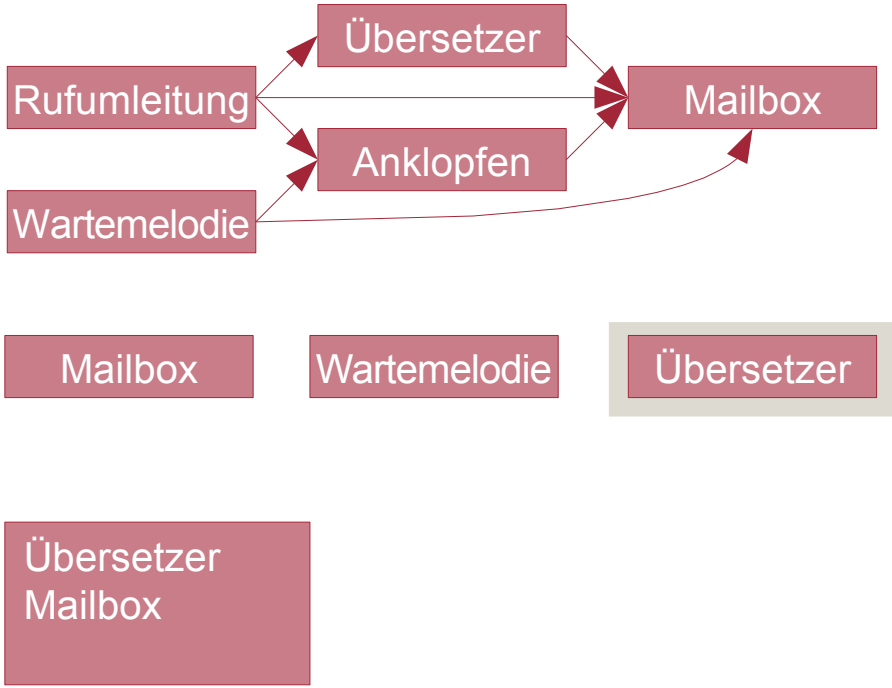
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste
  - Schrittweise Abarbeitung



Kooperation (3)

Bei verteilten Komponenten (3):

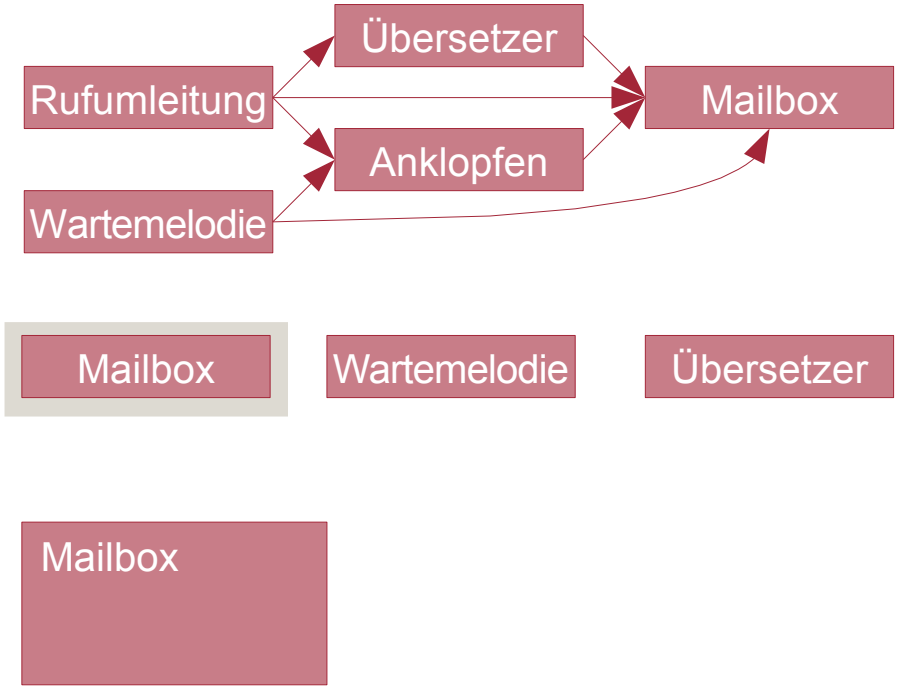
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste
  - Schrittweise Abarbeitung



