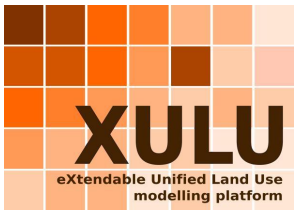
XULU – Eine JAVA-basierte Plattform zur Implementierung von Simulationsmodellen

Thamm, H.-P.; Bode, Th.; Schmitz, M. & A.B. Cremers



Zusammenfassung

Für viele Fragestellungen im Bereich der Entscheidungsunterstützung („decision support“) für nachhaltige Entwicklung ist es notwendig, Szenarien zukünftiger Landbedeckung und Landnutzung (LUC) räumlich explizit zu modellieren. Die von uns entwickelte *eXtenable Unified Land Use modelling platform* (XULU) erleichtert einerseits die Entwickung neuer Modellierungsalgorithmen, andererseits bietet sie durch ihre leicht bedienbaren und flexibel erweiterbaren Funktionalitäten die Möglichkeit, relativ problemlos auch alternative Modellierungsansätze auf einem gegebenen Datensatz zu evaluieren. XULU ist so aufgebaut, dass das modell-unabhängige XULU-Framework mit vielen Standard-Routinen (Datenverwaltung, I/O-Routinen, usw.) strikt von den spezifischen Modell-Implementierungen abgegrenzt ist und die meisten Funktionalitäten durch Plugins in das Framework integriert sind. Unterschiedliche Landnutzungsmodelle sind bereits implementiert, neue können leicht integriert werden. XULU wird erfolgreich für die LUCC[1]-Modellierung in Westafrika und Mitteleuropa eingesetzt und eignet sich auch sehr gut als „Decision Support System“ (DSS). Durch seine Leistungsfähigkeit, sowie die Flexibilität und Erweiterbarkeit, ist XULU ein interessantes Werkzeug für Anwendung, Forschung und Lehre.

1. Einleitung

Insbesondere der Bereich der Entscheidungsunterstützung für nachhaltige Entwicklung, wie z.B. die Erstellung von Landbedeckungs- und Landnutzungsplänen, Stadt- und Infrastrukturplanung, Abschätzung der Versorgungssicherheit und die Erkennung von Brennpunkten zukünftiger Entwicklung erfordert die räumlich explizite Modellierung zukünftiger Landbedeckungs- und Landnutzungsszenarien unter unterschiedlichen Randbedingungen. Aus diesem Grund ist auf dem relativ jungen Gebiet der LUC-Modellierung in den letzten Jahren ein sehr dynamischer Fortschritt zu verzeichnen. Dieser

wird durch immer leistungsfähigere Rechner und die wachsende Verfügbarkeit von räumlich expliziter Information durch Satellitendaten und raumbezogene Datenbanken verstärkt (GEIST & LAMBIN 2001, VERBURG ET.AL. 2002, BRIASSOULIS 2000, PARKER ET AL. 2003). Es wurden zahlreiche Modellansätze entwickelt und in entsprechenden Programmen umgesetzt, die jedoch sowohl die Modell-Anwender, als auch die Entwickler neuer Modelle oft vor grosse Probleme stellen. Meist sind die Programme sehr statisch aufgebaut sind und können nicht oder nur schlecht erweitert werden. Zudem mangelt es vielfach an nutzerfreundlicher Bedienung und guter Dokumentation, so dass diese Systeme nur von Experten und nach langer Einarbeitungszeit bedienbar sind. Unterschiedliche Datenstrukturen erfordern aufwändiges Umformatieren und Umrechnen, um die Ausgaben eines Modells als Eingaben für ein anderes Modell zu nutzen. Dies alles schränkt den allgemeinen Einsatz bestehender Modell-Anwendungen sehr stark ein (U.S. EPA 2000).

XULU wurde entwickelt, um diese unbefriedigende Situation sowohl für die Anwender, als auch die Entwicker neuer Modelle zu verbessern. Im folgenden Abschnitt 2 wird zunächst auf die Architektur von XULU hinsichtlich der Erweiterbarkeit eingegangen und anschließend in Abschnitt 2.3 XULU aus dem Blickwinkel des Anwenders betrachtet. Teil 3 zeigt abschließend auf, wie XULU in Zukunft weiterentwickelt werden soll.

2. Die Architekt von XULU

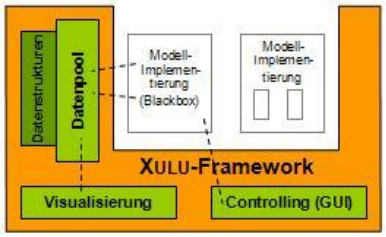
2.1. Trennung von Modell und allgemeinen Anwendungsfunktionen

Das zentrale Konzept von XULU besteht darin, dem Modell-Entwickler möglichst viel (Programmier-) Arbeit abzunehmen, bzw. zu ersparen. Um ein neues Modell oder eine neue Modellierungsidee auszuprobieren, sollte der Entwickler lediglich den Modellalgorithmus programmieren und nicht jedesmal eine vollständige Modell-Anwendung erstellen müssen. Aus diesem Grund erfolgt in XULU eine semantische Trennung zwischen dem generischen *XULU-Framework* und den individuellen *Modell-Implementierungen*[2] (Algorithmen).

