Test

Bash Vi, Shylux, Kaleb Tschabolt

February 27, 2011

Status	**In Arbeit**/In Pruefung/Abgeschlossen
Projektname	Projektexplorer
Projektleiter	Niklaus Hofer
Auftraggeber	M. Frieden, GIBB
Autoren	Niklaus Hofer
Verteiler	Lukas Knoepfel, Kaleb Tschabolt, Niklaus Hofer

Aenderungskontrolle, Pruefung, Genehmigung

Version	Datum	Beschreibung, Bemerkung	Name oder Rolle

Definitionen und Abkuerzungen

Begriff/ Abkuerzung	Bedeutung

Referenzen

Referenz	Titel, Quelle
[1]	
[2]	
[3]	

Contents

1 Zweck des Dokuments **4**

2 Ist-Aufnahme und Ist-Analyse **4**
2.1 Beschreibung des Ist-Zustands (Ist-Analyse) 4
2.2 Schwachstellenanalyse 4
2.3 Sicherheit 4
2.4 Zukünftige Entwicklung 4

3 Systemziele **6**

4 Fachlicher Soll-Zustand **6**
4.1 Hauptaufgaben der neuen Lösung 6
4.2 Übersicht über die erforderlichen Informationen / Daten 7
4.3 Anforderungen bezüglich Informationssicherheit und Datenschutz 7

5 Lösungsvarianten **8**
5.1 Zu verwendende Programmiersprache 8
5.1.1 Lösungsvariante 1: Java 8
5.1.2 Lösungsvariante 2: Python 8
5.2 Zu verwendende Ablage für die Metadaten 8

1 Zweck des Dokuments

In der Voranalyse geben wir einen groben Einblick in die kuenftigen Funkionen unseres Programmes. Hier werden die Ziele festgesetzt und Entscheidungen über die zu verwendeten Technologien begruendet und festgehalten.

2 Ist-Aufnahme und Ist-Analyse

2.1 Beschreibung des Ist-Zustands (Ist-Analyse)

- Da das Projekt neu gestartet wird, existiert keine Vorarbeit, auf der wir aufbauen könnten.
- Einige der Ideen können aber mit denjenigen von Gnome Zeitgeist [1] und Apple timemachine [2] verglichen werden.
- Als bestehende Lösung könnte man die herkömmliche, hierarchische Struktur von Dateisystemen ansehen. Allerdings wollen wir diese nicht ersetzen, sondern lediglich um weitere Möglichkeiten ergänzen.

2.2 Schwachstellenanalyse

- Es existiert zwar kein Vorsystem, dessen Schwächen wir hier beschreiben könnten. Ich möchte hier aber auf die Schwächen herkömmlicher Dateiverwaltungssysteme eingehen.
- Diese haben vorallem das Problem, dass sich die Anwender herkömmlicher Dateisysteme eine Dateistruktur ausdenken und merken müssen. Sollten sie die Struktur jemals vergessen, so ist es schwierig Dateien zu finden. Auch ist das System recht unflexibel, wenn es darum geht Dateien, die zum gleiche Thema gehören, die aber weit über die hierarchische Struktur verteilt sind, zusammen zu verwalten.

2.3 Sicherheit

Es existieren keine Vorsysteme, die daher auch keine Sicherheitslücken aufweisen können.
Erwähnenswert scheint hier aber, einmal mehr, das cloud computing. Alle Daten bei einem storage Anbieter zu lagern löest nicht nur das Problem der Backups, sondern erlaubt auch das blitzschnelle Durchsuchen der Dokumente, da viele der online Anbieter (insbesondere Google mit seinem online Office) die Dokumente indexieren und so das Durchsuchen mit komplexen Algorithmen erlauben. All diese Vorteile haben aber einen entscheidenden Nachteil: Die Kontroller über die Dateien geht verloren. Der Storage-Anbiter hat, sofern man die Daten nicht manuell verschlüsselt, vollen Zugriff darauf. Zudem besteht hier der Nachteil dass ein Angreiffer alle die Daten bequem einsehen wenn er an das Passwort kommt.

2.4 Zukünftige Entwicklung

Das Gnome Zeitgeist Projekt [1] setzt einige innovative Ideen um, die das verwalten von Dateien komfortabler machen sollen. Dabei geht es darum, dass dokumentiert wird wann welche Dateien geöffnet wurden und welche anderen zur selben Zeit geöffnet waren. Es ermöglicht dann nicht nur die chronologische Ansicht der Dateien, sondern auch Abfragen wie “welches war der Song, den ich während dem Betrachten jener Bilder gehört habe?”.

