pydica-docs

Release 1.0

Daniel Szoska, Christian Tismer, Veit Schiele

May 05, 2014

Contents

Inhalt:

CHAPTER 1

The Big Picture

- Distributed Document Capture
- Plattform für verteiltes Scannen, Erkennen, automatisierte und manuelle Überprüfung, Archivierung
- Skalierbare Client-Server-Architektur
- Verteilung der Aufgaben
- Administration und Monitoring

CHAPTER 2

Komponenten

Inhalt:

2.1 Konfiguration

Der Configuration-Editor erlaubt die Konfiguration der Komponenten *OCR*, *Server-Validator* und *Client-Validator* über `*.ini`-Dateien.

Im Einzelnen enthalten diese Dateien folgende Angaben:

2.1.1 Form Sets

- Die Konfiguration kann mehrere Arten von Formularen (Form Sets) enthalten.
- Jedes Form Set kann von einem anderen abgeleitet werden und nur die abweichenden Angaben enthalten.
- Jedes Form Set besteht aus folgenden Angaben:
  - eine oder mehrere Seiten
  - Feldern
  - Feldübergreifende Validatoren
  - Feldübergreifende Ersetzungen

2.1.2 Formularfelder

Folgende Angaben sind für Felder möglich

. . . für den Client-Validator

- ID und lokalisierbares Label eines Feldes
- Jedes Formularfeld enthält die exakte Position des Feldes, deren Höhe und Breite. Zum aktuellen Zeitpunkt ist Pixel (px) die einzig zulässige Maßangabe.
- Darüberhinaus kann für jedes Feld angegeben werden, in welchem Bereich um das Feld die OCR durchgeführt werden soll.
- Konfiguration der Anzeige

  - Standardschrift (ist keine angegeben, wird die Standardschrift des Betriebssystems verwendet).
  - Für jedes Feld soll die Standardschrift mit einer eigenen Schriftangabe überschrieben werden können
  - Mindestbreite eines Feldes in Anzahl der Zeichen

... für den Server-Validator

- Liste der Klassennamen der Validatoren für ein einzelnes Feld.

... für die Ersetzungen

- Liste der Klassennamen der Ersetzungen, wobei die Reihenfolge bedeutend ist.

  Dabei wird unterschieden zwischen Ersetzungen von Feldern und Form Sets.

2.2 OCR

2.2.1 Tesseract

Tesseract ist eine freie Implementierung, die von HP entwickelt wurde und jetzt von Google übernommen wurde.

- Home: https://code.google.com/p/tesseract-ocr/
- Python-Bindings: http://code.google.com/p/python-tesseract/

weitere Links:

- http://isbullsh.it/2012/06/Automatic-tesseract-training/
- http://stackoverflow.com/questions/4763956/performance-issues-using-tesseract-ocr-from-a-python-application
- http://wiki.ubuntuusers.de/OCRFeeder

2.3 Server-Validator

The valdiation of the forms generates tasks that are passed to the task manager.

2.3.1 Order

Die Reihenfolge der Validierungen ist eindeutig, es muss nicht zwischen Pre- und Post-Replacements unterschieden werden:

The sequence of the validation is unique. A distinction between prior and subsequent substitution is not required.

1. Field replacements
2. Validation
3. Form set choice

   The form type is chosen for which a minimum of tasks are available.

2.3.2 Validators

Methods of the field validators:

- Boolean
- Date
- Integer
- Decimal Numbers
- Regular expression
- Minimalwert
- Maximalwert

Methods for the form validators:

- Sum
- Same date or earlier
- Same date or later

2.3.3 Messages

In the messages the reasons for the manual checking should be localizable.

2.3.4 Traceability

In a format to be specified more precisely all validations, replacements and manual changes has to be saved. Conventional services should be used for analysis.

2.3.5 Allgemeine Prüfungen

Sonderfälle, wie zum Beispiel komplette weiße oder schwarze Images oder OCR-Ergebnisse, die weit außerhalb des "Normalen" liegen, sollten ein extra Event werfen.

Innerhalb eines Belegstapels soll geprüft werden, ob es die gleiche PIC-Nr. mehrmals gibt - das darf auf keinen Fall passieren (kommt aber trotzdem selten vor, zuletzt im Abrechnungsmonat 04/2012). Gibt es eine PIC-Nr. mehrmals, so müssen sicherheitshalber alle Belege mit dieser PIC-Nr. gelöscht werden. Die Prüfung auf doppelte PIC-Nr. wird in nachfolgenden Prozessen über alle Monatsrezepte durchgeführt, je eher ein solcher Fehler auffällt, desto besser. Das Löschen der betroffenen Rezepte muß interaktiv erfolgen.

Für Felder, die nur Text enthalten und sonst nicht weiter überprüft werden, muß es trotzdem eine Prüfung auf gültige Zeichen geben. Für die Felder Name, Straße und Ort sind zum Beispiel die Zeichen `:`, `+`, `?` und `'` nicht zulässig, da diese Zeichen bei der Datenlieferung an die Krankenkassen eine besondere Bedeutung haben (Feldtrenner etc.) und nicht an allen Stellen in der bestehenden Software sichergestellt ist, daß diese Zeichen ordentlich escaped werden.

2.3.6 Feldübergreifende Prüfungen

PZN mehrfach auf dem Rezept

Kommt eine PZN innerhalb eines Belegs mehrfach vor, so muß darauf hingewiesen werden und dem Benutzer überlassen werden, ob das in Ordnung ist. Nur in sehr seltenen Fällen kommt das vor, meistens ist es ein Fehler der

aktuellen OCR, die eine Zeile verdoppelt hat.

Sonderbehandlung Wirkstoff Isotretinoin

Arzneimittel, die den Wirkstoff Isotretinoin enthalten(zum Beispiel Isoderm) dürfen für Frauen im gebährfähigen Alter nur abgegeben werden, wenn das Ausstellungsdatum des Rezepts nicht älter als 7 Tage ist.

