The STDM Development: Strategic Choices and Design Features

Danilo ANTONIO, John GITAU and Solomon NJOGU, Kenya

Key words: STDM, Security of Tenure, GLTN, GIS, Open Source Software

SUMMARY

Recent developments in Information and Communication Technology (ICT) have had a positive impact on the establishment of land administration and cadastral systems and geospatial data infrastructures (GSDI). These developments such as database management systems (DBMS), information system modeling standard UML (Unified Modeling Language), free and open source software (FOSS), positioning systems and emerging powerful computers and networking have greatly improved the quality, cost effectiveness and performance of land administration and recording systems (Lemmen and van Oosterom, 2002). However, there still exists a critical gap in the development of appropriate tools and approaches in providing tenure security to the majority of the people including the modeling of people-land relationships independently from the level of formalization, or legality of these relationships.

This paper describes the strategic design choices and processes in the development of the Social Tenure Domain Model (STDM), a pro-poor land information system that supports the continuum of land rights approach. STDM provides a land information management framework that integrates formal, informal and customary land systems, as well as the corresponding administrative and spatial components. By doing so, the model describes relationships between people and land in an unconventional manner, and as such it has the power to tackle land administration needs in communities, such as people in informal settlements and customary areas.

In particular, the paper highlights the development and implementation of STDM version 0.9.5 tool, which has been specifically customized to address land information requirements of the urban poor in the context of undertaking settlement upgrading initiatives.

The STDM Development: Strategic Choices and Design Features

Danilo ANTONIO, John GITAU and Solomon NJOGU, Kenya

1. BACKGROUND

Most developing countries have less than 30 percent cadastral coverage; this means that over 70 percent of the land in many countries is generally outside the land register (Lemmen, Augustinus, Haile and van Oosterom, 2009). This has caused enormous problems for example in cities, where over one billion people live in slums without proper water, sanitation, community facilities, security of tenure or quality of life. This has also caused problems for countries with regard to food security and rural land management issues.

The Global Land Tool Network (GLTN), facilitated by UN-Habitat, has taken up this challenge. GLTN partners has taken the lead in the development of the Social Tenure Domain Model (STDM), to address the identified technical gaps associated with securing tenure of urban and rural poor, amongst others.

STDM is meant specifically for developing countries where there is very little cadastral coverage in urban areas with slums, in rural customary areas or in complex situations like post crisis context areas. The focus of STDM is on all relationships between people and land, independently from the level of formalization, or legality of those relationships. This principle makes the STDM tool to be more flexible for other applications and contexts.

The development of STDM can be summarized in three distinct perspectives, these are:

- *STDM as a concept* - It provides a standard for representing flexible 'people-land' relationships. These relationships can be expressed in terms of persons (or groups of persons) having 'social tenure relationships' to spatial units, where a spatial unit can represent a parcel of land, building or a given natural resource such as a river or forest. Figure 1 represents the conceptual model of STDM.
- *STDM as a model* – It is a specialization of the ISO-approved Land Administration Domain Model (LADM) (ISO, 2012). Specialization means that there are some differences, which are mostly in the terminology and application area. Any form of right, responsibility or restriction in a formal system is considered as a social tenure relationship in STDM. Please see Table 1 for further explanation on the comparison between LADM and STDM.
- *STDM as an information tool* – This represents the implementation of the model as a software package that enables the recordation and visualization of the 'people-land' relationships and other information.