- Welche Trends zeichnen sich ab und sind sie relevant?
  - Ein Trend ist dahingehend fest zu stellen, dass immer mehr Volltextsuchen für den Desktop zu Einsatz kommen, zum Beispiel Google Desktop search [3].
- Wie ist die Bereitschaft zu Veränderungen?
  - Die Bereitschaft grosse Änderunge ein zu gehen scheint nicht besonders gross zu sein.
    So wird das aktuelle System schon seit Jahrzehnten nahezu unverändert verwendet.
    Um die User nicht zu verwirren, vermeiden die Hersteller grundlegende Änderungen und verhalten sich eher zurückhaltend.
    Auch Gnome Zeitgeist wurde aus dem Gnome 3.0 Release gestrichen [4] und wird wohl vom Anwender selbst installiert werden müssen.
- Welche Anforderungen muss das System in Zukunft erfüllen?
  - Die User möchten Files mit anderen über die Ordner hinaus verknüpfen. Weg von der hierarchischen Struktur.

  - Das System muss in der Lage sein auch sehr grosse Mengen an Dateien so zu verwalten, dass einzelne Dateien schnell und ohne Umwege gefunden werden können, ohne dass sich der User lange Dateipfade merken muss.

- Wie wird sich das System entwickeln, wenn nichts geändert wird?
  - Die Betriebsysteme werden eingebaute Funktionen zum tagen mitbringen. Diese Funktion gibt es bereits in Gnome Zeitgeist.
  - Zum Versionieren gibt es heute schon einige einige Tools (meist Kostenpflichtig), die aber meist sehr undurchsichtig sind.
  - Bequeme backup-timeline-Lösungen ähnlich Apple's Zeitgeist werden wohl vermehrt eingesetzt werden, da sie das Erstellen und Verwalten von Backups vergleichsweise sehr einfach machen.

- Wo wurden ähnliche Probleme schon gelöst?
  - Windows 7 hat ein neues System zum erstellen 'virtueller' Ordner mitgebracht. Hier können Diese fassen mehrere Ordner und Dateien, unabhängig von deren Speicherort, in einem virtuellen Verzeichnis zusammen.
  - Es gibt schon einige Taggingsysteme, auch auf Basis des Filesystems.
  - Abgesehen von Gnome Zeitgeist hat sich seit Jahren nur sehr wenige getan bei der Verwaltung von Dateien.

- Welche Vorstellungen hat man von einer neuen Lösung?
  - In dem Bereich gibt es einige interessante Experimente und Untersuchungen, ein allgemeiner Tenor über eine 'Lösung der Zukunft' ist aber nicht in Sicht.
  - Evtl. wird sich das Problem auch von selbst lösen, falls die Verschiebung in die Cloud so von statten geht wie es sich einige Internetdienstleiste (Stichwort Google) vorstellen.
  - Auch aufwendige Dokumentenverwaltungssysteme könnten in Zukunft öfters zum Einsatz kommen, besonders beim Einsatz in Firmen, die grosse Datenmengen sinnvoll verwalten müssen.

- Was oder wer kann das zukünftige System beeinflussen?
  - Das System könnte durch zwei Dinge stark beeinflusst werden. Eine Möglichkeit wäre, dass ein Startup, oder ein freies Projekt eine Lösung bringt, die derart genial ist, dass sie sich durchsetzt und dann auch in der Industrie (Windows, OS X) integriert wird.
  - Die andere Möglichkeit wäre, dass der Branchenriese Microsoft eine Lösung in künftige Versionen von Windows integriert und so die Anwender auf ein neues System 'zwingt'.

- Welche kritischen Erfolgsfaktoren gibt es?
  - Das System darf nicht kompliziert sein und die Nutzer verwirren. Viele Nutzer hatten bereits Probleme damit das aktuelle System zu erlernen. Falls ein neues nicht sofort verständlich ist, so werden sie es ablehnen.