Anhand der Daten auf dem Rezept ist nicht eindeutig bestimmbar, ob es sich um eine Frau handelt. Deswegen sollte ein Rezept zur Anzeige gebracht werden, wenn alle folgenden Bedingungen erfüllt sind:

- Wirkstoff einer PZN auf dem Rezept ist Isotretinoin
- Zwischen Abgabedatum und Ausstellungsdatum des Rezepts liegen mehr als 7 Tage
- Anhand des Geburtsdatums wird ein Alter ermittelt, mit welchem sich eine Frau im gebärfähigem Alter befindet

Weitere Infos gibt es unter http://www.hautzone.ch/dermatologie/Seborrhoic/aknebehandlung.htm unter dem Stichwort “Tetragenes Risiko”.

Als Beispielrezept, welches von der Kasse wegen Überschreitung der Abgabefrist abgesetzt wurde, kann die PIC-Nr. 10518100436 dienen.

2.4 Ersetzungen

2.4.1 Ersetzungen innerhalb von Feldern

- Ersetzen von Werten, z.B. ‘G’ durch ‘6’
- Ersetzen von regulären Ausdrücken

2.4.2 Ersetzungen innerhalb von Formularen

- Ersetze Werte eines Feldes durch die Werte eines anderen Feldes

2.4.3 Ersetzung von Formularen

- Ersetze den aktuellen Formulartyp durch einen anderen

2.5 Client-Validator

2.5.1 Configuration

The client can be configured so that it can view and edit only tasks with certain classifications.

2.5.2 User Interface

Dialogs

Open

1. Only those files are available for selection, for which tickets of the configured classifications are available.
2. When opening the first page is displayed for which a ticket with an appropriate classification will be available.

Keyboard shortcuts

Ctrl-O (Strg-O) Oeffnen einer RDB- bzw. CDB-Datei, die alle Daten enthaelt

F3 Springt innerhalb der gerade geoffneten RDB- bzw. CDB-Datei zu einem bestimmten Beleg. Dazu wird das Feld unten in der Statusleiste, in dem die aktuelle Belegnummer innerhalb der Datei angezeigt wird, aktiviert, so das dort Belegnummer eingegeben werden kann. Durch druecken von Enter wird dann der entsprechende Beleg geladen und angezeigt.

PgUp (Bild-nach-oben) springt zum vorhergehenden Bele

PgDn (Bild-nach-unten) springt zum naechsten Beleg

Cursor up (Cursor-nach-oben) springt zum vorhergehenden Feld innerhalb des Belegs; ist das erste Feld im Beleg erreicht, so wird der vorhergehende Beleg geladen und dort in das letzte Feld gesprungen

Cursor down (Cursor-nach-unten) springt zum naechsten Feld innerhalb des Belegs; ist das letzte Feld im Beleg erreicht, wird der naechste Beleg geladen und dort in das erste Feld gesprungen

CR (Enter) gleiche Wirkung wie "Cursor down"

Misc

- Ein großer Abstand zwischen Scan und Text verlangsamt die Überprüfung.
- Heute besitzen die Monitore üblicherweise eine deutlich höhere Auflösung, sodass ein besserer Überblick gegeben werden kann.
- So kann das Originalbild links dargestellt werden und rechts daneben der zu korrigierende Text im annähernd selben Layout.

  Hier würde sich ein Configuration-Editor empfehlen um Layoutanpassungen schnell vornehmen zu können.
- Anzeige von leicht verständlichen Statusmeldungen mit den Gründen für die manuelle Überprüfung.

Sonderbefehle

PZN / Faktor / Taxe ab 4. Zeile ergänzen

Aktuell werden von der OCR nur die ersten drei Zeilen erkannt. Bei vielen Rezepten, die 4 Positionen oder mehr haben, stimmt dadurch die Bruttosumme nicht (wenn nicht eine Taxe durch die OCR automatisch korrigiert wurde - das sollte unbedingt bald abgeschaltet werden!).

Es sollte in einem solchen Fall eine einfache Möglichkeit geben, um PZN, Faktor und Taxe ab der 4. Zeile nachzuerfassen.

Ein Sonderfall tritt ein, wenn ein Betrag von 26, 52 oder 78 Cent am Brutto fehlen - dann ist es mit ziemlicher Sicherheit ein BTM-Rezept mit der Sonder-PZN 02567024 und dem Faktor 1, 2 oder 3. Hierfür sollte es noch einen besonderen Modus geben, der einem die Tipparbeit für diesen recht häufig vorkommenden Fall so weit wie möglich abnimmt. Ein gutes Beispiel ist der Stapel 55/304 im Abrechnungsmonat 02/2013.

2.6 Validatoren

2.6.1 Prüfziffernverfahren

Institutionskennzeichen (IK)

Zitat von http://de.wikipedia.org/wiki/Institutionskennzeichen:

> Die Institutionskennzeichen (kurz: IK) sind eindeutige, neunstellige Zahlen, mit deren Hilfe Abrechnungen im Bereich der deutschen Sozialversicherung einrichtungsübergreifend abgewickelt werden können. Hierbei erhalten alle Einrichtungen, die Leistungen nach dem Sozialgesetzbuch (SGB) erbringen, auf Antrag ein IK.

Im konkreten Fall der Rezeptabrechnung wird das IK für die Leistungserbringer (Apotheken) und die Kostenträger (Krankenkassen) verwendet.

Das Prüfziffernverfahren ist gut auf Wikipedia dokumentiert: http://de.wikipedia.org/wiki/Institutionskennzeichen#Pr.C3.BCfverfahren

Krankenversichertennummer (KVNR)

Einstiegspunkte:

- http://de.wikipedia.org/wiki/Krankenversichertennummer
- https://kvnummer.gkvnet.de/(S(okkxpu55jdhixt452nidvaqj))/pubpages/krankenversichertennummer.aspx

Relevant für die Prüfung im Rahmen der Rezeptabrechnung sind nur die ersten 10 Stellen der Krankenversichertennummer, da nur diese auf dem Rezept stehen.

