**Ergebnisprotokoll vom Teammeeting am 08.05.2015**

**Anwesend:** Arno Däuper, Niklas Kröger, Inga Quatuor, Sebastian Schwanewilms, Jan Zimmer

**Protokollantin:** Inga Quatuor

**Thema:** Spezifikation des Projekts im Kurs *Embedded Multimedia*

**Inhalt:**

1. Das Projekt
2. Grundsätzliche Struktur
3. Details
   1. Netzwerk
   2. Output
   3. Präsentation
   4. Phasen
   5. Messages
   6. Sonstiges
4. Aufgabenverteilung / Arbeitspakete
5. Meilensteinplanung
6. Dokumentation
7. Lizenz

**1. Das Projekt**

Unser Projekt entspricht im Grundsatz dem Default-Projekt aus dem Kurs *Embedded Multimedia.* Als Ergänzung sollen zusätzlich Audioaufnahmen des Vortrags mitgeschnitten werden.

Arbeitstitel des Projekts ist PräsiBert.

**2. Grundsätzliche Struktur**

Unser Softwaresystem soll aus den folgenden unabhängigen Komponenten bestehen:

* Server: Die Server-Applikation verwaltet die angemeldeten Clienten, die Präsentation und die Datenhaltung.
* MasterClient: Der MasterClient lädt eine Präsentation beim Server hoch und kann diese Präsentation dann steuern (Folie vor / Folie zurück) und beenden. Der MasterClient schneidet während des gesamten Vortrags Audio mit. Es kann immer nur einen MasterClient geben.
* ListenerClient: Die ListenerClients bekommen die Präsentation während des Vortrags übertragen und können Redezeit beantragen. Wird die Redezeit vom MasterClient genehmigt, schneidet der ListenerClient Audio mit und versendet es anschließend an den Server.
* OutputClient: Jeder Client verfügt über die Funktionalität zum OutputClient zu werden. Der OutputClient ist mit dem Ausgabemedium (z. B. Beamer) verbunden, bekommt vom Server die zu übertragenden Bilder gesendet. Es kann mehrere OutputClients geben.

Die Komponenten können entweder auf demselben Device laufen oder verteilt. Einschränkung: Auf demselben Device kann nicht gleichzeitig ein MasterClient und ein ListenerClient laufen.

Begründung

Durch die funktionale Trennung nach Zuständigkeit hat unser System folgende Vorteile:

* Es kann wechselnde Vortragende geben, ohne dass alles neu initialisiert werden muss. Es muss sich lediglich der alte MasterClient abmelden und ein neuer anmelden.
* Unser System ist flexibel in Bezug auf die Art und Anzahl der Ausgabemedien.
* Eine spätere Portierung ist besser möglich (aber im Rahmen des Projektes nicht vorgesehen).
* Ausfallsicherheit durch Entkoppelung: Solange der Server nicht abstürzt, läuft das System.

**3. Details**

**3.1 Netzwerkaufbau**

* Voraussetzung für den Aufbau des Netzwerks ist, dass der Server läuft und seine IP-Adresse bekannt ist.
* ListenerClients können sich jederzeit an- oder abmelden
* Ein MasterClient kann sich nur dann anmelden, wenn noch kein MasterClient beim Server registriert ist.
* Netzwerkprotokoll: TCP

**3.2 Output**

* Als Testaufbau wird zunächst betrachtet: Ein Device ist per HDMI-Kabel an den Beamer angeschlossen. Der Client auf diesem Device kann dann vom MasterClient in der Setup-Phase vor Vortragsbeginn als OutputClient bestimmt werden.
* Funktion des OutputClient: Empfängt die aktuelle Ausgabe vom Server und sendet diese an das Output Device weiter.
* Optional: Der OutputClient könnte zusätzlich dazu genutzt werden, in größeren Räumlichkeiten das Audio des Vortrags weiterzuleiten. (Das ist im Rahmen des Projekts zunächst nicht vorgesehen.)

**3.3 Präsentation**

* Die Präsentation ist zunächst lokal beim MasterClient gespeichert.
* Vor dem Vortrag wird die Präsentation dann vom MasterClient zum Server hochgeladen.
* Die Präsentation darf nur aus Bildern bestehen.
* Dateiformat: ist noch nicht festgelegt, evtl. Zip-Archiv mit JPG-Dateien. Hier muss recherchiert werden, was von QT problemlos verarbeitet werden kann. Wer recherchiert?

