



Contact: **David Burggraf**
Suite 1300, 409 Granville Street
Vancouver, BC V6C 1T2
Canada

Phone: +1 (604) 484-2750

DIGGS V2.0.a Documentation

Date: June 30, 2012

Prepared for: DIGGS



Table of contents

Contents

Glossary of Terms and Abbreviations	9
1 Executive Summary	12
2 Introduction	21
2.1 Document Purpose	21
2.2 DIGGS Scope	21
2.3 DIGGSM Overview	22
2.3.1 Feature Model Overview	23
2.3.2 DIGGS Applications	26
2.4 DIGGSM Repository Location	26
3 DIGGS Revision History from V1.0a to V2.0a	27
3.1 DIGGS 1.0a	27
3.1.1 Independent Review of DIGGS 1.0a Schemas	27
3.2 DIGGS 1.1	28
3.2.1 Validaiton Performance Assessment	28
3.2.2 Complexity Assessment	29
3.2.3 Overview of Changes in V1.1	29
3.2.4 Specific XML Changes in V1.1	30
3.3 DIGGS 1.2	37
3.3.1 Summary of Changes in V1.2	37
3.4 DIGGS 2.0a	38
3.4.1 Summary of Changes in V2.0a	38
4 DIGGS 2.0a Structures and Organization	40
4.1 Naming and Identification	40
4.1.1 Authorities and Namespaces	40
4.1.2 DIGGS URN Identifier Structure	42
4.1.3 DIGGS Feature Identifiers and References	44
4.2 DIGGS Repository Organization	46
4.2.1 Official Schemas	46
4.2.2 Data	49
4.2.3 Dictionaries	50

4.2.4	CodeLists	56
5	DIGGS 2.0a Feature Model.....	59
5.1	Feature Properties and Attributes	60
5.1.1	Named Features and Inherited Properties.....	60
5.2	DIGGS Feature Classes and Objects	62
5.2.1	Projects	63
5.2.2	Sampling Features	66
5.2.3	Measurement	93
5.2.4	SamplingActivity.....	98
5.2.5	Sample	101
5.2.6	LayerSystem	105
5.2.7	Group	106
5.3	Feature Metadata	107
6	DIGGS 2.0a Schema Complexity Evaluation.....	113
6.1	Schema Load and Validate Performance.....	113
6.1.1	Results	114
6.2	Model Complexity assessment	115
6.2.1	Options Used for Oxygen V14.0 Random Instance Generation.....	115
6.2.2	Results	118
7	Specialized DIGGS Tools.....	122
7.1	Overview.....	122
7.1.1	DIGGS Excel Tool.....	122
7.1.2	DIGGS KML Tool	123
7.2	DIGGS Excel Tool Installation.....	124
7.2.1	DIGGS Excel Tool Operating Environment	124
7.3	Using the DIGGS Excel Tool.....	127
7.3.1	Convert a DIGGS GML File to an Excel Spreadsheet.....	127
7.3.2	View the DIGGS Data in Excel	128
7.3.3	Navigating the DIGGS Data Worksheets.....	138
7.3.4	Save the Converted Excel Spreadsheet.....	139
7.3.5	Clear the Excel Spreadsheet	140
7.3.6	Configuration Options	141
7.4	DIGGS KML Tool Installation	143
7.4.1	DIGGS KML Tool Operating Environment	143

7.5	Using the DIGGS KML Tool	146
7.5.1	Open the DIGGS KML Tool	146
7.5.2	Process a DIGGS File.....	147
7.5.3	View the KML File	149
7.5.4	Exit the Converter	149
7.5.5	Configuration Options	150
7.5.6	Defining Styling for the Output KML.....	151
8	Future Enhancements.....	165
9	Bibliography	166
Appendix A.	DIGGS URN Registration RFC to IANA	167
	Introduction	167
	Namespace ID	167
	Registration Information	167
	Declared registrant of the namespace.....	167
	Designated contact person	168
	Declaration of syntactic structure	168
	Relevant ancillary documentation	168
	Identifier uniqueness considerations	168
	Identifier persistence considerations	169
	Process of identifier assignment	169
	Process for identifier resolution	169
	Rules for Lexical Equivalence	169
	Conformance with URN Syntax.....	170
	Validation mechanism.....	170
	URN Scope.....	170
	Example.....	170
	Namespace Considerations	170
	Community Considerations	171
	Security Considerations	171
	Informative References	171
	Full Copyright Statement.....	171
	Intellectual Property	172
Appendix B.	GML 3.2 Practices Adopted by DIGGS.....	173