Aus http://www.gematik.de/cms/media/dokumente/release_0_5_2/release_0_5_2_egk/gematik_eGK_Spezifikation_Musterkarten_und_
wurden von Seite 60 folgende Information entnommen:

> Die erste Stelle der 10 Stellen ist ein Großbuchstabe (A-Z), die nächsten 8 Stellen sind Ziffern (0-9) und die letzte Ziffern ist eine Prüfziffer (0-9).
>
> Der Buchstabe und die 8 Ziffern sind für jede Person „zufällig“, aber eindeutig, vergeben. Werte mit mehr als drei aufeinander folgenden gleichen Ziffern werden ausgeschlossen. „Zufällig“ meint hier, dass keine weitere Semantik enthalten ist. ...
>
> Die Prüfziffer wird mit dem Modulo-10-Verfahren und den Gewichtungen 1-2-1-2-1-2-1-2-1-2 berechnet. Der Buchstabe wird dabei durch eine zweistellige Zahl ersetzt, das A mit 01, das B mit 02, . . . , und das Z mit 26.

Beispiel: Aus Z629410041 wird für die Prüfung 2662941004, woraus sich dann die Prüfziffer 1 ergibt.

lebenslange Arztnummer (LANR)

Auszug aus http://de.wikipedia.org/wiki/LANR:

> Die lebenslange Arztnummer, kurz LANR, ist eine neunstellige Nummer, die die zuständige Kassenärztliche Vereinigung bundesweit an jeden Arzt vergibt, der an der vertragsärztlichen Versorgung (siehe auch GKV) teilnimmt.

Das Prüfziffernverfahren ist unter http://de.wikipedia.org/wiki/LANR#Aufbau dokumentiert.

Pharmazentralnummer (PZN)

Auszug aus http://de.wikipedia.org/wiki/Pharmazentralnummer:

> Die Pharmazentralnummer (PZN) ist ein in Deutschland bundeseinheitlicher Identifikationsschlüssel für Arzneimittel und andere Apothekenprodukte.

Seit 01.01.2013 ist die PZN 8-stellig, vorher war sie 7-stellig. Aus den bisherigen 7-Stellern wurde ein 8-Steller, in dem eine 0 vorangestellt wurde. Durch den Aufbau des Prüfziffernverfahrens ist sichergestellt, dass die Prüfziffer bei der Erweiterung von 7 auf 8 Stellen nach wie vor die gleiche ist.

Das Prüfziffernverfahren ist in http://www.ifaffm.de/download/IFA-Info_Funktion_und_Aufbau_PZN.pdf dokumentiert.

Transaktionsnummer (TID)

Im Rahmen der Rezeptabrechnung erhält ein Rezept eine Transaktionsnummer, wenn parallel zum Rezept eine Datenlieferung von der Apotheke an das Rechenzentrum erfolgt. Diese Transaktionsnummer wird auf das Rezept aufgedruckt. Im Rechenzentrum können dann anhand dieser aufgedruckten Transaktionsnummer die elektronisch gelieferten Daten dem Rezept zugeordnet werden.

Das Prüfziffernverfahren für die Transaktionsnummer ist in der Technischen Anlage 1 zur Vereinbarung über die Übermittlung von Daten im Rahmen der Arzneimittelabrechnung gemäß § 300 SGB V beschrieben. Die aktuelle Version des Dokuments ist immer unter http://www.gkv-datenaustausch.de/, dort unter Leistungserbringer - Apotheken zu finden.

Die zum Zeitpunkt aktuelle Version der Technischen Anlage 1 ist unter http://www.gkv-datenaustausch.de/media/dokumente/leistungserbringer_1/apotheken/technische_anlagen_aktuell/TA1_023_20121127.pdf zu finden. Dort wird im Abschnitt 7 auf Seite 25 das Prüfziffernverfahren beschrieben.

2.7 Task Manager

Initially, the task manager shall provide the paths to the data bases and some useful methods.

2.7.1 Paths

The following attributes are configurable according to the Uniform Naming Convention (UNC):

Protocol This may be for instance `file:` for access to the local file system, but may also set conventions for real database access.

Storage Server In case of the `file:` protocol, it may be the identifier of the drive (i.E. drive letter on windows).

Path Components Path to the data base as specification of the partial component

File name Calculated name of the data base file

2.8 Categories

Categories are extensible. The set of available categories is individual to a context.

The lowest set of categories has the following default hierarchy:

- `batch_ident`

  identification of a batch of forms

- `form_ident`

  identification of a form in a batch

- `page_ident`

  identification of a page in a form

2.9 Contexts

A context is associated with a name space, typically negotiated with a customer.

The categories in a context define the way how this context is organized.

Examples for categories in a context are:

- `dsz.pzn`
- `dsz.date_format`
- `dsz.date_of_issue`

2.10 Generic API

The generic api is flexible and gets its specialization by a customer plugin. Every plugin has a set of required categories which need to be inquired in order to use the plugin.

2.11 API

The current API is oriented by a hierarchy of categories which is extensible.

The Task Manager provides the following interface for enquiring individual contexts, batches and forms:

`batch_list` lists the data bases of batches in a path

`form_list` lists the form identifiers in a data bases of batches in a path

`open_form` opens a form for processing

`form_last_editor` returns the ID of the last editor of a form

`form_state` returns the current state of the form

Possible states are:

- initial *
- processing/locked

- finished

CHAPTER 3

Installation und Konfiguration

Inhalt:

3.1 Sphinx Documentation Server

Nachdem das Projekt kopiert wurde, kann die Dokumentation einfach erstellt werden mit:

```
$ cd pydica/doc
$ python bootstrap.py
$ ./bin/buildout
```

1. Erstellen der Dokumentation:

   ```
   $ ./bin/sphinxbuilder
   ```

3.1.1 Weitere Informationen

- Sphinx documentation
- Sphinx reStructuredText Primer
- collective.recipe.sphinxbuilder
- reStructuredText - Markup Syntax and Parser Component of Docutils

3.2 Mercurial

3.2.1 Installation

Windows 7

1. mercurial-2.1.2-x86.msi installiert
2. `C:\Users\daniel\mercurial.ini` erzeugt und Extensions aktiviert

OS X

Stand: 24.02.2013

Unter OS X ist folgendes wichtig bzw. wird es ab jetzt:

- Python (2 oder 3) kann mit Homebrew installiert werden.

```
$ brew install python
$ brew install python3 # optional if you are a python3-wheenie
$ brew install qt
```

- Mercurial

  Installieren Sie es _nicht_ mit Homebrew, oder es wird mit virtualenv schief laufen. Statt dessen:

```
$ pip install mercurial
```

Alle anderen, hier nicht erwaehnten Module sollen mit *pip* installiert werden.