Kooperation (3)

Bei verteilten Komponenten (3):

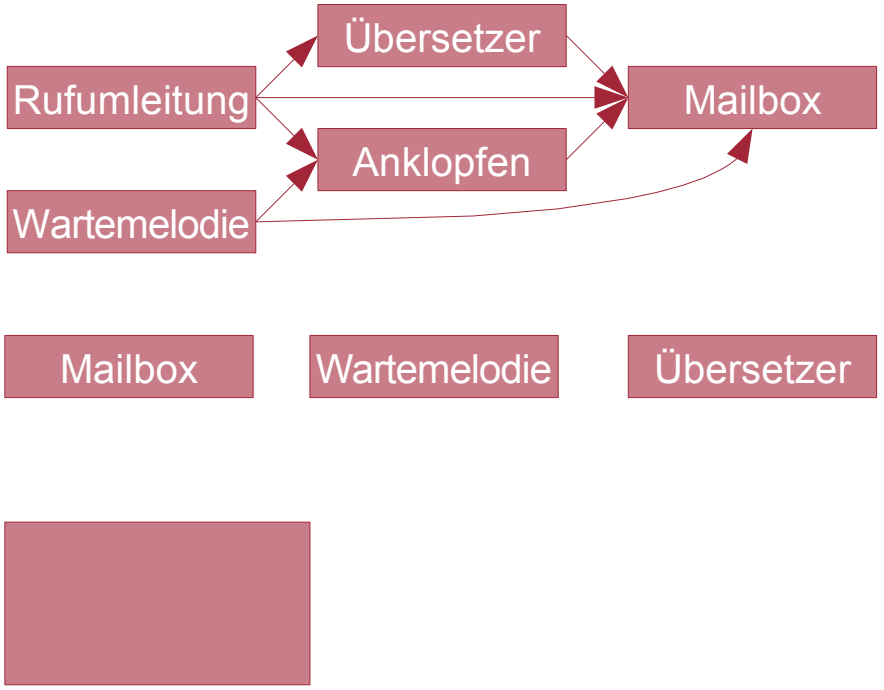
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste
  - Schrittweise Abarbeitung



Kooperation (3)

Bei verteilten Komponenten (3):

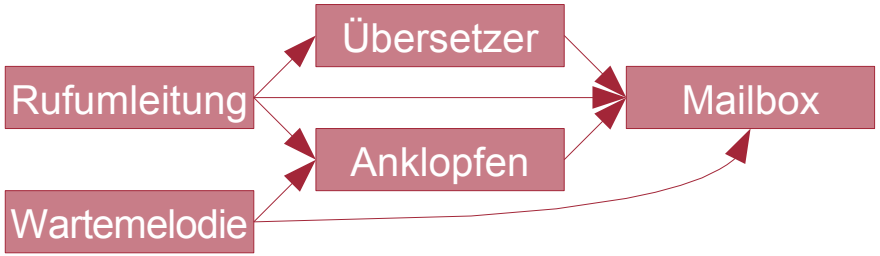
- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste
  - Schrittweise Abarbeitung



Kooperation (3)

Bei verteilten Komponenten (3):

- DFC: Routing (Jackson u. Zave)
  - Basis: Partielle Ordnung und gewählte Features
  - Erstellung einer Routingliste
  - Schrittweise Abarbeitung
  - Veränderung des Ziels → Neuberechnung



Datenfluss

Klassisch: Trennung von Signalisierung und Datenfluss

- DFC: Sprachverbindung zwischen Komponenten folgt Routing
  - Daten passieren Filter in Routing-Reihenfolge
- WS-AMUSE (Scholz et al.):
  - Datenübertragung klassisch über RTP

Datenfluss (2)

Kooperation und Datenübertragung auf selbem Weg:

- Auch Datenübertragung über Methodenaufrufe (Posnak et al.)
- NMM: Jacks zur Daten- und Signalübertragung (Lohse et al.)
  - Verbindung durch austauschbare Communication Channels
  - Buffer Messages für Daten und Composite Events für Signalisierung