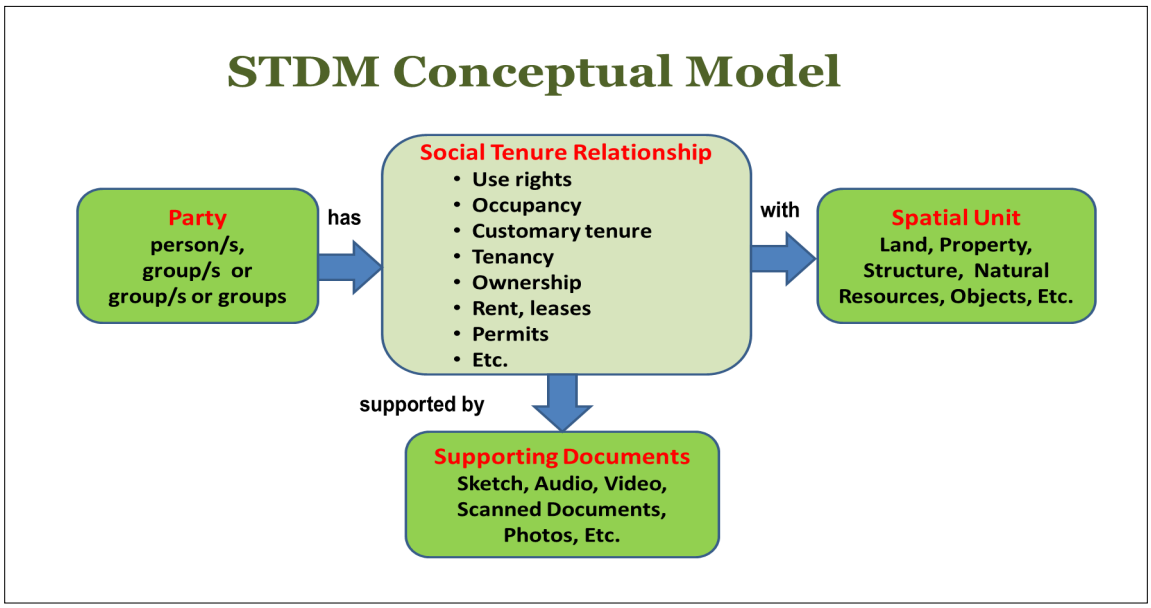

Figure 1: STDM Conceptual Model

LADM Class Name	Corresponding Class Name in STDM	Description
Party	*Similar name*	This is a person or organization that plays a role in a rights transaction. An organization can be a company, municipality, state, tribe, farmer cooperation or even church community where each organization can be represented by a delegate, director, chief.
SpatialUnit	*Similar name*	A single area (or multiple areas) of land and/or water, or a single volume (or multiple volumes) of space. Spatial units may be described in text (e.g. 'from this tree to that river'), or based on a single point, or represented as a set of unstructured lines.
RRR (Right, Responsibility, Restriction)	SocialTenureRelationship	This can be a: • Right - An action, activity or series of actions that a party may perform on or using an associated resource such as grazing, fishing or ownership. • Responsibility - Formal or informal obligation to do something such as to maintain a monument. • Restriction - Formal or informal obligation to refrain from doing

		something.
SpatialUnitGroup	AdminUnitSet	This is made up of one or more spatial units, or constitute a larger spatial unit group or can even be a combination of spatial units and spatial unit groups.

Table 1: Terminology differences between LADM and STDM

2. KEY PRINCIPLES AND STRATEGIC CHOICES

The need for efficient and effective land rights' recordation tools and use of IT has become a necessity as emphasized by (Steudler, Törhönen and Pieper, 2010), hence in selecting the right combination of tools for the STDM framework, it was important to ensure that these tools incorporated the latest and most relevant technologies and standards, while at the same time ensuring that they adhered to GLTN's core values (i.e. affordability, systematic large scale, gender responsive and pro-poor) and was based on free and open source software (FOSS) i.e. customizable such that there would be no acquisition costs, no license fees (purchase or maintenance), and no upgrade fees.

The second criteria was that the platform had to be GIS-based as one of the core entities of the STDM is 'spatial units' – these are geometry objects (or textual descriptions) that can either be in 2D or 3D representations (LADM, ISO 2012). The role of GIS as the core framework of the tool enables the linkage and visualization of the spatial units with regard to land tenure; land and property taxation; planning and management of utilities such as water, sewerage, electricity, telecommunication; and linkages to existing forms of land-use (Dale and McLaren, 2005).