3.2.2 Konfiguration

Einträge in der Konfigurationsdatei (`~/.hgrc` unter Linux/MacOS, `%USERPROFILE%\Mercurial.ini` unter Windows):

```
[ui]
username = Vorname Nachname <Vorname@Nachname.de>
ssh = ssh -C

[extensions]
# manage different line endings between Linux/MacOS and Windows
eol =

# show progress bar for various operations
progress =

# hg glog shows a semi-graphic log
graphlog =

# colors the output, for instance of hg diff
color =

# handle large and/or binary files
largefiles =

[largefiles]
# hint: the first large file has to be added by hand with hg add --large filename and hg add --normal
minsize = 10
patterns = *.CDB *.RDB *.IBF *.jpg *.jpeg *.JPG *.TIF *.tiff *.bmp *.zip *.bz2 *.gz *.tgz *.bmp *.png
```

3.2.3 Projekt kopieren

```
$ hg clone https://bitbucket.org/pydica/pydica
```

3.2.4 Weitere Informationen

- Mercurial CACertificates

3.3 Client Validator

Note: Zur Installation von Mercurial siehe *Mercurial*.

3.3.1 Windows 7 64 Bit

1. python-2.7.3.msi installiert (32 Bit)
2. setuptools-0.6c11.win32-py2.7.exe installiert
3. PySide-1.1.0qt474.win32-py2.7.exe installiert
4. `$ python get-pip.py`
5. `$ pip install virtualenv`
6. `$ hg clone https://hg.veit-schiele.de/private/pydica`
7. `$ cd pydoca`
8. `$ virtualenv .`
9. `$ Scripts\active.bat`
10. `$ pip install pycparser configobj pytest mock`
11. die Verzeichnisse `C:\Python27\Lib\site-packages\PySide` und `C:\Python27\Lib\site-packages\PySide-1.1.0qt474-py2.7.egg-info` in das site-packages-Verzeichnis des virtualenv kopiert
12. die Datei `tiffcp.exe` ins Scripts-Verzeichnis des virtualenv kopiert
13. ausserhalb des virtualenv ausgefuehrt: `$ pip uninstall pyside`
14. im globalen site-packages noch alles gelöscht, was nach pyside aussieht
15. wieder innerhalb des virtualenv die Umgebungsvariable `QT_PLUGIN_PATH` auf den Pfad zum Plugin-Verzeichnis des pyside im virtualenv gesetzt
16. jetzt klappen sowohl die Tests als auch der Client-Validator

3.3.2 Mac OS X

1. `$ brew install qt`
2. `$ brew install python`
3. `$ pip install mercurial # do _not_ use the brew version -- wrong !!`
4. `$ brew install python3`
5. `$ brew install virtualenv`
6. `$ virtualenv -p python3`
7. `$ source bin/activate`

8. `(pydica) $ wget https://bitbucket.org/pydica/pyside-setup/downloads/PySide-1.1.2-py3.3`
9. `(pydica) $ easy_install PySide-1.1.2-py3.3.egg`
10. `(pydica) $ bin/pyside_postinstall.py -install`
11. `(pydica) $ pip install tiffany pytest six`

CHAPTER 4

Config Design

4.1 About Configuration

Configuration is a concept that has several different aspects. Here are some general thoughts, which are currently in preparation for the application.

- config of individual GUI objects

  Despite of general decisions about when, where and how to specify confinuration info, the configuration of certain objects needs to be done, regardless.

  Before driving any global decisions about the application, the following structure makes sense:

  - a GUI object has a config object, where its default options come from (font size, colors, font family, ...)
  - these settings do not need to be specified in the first place. Instead, every GUI entity initializes itself by using settings stored in some Python class. The settings can be supplied at object creation time. A default settings object should be defined in the same module. The GUI object should use this settings object as default, unless a different one is supplied by the invoking application.

- config of the whole application

  The application needs to know certain settings. While for many of the settings some defaults can be provided, certain things need to be defined, somewhere. Our approach is to use .ini files to make config data persistent.

  There are many different approaches, including configparser, ConfigObj, Qt's QSettings and others. We try to use a minimalistic compromize which does not rely on a certain feature of one implementation.

- storage of configuration

  We are using the ideas of Qt's QSettings objects, without being dependent of the Qt implementation. For the location of config files, the Qt documentation can be consulted. We use a similar heirarchy of .ini file lookups, stored in OS dependant default locations.

  We use a flat heirarchy of config sections this way to build trees: A config section may be named [default], or it has a special name syntax. There is no deeper nesting level than one.

  [to be continued]

CHAPTER 5

Notizen

Inhalt:

5.1 OS X Hints

Bei OS X mit QtSDK 4.8.0 funktioniert PySide nicht. Abhilfe:

```
sudo cp -pvR /Developer/QtSDK/Desktop/Qt/4.8.0/gcc/lib/*.framework /Library/Frameworks
```

5.1.1 GUI Design Notizen

Das Gui wird mit Qt Designer entworfen, solange dies effizient moeglich ist. Der Einstieg in den Designer ist etwas schwierig; der Designer ist teilweise problematisch zu handhaben.

Die meiste Dokumentation bezieht sich auf C++ - Code, und eine groessere Huerde ist, die Umsetzung in PySide zu finden.