Gliederung

Zusammenfassung und Ausblick

Dynamische Komponentenframeworks für Multimediasysteme

- Pipes and Filters
- Erweiterungen

Mögliche Erweiterungen

- Automatisches Deployment
- Sicherheitsaspekte
- Automatische Adaption an Netzwerkeigenschaften
  - Ständige Überwachung notwendig

Bildquellen

Titelbild

- „Pipes and gauges“ von „kqedquest“, freigegeben unter CC-by-nc-Lizenz. Quelle: http://www.flickr.com/photos/kqedquest/770678272/

Literatur

- [1] Microsoft Corporation: *DirectShow*. http://msdn.microsoft.com/en-us/library/ms783323.aspx – 12.11.2008.
- [2] Apple Inc.: *QuickTime*. http://www.apple.com/quicktime/ – 12.11.2008.
- [3] GStreamer Project: *GStreamer*. http://gstreamer.freedesktop.org/ – 12.11.2008.
- [4] Xine Project: *Xine*. http://xinehq.de/ – 12.11.2008.

Literatur (2)

- [5] Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad und Michael Stal: *Pattern-orientierte Software-Architektur*, Pipes-and-Filters, 54–71. Addison-Wesley, 2000.

- [6] Bengel, Günther: Grundkurs Verteilte Systeme: Grundlagen und Praxis des Client-server-computing, Kapitel 3.2, Seiten 137–145. Vieweg+Teubner Verlag, 2004.

- [7] Comer, Douglas: Internetworking with TCP/IP: Principles, Protocols and Architecture, Kapitel 21, Seiten 373–401. Prentice Hall, 2006.

- [8] Posnak, Edward J., R. Greg Lavender und Harrick M. Vin: *Adaptive Pipeline - an Object Structural Pattern for Adaptive Applications*. *Proceedings of the Third Pattern Languages of Programming Conference*, Monticello, Ill., Sept 1996.

Literatur (3)

- [9] Wijnants, Maarten, Bart Cornelissen, Wim Lamotte und Bart De Vleeschauwer: *An overlay network providing application-aware multimedia services*. *AAA-IDEA '06: Proceedings of the 2nd international workshop on Advanced architectures and algorithms for internet delivery and applications*, New York, NY, USA, 2006. ACM.

- [10] Xu, Dongyan und Xuxian Jiang: *Towards an integrated multimedia service hosting overlay*. *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, 96–103, New York, NY, USA, 2004. ACM.

- [11] Wagner, Matthias und Wolfgang Kellerer: *Web services selection for distributed composition of multimedia content*. *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, 104–107, New York, NY, USA, 2004. ACM.

Literatur (4)

- [12] Scholz, Andreas, Christian Buckl, Alfons Kemper, Alois Knoll, Jörg Heuer und Martin Winter: *WS-AMUSE - web service architecture for multimedia services*. *ICSE '08: Proceedings of the 30th international conference on Software engineering*, 703–712, New York, NY, USA, 2008. ACM.

- [13] Lohse, Marco, Michael Repplinger und Philipp Slusallek: *An Open Middleware Architecture for Network-Integrated Multimedia*. *Protocols and Systems for Interactive Distributed Multimedia Systems, Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems, IDMS/PROMS 2002, Proceedings*, 2515 *Lecture Notes in Computer Science*, 327–338. Springer, 2002.

- [14] Jackson, M. und P. Zave: *Distributed feature composition: a virtual architecture for telecommunications services*. IEEE Transactions on Software Engineering, 24(10):831–847, Oct 1998.

Literatur (5)

- [15] Cheung, E. und K.H. Purdy: *An Application Router for SIP Servlet Application Composition*. *IEEE International Conference on Communications, ICC '08*, 1802–1806, May 2008.

- [16] Gibbs, Simon: *Object-Oriented Software Composition*, Multimedia component frameworks, 305–319. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995.

- [17] Fuentes, Lidia und Josée M. Troya: *Towards an open multimedia service framework*. ACM Comput. Surv., 32(24):24–29, March 2000.