Thirdly, the platform had to provide an extensible framework which enabled it to be customized using a multiplatform scripting language such as Java, Python, C++. The initial platform was Microsoft Windows but in future, the system should be easy to deploy in Macintosh and UNIX (multiple flavors) platforms.

The fourth was that the platform had to be scalable in terms of:

- supporting seamless integration of external generic or legacy systems such as web mapping applications or ERP systems using appropriate web services.
- enable terabyte-level storage of data without compromising on performance.

This capability enables the system to be used in a variety of situations from community level to national level. The organizational setups could range from standalone client-server configurations; dedicated server installations where multiple clients can access the database repository in LAN environment, which is the typical setup in small organizations; and finally, large national distributed cadastral databases.

With these principles in mind, the final release of STDM version 0.9.5 was deployed as Windows desktop software that was built on top of existing free and open-source geographic information systems (GIS). The software provides a framework for recording and visualizing social tenure relationships in the context of informal settlements in Uganda. It is clear that the framework can adopt to other context and situations.

The STDM version was piloted in Mbale Municipality in 2011-2012 in partnership with Shack/Slum Dwellers International, Cities Alliance and International Federation of Surveyors, national and local authorities and stakeholder groups.

3. DESIGN FEATURES AND IMPLEMENTATION

3.1. Application Architecture

Conventional land information systems have in common the need for a spatial data store that is responsible for recording and maintaining all cadastral and boundary information, and graphical editing tools for creating and updating boundary information in a map. In this regard, the implementation of STDM is based on the same principles for storing 'people-land' relationships as well as the corresponding social tenure relationships. It consists of three basic software components – a spatially-enabled data repository provided by PostgreSQL/PostGIS, Quantum GIS (QGIS) as the GIS client and the STDM plugin that is built on top of QGIS.

3.1.1. PostgreSQL/PostGIS Database Repository

The core of the STDM system is the data repository which stores all the data (both spatial and textual). It is very important that the data storage is reliable and safe. As is a basic requirement for any relational database management system to be able to manage large amounts of data while restricting unauthorized access to the information, textual land tenure data can be stored in PostgreSQL 8.4 but in order to be able to store and maintain spatial data representing spatial units (either as points, lines or polygons), PostGIS 1.5 spatial database engine extends PostgreSQL with this functionality.

Some of the reasons that make PostgreSQL appealing as the core data repository for STDM are: it provides the best option with regard to the total cost of ownership (TCO) when compared to other DBMS products (Web Commerce Group, 2002); it supports database replication for improved reliability; it enables database administrators (DBAs) to write custom functions using multiple languages inside the database such as Python, Ruby, R, C, V8 JavaScript; DBAs can incorporate NoSQL capabilities; there are short release cycles of approximately 15 months for new versions; it has the capability of connecting to external data systems as if they existed locally within the database using foreign data wrappers; and, its use by around 30% of tech companies globally for core applications such as Microsoft for Skype, Apple for Apple Remote Desktop 2, Fujitsu, etc.

PostgreSQL/PostGIS provides numerous geometry and topology functions for creating and maintaining spatial units in STDM. One of the strengths of PostgreSQL/PostGIS is that it has become the standard spatial database for all open source GIS packages (Ramsey, 2007).

The PostgreSQL installer comes with pgAdmin, a graphical user interface that caters for the needs of all users, ranging from writing simple SQL queries to developing complex databases.

3.1.2. The GIS Engine: Quantum GIS

Open-source GIS software has really taken off during the last couple of years and there is a full range of desktop GIS products that are useful for land information systems. New versions are coming out regularly and there have been major improvements in vector editing, database connections and spatial operations.

