## PEGASUS

Projekt der FOTEC

Aufgaben:

* SW-Architektur für das Projekt erstellen
* Evaluierung wie gut die beiden Systeme (freeIrgendwas und NochIrgendwas) wirklich miteinander einsetzbar sind

## CLAM – Komponenten-Repository

* Audio-Analyse-Tool
* Einzelkomponenten können zusammengestellt werden (werden mit CLAM fix ausgeliefert, was auch schon das eigentliche Problem darstellt)
* basiert auf C++-Bibliothek, Editor in Python

Aufgaben:

* Erweiterung des Projekts um Komponenten-Repository
  + Überlegung wie das geschehen soll
  + Komponenten auf Basis des Komponenten-Systems hinterlegt
  + API (REST-Service, etc.)
  + Notification, Queries an das Repository
  + Repository kann beliebig mächtig gemacht werden (Verwendungshinweise, etc.)
* Verteilung von Komponenten
  + Redundante Verteilung

## P2P Systeme evaluieren

Es gibt verschiedene P2P-Systeme, welche alle unterschiedliche Stärken und Schwächen haben und unterschiedliche Herangehensweisen verfolgen.

Aufgaben:

* Aufwand für Umsetzung
* Welche Eignung für welche Aufgaben
* Sicherheitsaspekte
* Use Cases und Szenarien finden und Prototyp erzeugen und vergleichen

## Software-Lego

Verschiedenste Interpretationen zu Komponenten-Modell => verschiedenste Realisierungen (Frameworks)

Komponente = static abstraction with plugs (Analogie zu Lego)

Aufgaben:

* Welche Technologien gibt es
* Technische Faktoren bewerten (Entwicklungsaufwand & -kosten, Usability, Portierbarkeit, Interoperabilität)
* Methodischen Faktor bewerten (gibt es Repositories?, welche Programmiersprachen werden unterstützt?, etc.)
* Use Cases und Prototypen

## Parallele Programmiermodelle

Imperative Programmierung: Problem -> Alogorithmus -> Programm als Folge von Anweisungen; Varbiablenzuweisung; Seiteneffekte (Werte können sich jederzeit ändern, Verifizierbarkeit von Programmen schwierig); quasi voll und ganz im Solution-Space einzuordnen; bricht Separation-Of-Concern-Prinzip bzw. Single-Responsibility-Prinzip

Deklarative Programmierung: Problem-Space (was will ich haben?), Solution-Space (wie kann ich Problem lösen? Welchen Algorithmus?); Problembeschreibung in Reinstform (SQL, Prolog, DLV, Haskall); eignen sich gut für Parallelisierung; vermeidet viele Probleme der imperativen Programmierung;

Aufgaben:

* Welche (deklarativen) Programmiersprachen gibt es überhaupt?
* Wo sind die Unterschiede?

## Evaluierung von Object-Storages

Momentaner interessanter Trend; Kein Block im Filesystem, sondern Object;

Aufgaben:

* Was verbirgt sich wirklich hinter den Object-Storages
* Sicherheitsaspekte (speziell bei P2P-Modellen)