B.1	Coverage Encodings	173
	Domain Set.....	174
	Range Set.....	175
Appendix C.	GML 3.3 Extensions Adopted by DIGGS.....	180
C.1	Simple MultiPoint Encoding	180
C.2	Linear Referencing	180
	C.2.1 Linear Spatial Reference System	180
C.3	Linear Referencing Offset Vectors	182
	C.3.1 Target namespace	182
	C.3.2 Introduction	183
	C.3.3 Vector Offset Linear Spatial Reference System	183
Appendix D.	DIGGS Change Release Log.....	189
D.1	Version 1.1	189
D.1.1	Change Log 2010-05-18T14:30	189
D.2	Version 1.2a	189
D.2.1	Change Log 2010-05-20T12:25	189
D.3	Version 1.2.1	190
D.3.1	Change Log 2010-06-10	190
D.4	Version 1.2.2	193
D.4.1	Change Log 2010-07-01T13:30	193
D.5	Version 1.2.3a	194
D.5.1	Change Log 2010-07-06T11:31	194
D.6	Version 1.2.3.b	195
D.6.1	Change Log 2010-07-05T12:59	195
D.7	Version 1.2.4.a	196
D.7.1	Change Log 2010-07-07T11:20	196
D.8	Version 1.2.4.b	201
D.8.1	Change Log 2010-07-08T13:03	201
D.8.2	Change Log 2010-07-12T19:14	202
D.8.3	Change Log 2010-07-20T17:50	203
D.8.4	Change Log 2010-07-23T18:49	205
D.9	Version 1.2.4.c.....	207
D.9.1	Change Log 2010-07-28T17:38	207

D.10	Version 1.2.4.d	209
D.10.1	Change Log 2010-08-06T18:09	209
D.11	Version 1.2.4.e	210
D.11.1	Change Log 2010-08-09T13:08	210
D.11.2	Change Log 2010-08-11T12:21	210
D.11.3	Change Log 2010-08-12T12:25	210
D.11.4	Change Log 2010-08-12T16:10	211
D.11.5	Change Log 2010-08-12T16:16	211
D.11.6	Change Log 2010-08-16T11:12	212
D.12	Version 1.2.4.f	213
D.12.1	Change Log 2010-08-17T15:21	213
D.13	Version 1.2.4.g	213
D.13.1	Change Log 2010-08-26T14:30	213
D.13.2	Change Log 2010-08-26T14:36	214
D.14	Version 1.2.4.h	215
D.14.1	Change Log 2011-01-11T10:52	215
D.14.2	Change Log 2011-01-13T09:20	216
D.14.3	Change Log 2011-02-11T14:32	218
D.15	Version 1.2.4.j	219
D.15.1	Change Log 2011-05-12T22:36	219
D.15.2	Change Log 2011-05-12T09:15	220
D.15.3	Change Log 2011-05-19T07:15	220
D.16	Version 1.2.4.k	220
D.16.1	Change Log 2011-05-19T07:58	220
D.17	Version 2.0a	222
D.17.1	Change Log 2012-02-10T12:55	222
D.17.2	Change Log 2012-03-01T14:04	225
D.17.3	Change Log 2012-03-14T00:44	226
A	Appendix E. DIGGS Meeting Notes	227
E.1	Teleconference Meeting Notes 2010-01-06	227
E.2	Teleconference Meeting Notes 2010-01-12	228
E.3	Teleconference Meeting Notes 2010-01-14T07:30	230
E.4	Teleconference Meeting Notes 2010-01-14T08:30	231