For implementing the GIS engine for STDM, GLTN selected Quantum GIS (QGIS) 1.8 as the core client software due to:

- Its capability in providing a rich feature set of vector editing functions for creating and maintaining spatial units (whether in point, line or polygon form), topology validation, native spatial database support, and support for common vector and raster data formats (through its utilization of GDAL/OGR library).
- Its growing community of users and developers, which means that there is more widespread usage; extended functionality and shorter release cycles of newer versions.
- The huge volume of QGIS support and tutorials that are easily accessible through various websites.
- Its built-in support of PostGIS layers stored in a PostgreSQL database. In QGIS, PostGIS layers provide cutting-edge and more accurate spatial capabilities such as spatial indexing, filtering and querying capabilities.
- Most importantly, it offers a sophisticated plugin architecture that supports customization using C++ or Python programming languages. This allows new features/functions to be easily added to the application. Many of the features in QGIS are actually implemented as core or external plugins. STDM client has been implemented as a QGIS plugin using Python programming language.
-

3.2. Interaction of STDM Plugin with PostgreSQL/PostGIS

QGIS comes with the Psycopg2 library, which is the most popular PostgreSQL database adapter for the Python programming language. Psycopg2 is small, efficient and secure, and supports Python data types which are automatically adapted to matching PostgreSQL data types.

The STDM QGIS plugin heavily utilizes Psycopg2 for carrying out CRUD (Create, Read, Update, Delete) operations of both spatial and textual records stored in the PostgreSQL/PostGIS database repository.

The STDM QGIS plugin uses the STDMPGProvider class, which provides the business layer methods for accessing the STDM PostgreSQL/PostGIS database. The code snippet below illustrates the implementation of deleting a spatial unit using the primary key of the spatial unit:

```
import psycopg2

class STDMPGProvider(object):
    '''
    STDM PostgreSQL/PostGIS data provider.
    '''

    def deleteSpatialUnit(self,gid):

        #Method for deleting for the spatial unit record

        conn = psycopg2.connect(self.connStr)
        cur = conn.cursor()
        cur.execute('DELETE FROM spatial_unit WHERE gid=%s',(gid,))
        conn.commit()
        cur.close()
        conn.close()
```

Where applicable, the methods in the STDMPGProvider class always pass parameters to SQL queries using the *%s* argument (as shown in the snippet above) to help prevent against an SQL injection attack. This can occur if the variable containing the data to be sent to the database comes from an untrusted source (e.g. a form published on a web site) an attacker could easily craft a malformed string, either gaining access to unauthorized data or performing destructive operations (such as table deletions) on the database (Cisco, 2014).

3.3. The Graphical User Interface

QGIS is built on the Qt framework – a cross platform and open source application framework that is widely used for building graphical user interfaces (GUIs) as well as non-GUI programs such as command-line tools and consoles for servers. Qt software is built using the C++ programming language, though other language bindings exists, specifically PyQt.

PyQt refers to the Python bindings that wrap the C++ Qt libraries, which means that now Python can be used to build Qt applications instead of learning C++. Hence, using the

PyQt wrappers to access QGIS libraries was a practical solution for implementing the STDM QGIS plugin because QGIS was already built on top of Qt libraries.

STDM plugin GUI controls were built using Qt Designer – a tool for designing and building GUIs from Qt widgets using drag-and-drop operations. Qt Designer use XML *.ui* files to store designs and does not generate any code itself. Below is a code snippet of Qt Designer XML structure for STDM's login window:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
            <class>frmLogin</class>
 <widget class="QDialog" name="frmLogin">
  …
  <property name="windowTitle">
   <string>STDM Login</string>
  </property>
  …
      <property name="text">
       <string>UserName</string>
      </property>
     …
     <widget class="QLineEdit" name="txtPassword">
      <property name="minimumSize">
       <size>
        <width>0</width>
        <height>30</height>
       </size>
      </property>
      <property name="echoMode">
       <enum>QLineEdit::Password</enum>
      </property>
            …
</ui>
```

In order to use the XML structure defining the GUI in PyQt, it needs to be converted to the corresponding Python code using pyuic4 - a Python script that compiles the QT Designer XML layouts into a Python module. The generated Python output file for the login window is as shown below:

