Schnelles Prototyping für die medizinische Bildverarbeitung

Danial Bin Mohamed Saruji, Michael Müller, Hans-Peter Meinzer

Abteilung Medizinische und Biologische Informatik, Deutsches Krebsforschungszentrum (DKFZ)
d.saruji@dkfz-heidelberg.de

Kurzfassung. Prototypen spielen in der medizinischen Bildverarbeitung eine wichtige Rolle, da sie eine schnelle Evaluation eines neu entwickelten Algorithmus erlauben. Hierfür bietet sich die Nutzung einer interpretierten Sprache an, da die Entwicklungszeit, im Vergleich zu kompilierten Sprachen, kürzer ist. Um diese Möglichkeiten auch für C++ Bibliotheken zu erhalten, können Schnittstellen erzeugt werden mit denen der Zugriff auf Funktionen der Bibliothek, von einer interpretierten Umgebung aus, möglich wird. Es existieren verschiedene Verfahren um automatisch Schnittstellen für C/C++ Code zu generieren. In diesem Paper wird die Integration eines solchen Verfahrens examplarisch an dem Open-Source Toolkit MITK beschrieben.

1 Einleitung

Zeit spielt in der heutigen Softwareentwicklung eine wichtige Rolle. Aufgrund dessen kann es sinnvoll sein Prototypen zu erstellen, um schnell erste Ergebnisse zu erhalten. Dies bietet die Möglichkeit frühzeitig entscheiden zu können, ob sich der jeweilige Lösungsansatz eignet. Kompilierte Sprachen unterliegen dem Write-Compile-Run Zyklus, d.h. ein Programmierer muss den Quelltext, den er geändert hat, kompilieren, bevor er ihn ausführen kann. Bei größeren Projekten mit mehreren Modulen, die voneinander abhängen, kann dies einige Zeit in Anspruch nehmen. Bei interpretierten Sprachen ist dies nicht der Fall. Der Programmierer hat hier die Möglichkeit den Quelltext zur Laufzeit zu ändern. Mittels eines Interpreters werden die Instruktionen direkt in Maschinensprache übersetzt. Bei rechenintensiven Algorithmen aus der Bildverarbeitung kann dies ein Nachteil sein, da die Laufzeitgeschwindigkeit darunter leidet. Dies bringt den Schluss nahe, die Vorteile beider zu verbinden, also die kurze Entwicklungszeit der interpretierten Sprache mit der schnellen Ausführungszeit der kompilierten Sprache. Um Programmteile einer kompilierten Programmiersprache in einer interpretierten verwenden zu können, können sogenannte Wrapper verwendet werden. Wrapper bilden eine Schnittstelle und ermöglichen es Funktionen der kompilierten Sprache aufzurufen. Parameter können von der Zielsprache aus übergeben und Ergebnisse an diese zurück gegeben werden. Bei umfangreichen Bibliotheken ist die manuelle Erstellung von Wrappern sehr aufwendig. Hierfür stehen Schnittstellen-Generatoren zur Verfügung die automatisch Wrapper

Code generieren. In diesem Paper wird die Integration eines generischen Wrappingverfahrens, beispielhaft am Medical Image and Interaction Toolkit (MITK), beschrieben, um Teile der Klassen-Bibliothek skriptfähig zu machen und somit schnelles Prototyping zu ermöglichen. Dazu wurde ein automatisiertes Verfahren gesucht um die Möglichkeit zu bieten einfach und schnell Klassen skriptfähig zu machen. Als Zielsprache wurde hierbei zunächst die Skriptsprache Python [1] verwendet, jedoch sollte auch die Möglichkeit geboten werden andere Zielsprachen zu integrieren.