E.5	Teleconference Meeting Notes 2010-01-28T07:30	232
E.6	Teleconference Meeting Notes 2010-01-28T09:00	234
E.7	Teleconference Meeting Notes 2010-02-04	236
E.8	Teleconference Meeting Notes 2010-02-11	237
E.9	Teleconference Meeting Notes 2010-02-18	239
E.10	Teleconference Meeting Notes 2010-02-25	243
E.11	Teleconference Meeting Notes 2010-03-04	246
E.12	Teleconference Meeting Notes 2010-03-16	248
E.13	Teleconference Meeting Notes 2010-03-18	249
E.14	Teleconference Meeting Notes 2010-03-25T07:30	251
E.15	Teleconference Meeting Notes 2010-03-25T10:30	256
E.16	Teleconference Meeting Notes 2010-04-01	262
E.17	Teleconference Meeting Notes 2010-04-08	263
E.18	Teleconference Meeting Notes 2010-04-15	266
E.19	Teleconference Meeting Notes 2010-04-22	268
E.20	Teleconference Meeting Notes 2010-04-29	268
E.21	Teleconference Meeting Notes 2010-05-13	269
E.22	Teleconference Meeting Notes 2010-05-20	270
E.23	Teleconference Meeting Notes 2010-05-27	271
E.24	Teleconference Meeting Notes 2010-06-03	272
E.25	Teleconference Meeting Notes 2010-06-10	272
E.26	Teleconference Meeting Notes 2010-06-24	282
E.27	Teleconference Meeting Notes 2010-07-01	282
E.28	Teleconference Meeting Notes 2010-07-06	287
E.29	Teleconference Meeting Notes 2010-07-08	290
E.30	Teleconference Meeting Notes 2010-07-29	295
E.31	Teleconference Meeting Notes 2010-08-05	296
E.32	Teleconference Meeting Notes 2010-08-11	298
E.33	Teleconference Meeting Notes 2010-08-18	299
E.34	Teleconference Meeting Notes 2010-08-26	302
E.35	Teleconference Meeting Notes 2010-09-16	304
E.36	Teleconference Meeting Notes 2010-09-30	305
E.37	Teleconference Meeting Notes 2010-10-07	307

E.38	Teleconference Meeting Notes 2010-11-03	309
E.39	Teleconference Meeting Notes 2010-12-15	310
E.40	Teleconference Meeting Notes 2010-12-09	321
E.41	Teleconference Meeting Notes 2011-01-06	323
E.42	Teleconference Meeting Notes 2011-01-28	334
E.43	Teleconference Meeting Notes 2011-02-03	335
E.44	Teleconference Meeting Notes 2011-02-09	335
E.45	Teleconference Meeting Notes 2011-02-17	338
E.46	Teleconference Meeting Notes 2011-03-24	340
E.47	Teleconference Meeting Notes 2011-04-01	342
E.48	Teleconference Meeting Notes 2011-04-07	343
E.49	Teleconference Meeting Notes 2011-05-05	344
E.50	Teleconference Meeting Notes 2011-05-19	347
E.51	Teleconference Meeting Notes 2011-11-02	349
E.52	Teleconference Meeting Notes 2012-01-24	354
E.53	Teleconference Meeting Notes 2012-02-03	356
E.54	Teleconference Meeting Notes 2012-02-10	359

Glossary of Terms and Abbreviations

The following lists the abbreviations, acronyms, and terms used in this document:

Term or Abbreviation	Description
AGS	Association of Geotechnical and Geoenvironmental Specialists (United Kingdom)
COTS Software	Custom Off The Shelf Software
CPT	Cone Penetration Test
DIGGS	Data Interchange for Geotechnical and GeoEnvironmental Specialists
DIGGSML	DIGGS Markup Language
DIGGSNA	DIGGS Naming Authority
ebRIM	electronic business Registry Information Model
Feature	Abstraction of real world phenomena
Feature Attribute or Property	Captures the defining characteristics of a feature
Feature Metadata	Contextual information about the defining Attributes/Properties of Features
FPS	OGC Feature Portrayal Service – a component WMS that retrieves its data from a WFS
GML	Geography Markup Language – A joint OGC and ISO 19136 standard
Href	Hypertext REference – An attribute used for linking in both the W3C's HTML and XLink standards whose value is the URI of the web resource pointed to
HTML	Hyper Text Markup Language – A W3C standard to describe web pages
HTTP	HyperText Tranfer Protocol – an application layer network protocol built on top of TCP. Web clients and servers communicate via HTTP request and response messages
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol – used with TCP, IP takes care of handling the actual delivery of the message data
ISO	International Organization for Standardization