```
-*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'ui_login.ui'
#
# Created: Tue Feb 18 17:45:33 2014
#      by: PyQt4 UI code generator 4.10.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
```

```
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
            def _translate(context, text, disambig):
                return QtGui.QApplication.translate(context, text, disambig, _encoding)
        except AttributeError:
            def _translate(context, text, disambig):
                return QtGui.QApplication.translate(context, text, disambig)

        class Ui_frmLogin(object):
            def setupUi(self, frmLogin):
                frmLogin.setObjectName(_fromUtf8("frmLogin"))
                frmLogin.resize(320, 208)
                frmLogin.setMaximumSize(QtCore.QSize(320, 16777215))
                …
                self.txtUserName = QtGui.QLineEdit(frmLogin)
                self.txtUserName.setMinimumSize(QtCore.QSize(0, 30))
                self.txtUserName.setObjectName(_fromUtf8("txtUserName"))
                …
                self.gridLayout.addWidget(self.label_2, 1, 0, 1, 1)
                self.txtPassword = QtGui.QLineEdit(frmLogin)
                self.txtPassword.setMinimumSize(QtCore.QSize(0, 30))
                self.txtPassword.setEchoMode(QtGui.QLineEdit.Password)
                self.txtPassword.setObjectName(_fromUtf8("txtPassword"))
                self.gridLayout.addWidget(self.txtPassword, 1, 1, 1, 1)
                self.gridLayout_2.addLayout(self.gridLayout, 1, 0, 1, 1)
                self.btnBox = QtGui.QDialogButtonBox(frmLogin)

        self.btnBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)
                self.btnBox.setObjectName(_fromUtf8("btnBox"))
                self.gridLayout_2.addWidget(self.btnBox, 2, 0, 1, 1)
                self.vlNotification = QtGui.QVBoxLayout()
                self.vlNotification.setObjectName(_fromUtf8("vlNotification"))
                self.gridLayout_2.addLayout(self.vlNotification, 0, 0, 1, 1)

