



Deep Foundation Data Capabilities of the Data Interchange for Geotechnical and Geoenvironmental Specialists (DIGGS) Mark-up Language

Mark Styler

*Dept. of Civil and Coastal Engineering, University of Florida,
Gainesville, Florida, USA*

styler@ufl.edu

and

Marc Hoit

*Professor, Dept. of Civil and Coastal Engineering, University of Florida,
Gainesville, Florida, USA*

mhoit@ufl.edu

and

Mike McVay

*Professor, Dept. of Civil and Coastal Engineering, University of Florida,
Gainesville, Florida, USA*

mmcva@ce.ufl.edu

ABSTRACT

Data Interchange for Geotechnical and Geoenvironmental Specialists (DIGGS) is a data interchange format that will have global application and allow software vendors and users in the geotechnical community to seamlessly exchange data. The paper addresses the format being developed for driven pile and drilled shaft construction and testing data.

Keywords: data exchange; XML; foundations; piles; drilled shafts.

INTRODUCTION

Foundations constitute a significant and primary role in the performance and lifespan of bridges across the United States and the world. The costs involved in the design, construction, maintenance, and rehabilitation of substructures are the driving force behind the need for a universal data format. The goal of asset management through shared and universal data is a direction many industries are going.

A large amount of money is spent collecting necessary data for the design of a bridge foundation. This data consists of subsurface explorations, laboratory tests, and often foundation load tests. Through immediately available and used for design, the expense of this data justifies it being digitally archived. Future use of this data includes research, neighboring projects, and compiling detailed geological and geotechnical profiles.

Currently, formats are fragmented by both proprietary file formats and the boundaries of various governmental agencies. Programs can often store a fraction of the desired foundation data in non-interoperable binary or ASCII formats. Attempts to develop and adopt a format have been performed by the Florida Department of Transportation (FDOT) with the University of Florida, Consortium of Organizations for Strong-Motion Observation Systems (COSMOS) and Association of Geotechnical and Geoenvironmental Specialists in the United Kingdom (AGS) for some portions of the data. FDOT is currently maintaining an XML format, an internet based database, and various database interfaces that include both piling and the geotechnical information.

DIGGS OVERVIEW

The Data Interchange for Geotechnical and Geoenvironmental Specialists (DIGGS) Mark-up Language attempts to solve these pressing problems. DIGGS is being developed through the Transportation Pooled Fund Study (TPF 5(111)) coordinated by the Ohio Department of Transportation (DOT). The effort is a collaboration between Federal Highway Administration (FHWA), UK Highways, United States Environmental Protection Agency (US EPA), US Army Corps of Engineers, US Geological Survey (USGS), Eastern Federal Lands Highway Division (EFLHD) and the following state DOT's: California, Connecticut, Florida, Georgia, Kentucky, Minnesota, Missouri, Ohio, Tennessee. The initial release was developed by combining the existing standards developed by Association of Geotechnical and Geoenvironmental Specialists in the United Kingdom (AGS), University of Florida, Department of Civil Engineering (UF) and the Consortium of Organizations for Strong-Motion Observation Systems (COSMOS).

DIGGSML implements the eXtensible-Markup Language (XML) data format (<http://www.w3.org/XML/>). Generally, XML files will be significantly larger than equivalent representations of the same data in an ASCII file format or binary file. This is, however, recognized as one of the only disadvantages. With the current abundance of hard drive, broadband, and USB flash drive capacities this disadvantage is negligible. The advantages of XML over binary and ASCII formats are principally derived from the XML Schema Definition (XSD) files, application programming interfaces (APIs), and the overall popularity of XML as a data format for messages on the internet. XSD files define the rules for a valid instance of a specific type of an XML file. The DIGGS group has created XSD files governing DIGGS. Due to the popularity of XML many APIs have been developed to make the use of XML easier (Skonnard, 2000). DIGGS also implements the Geography Markup Language (GML) namespace, and can be classified as a GML application schema. GML provides the means by which DIGGS conveys geometries and features, non-tangible objects, dictionaries of

codelists, implementation of Coordinate Reference Systems (CRS), metadata, and measurements.

The transmission by electronic media of most of the data currently presented on forms such as Borehole Records, Trial Pit Records, in situ Test Data and Laboratory Test Summaries, is considered part of this first release. However, the transmission of **all** data, particularly from more complex testing, is accounted for but not covered explicitly by this document. For data not directly included in the standard, methods for attaching files and a table for including custom defined data are included. The standard also allows for custom additions to the standard and a way to share these customizations so others can interpret the data. The format of the transmission of large bodies of text and drawings, if required, is covered by other means. The DIGGS transfer format allows reference to these documents so that reports, drawings and photographs may also be transferred separately with recorded information about the file.