2 Material und Funktionen

2.1 Medical Image and Interaction Toolkit (MITK)

MITK [2] ist ein Open-Source Framework zur Entwicklung von interaktiver medizinischer Bildverarbeitungssoftware. Es baut auf dem Insight Segmentation and Registration Toolkit (ITK) und dem Visualizationtoolkit (VTK) auf und erweitert diese um weitere Funktionalitäten. ITK bietet Algorithmen für die Segmentierung und Registrierung von mehrdimensionalen Daten. VTK ist ein Visualisierungssystem und bietet eine große Zahl von Visualisierungsalgorithmen. Als Build System verwenden alle drei Toolkits CMake, welches ein plattformübergreifendes Buildsystem für C/C++ Projekte ist, um Projekt-Dateien für verschiedene Entwicklungsumgebungen zu erzeugen. Sowohl VTK als auch ITK verwenden ein generisches Wrappingverfahren um Wrapper für die Sprachen Python, Java und TCL zu generieren. VTK verwendet dabei ein vollständig selbst entwickeltes Verfahren unter der Verwendung von Yacc und Lex. ITK verwendet CSWIG, welches eine modifizierte Version des Entwicklungswerkzeugs Simplified Wrapper and Interface Generator (SWIG) ist, um Schnittstellen zu erzeugen.

2.2 Wrapper

Um die Funktionen aus der MITK Bibliothek in Python aufrufen zu können, wurden Wrapper erzeugt. Diese bilden Schnittstellen über die Funktionen aus einer anderen Programmiersprache aufgerufen werden können und enthalten die notwendigen Datentypkonvertierungen. Um Wrapper manuell zu erzeugen muss für jede Funktion, auf die Zugriff gewährt werden soll, eine Wrapper Funktion erstellt werden. Hierfür müssen die notwendigen Typkonvertierungen implementiert, mögliche Fehler abgefangen und teilweise Klassen/Strukturen erweitert werden. Zudem muss die Speicherverwaltung berücksichtigt und Zugriff auf Variablen, Strukturen und Klassen ermöglicht werden. Dies ist bei umfangreichen Bibliotheken sehr zeitaufwändig. Es existieren jedoch verschiedene Verfahren, um Python Schnittstellen automatisch für C/C++ Code zu generieren.

2.3 CableSWIG

Für die Erzeugung von Python Schnittstellen in MITK wurde CableSWIG [3] verwendet. CableSWIG ist eine Sammlung von Werkzeugen zur Erzeugung von

Wrappern, welche C/C++ Module für andere Sprachen verfügbar macht. Dabei werden 19 verschiedene Zielsprachen unterstützt. Es besteht aus drei Werkzeugen: CSWIG, CableIDX und GCCXML. In Abb. 1 wird das CableSwig Wrappingverfahren dargestellt. Die Cable Konfigurationsdatei definiert für welche Klassen Schnittstellen erzeugt werden sollen. Es kann für jede Klasse eine Cable Konfigurationsdatei erstellt werden, es ist aber auch möglich mehrere Klassen in einer zu definieren. Zudem ist es möglich mehrere Konfigurationsdateien in einem gemeinsamen Packet/Modul zu kombinieren. GCCXML ist ein Entwicklungstool das XML Dateien aus C++ Code generiert. CableIDX erzeugt aus der XML Beschreibung eine sogenannte Indexdatei, welche Informationen darüber enthält für welche Klassen Schnittstellen erzeugt werden sollen und in welchen Bibliotheken sich diese befinden. Die erstellten XML und Index Dateien werden als Input an CSWIG übergeben, der daraus die jeweiligen Wrapperklassen erstellt. Zusätzlich können noch SWIG Konfigurationsdateien übergeben werden mit denen die meisten Aspekte der Wrappergenerierung, wie z.B. individuelle Typkonvertierung, Klassen-/Strukturenerweiterung, Speicherverwaltung oder Exception Handling, individuell angepasst werden können.