Einen Einstieg bietet zum Beispiel

> http://developer.qt.nokia.com/doc/qt-4.8/gettingstartedqt.html

Zum Schreiben kleiner erster Programme eignet sich:

> http://zetcode.com/gui/pysidetutorial/firstprograms/

Eine entsprechende PySide-Dokumentation findet man unter

> http://developer.qt.nokia.com/wiki/PySideDocumentation/ http://developer.qt.nokia.com/wiki/PySide_Tutorials_by_Experience_I

5.1.2 Das Wichtigste im Kürze

Es gestaltet sich relativ einfach, eine Oberflaeche Zusammenzuklicken.

Layouts:

> Schwieriger wird es, wenn sich Widgets aneinander ausrichten sollen. Die groesste Huerde ist das Verstehen der Layouts. Layouts funktionieren nur dann richtig, wenn die Kette von innen nach aussen zusammenhaengt.

Es ist hilfreich, diese Dokument durchzuarbeiten:

> http://developer.qt.nokia.com/doc/qt-4.8/designer-layouts.html

5.1.3 Splitter:

Es dauert eine Weile, bis man versteht dass Splitter im Designer nicht direkt zu sehen sind. Stattdessen sind sie eine spezielle Art Layout.

Stretch-Faktoren:

Jedes Widget hat einen vertikalen und horizontalen Stretch. Das ist die Gewichtung der Widgets beim Layout. Sehr wichtiger Punkt: innerhalb eines Layouts schnurren widgets mit Stretch Faktor 0 auf das Minimum zusammen, wenn andere mehr als 0 haben.

5.1.4 Tips zu Layouts:

Wenn ein Widget in einem uebergeordneten Layout sitzt, sind die von diesem Layout kontrollierten Attribute gesperrt. Auch ist es oft unmoeglich, in diesem Zustand neue Widgets hinzuzufuegen oder deren Anordnung zu aendern.

Dazu gibt es als Kontext-Menue und auch als Tool Button oben “Break Layout”. Nach “Break Layout” kann es passieren, dass die Controls zu einem Punkt zusammenschnurren, das macht aber nichts da die Controls jetzt wieder bewegt werden koennen.

Beachte dass es beliebige Undos gibt. Es ist oft einfacher, eine Reihe von Aenderungen zu wiederholen, als es nachtraeglich zu korrigieren.

Es lohnt sich, etwas fummelid zu erstellende Layouts als Template zu speichern, um einen definierten Anfang zu haben.

5.1.5 Projektstruktur

Das GUI soll mit Designer bearbeitbar bleiben, aber im Python/PySide entwickelt werden. Damit der Designer weiterbenutzt werden kann, soll sein Output nicht direkt bearbeitet werden.

Konvention: Der Designer erzeugt name.ui Mit pyside-uic erzeugt man

pyside-uic -x -o name_ui.py name.ui

Man schreibt dann von Hand eine

name.py. (nicht zwingend, empfehlung)

Es gibt verschiedene Möglichkeiten, den Hybrid zu bauen.

Einfach und etwas “hackish”:

Zusätzliche Methoden definieren und an die Ui Klasse dran-definieren. Siehe das letzte Beispiel.

Bei grösseren Sachen:

Klasse ableiten, Methoden überschreiben.

Als einfaches Template kann das Hauptprogramm kopiert werden, welches pyside-uic durch “-x” generiert.

Davor importiert man das generierte Form und leitet ein neues ab.

Um das Zusammenspiel zw. QtDesigner und Python zu vereinfachen, folgender Vorschlag:

Benennung bei allen Widgets veraendern, auf die von Python Bezug genommen wird. Benennung in Englisch. Namen sind small_with_underscore

Widgets die bei der Erzeugung gelöscht werden sollen, haben eine Kennung tmp_ vornedran.

Dieser Punkt ist noch nicht ausgereift. Es scheint leicht zu sein, Strukturen zu kopieren, etwa als kleine Widget-Datei.

5.1.6 Design der dynamischen LineEdits

Qt hat bei allen Widgets ein Style-Sheet. Dieses konfiguriert das Aussehen der Widgets total und erlaubt zum Beispiel Ändern der Rahmenfarbe.

Beispiel zum Einstieg:

http://developer.qt.nokia.com/doc/qt-4.8/stylesheet-examples.html

Alles Weitere dazu:

http://developer.qt.nokia.com/doc/qt-4.8/stylesheet.html

Funktionierender Prototyp in PySide:

https://hg.veit-schiele.de/dsz/dsz/file/2407ebb6bf2f/src/designer_test

5.1.7 Neues Markierungsverfahren mit Stylesheets

Die Verwendung von Stylesheets ergibt eine bessere Moeglichkeit zur Markierung von einzelnen Widgets:

Jedem Widget kann ein Stylesheet zogeordnet werden. Auch wenn gar kein Style benutzt wird, eroeffnet dies beliebige Parametrisierung ueber Kommentare.

Beispiel eines Stylesheets mit Kommentar:

```
/* BEGIN tmp_line_input_1 */
QLineEdit {
     border: 2px solid green;
}
QLineEdit:focus {
     border: 2px solid lightgreen ;
        margin: 1px ;
}
QLineEdit:hover {
     border: 2px solid green ;
        margin: 1px ;
}
```

Zum Parsen dieses Strings greift man ueber das Codeobjekt zu, siehe css_test.py:

```
css_dict = {}

# find css arguments with a 'BEGIN' comment
for const in Ui_Form.setupUi.im_func.func_code.co_consts:
    if isinstance(const, str) and '/* BEGIN' in const:
        widget_name = const.split()[2]
        css_dict[widget_name] = const
```

5.2 Tools

Hier sind Merkberger zum Projekt. Weiss noch nicht wohin damit.