The intent of the initial release of DIGGS is to capture the commonly reported information that is most often required in project reporting. Special or additional information can always be reported in the remarks or description fields of tables if specific values or types are not available.

A key concept is that DIGGS is a transfer standard and NOT a database standard. The difference is crucial. The working database needs to be designed to best meet the needs of the users. A general purpose interchange standard will rarely, if ever, meet the usage requirements of any specific user group. Databases also can contain custom functions and business rules that may be important for the operation of a specific project or entity, but are not relevant in a transfer standard. Using an interchange structure as a working database can lead to awkward data entry procedures, difficulty in data validation, reduced querying capabilities, and loss of information.

An important feature of the DIGGS standard is the implementation of codelists or picklists. Almost all enumerated lists (picklists) are intentionally not defined within the XSD schema files. Codelists are external XML files that contain the valid options for codelists within valid DIGGS files. The benefit of using codelists is that it allows the end user or developer to essentially add to the DIGGS standard without the complications of extending the schema. This also increases the lifespan of versions of the DIGGS schema which encourages adoptions of the standard.

DIGGS FOUNDATION SCOPE

DIGGS follows a hierarchal design for the location of data. The root of this hierarchy is an object concerning the specifics of data transmission. This root object can contain multiple projects, each of these projects can in turn contain multiple foundation groups. For example, a foundation group could contain all of the piles within a pile cap, a bridge bent or pier, or it may contain all of the driven piles with attached PDA equipment.

In the first version of DIGGS DrivenPiles and CastShafts are implemented. A driven pile is constructed above ground and then driven into place. Conversely, a CastShaft is constructed in the ground. This is an important distinction, and there is some apparent confusion in the terminology used in the United Kingdom and

the United States (cast shafts are referred to as bored piles in the United Kingdom).

