Angabe für das Abschlussprojekt in WS2014 / MIT / SAD / Programmierung

1 Einführung

Das vorliegende Dokument beschreibt die Aufgabenstellung[1] für das Abschlussprojekt. Es handelt sich um ein Programm, dass in der Lage ist, Datenbestände auf verschiedenen Speicherorten synchron zu halten. Nachfolgend wird hierfür der Name DataSync verwendet.

2 Anforderungen

Nachfolgend werden die **Mindestanforderungen** beschrieben. Die Nichteinhaltung dieser Mindestanforderungen resultiert automatisch in einer negativen Note.
Das Programm **muss** aus Enums, Klassen, Interfaces, Delegates und Events bestehen. Die Klassen und deren Methoden müssen sinngemäß gekapselt werden, so dürfen in Methoden die z.B.: der Berechnung von Werten dienen keine Eingaben von bzw. Ausgaben am Bildschirm vorgenommen werden. Entwickeln Sie das Programm in Hinblick auf Erweiterbarkeit.

2.1 Coding Guidelines

Um die Lesbarkeit und Wartbarkeit des Codes zu gewährleisten sind ALLE StyleCop-Regeln einzuhalten. Automatisch generierte Dateien, wie z.B.: Assembly.cs, sind von dieser Regel nicht betroffen, Änderungen an diesen Dateien müssen also nicht vorgenommen werden.

2.2 Bedienbarkeit

Das Programm muss für den Benutzer intuitiv bedienbar sein, Fehleingaben müssen so weit wie möglich toleriert werden. Abstürze sämtlicher Art sind zu vermieden. Das bedeutet, dass sämtliche Laufzeitfehler abgefangen, entsprechend behandelt und ggf. dem Benutzer in sinnvoller Art und Weise zum Quittieren übergeben werden müssen.

3 Mindestanforderungen pro Notengrad und Überprüfung

Nachfolgend werden die jeweiligen Mindestanforderungen pro Notengrad beschrieben. Für eine positive Note müssen mindestens die Anforderungen der Note „Genügend“ vollständig und fehlerfrei umzusetzen. In Abhängigkeit der Anzahl an zusätzlich umgesetzten Funktionalitäten (Erweiterungen A – C) ergeben sich die Notengrade „Befriedigend“, „Gut“ und „Sehr Gut“. Für eine zusätzlich umgesetzte Funktionalität kann bestenfalls die Note

[1] Allgemeiner Hinweis zur Abgabe: Es wird eine Abgabe inklusive Abgabe-Prüfungsgespräch geführt in dem nachgewiesen werden muss, dass sämtliche Inhalte selbstständig erarbeitet und verstanden wurden. Dies wird unter Umständen auch durch implementieren von Funktionalität während des Prüfungsgesprächs überprüft.

„Befriedigend“, für zwei zusätzlich umgesetzte Funktionalitäten die Note „Gut“ und für drei zusätzlich umgesetzte Funktionalitäten die Note „Sehr Gut“ erreicht werden. Die Reihenfolge der zusätzlich umgesetzten Funktionalitäten ist hierbei von keiner Relevanz.

3.1 Genügend

Implementieren Sie eine Consolen Applikation, die alle Mindestanforderungen und alle nachfolgenden Anforderungen erfüllt:

- Der Benutzer kann eine beliebige Anzahl von Verzeichnissen (sowohl lokale, als auch über FileShare erreichbare) angeben. Ein solches Verzeichnis wird im Folgenden als „Quellverzeichnis“ bezeichnet.
- Zu jedem Quellverzeichnis kann eine beliebige Anzahl von Verzeichnissen (sowohl lokale, als auch über FileShare erreichbare) angegeben werden. Ein solches Verzeichnis wird im Folgenden als „Zielverzeichnis“ bezeichnet.
- Das Programm muss Änderungen an Daten (auch löschen von Dateien) innerhalb des Quellverzeichnisses feststellen und diese in die definierten Zielverzeichnisse synchronisieren können. Hierbei sollen die Dateien nach abgeschlossenem Synchronisationsvorgang in jeder Hinsicht (also nicht nur inhaltlich sondern auch attributiv) ident sein.
- Es muss möglich sein einzustellen, ob nur Änderungen innerhalb des Quellverzeichnisses oder auch Änderungen in hierarchisch darunterliegenden Verzeichnissen überwacht werden sollen.
- Das Programm muss dem Benutzer die Möglichkeit bieten eine beliebige Anzahl an Ausnahmeverzeichnissen für jedes Quellverzeichnis zu definieren.
  Beispiel: Als Quellverzeichnis wurde „D:\Daten“ rekursiv definiert. In diesem Verzeichnis befinden sich die Unterverzeichnisse „A“, „B“, und „C“ wobei „B“ wiederum das Unterverzeichnis „B1“ besitzt. Als Ausnahmeverzeichnis wurde „B“ angegeben. Änderungen in den Verzeichnissen „A“ sowie „C“ sind also zu berücksichtigen, Änderungen im Verzeichnis „B“ (und daher auch in dessen Kindverzeichnissen – in diesem Fall „B1“) sind zu ignorieren.
- Sämtliche Aktionen müssen geloggt/protokolliert werden. Dies soll einerseits als Textausgabe in der Console geschehen und andererseits optional zusätzlich in eine spezifizierbare Logdatei.
- Es muss möglich sein eine maximale Dateigröße für die Logdatei zu definieren. Sobald die Logdatei die definierte Größe erreicht wird der Name dieser mit der Endung „.bak“ erweitert und eine neue Datei mit dem Originalnamen angelegt. Sollte bereits eine Datei mit der Endung „.bak“ vorhanden sein, so wird diese vor dem Logdateiüberlauf gelöscht.
  Beispiel: Als Logdatei wurde „D:\Log.txt“ und die maximale Dateigröße wurde mit 10Mb definiert. Die derzeitige Dateigröße beträgt 9.9Mb. Auf Grund durchgeführter Aktionen werden weitere Einträge (Zeilen) in die LogDatei geschrieben, daher wächst die Größe und erreicht (bzw. überschreitet) die definierte maximale Dateigröße. Nun wird überprüft, ob bereits eine Datei mit dem Namen „D:\Log.txt.bak“ existiert. Ist dies der Fall, so wird diese Datei gelöscht. Anschließend wird die Datei „D:\Log.txt“ in „D:\Log.txt.bak“ umbenannt und eine neue Datei mit dem Namen „D:\Log.txt“ erstellt.