5.2.1 Allgemeiner Tip zum Kommando *locate* unter Mac OS X:

das Aktualisieren der Locate-Datenbank geht so:

sudo /usr/libexec/locate.updatedb

(aus http://superuser.com/questions/109590/whats-the-equivalent-of-linuxs-updatedb-command-for-the-mac)

5.2.2 Über Installation und __main__.py

http://sayspy.blogspot.de/2010/03/various-ways-of-distributing-python.html

http://docs.python.org/2/using/cmdline.html#interface-options

http://www.boredomandlaziness.org/2011/03/what-is-python-script.html

5.2.3 Wichtiger Hint zum Bauen von PySide:

cmake hat einen Bug in Version 2.8.10.1:

```
Linking CXX executable shiboken
ld: framework not found QtCore
clang: error: linker command failed with exit code 1 (use -v to see invocation)
make[2]: *** [generator/shiboken] Error 1
make[1]: *** [generator/CMakeFiles/shiboken.dir/all] Error 2
make: *** [all] Error 2
error: Error compiling shiboken
```

Dieser Fehler tritt auf, wenn man mit der aktuellen Version von *homebrew* arbeitet.

Auf die Lösung bin ich durch folgenden Link von Matthew Brett gestossen:

http://permalink.gmane.org/gmane.comp.programming.tools.cmake.user/44595

Der Fehler kann vermieden werden, wenn man die Version folgendermassen zurücksetzt:

```
cd /usr/local
brew versions cmake

  2.8.10.1 git checkout b5942ec Library/Formula/cmake.rb
  2.8.9    git checkout 54ff55c Library/Formula/cmake.rb
  2.8.10   git checkout d6d8b3e Library/Formula/cmake.rb
  ...

git checkout 54ff55c Library/Formula/cmake.rb
brew switch cmake 2.8.9

cmake --version

  cmake version 2.8.9
```

Der Hinweis stammt aus

http://stackoverflow.com/questions/3987683/homebrew-install-specific-version-of-formula

Hinweis mit 5 Steps.

Eine Versionsprüfung wird in meine neue Version des PySide - Installers eingebaut. Dieser Installer wird unter

https://bitbucket.org/pydica/pyside-setup

verfügbar.

5.2.4 Zu installierende Pakete

tools zum Testen:

```
pip install mock pytest
```

tools zum Parsen:

```
pip install pycparser
```

Dabei wird automatisch das phantastische *ply* Packet von David Beazley mitinstalliert. Um die Beispiele von *pycparser* auszuprobieren, sollte man dennoch das Quellpaket laden:

```
http://code.google.com/p/pycparser/
```

Das Quellpaket enthält auch den Praeprozessor *cpp.exe*.

Zur Installation von *pip* siehe

```
http://www.pip-installer.org/en/latest/installing.html#prerequisites
```

5.2.5 Noch nicht endgültig entschiede Pakete

Lesen der Ini-Dateien geht ganz gut mit dem eingebauten *ConfigParser*. Eine gute Alternative ist evtl. *ConfigObj* von Michael Foord.

```
pip install configobj
```

5.3 Projektplan

Dies sind Überlegungen aus dem Flug nach USA. Nicht mehr ganz aktuell und vermutlich etwas zu kompliziert gedacht.

5.3.1 Stichworte und Kommentare:

- CSS Struktur Abkehr von CSS-Tricks CSS - Modul mit eingebetteter Demo-App die das anzeigt Introspection-Tricks moeglich aber nicht notwendig -> weg damit
- Fields-INI Analyse ist notwendig Verifikation anhand vorhandener Daten - die einzelnen Felder muessen verifizierbar erkannt werden

Update 05.04.12: Struktur vereinfacht, Reduktion auf zunaechst das Noetigste.

5.3.2 Eingrenzung

Die Aufgabenstellung ist implizit schon klar, nicht klar ist der zur Umsetzung notwendige Aufwand. Um einen eindeutigen Einstieg zu haben, zunaechst Reduktion auf Attribute “LEFT, TOP, RIGHT, BOTTOM”. Diese Attribute sind erstmal genug um die Positionen zu finden, und es soll sich verifizieren lassen, dass die Sonderfaelle “0 1 1 0” korrekt aus Datei APO_FSRH.C rekonstruiert werden koennen. Erschwerend kommt hinzu, dass dazwischen ein Koordinaten-Wechsel stattfindet. Seiteneffekt: Die Korrektheit der TIFF-Abmessungen wird dadurch leider wichtig. Anmerkung: Die Messung ab linker unterer Ecke ist typisch durch Tiff entstanden. Offensichtlich wurde APO-FSRH.C bereits vor der Adoption von Tiff konzipiert.

FSEARCH.INI (siehe Ende von README.txt) scheint unklar zu sein bezueglich der Frage, was wofuer wie ausgewertet wird. Es ist unklar, ob die vorhandene Analyse ausreicht, oder ob zur endgueltigen Klaerung eine Quellcodeanalyse notwendig ist.

Ich schlage vor, den eingeschlagenen Heuristik-Weg fortzusetzen, solange die dabei ermittelten Ergebnisse plausibel sind. Sollten dabei Dinge unklar bleiben, schlage ich eine Totalanalyse vor, die aber aufwendig ist. Zunaechst verfolge ich den optimistischen Heuristik-Ansatz.

Update 05.04.12: Einfache, pragmatische Vorgehensweise

5.3.3 Ziele des Projektplans

Generell ist ein totaler Ersatz der existierenden Lösung gewünscht. Gleichzeitig sollen die vorhandenen MA vom Neuansatz ueberzeugt werden. Um moeglichst viel fruehzeitigen Feedback zu erlangen, sollen die ersten Schritte gleichzeitig zu - Extraktion und Validierung der Feldpositionen fuehren - vorhandene Scans mit richtiger Position dargestellt werden - das GUI ansprechend und korrekt die Felder darstellen.

5.3.4 Definition Cursorposition

Es gibt einen Cursor, der sich in einem Feld befindet. Das klingt zunaechst trivial, bedeutet aber mehr: Der Cursor ist virtuell, d.h. er befindet sich von der Logik her in einem Feld, um das es gerade geht. Physikalisch bedeutet es aber mehr, weil der Bildschirm gleichzeitig verschiedene Darstellungen besitzt.