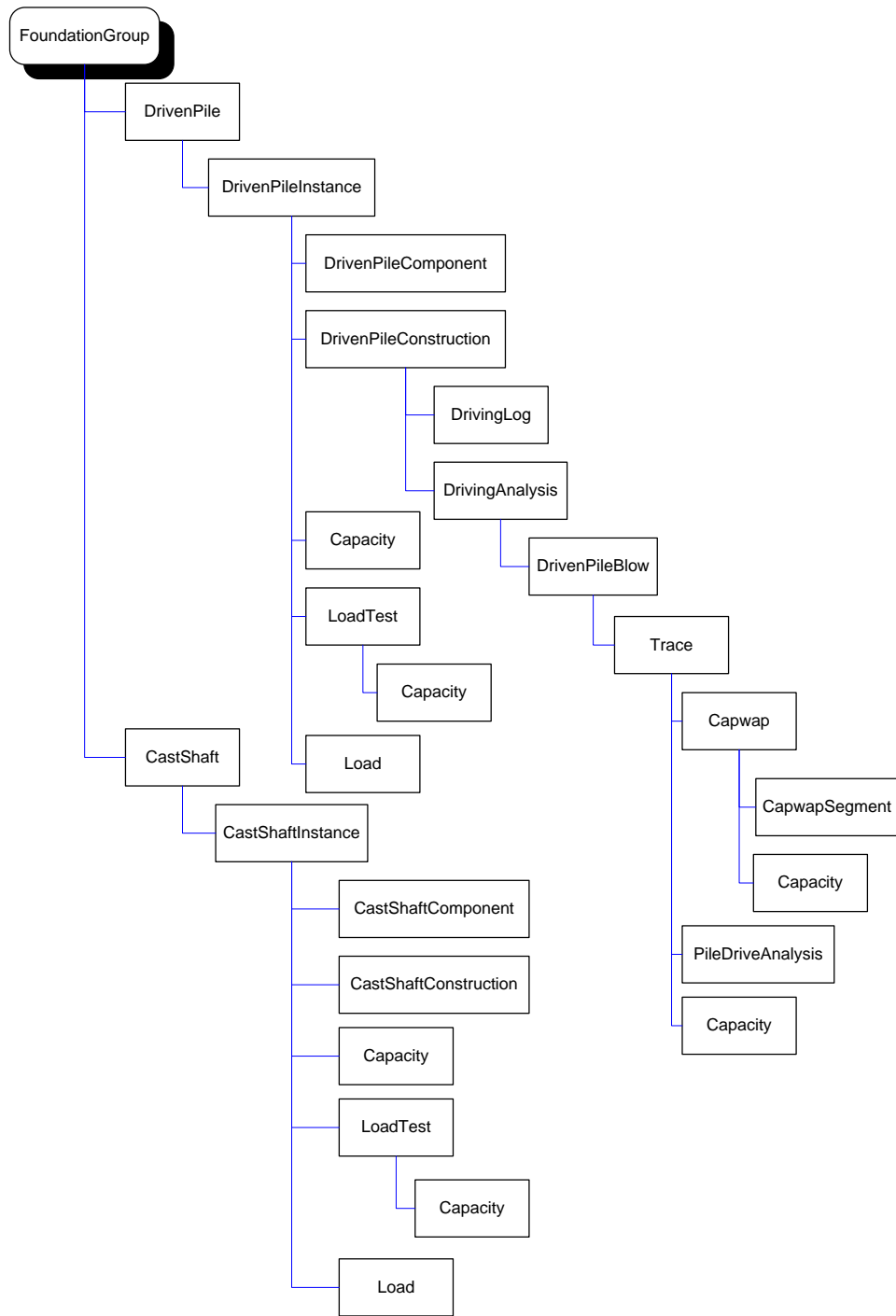


Figure 1. DIGGS foundation schema outline

Instances represent various stages of a foundations existence. At the most basic level, a pile will have a single design instance and a single as built instance. An

infinite number of instances can be created in order to contain sequential redesigns, damaged or current states, etc, while preserving previous models.

The capacity property is used to contain lateral and vertical capacities as well as predicted and measured capacities. Additionally, the capacity property also stores the method and type of capacity being represented. An instance can contain multiple capacity instances in order to contain various prediction methods as well as ultimate and design capacities.

The design of a deep foundation is represented with multiple components. Each component generally represents a single material. For example, a driven pile could be represented with three components, prestressed steel, reinforcing steel, and concrete. Components also contain upper and lower limits so that components can be defined that do not cover the range of the pile.

Arguably, the most important data covered by the foundation side of DIGGS after the foundation geometry and components is the load test. The load test property essentially contains a generic table. The intention of this load test table is to be able to contain the data concerning various types of load tests. Using the table analogy, each row of data in a load test can contain an elapsed time (from an arbitrary starting time), multiple load applications, multiple displacement measurements, multiple strain measurements, and multiple derivations of the load transferred. Each column of data is with respect to a certain depth along the pile. A basic static load test would not need to record the elapsed time, but would record the load applied at the top of the pile and the displacement measured at the top of the pile. To measure creep the elapsed time and displacement at the top of the pile can be recorded without increasing the applied load. If an Osterberg test is performed than multiple applied loads would be used to record pressures, each at the depth of the respective Osterberg cell. If a lateral load test is performed, multiple strain gauges with their depths and exact location can be used to calculate the bending moments.

DIGGS IMPLEMENTATION

The main purpose of creating the DIGGS standard was to create the means by which independent software developers can create interoperable programs. Most of the implementation rules have been devised with this goal in mind. The means by which software can become certified as DIGGS compatible will eventually be available on the website. In the meantime, the following rules should be followed:

1. Generated DIGGS is verified against the DIGGS schema files. No file should be passed off as a DIGGS file if it fails any part of the schema.
2. Codelist values are checked against the correct codelists.
3. All ID fields are generated UUIDs. It might not be necessary for a specific application that all the IDs be unique, however it is for DIGGS compliance.
4. The UUID fields are not editable by hand. To preserve the uniqueness they should not be modified after they have been generated.
5. UUID fields are preserved. If a DIGGS file is opened the UUID values for all unique DIGGS objects should not change. This rule can be

broken if the same object ends up with different UUIDs. See the UUID discussion later in this paper.

6. Speciality programs cannot lose unrelated DIGGS data. For example, if an SPT editing program opened up a DIGGS file that contained both SPT and deep foundation data, then the deep foundation data must still be in the file when it is saved. No DIGGS data can be lost when a program opens and saves a DIGGS file.

The DOM (Document Object Model) API can be used to easily verify DIGGSML instance files against the DIGGS schema files. However, the DOM cannot verify that codelists have been used correctly. Codelist verification must be programmatically implemented. This can be done by loading the DIGGS file and the codelist file into separate DOM objects. Each time a codelist is referenced by the DIGGS file the codelist DOM can be checked to see if the id exists.

The universal unique identifier is an integral part of the DIGGS standard. The advantages and disadvantages of its use are fully understood by the DIGGS committee. It is important that a developer be aware of the disadvantages in order to work through them.