- Es muss möglich sein die oben angeführten Einstellungen auch über Parameter in der Kommandozeile anzugeben. In welcher Form dies geschieht bleibt Ihnen überlassen, denken Sie aber daran, dass die Form der Angabe für Benutzer intuitiv und einfach sein soll.
- Wenn das Programm beendet wird und ein nicht synchroner Zustand vorherrscht muss der Benutzer darauf hingewiesen werden und das Beenden des Programms erneut bestätigen.
- Wenn das Programm gestartet wird muss überprüft werden, ob alle angegebenen Zielverzeichnisse mit allen angegebenen Quellverzeichnissen synchron sind. Ist dies nicht der Fall so müssen die Verzeichnisse synchronisiert werden.

3.2 Erweiterung A

Zur Umsetzung der *Erweiterung A* sind nachfolgende Anforderungen zu erfüllen:

- Ihre Applikation muss Synchronisationsvorgänge über „Jobs“ abarbeiten und diese in „Queues“ verwalten. Unter Job ist ein Vorgang, der in einer Queue gereiht wird, zu verstehen.
- Es muss möglich sein, sämtliche in der Applikation vorhandenen JobQueues zu visualisieren wobei mindestens die nächsten 10 anstehenden Jobs visuell darzustellen sind.
- Pro visualisiertem Job müssen mindestens dessen Quellverzeichnis, Dateiname, Zielverzeichnis und Status („queued“ oder „processing“) erkennbar sein, zu lange Verzeichnis- und Dateinamen dürfen sinnvoll abgekürzt werden.

3.3 Erweiterung B

Zur Umsetzung der *Erweiterung B* sind nachfolgende Anforderungen zu erfüllen:

- Unterschiede in Dateien werden auf Blockebene ermittelt. Hierfür kann der Benutzer definieren, ab welcher Dateigröße der Blockvergleich zum Einsatz kommt.
- Wenn eine Datei, für die der Blockvergleich gilt, modifiziert wurde, so wird die Quelldatei mit den Zieldateien anhand der definierten Blockgröße verglichen und es werden ggf. nur modifizierte Blöcke in den Zieldateien angepasst.
- Sowohl Dateigröße als auch Blockgröße müssen auch über Parameter in der Kommandozeile angegeben werden können.

3.4 Erweiterung C

Zur Umsetzung der *Erweiterung C* sind nachfolgende Anforderungen zu erfüllen:

- Es handelt sich um ein Feature, das künftig als „ParallelSync“-Feature bezeichnet wird.
- Bei Änderungen in Quellverzeichnissen und deren Synchronisation in die Zielverzeichnisse wird überprüft, ob sich die Zieldateien auf dem gleichen logischen Laufwerk befinden. Existieren Zieldateien auf verschiedenen logischen Laufwerken, so sollen die Schreiboperationen parallel für alle Zieldateien auf unterschiedlichen logischen Laufwerken abgesetzt werden.

- Es muss möglich sein, diese Funktion auch mittels Parameter in der Kommandozeile ein- bzw. auszuschalten.
  <u>Beispiel:</u> Für das Quellverzeichnis „D:\Daten“ wurden folgende Zielverzeichnisse definiert: „E:\DatenBackup“ und „F:\DatenBackup“. Wird nun eine Datei im Quellverzeichnis, z.B.: „D:\Daten\Quelldatei.txt“ modifiziert, so werden bei aktiviertem ParallelSync-Feature die Änderungen gleichzeitig in die Zielverzeichnisse übertragen. Wurde das ParallelSync-Feature nicht aktiviert, so erfolgt die Synchronisation der Zieldateien sequentiell (also z.B.: erst im Verzeichnis „E:\DatenBackup“ und anschließend im Verzeichnis „F:\DatenBackup“).

3.5 Überprüfung

Es wird eine Abgabe inklusive Abgabe-Prüfungsgespräch geführt in dem nachgewiesen werden muss, dass sämtliche Inhalte selbstständig erarbeitet und verstanden wurden. Dies wird unter Umständen auch durch implementieren von Funktionalität während des Prüfungsgesprächs überprüft.