3 Systemziele

Bezeichnung	Beschreibung	Priorität (0-3)	Zielhierarchie	Zielkategorie	Kriterien zur Bewertung
Performance	Läuft das program schnell?	2			Das Programm sollte ähnlich schnell reagieren, wie die von den verschiedenen Systemen mitgelieferten Dateimanager. Keinenfalls darf sich die Benutzung so träge anfühlen, dass man lieber auf die Vorteile des Systems verzichtet.
Qualität	Kann das Programm einfach erweitert werden?	2			Das Programm ist objektorientiert aufgebaut. Die Benennung von Objekten und Variablen ist einheitlich. Das Programm ist klar und verständlich dokumentiert.
Sicherheit	Das Programm reist keine Sicherheitslücken in die Mechanismen der jeweiligen Betriebssysteme.	3			Firewalls und Rechteverwaltungssysteme werden nicht umgangen.
Kompatibilität	Läuft das Programm auf allen wichtigen Betriebssystemen.	2			Das Programm soll auf folgenden OS laufen: Windows, Linux, OS X
Datenkonsistenz	Die Metadaten werden ohne ueberfluessige Redundanze	1			Die Metadaten Ablage verhindert vom Design her widersprüchliche oder doppelte Angaben.

4 Fachlicher Soll-Zustand

4.1 Hauptaufgaben der neuen Lösung

- Hauptaufgaben
  - Dateien verwalten
  - Dateien können mit Tags versehen werden.
  - Dateien können nach Tags sortiert werden.
  - Operationen können auf eine Auswahl an Dateien (z.B. alle Dateien eines bestimmten Tags) angewandt werden.
  - Das Programm kann Dateien Versioniren.
- Informationsflüsse
  - Informationen zu den Dateien (Tags) werden dauerhaft abgelegt.
  - Werden Dateien mit Hilfe anderer Programme verschoben/erstellt, sollte das Programm (zumindest falls es läuft, vorzugsweise aber immer) davon in Kenntnis gesetzt werden.
- Schnittstellen nach aussen, zu externen Systemen
  - Das Programm muss Zugriff auf ein System haben, in dem es die Tags ablegen kann.
  - Zugriff auf das Dateisystem muss vorhanden sein, um das Verschieben von Dateien zu registrieren und um Dateien zu archivieren/versioniren.
  - Falls das Programm ein CLI erhält, so kann es natürlich mit Scripts bedient und automatisiert werden.

- Datenfluss

4.2 Übersicht über die erforderlichen Informationen / Daten

- vom Dateisystem
  - Name der Datei
  - Wo liegt die Datein (Verzeichnis)
  - Andere Metadaten wie Zugriffsrechte, Datum der letzten Veränderung, ...
- von der Metadaten-Ablage
  - Welche Tags wurden der Datei X zugeordnet
- Dateisystem-Events
  - Datei(en) wird/werden gelöscht
  - Datei(en) wird/werden verschoben
  - Datei wird verändert

4.3 Anforderungen bezüglich Informationssicherheit und Datenschutz

- Informationssicherheit: Verfügbarkeit, Vertraulichkeit, Integrität
  - Die Verfügbarkeit des Programms hängt lediglich von der des Dateisystems ab und kann mit dieser gleichgesetzt werden.
  - Das Vertrauen des Users wird gewonnen, indem nichts nach aussen (internet) geschickt wird.
  - Die Integrität der Daten zu garantieren ist von daher eine Herausforderung, als dass das Programm mitbekommen muss, wenn Dateien verschoben werden und das registrieren muss. Ansonsten verweisen Metadaten ins Leere und Dateien verlieren die Ihnen zugeordneten Metadaten.
    Dieses Problem kann durch das Abhören von Dateisystem Events behoben werden.
    Umgehen liesse sich das Problem nur dann, wenn die Tags in den Filesystem-metadaten gespeichert würden.

- Datenschutz
  - Da das Programm lokal auf dem Computer läuft und nicht in die Zugriffsmechanismen des vorhandenen Systems eingreift geht die einzige Gefahr davon aus, dass der Speicherort der Tags für andere User frei zugänglich ist.
    Um das zu verhindern müssen die Tags entweder direkt in der jeweiligen Datei abgelegt werden oder, falls eine Datenbank zum Einsatz kommt muss diese in einem Verzeichnis abgelegt werden, auf das nur der jeweilige Benutzer Zugriff hat.

5 Lösungsvarianten

5.1 Zu verwendende Programmiersprache

5.1.1 Lösungsvariante 1: Java

Beschreibung der Lösungsvariante Bei dieser Lösung würde Java als Programmiersprache für das Projekt verwendet. Java hat den Vorteil, dass der Interpreter eine sehr grosse Verbreitung geniesst und alle beteiligten Entwickler mit der Sprache vertraut sind.