The main disadvantage is the assigning of multiple UUIDs to the same DIGGS object. For example, in many cases a field boring will create a sample. This sample will end up having a UUID. When the boring log is created in DIGGS a UUID will be assigned. Now, the laboratory work may be started before or after the boring log is typed. It may also be done by another organization. From this, it seems extremely likely that the individual sample will end up having two UUIDs. One generated from the boring log and one generated from the laboratory. DIGGS will now recognize this as two different samples, even when they overlap in physical space. The merging of DIGGS files, either in a program or a database, should take this problem into consideration. Furthermore, it is recommended that if the DIGGS objects are merged then the obsolete UUID should be saved as an extra instance of the gml:name property.

It should be recognized that this problem stems from a disconnected paper trail. The use of the UUID is meant to encourage and allow the development of using the Internet to correlate DIGGS objects between organizations. The most desirable development would be assigning the UUID the moment the DIGGS object is physically created. In other words, the moment the sample is collected a UUID could be assigned then used for the lifetime of that sample. The moment a borehole is planned a UUID could be assigned and used throughout the subsurface investigation and when the resulting data is archived.

Another advantage of using UUIDs is that a DIGGS object can pass through multiple organizations, each with unique naming conventions, and still be a unique object. All of the internal names, state names, and laboratory sample names could be stored in the gml:name property while the gml:id maintained is the same constant UUID.

The following link is for a web-based UUID generator:

<http://kruithof.xs4all.nl/uuid/uuidgen>

UUID specifications:

<http://www.ietf.org/rfc/rfc4122.txt>

How to Use CoCreateGUID API to Generate a GUID with VB

<http://support.microsoft.com/default.aspx?scid=kb;EN-US;q176790>

The DOM API can be implemented by Visual Basic macros in Microsoft Excel. This allows Excel sheets to generate valid verifiable DIGGS transmission files with a familiar interface.

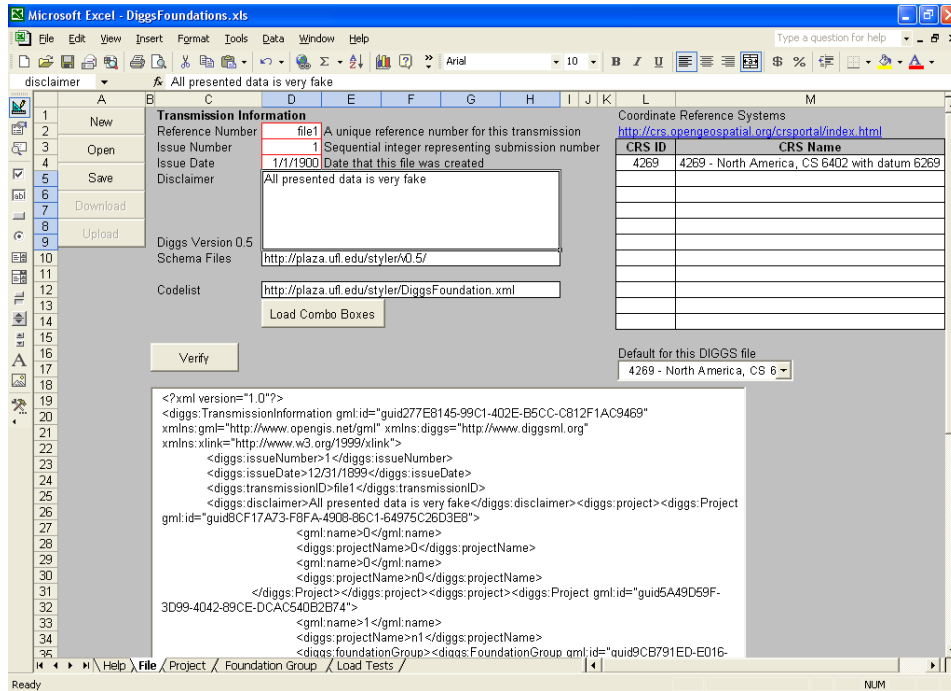


Figure 2. Sample DIGGS Microsoft Excel sheet

A sample Excel sheet (Figure 2) for deep foundations was created to store pile foundations and load tests. The combo box controls (enumerated lists) are filled out according to a code list. This results in the capability of the end user to modify the code list file in order to change the options presented in the combo boxes of the Excel sheet. Another feature of the program is that it clearly displays the resulting DIGGS file. In most cases the end user will not be interested in how the data is digitally represented, only that it is digital and that it is DIGGSML. However, for demonstrative purposes this program displays the generated DIGGSML and lets the end user directly edit it. It will not save the file if the edits make it invalid.