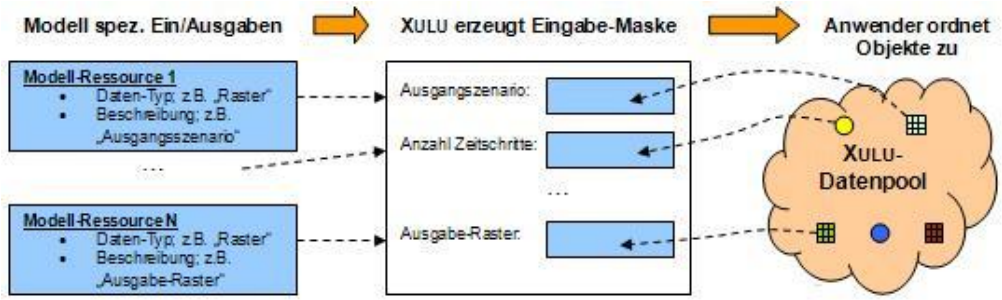
Das in JAVA 1.5 implementierte[3] XULU-Framework bildet einen autonomen Rahmen für verschiedenste Modelle. Daten-Verwaltung (Datenpool), Import- und Export-Routinen (kurz: I/O), Daten-Visualisierung und Modellsteuerung sind in einer generischen und modell-unabhängigen GUI-Anwendung realisiert. In diese können unterschiedlichste Modelle (auch gleichzeitig) integriert werden. Da alle Modelle denselben Datenpool

[1] LUCC = LUC change = **L**and **U**se land **C**over **C**hange
[2] im Folgenden auch kurz mit *Modell* bezeichnet
[3] und somit plattform-unabhängige!!

verwenden, wird der Datenaustausch zwischen den verschiedenen Modellen erheblich erleichtert.



Ein Modell wird in Form einer JAVA-Klasse implementiert[4] und als Plugin in das XULU-Framework eingebettet. Hierdurch steht die gesamte Mächtigkeit der Programmiersprache JAVA für die Implementierung von Modellen zur Verfügung. Aufgrund der Framework-Architektur reduziert sich der Implementierungsaufwand für ein Modell im Wesentlichen auf den Algorithmus. Selbst die zur Angabe der *Modell-Ressourcen* (Ein- und Ausgabedaten) benötigte GUI wird vom XULU-Framework automatisch bereitgestellt. In dieser ordnet der Anwender den Modell-Ressourcen individuell Datenobjekte aus dem Datenpool zu. Mit „einem Mausklick“ kann so zwischen alternativen Modell-Szenarien gewechselt werden, ohne dass aufwändig Konfigurationsdateien angepasst werden müssen.



Die Schnittstelle, über die ein Modell mit dem XULU-Framework interagiert, beschränkt sich auf einige wenige Funktionen, die die Art und Weise des Modell-Ablaufs jedoch in keinster Weise eingrenzen:

- Spezifizierung der für den Modell-Ablauf benötigten Modell-Ressourcen (Art der Daten-Objekte für die Modell-Ein/Ausgabe und temporäre Zwischenergebnisse)
- Modell-Initialisierung (Konsistenz-Prüfung der durch den Anwender zugeordneten Daten-Objekte)
- Starten des Modell-Ablaufs

Nach dem Start läuft ein Modell weitestgehend „autonom“ vom XULU-Framework ab. Ob und zu welchem Zeitpunkt bestimmte Daten visualisiert oder abgespeichert werden, bestimmt dabei nicht der Modell-Algorithmus, sondern der Anwender durch individuelle Steuerung des Frameworks. Das *Event-Konzept* von XULU sieht lediglich vor, dass das Modell an signifikanten, modell-spezifischen Punkten Ereignisse[5] auslöst (vgl. 2.3).