- Im linken Fenster wird das aktuelle Feld so exakt wie moeglich durch zarte farbliche Kennzeichnung markiert. Editieren erfolgt dabei nicht. Bei Cursorbewegung wandert diese Markierung entsprechend, und die Darstellung wird geaendert.
- Im rechten Feld erscheint ein Eingabefenster, welches sich relativ nahe an der linken Darstellung orientiert, aber mit realen LineEdit-Controls realisiert ist. Diese Controls erlauben eine Texteingabe und sind der Darstellung im linken Bereich weitgehend, aber nicht zwingend angepasst. Hier wird eine entsprechende Rundung, Vereinfachung und Rasterisierung vorgenommen, sodass ein bestmoeglicher Kompromiss zwischen Darstellung und Eingabemoeglichkeit erzielt wird.

Update 05.04.12: Zunächst nur eine Cursorposition, siehe Tasten.txt

5.3.5 Vorschlag Arbeitspakete und Zeitschaetzung

- CSS-Modul Das bisher gewonnene Qt-Wissen schlaegt sich in einem eigenen CSS-Modul nieder, welches ohne Introspection-Tricks die Formatierung der verschiedenen Feldklassen definiert. CSS in der Applikation wird ignoriert und existiert nur zur Darstellung des Prototypen. Die relevanten Controls werden mit Hilfe der Koordinaten-Information und dem CSS-Modul dynamisch erzeugt.
- INI-Analyse erster Teil Die Ini-Datei wird ausgelesen und fehlende Werte aus APO-FSRH.C ergaenzt. Die Struktur wird zunaechst nur im Speicher erzeugt; solange diese Struktur nicht vollstaendig klar ist, halte ich extra-Dateien fuer verfrueht, zumal der Rechenaufwand gering ist.
- Anzeige der Felder im Original (z.Zt. Links) Die Felder werden durch die ermittelten Feldpositionen im Originalbild markiert, auch zur Validierung und als positives Feedback. Die Markierung ist nur zart und nicht stoerend. Die Anzeige erfolgt nur, wenn sich der Cursor im entsprechenden virtuellen Feld befindet.
- Editieren der Felder als Eingabe (z.Zt. Rechts) Gleichzeitig mit der zarten Felddarstellung (links) erfolgt eine Darstellung aller Felder zum Editieren (rechts), wobei ein Edit-Cursor zwischen den Feldern bewegt wird. Das aktuelle Feld zeigt dabei durch CSS seinen Editier-Zustand durch dass CSS-Modul an. Je nach Tastendruck folgt der Cursor dabei der definierten Reihenfolge. Durch Enter wird eine logisch durch die Felddefinition festgelegte

Reihenfolge eingehalten. Durch Cursor-Tasten (Auf/Ab) wird das raeumlich naechstgelegene Feld angesteuert. Durch Mausklich wird ein Feld direkt angewaehlt.

Anmerkungen 09.04.2012 von Daniel:

- Die Cursor-Tasten sollten nicht die räumlich naechstgelegenen Felder ansteuern, sondern genau wie Enter das vorhergehende bzw. nachfolgende logische Feld. Was das "räumlich naechstgelegende" Feld ist, waere naemlich erst noch zu definieren ist meiner Ansicht nach nicht so ganz trivial und wird bei uns erst einmal nicht gebraucht.
- Der Punkt "durch Mausklick wird ein Feld direkt angewaehlt" ist momentan sicherlich recht einfach umzusetzen, wenn dann aber spaeter mal die Logik des Validators ins Spiel kommt, ist die Aufgabe der freien Feldauswahl mit der Maus alles andere als trivial, so zumindest meine bisherigen Erfahrungen auf diesem Gebiet. Die freie Positionierung mit der Maus wird momentan zum Ersetzen des aktuellen Programms bei uns nicht dringend benoetigt.

Update 05.04.12: Doppelte Reihenfolge weggelassen, siehe Tasten.txt

- Abspeicherung bei Feldwechsel Der Zustand eines Feldes wird bei Feldwechsel grundsaetzlich gespeichert. Alle Aenderungen werden grundsaetzlich in einer Session festgehalten. Ungeklaert ist noch, wie lange Sessions existieren sollen und wie lange sie offengehalten werden. Vorschlag ist erstmal, dass man sich in einem noch zu definierenden Batch befindet, den man irgendwann als beendet "abschliesst". Gespeichert wird die Arbeit aber auf jeden Fall erst einmal.

  Update 05.04.12: Batch ist durch Daniel definiert, grundsaetzlich eine gescannte Gruppe.

- Wechsel zwischen Dateien Beim Verlassen des logisch letzten Feldes wird die naechste Datei geladen und angezeigt. Dies wird in der Statusanzeige entsprechend kenntlich gemacht. Das Laden und Speichern von Dateien geschieht implizit, der Benutzer "bewegt" sich frei innerhalb eines Batches, das noch zu definieren ist. Vorerst gibt es ein Batch als Menge von Dateien, als einfacher Ansatz.

  Update 05.04.12: Siehe Oben, sowie einfacher Datei-Ansatz, testen ob Cache noetig.

- Lesen von TIFF-Dateien Die zugrundeliegenden Tiff-Dateien sind leider etwas komplexer als erwartet und muessen verarbeitbar gemacht werden. Dazu muessen existierende Tools und Biobliotheken analysiert und auf Verwendbarkeit geprueft werden. Im Zweifelsfalle favorisiere ich den dynamischen Ansatz, also Zugriff ueber ein zu schreibendes tiff-Modul, solange der Rechenaufwand klein ist.

  Update 05.04.12: Benutzen von TiffCP, externer Prozess.

5.4 Migration zu Python 3.3

Voraussetzung war, dass es problemlos mit Homebrew geht.