Load Test					
Project	Project Number	Bridge Name	Bridge Number	Pier Name	Pile Name
Sample Project	12345-6789	River Bridge	12345	Pier 7	Test Pile 1
Test Type	Static LT				
Test Date	8/21/2006				
Depth Datum	Ground Surface				
Load Test Data Table					
Depth	Elapsed Time	Applied Load	Displacement	Displacement	
Units	Seconds	kips	in	in	
		0	0	0	
		50	-0.0028	0	
		100	0.0022	0	
		150	0.0011	0	
		200	0.0027	0.00013	
		250	0.0116	0.00058	
		300	0.0134	0.00067	
		350	0.0237	0.00118	
		400	0.036	0.00180	
		450	0.0488	0.00243	
		500	0.0602	0.00300	
		550	0.0912	0.00455	
		600	0.1399	0.00698	
		650	0.2176	0.01085	
		700	0.3345	0.01668	
		750	0.5884	0.02934	
		800	0.7186	0.03583	
		850	0.733	0.03655	
		900	0.7418	0.03699	
		800	0.7562	0.03771	
		700	0.7142	0.03561	
		600	0.6852	0.03417	

Figure 3. Load test worksheet

At the very least a load test requires one column that is the applied load. The applied load is usually applied at the top of the pile, but generally that is not the case for Osterberg tests and lateral load tests. The above example (Figure 3) demonstrates displacements measurements at the top and bottom of a pile during a static load test. The units of the displacements are in inches, while the location of the measurement depends on the coordinate reference system that defined the ground surface elevation.

CONCLUSION

The goal of DIGGS is to provide a data interchange format that would have global application and allow software vendors and users in the geotechnical community to seamlessly exchange data. The initial release of DIGGS covers borehole logging data, geotechnical lab data, in situ testing data, site data (e.g. location, area, etc), driven pile and drilled shaft construction and testing data and some geophysical data.

The immediate challenge facing the DIGGS standard is the adoption by various software vendors and government agencies. This is being met by incorporating existing standards, working with groups interested in a long term solution, demonstrating the feasibility of the DIGGS standard, recognition of the DIGGS standard by ISO, and promoting the DIGGS standard to end users, vendors, and agencies. This paper specifically dealt with introducing the scope of DIGGS, promoting the adoption of DIGGS, and demonstrating the feasibility of specialized DIGGS applications.

An adopted international digital standard is long overdue, DIGGS is in a position to become the basis of the digital representation that will shape the industry for many years.

REFERENCES

- Skonnard, A., (2000), "SAX, the Simple API for XML." *Microsoft Developer Network Magazine*, April 2006.
<http://msdn.microsoft.com/msdnmag/issues/1100/xml/>.
- World Wide Web Consortium. (2006). "Extensible Markup Language (XML)." April 2006. <http://www.w3.org/XML/>.
- World Wide Web Consortium. (2005). "W3C Document Object Model." April 2006. <http://www.w3.org/DOM/>.
- World Wide Web Consortium. (2004). "XML Schema." April 2006.
<http://www.w3.org/XML/Schema> .
- UUID (GUID) Generator on the Web. (n.d.). April 2006.
<http://kruithof.xs4all.nl/uuid/uuidgen>.
- SAX (Simple API for XML). (n.d.). "The Official SAX Website." April 2006.
<http://www.saxproject.org/>.
- Microsoft Developer Network. (2006). "How Do I Use DOM?" April 2006.
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/332a15a2-430b-4c32-960b-d51cf2699018.asp>.
- Microsoft Help and Support. (2004). "How to Use CoCreateGUID API to Generate a GUID with VB." April 2006.
<http://support.microsoft.com/default.aspx?scid=kb;EN-US;q176790>.
- Leach, P., Mealling, M., and Salz, R. (2005). "A Universally Unique Identifier (UUID) URN Namespaces." *Network Working Group*. April 2006.
<http://www.ietf.org/rfc/rfc4122.txt>
- Lake, R., Burggraf, D., Trninic, M., and Rae, L. (2004). *Geography Mark-Up Language: Foundation for the Geo-Web*, John Wiley & Sons, West Sussex, England.
- Glados. (n.d.). "Glaodos CRS Client." April 2006.
<http://crs.opengis.org/crsportal/index.html>.
- Cox, S., Daisey, P., Lake, R., Portele, C., and Whiteside, A. (2004). "Open GIS Geography Markup Language Implementation Specification." *Open GIS Consortium*.