[4] JAVA-unerfahrene Entwickler unterstützt XULU durch einen Sourcecode-Generator (vgl. 2.2)
[5] z.B. „Zeitschritt i begonnen“ oder „Iterationsschritt j beendet“

2.2. Erweiterbarkeit von XULU

Da Datenverwaltung, I/O-Routinen und Visualisierung vollständig auf der Seite des Frameworks ablaufen und nicht Bestandteil eines Modells sind, können Erweiterungen und Optimierungen an XULU (z.B. speicherplatz-effizientere Datentypen, neue Dateiformate, Datenbankanbindung) zentral am Framework vorgenommen werden und sind automatisch für alle Modelle wirksam. Dabei ist das XULU-Framework sehr offen gestaltet. Nicht nur die Modelle, sondern auch die wesentlichen Komponenten, die das Anwendungsgebiet beeinflussen, werden in Form von Plugins in XULU eingebettet:

- Datentypen
- entsprechende Import/Export-Routinen (*Factorys*)
- Komponenten zur Daten-Visualisierung.

Neben der Unabhängigkeit von einem konkreten Modellalgorithmus oder Modelltyp ist XULU somit sogar unabhängig von einem konkreten Anwendungsgebiet. Zur Zeit stehen bereits Plugins zur Verfügung, mit denen LUCC-Modellierung betrieben wird: Räumliche Datentypen für Vektor- und Raster-Daten[6], I/O-Routinen für Shape-Files und verschiedene Raster-Dateiformate (z.B. ArcInfoAscii oder GeoTiff), sowie eine Layer-basierte Visualisierung von Karten[7]. Durch die Erweiterung mit entsprechenden Plugins kann XULU jedoch auch für gänzlich andere Anwendungsgebiete genutzt werden, ohne dass das XULU-Framework umprogrammiert werden müsste.

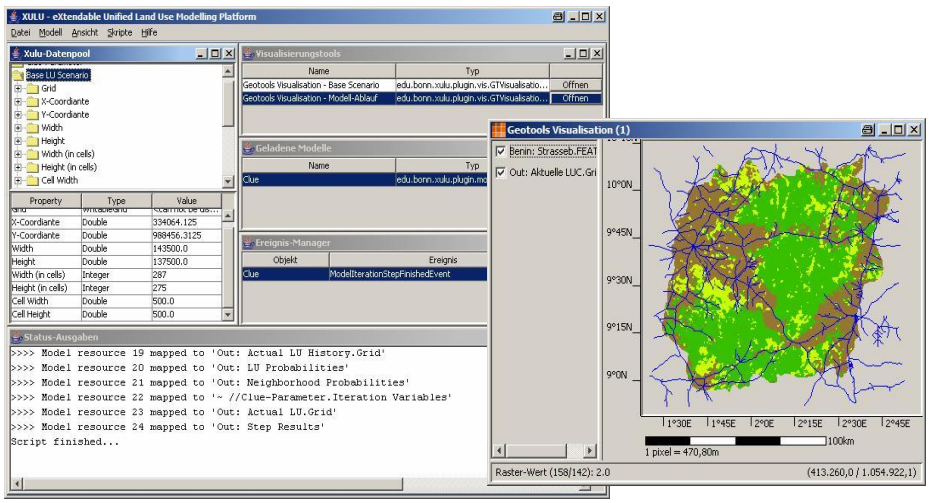
Darüberhinaus können auch vollkommen neue Funktionalitäten über die Plugin-Schnittstelle von XULU in das System integriert werden. So wurde beispielsweise ein spezieller **Quellcode-Generator** realisiert, der in JAVA ungeübte Entwickler bei der Implementierung neuer Modelle unterstützt. Dabei werden die für das Modell benötigten Ressourcen interaktiv über eine GUI spezifiziert. Der Quellcode-Generator erzeugt anschließend eine compile-fähige JAVA-Klasse, die das Grundgerüst der Schnittstelle zwischen Modell und Framework implementiert.

2.3. Arbeiten mit XULU

Nach dem Starten des XULU-Frameworks muss der Anwender zunächst die benötigten Daten in den Datenpool laden. Anschließend kann er ein oder mehrere Modell-Plugins öffnen und den Modell-Ressourcen konkrete Datenpool-Objekte zuordnen. XULU unterstützt den Anwender hierbei mit einer **Skript-Funktionalität**. Häufig wiederkehrende und zeitaufwändige Abläufe – wie z.B. bestimmte Daten in den Datenpool laden – können

[6] Für den sehr hauptspeicherplatz-kritischen Datentyp „Raster“ gibt es in XULU spezielle Implementierungsvarianten für kleine Datensätze, die permanent im Hauptspeicher verwaltet werden können, sowie für grosse (bzw. hochauflösende) Datensätze, die nur für den Zeitraum im Hauptspeicher gehalten werden, für den sie benötigt werden.

in Form eines Skripts (Makro-Datei) hinterlegt werden, das viele einzelne und „mühsame“ Aktionen ersetzt. Dies erleichtert das Arbeiten mit XULU erheblich. Während des Modellablaufs gestattet es XULU, Zwischenergebnisse visuell zu verfolgen. Dabei bestimmt nicht das Modell, sondern der Anwender, zu welchen Zeitpunkt und für welche Daten dies geschieht. Insbesondere kann der Anwender sog. *EventHandler* definieren (z.B. „Visualisierung von Objekt X aktualisieren“), mit denen der XULU-*EventManager* automatisch auf Modell-Ereignisse reagiert. Die folgende Abbildung zeigt eine konkrete Modellierungssituation für das IMPETUS-Untersuchungsgebiet des Oberen Ouémé in Benin/Westafrika. Zur Anwendung kommt der CLUE-S-Algorithmus (THAMM ET AL. 2005, VERBURG ET AL. 2002).