Leider trat bei

    brew install python3

Das Problem auf, dass immer diese Meldung kam:

Undefined symbols for architecture i386:

"_libintl_bindtextdomain", referenced from: _PyIntl_bindtextdomain in libpython3.3m.a(_localemodule.o)

"_libintl_dcgettext", referenced from: _PyIntl_dcgettext in libpython3.3m.a(_localemodule.o)

"_libintl_dgettext", referenced from: _PyIntl_dgettext in libpython3.3m.a(_localemodule.o)

"_libintl_gettext", referenced from: _PyIntl_gettext in libpython3.3m.a(_localemodule.o)

"_libintl_setlocale", referenced from: _PyLocale_setlocale in libpython3.3m.a(_localemodule.o)

> **"_libintl_textdomain", referenced from:** _PyIntl_textdomain in libpython3.3m.a(_localemodule.o)
>
> ld: symbol(s) not found for architecture i386 clang: error: linker command failed with exit code 1 (use -v to see invocation) make: *** [Python.framework/Versions/3.3/Python] Error 1 ==> Configuration HOMEBREW_VERSION: 0.9.3 HEAD: b14a896325a6be555462f11f12466cbdf2959e43 CPU: 8-core 64-bit sandybridge OS X: 10.8.2-x86_64 Xcode: 4.5.2 CLT: 4.5.0.0.1.1249367152 X11: 2.7.4 => /opt/X11 ==> ENV CC: cc CXX: c++ MAKEFLAGS: -j8 CMAKE_PREFIX_PATH: /usr/local/opt/readline:/usr/local CMAKE_INCLUDE_PATH: /usr/include/libxml2:/System/Library/Frameworks/OpenGL.framework/Versions/Current/Headers/ CMAKE_LIBRARY_PATH: /System/Library/Frameworks/OpenGL.framework/Versions/Current/Libraries PKG_CONFIG_PATH: /usr/local/lib/pkgconfig:/usr/local/Library/ENV/pkgconfig/mountain_lion ACLOCAL_PATH: /usr/local/share/aclocal OBJC: cc PATH: /usr/local/Library/ENV/4.3:/usr/local/opt/pkg-config/bin:/usr/local/opt/sqlite/bin:/usr/local/opt/gdbm/bin:/usr/local/opt/python/bin:/usr/bin:/bin:/usr/sbin:/sbin
>
> Error: python3 did not build Logs: /Users/tismer/Library/Logs/Homebrew/python3/config.log

Auf die Lösung bin ich indirekt hier gestossen: http://mail.python.org/pipermail/python-dev/attachments/20100307/3fa5390b/attachment.html

Das Problem was gettext, was schon von irgendwo installiert war. Also:

> brew install gettext

Homebrew erzählt dann, dass das Paket schon da sei und es nicht gelinkt würde! Aha, also

> brew link gettext

Das hat sich einigermassen gesträubt, bis ich gettext weggeputzt hatte.

Danach kamen diverse Fehler durch Überreste einer vorherigen python3 Installation, aber das war alles schrittweise behebbar.

5.5 Abstract Preliminary Relational Interface Layer

This is a couple of random thoughts, crammed together after a fair bit of hacking on the CDB.

5.5.1 Database Abstraction

Databases are widely used. Everybody needs them, nobody likes them, and there will never be a perfect solution.

Instead of trying to solve this, we do an abstract solution by providing a way to describe a particular database by its properties in an unspecified abstract property language.

The goal is to describe the currenty existing simple database scheme with a few properties and to make the existing implementation abstract enough to implement one for a different database system.

As far as this current approach goes, it just tries to describe the relevant properties of the database. This should be re-iterated when we get more examples of databases.

We use a simple, preliminary notation of .INI files.:

```
[DEFAULTS]
active-db           : RDB/CDB demo
descriptive-section : [concrete APRIL]

[concrete APRIL]
customer            : Stein-Reichwald GmbH
vendor              : Walther GmbH
```

```
version             : unknown
dbtype              : RDB/CDB
meta-data           : [database meta-data]

[database meta-data]
type                : RDB/CDB
constraints         : [rdb-constraints]
db-properties       : [rdb-properties]
sys-interface       : [system interface]

[rdb-properties]
sql-like            : no
sql-interface       : no
dbase-like          : yes
dbase-interface     : no
object-like         : no
object-interface    : no

[system interface]
native-os           : windows
access-type         : [rdb-files]
network-layer       : novell netware
network-speed       : [novell-netware-speed]

[rdb-files]
files-per-rdb       : 2
file-path-1         : {ident}.CDB
file-path-2         : 00000001/{ident}.IBF
file-type-1         : rdb-form-batch
file-type-2         ; rdb-image-batch

[novell-netware-speed]
open                : fair
close               : fair
read                : slow
write               : slow

[rdb-form-batch]
batch-size          : varying
batch-header-size   : fixed
batch-layout        : [rdb-forms]

[rdb-image-batch]
batch-size          : varying
batch-header-size   : fixed
batch-layout        : [rdb-images]

[rdb-form]
form-header-size    : fixed
form-header-funcs   : (compute_form_header, compute-form-size)
form-size           : fixed
form-size-compute   : function(...)
form-attribute      : [rdb-form-field]

[rdb-form-field]
field-size          : fixed
field-count         : fixed
field-names         : fixed
```

```
field-properties   : [rdb-field-attribute]
field-encodings    : (cp850)

[rdb-field-attribute]
attribute-count    : fixed
attribute-type     : string, int32
attribute-size     : fixed(attribute-name)
encoding           : (cp850)

[rdb-image]
image-header-size  : fixed
image-header-funcs : (compute-image-header, compute-image-size)
image-size         : varying
image-size-compute : function(...)
image-attribute    : [rdb-form-field]
image-format       : fax G3/G4
```

CHAPTER 6

Links

6.1 Home

- pydica

6.2 mailing lists

- Developer's List
- Commit messages from Bitbucket

6.3 bitbucket

- pydica
- Tiffany

6.4 ohloh

- pydica
- Tiffany

CHAPTER 7

Index und Suche

- *genindex*
- *search*