3 Ergebnisse

3.1 Wrappen des MITK Core

Mit diesem Wrappingverfahren wurden Wrapperklassen für den Core Teil von MITK generiert. Dieser besteht aus Klassen die von (nahezu) allen MITK Applikationen benötigt werden. Es handelt sich dabei um Klassen für die Datenverwaltung, Synchronisation mehrerer Ansichten auf die gleichen Daten, Input und Output verschiedener Bildtypen und Interaktionen. Dabei sollte das Wrappingverfahren möglichst einfach in das CMake Build System integriert werden, damit auch Python Schnittstellen für neue Klassen auf schnelle und problemlose Weise generiert werden können (Abb. 2). Mit Hilfe von CableSwig wurde das MITK Core Erweiterungsmodul für Python erzeugt. Die normalen Python Module bestehen aus einer schlichten Textdatei mit Python-Programmanweisungen. Im Gegensatz dazu ist das Erweiterungsmodul ein Shared Object, bzw. eine DLL,

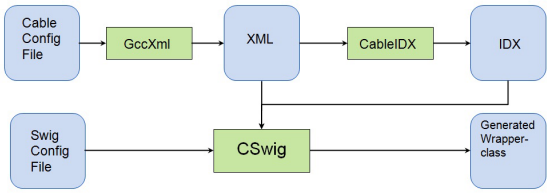


Abb. 1. CableSwig Wrappingverfahren.

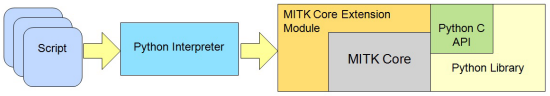


Abb. 2. MITK Core Wrapping.

die eine Initialisierungsfunktion für Python exportiert. Mit dem Modul besteht nun die Möglichkeit in Python Zugriff auf die MITK Core Klassen zu erhalten, wobei das Erweiterungsmodul als Schnittstelle dient und die notwendigen Wrapper für die Funktionen der Ausgangsbibliothek enthält. Dabei greift das Modul auf Funktionen der Python C Api zu, welche bei Python standardmäßig mitgeliefert wird. Diese API dient C/C++ Programmierern dazu, Erweiterungsmodule zu schreiben oder Python in ihre eigene Applikation einzubinden. Sie bietet beispielsweise Funktionen an, um Python Objekte in C Objekte zu konvertieren und auch umgekehrt. Der Entwickler kann nun die C++ Klassen, die sich im MITK Core befinden, in einer interpretierten Umgebung aufrufen. Dadurch kann er z.B. einen Algorithmus schnell und einfach implementieren und testen, ohne diesen vorher noch kompilieren zu müssen. Zudem existieren die C++-Klassen bereits als Maschinencode, wodurch die Laufzeitgeschwindigkeit nicht gemindert wird.