Um eine Vorprozessierung von Modell-Daten zu erleichtern, wurde für die Arbeit mit Raster-Daten das XULU-Plugin „**Raster-Calculator**“ entwickelt. Hiermit können arithmetische Funktionen (+, - , * , /, round, random, usw.) auf und zwischen Rastern (d.h. auf jeder einzelnen Zelle) ausgewertet werden. Auch das Anwenden von Filter-Matrizen ist möglich. Der Raster-Calculator ist so konzipiert, dass seine Funktionalitäten auch innerhalb einer Modell-Implementierung (Algorithmus) genutzt werden können..

3. Ausblick

Da die statistische Vorprozessierung von Eingabe-Daten für die Modellierung sehr zeitaufwändig ist, wird zurzeit daran gearbeitet, das freie Statistikprogramm R (www.r-project.org) in XULU einzubinden. Zum einen entfällt hierdurch, eine Umcodierung der Daten für exter-

[7] basierend auf der JAVA-Bibliothek GEOTOOLS 2 (http://www.geotools.org)

ne Programme vornehmen zu müssen. Zudem wird eine Kopplung unterschiedlicher Modelltypen vereinfacht. So könnte z.B. für jeden Zeitschritt zuerst mit agenten-basierten Modellen Siedlungswachstum simuliert werden, dann eine statistische Analyse erfolgen, die anschließend als Grundlage für statistisch-dynamische LUCC-Modellierung des gesamten Untersuchungsgebiets verwendet wird.

LUC-Modellierung stößt für größere oder hochaufgelöste Untersuchungsgebiete schnell an die Grenzen der zur Verfügung stehenden Rechner-Kapazitäten. Um auch für solche Gebiete überschaubare Rechenzeiten zu erhalten, wird aktuell eine parallelrechenfähige XULU Version entwickelt. Zudem laufen Arbeiten, um XULU als Monitoring System und Entscheidungsunterstützungssystem (DSS) einzusetzen, z.B. für ein effizientes Feuermanagement in West-Afrika. Desweiteren soll mit XULU eine Modellierung des CO_2-Umsatzes und der Nährstoffflüsse unter unterschiedlichen Feuermanagementoptionen erfolgen. Die Generierung von kombinierten Modellansätzen (statistisch-dynamisch und agent-based) ist ein Ziel für zukünftige Arbeiten. Dadurch können die Vorteile unterschiedlicher Modellierungsansätze vereint und komplexe Systeme leichter beschrieben werden. Weiterhin soll an einer lokalen (Geo-)Datenbank-Integration gearbeitet werden, über die auf die unterschliedlichen Eingabedaten zugegriffen werden kann.

4. Literatur

BRIASSOULIS (2000), Analysis of Land Use Change: Theoretical and Modeling Approaches. In: The Web Book of Regional Science (www.rri.wvu.edu/regscweb.htm).

GEIST H.J. & E.F. LAMBIN (2001): What Drives Tropical Deforestation? A meta-analysis of proximate and underlying causes of deforestation based on subnational case study evidence. LUCC Report Series No. 4, Louvain-la-Neuve.

PARKER, D.C.; MANSON, S. M.; JANSSEN, M. A.; HOFFMANN, M. J. & P. DEADMAN (2003), Multi-Agent Systems for the Simulation of Land-Use and Land-Cover Change: A Review. In: Annals of the Association of American Geographers 93 (2), S. 314-337.

SCHMITZ, M. (2005), XULU - Entwicklung einer generischen Plattform zur Implementierung von Simulationsmodellen am Beispiel der Landnutzungsmodellierung – Diplomarbeit am Institut für Informatik III der Universität Bonn.

Thamm H.-P.; Judex J. & G.Menz (2005): Modelling of Land-Use and Land-Cover Change(LUCC) in Western Africa unsing Remote Sensing

U.S. EPA (2000): Projecting Land-Use Change: A Summary of Models for Assessing the Effects of Community Growth and Change on Land-Use Patterns. EPA/600/R-00/098. U.S. Environmental Protection Agency, Cincinnati, OH. 260 pp.

VERBURG P., VELDKAMP A. & V. ESPALDON (2002), Modelling the Spatial Dynamics of Reginal Land Use: The CLUE-S Model. Springer-Verlag New York.