**3.4 Phasen**

Einen Vortrag mit PräsiBert zu halten, gliedert sich grundsätzlich in vier Phasen:

1. Vorbereitung / Setup
   * Netzwerk aufbauen
   * Präsentation hochladen
   * OutputClient wählen
2. Start
   * Server sendet Startfolie an alle Clients
   * Audioaufnahme beim MasterClient startet
3. Währenddessen
   * MasterClient kann die Befehle previous bzw. next auslösen, entweder per GUI oder per Gestensteuerung
   * Server reagiert auf dies Befehle previous bzw. next und schaltet ein Bild zurück bzw. vor (und sendet das an alle Clients)
   * ListenerClients können per GUI Redezeit beantragen
   * Der MasterClient bekommt dies per GUI angezeigt und kann die Anfrage ablehnen oder annehmen.
   * Falls die Anfrage angenommen wird, startet der ListenerClient eine Audioaufnahme.
   * Die Redezeit endet, wenn der MasterClient oder der ListenerClient dies (per GUI) bestimmen. Dann wird die Audioaufnahme beendet und vom ListenerClient an den Server geschickt.
   * Der MasterClient kann (per GUI) den Vortrag beenden.
4. Stop
   * Sobald der Vortrag beendet ist, wird das Audiofile des MasterClient an den Server geschickt.
   * Der Server sichert die finale Datenstruktur mit Audiofiles, Präsentation und Logs.
   * optional: Der MasterClient bekommt die Audiodaten vom Server übertragen. (Zunächst nicht vorgesehen.)
   * Der MasterClient bekommt einen LogOut-Screen mit Countdown bis zum automatischen LogOut (Begründung: Mögliche Erweiterungen, welche jetzt noch eine Verbindung zwischen MasterClient und Server benötigen.)

**3.5 Messages**

Alle Prozesse sollen in einem einheitlichen Message-Format miteinander kommunizieren. Konkret betrifft das die folgenden Befehle (evtl. unvollständig):

* + Start der Präsentation
  + Stop der Präsentation
  + Previous (vorige Folie)
  + Next (nächste Folie)
  + Redeanfrage eines ListenerClients
  + Annahme/Ablehnung des MasterClient
  + Stop der Redezeit des ListenerClients
  + alle Datenübertragungen

Vermutlich kann man hier mit XML arbeiten, Jan recherchiert das genauer.

**3.6 Sonstiges**

* alle Audiodateien müssen mit Timestamps versehen werden. Hierzu ist notwendig, dass eine „Uhrenkontrolle“ implementiert wird, die dafür sorgt, dass alle Clienten und der Server in dieser Hinsicht synchron sind.
* optional: PräsiBert könnte um Zugriffskontrollen ergänzt werden, z. B. nur registrierte Clienten können sich anmelden. (Zunächst nicht vorgesehen)

**4. Aufgabenverteilung / Arbeitspakete**

Reminder: Eine zugewiesene Aufgabe bedeutet, dass das oder die Teammitglieder für diese Aufgabe hauptverantwortlich sind. Eine Aufgabenzuteilung bedeutet nicht, dass das oder die Teammitglieder diese Aufgabe vollständig alleine bearbeiten müssen.

* Server: Sebastian
* Netzwerk: Niklas
* Messages (Kommunikationsformat der Prozesse): Jan
* GUI: N. N.
* Clients – Basis: Sebastian und Jan
* Clients – Details: N. N.
* Audio Processing: Arno
* Gestensteuerung: Inga
* Dokumentation: Inga

Für alle Teammitglieder gilt:

* Wer fertig ist, meldet sich!
* Die eigene Arbeit ist zu dokumentieren (Details siehe 6. Dokumentation)

**5. Meilensteinplanung**

* Mittwoch, 20.05.2015: zugeteilte Aufgaben sind vom Verantwortlichen detailliert zu spezifiziert. (Stichworte: Klassendiagramm, Dokumentation von Design-Entscheidungen)
* Freitag, 12.06.2015: (Erste) Implementierungen sind abgeschlossen 🡪 Zusammenführung der Einzelteile zu einem Softwaresystem ist möglich
* Mittwoch, 01.07.2015: Lauffähiges Softwaresystem

**6. Dokumentation**

Bestandteile:

* Dokumentation mit den Requirements Engineering Methoden (Lastenheft, Pflichtenheft, Use Cases, etc.) – Grundlage hierfür stellt dieses Protokoll dar
* automatisch generierte Dokumentation, Stichwort: Doxygen 🡪 alle Teammitglieder haben darauf zu achten, dass ihr Code diesem Standard genügt
* im Rahmen der Versionskontrolle mit GIT: Dokumentation der Commits, d. h. was wurde gemacht und warum wurde es gemacht.

**7. Lizenz**

PräsiBert soll unter GNU General Public License (GPL) gestellt werden.