Struktur (Grobe Architektur) Java hat sehr viele Module, die nicht nur die Entwicklung erleichtern, sondern auf den Zugriff auf verschiedene Formate, die zum Abspeichern der Daten verwendet werden können (Siehe unten <LINK ZU KAPITEL>).

Externe und interne Schnittstellen Wie bereits erwähnt bietet Java viele Schnittstellen zu gängigen Datenbanken und Dateiformaten an.
Zudem hat Java verschiedene Frameworks die zur Erstellung des GUIs verwendet werden können und sich zum Teil stark an den nativen Look des jeweiligen Systems anpassen.
Java's Ausgabe in die Konsole hingegen ist eher langsam.

Abdeckung der Anforderungen

- **Performance:** Java gilt heute als recht schnell.
- **Flexibilität:** Java gilt zur Zeit als die meist verwendete Programmiersprache überhaupt, was den Vorteil mit sich bringt, dass sehr viele Programmierer in der Lage sind damit geschriebene Programme weiter zu entwickeln oder an zu passen.
- **Sicherheit:** Dadurch dass Java code in der JVM ausgeführt wird, wird das ohnehin geringe Sicherheitsrisiko, dass durch den Einsatz der Software entsteht, noch weiter verringert.
- **kompatibilität:** Java ist für nahezu alle computer (egal ob Desktop oder handheld) verfügbar, in verschiedenen Varianten, die teils selbst den hohen Offentheitsansprüchen des Debian Projektes genügen. Die Verwendung vieler verschiedener Interpreter bringt aber auch Probleme. So sehen etwa Programme die für Sun/Oracle Java entwickelt wurden im Betrieb unter IcedTea etwas anders aus.

Realisierbarkeitsbetrachtung Die beteiligten Programmierer haben alle bereits Erfahrungen mit Java. Zudem findet man im Internet schnell Hilfe zu fast allen Bereichen der Programmiersprache.
Die Realisierbarkeit ist also durchaus gesichert wenn Java zum Einsatz kommt.

5.1.2 Lösungsvariante 2: Python

Beschreibung der Lösungsvariante Zur Implementierung des Programmes würde hier Python verwendet. Die moderne Programmiersprache kommt heute immer öfters zum Einsatz und läuft auf allen gängigen Betriebssystemen.

Struktur (Grobe Architektur) Auch Python bietet Schnitstellen zu fast allen wichtigen Formaten und Datenbanken. Zudem bietet Python aber noch den Vorteil, dass es auch sehr viele low-level-Schnittstellen gibt, die auch sehr hardware (oder, hier von grösserem Interesse, Dateisystem)nahe Operationen erlauben.

Externe und interne Schnittstellen Python braucht ebenfalls einen Interpreter. Dieser ist aber für alle gängigen Betriebssysteme verfügbar. Zudem kann Python code auch in der weit verbreiteten Java Virtual Machine ausgeführt werden [5].

Abdeckung der Anforderungen

- **Performance** Python gilt als sehr performante Programmiersprache. Die Performance würde für unseren Einsatzzweck vollauf genügen.
- **Flexibilität** Python hat ein ausgeprägtes Objekt Modell und ist gut Verständlich. Zudem steigt die Zahl an Python Entwicklern seit einger Zeit sehr rasch. Auch eine in Python geschriebene Implementierung könnte also später gut von einem anderen Programmierer erweitert oder geändert werden.
- **Kompatibilität** Der Python Interpreter hat eine deutlich weniger grosse Verbreitung als Java, ist aber auf Unix-artigen Systemen meist enthalten und kann unter Windows einfach installiert werden. Es lässt sich zudem spekulieren, dass Oracles aktueller Umgang mit Java und insbesondere mit der community der Verbreitung von Java starch schaden wird, wodurch dessen Verbreitung evtl. hinter die von Python zurückfallen könnte.

Realisierbarkeitsbetrachtung Der Einsatz von Python hat den Nachteil, dass die beteiligten Entwickler damit weniger vertraut sind als mit Java. Allerdings ist dadurch der Lerneffekt grösser. Zudem bietet Python mächtige Module, die die Arbeit vereinfachen.
Python ist auch insbesondere für die Entwicklung von GUIs gut geignet. Unter Linux gibt es eine Vielzahl von Projekten, die lediglich ein Python GUI für Tools darstellen, die ansonsten nur für die Konsole verfügbar sind.

5.2 Zu verwendende Ablage für die Metadaten