                self.retranslateUi(frmLogin)
                …

            def retranslateUi(self, frmLogin):
                frmLogin.setWindowTitle(_translate("frmLogin", "STDM Login", None))
                self.label.setText(_translate("frmLogin", "UserName", None))
                self.label_2.setText(_translate("frmLogin", "Password", None))
```

4. FUNCTIONAL DESIGN

Like any other information system or software, STDM version 0.9.5 incorporated upgrades to the core functionality, ease of use in the installation process as well improved user experience of the tool, from its previous version 0.8. The prior process of gathering information regarding its feature set and capabilities was based on:

a. Internal peer review and recommendations by the GLTN team working directly on the tool.
b. Feedback received by participants during the training events on the use and application of STDM.
c. Assessment of the realities on the ground during the customization of STDM in different thematic areas. This was especially so if the functional requirement was common across different contexts.

Figure 2 below summarizes the main activities that helped identify its feature set and capabilities.

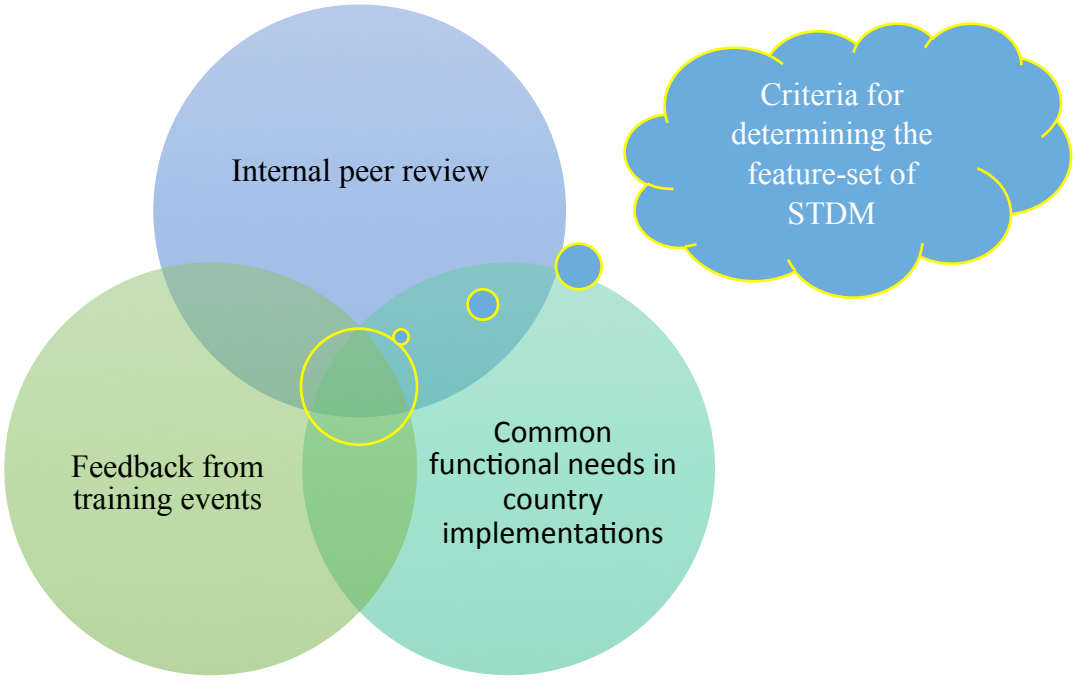

Figure 2: Activities used to determine functional requirements of STDM v0.9.5

4.1. The Feature List

Based on an evaluation of the software requirements resulting from Figure 2 above, the following main features have been incorporated in STDM version 0.9.5:

- CRUD operations of STDM data model entities: project area, household, spatial unit (represented as structures), person, social tenure relationship, enumerator, respondent.

- Importation of textual and spatial data from CSV and shapefile file formats respectively into the STDM database. One unique capability of this tool is that it provides source- and destination-column matching for specifying which values from given columns will be imported into the STDM database repository.
- Modular and logical arrangement of entity data management windows.
- Support for most file formats as documentary evidence for either spatial unit (e.g. photo of the structure), persons (e.g. photo of a person or government national identity card), household or social tenure relationship (photo or scan of lease agreement or tax receipt).
- Hierarchical archiving of the supporting documents in the folders and workspaces.
- Transferable workspaces between different projects.
- Basic reporting using a generic report builder.
- Generation of certificate of residency using an intuitive certificate composer module.
- Generic chart generation facility for generating plots and graphs.
- Standalone server or client installers for installation of server or client software components respectively.
- Deployable in a LAN environment for simultaneous multiuser connections.
- Support for 32- and 64-bit MS Windows (XP, Vista, 7, 8).
- Customizable post-installation user options through an intuitive wizard.

4.2. Module Design Using Unified Modelling Language

The design of the STDM plugin is based on object-oriented principles, which integrates seamlessly with the model driven architecture of STDM's data model. In order to simplify the design process, Unified Modelling Language (UML) has been used to document the different various perspectives of each module.

UML is a general-purpose modelling language used extensively for specifying constructing and documenting software systems. It includes specifications for fourteen different diagrams used to document various perspectives of a software solution from projection inception to installation and maintenance (Sparx, 2008).

Depending on the complexity of the design, each STDM module utilizes at least two of the UML diagrams, that is Use Case and Class diagrams that represent functional and static views respectively. The use of views in UML provides a means of organizing features and applications, thus making it easier to identify the right tool for the right job during the requirements and design stages of each module. See Figures 3 and 4 for illustrations.