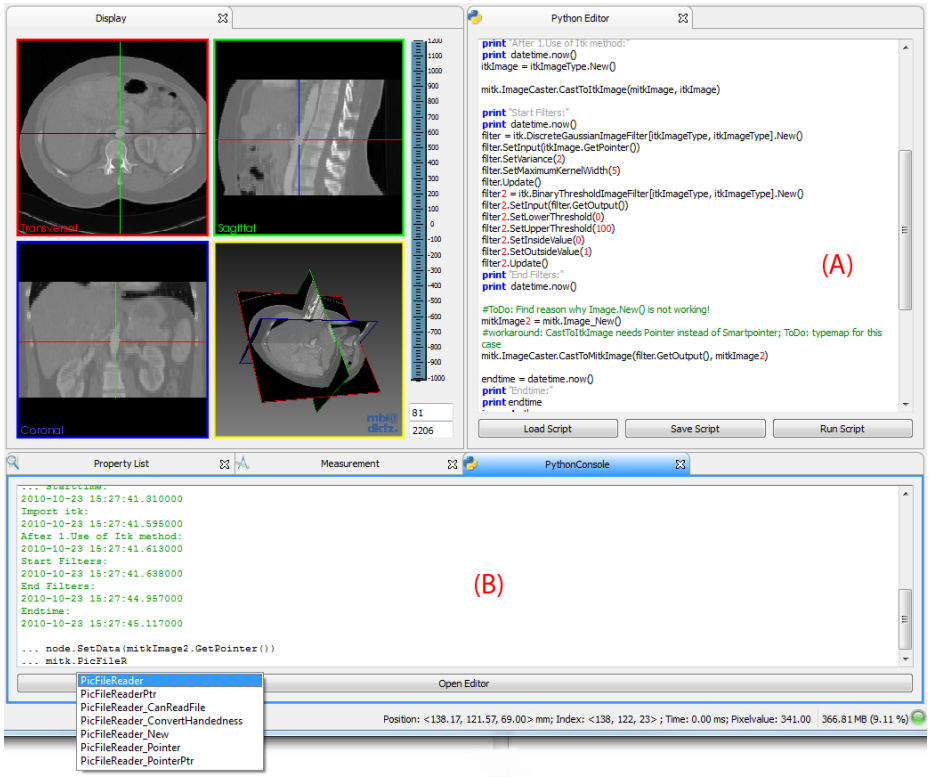
3.2 MITK Python Modul

Zur Entwicklung mit Python in MITK wurde ein MITK Python Modul implementiert, welches eine GUI mit Python Konsole (Abb. 3, A) und Skript Editor (Abb. 3, B) liefert. Die Konsole bietet Tab Completion und Texthighlighting. Mit dieser kann der Entwickler interaktiv arbeiten, d.h die eingegebenen Befehle werden sofort ausgeführt. Dies kann vor allem zum Testen von kleineren Abschnitten des Quelltextes nützlich sein. Mit dem Skripteditor ist es möglich Skripte zu erstellen, zu speichern und auch wieder zu laden. Der Editor bietet ebenfalls Texthighlighting an.

4 Diskussion

Durch die Integration des CableSwig Wrappingverfahrens in das MITK Build System ist es möglich einfach und schnell Erweiterungsmodule für Python zu generieren. Das Verfahren ist ein voll automatisierter Prozess, der einen großen Teil der Arbeit übernimmt. Es ist leicht erweiterbar, um weitere Zielsprachen zu nutzen. Durch das MITK Python Modul besteht nun die Möglichkeit interaktiv zu entwickeln, dies ist vor allem bei schnellem Prototyping von Vorteil. Es kann beispielsweise ein Bild geladen, verschiedene Filter daran ausprobiert und angezeigt werden, ohne C++ Code schreiben und kompilieren zu müssen. Die Laufzeit wird nicht gemindert, da die C++ Algorithmen bereits als Maschinencode vorhanden sind. Es existieren zudem umfangreiche Python Module aus den unterschiedlichsten Bereichen, auch aus der Bildverarbeitung, die der Entwickler einfach importieren und nutzen kann. Nachteilig an dem Wrappingverfahren ist, dass Template Klassen explizit instaziiert werden müssen. Daneben wird viel Code für die Wrapper generiert, was insbesondere bei Template Instanziierungen zu einer sehr großen Datei führen kann. Um die Entwicklung mit Python noch einfacher zu gestalten, könnte eine GUI zur grafischen Programmierung implementiert werden. Hierdurch könnte Quelltext anhand von strukturierten

Abb. 3. MITK Python Modul mit Skripteditor (A) und Konsole (B).



Modellen erzeugt werden. Auch zur testgetriebenen Entwicklung könnte Python genutzt werden. Die Python Standardbibliothek stellt zwei Module bereit mit denen Abschnitte des Programms durch automatisierte Testdurchläufe auf Fehler überprüft werden können.

Literaturverzeichnis

1. Sanner MF. Python: a programming language for software integration and development. J Mol Graph Model. 1999;17:57–61.
2. Wolf I, Vetter M, Wegner I, et al. The medical imaging interaction toolkit. Med Image Anal. 2005;9(6):594–604.
3. Lehmann G PZ, B R. WrapITK: enhanced languages support for the insight toolkit. Insight J. 2006; p. 4–35.