Term or Abbreviation	Description
KML	An earth browser visualization and styling language originally created by Google, formerly known as Key Markup Language, now an international standard maintained by OGC.
NID	Namespace Identifier
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OMG	Object Management Group –a not-for-profit computer industry specifications consortium known for defining and maintaining the UML specification
RDF	Resource Description Framework –a semantic web standard by W3C
RIM	Registry Information Model
SI	Système International d'unités – International System of Units
TCP	Transmission Control Protocol – a set of rules (protocol) used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP takes care of keeping track of the individual data packets that a message is divided into for efficient routing through the Internet
UML	The Unified Modeling Language™ – OMG's most-used specification used for conceptual modeling of data structures and business processes
URI	Uniform Resource Identifier – A unique identifier for a resource, structured in conformance with IETF RFC 2396
URL	Uniform Resource Locator – A location dependent URI
URN	Uniform Resource Name – A location independent URI
W3C	World Wide Web (W3) Consortium
WFS	OGC Web Feature Service
WITSML	Wellsite Information Transfer Standard Markup Language – A petroleum industry standard for technical data transfer
WKT	Well-Known Text (WKT) – text based language by OGC/ISO for representing vector geometry, spatial reference systems, and transformations between spatial reference systems

Term or Abbreviation	Description
WMS	OGC Web Map Service
Xerces J	A well-known and rigorous validating XML Parser implemented in Java and developed by the the Apache Xerces™ open source project (xerces.apache.org).
XSD	XML Schema Definition
XML	Extensible Markup Language
XLink	XML LINKing Language – A W3C standard that is used to link XML documents
XPointer	XML Pointer Language – a language for locating data within an XML document
XMLNS	XML NameSpace
XSLT	Extensible Stylesheet Language Transformation

1 Executive Summary

The main objectives in the development of DIGGS V2.0a from V1.0a were to:

1. Bring DIGGS into compliance with Geography Markup Language (GML),
2. Address the schema complexity of DIGGS 1.0a and corresponding performance issues with XML software to remove the technical barriers to adoption,
3. Apply modeling best practices and improve the usability of DIGGS,
4. Update to the latest version of GML,
5. Enrich/refine the geotechnical and geoenvironmental domain model,
6. Improve/test implementability of DIGGS

Much of the technical developments toward the listed objectives were accomplished between Jan 2011 and June 2012. A brief description of the results for each objective is summarized in the following table:

DIGGS V2.0a Objective	Brief Summary of Results
GML conformance	<p>Passed all GML validation tests (see Section 3.2.3 for details)</p> <ul style="list-style-type: none"> • Inspection against GML 3.2 and 3.3 conformance clauses • Automated scan for conformance violations and syntactic conventions using Galdos GML Software Development Kit (SDK)
Address complexity and performance	<ul style="list-style-type: none"> • Several cases of infinite recursion were detected and removed from the schemas • A GML profile was created to eliminate/restrict unused GML elements and types • Dependency on WITSML was restricted to just the unit and value types
Modeling best practices and usability	<ul style="list-style-type: none"> • Defined and extensible and globally unique identification scheme in accordance with web identification practices for DIGGS data, codelists, dictionaries, and other relevant resources (see 4.1.2). • Used GML 3.2.1 Coverage model for table data in Tests. • Adopted OGC Observation and Measurement patterns for Monitoring and Test features
Update to latest GML version	<ul style="list-style-type: none"> • DIGGS 2.0a is now based on the Joint OGC (GML 3.2)/ISO (19136:2007) standard • DIGGS 2.0a also leverages selected new features of GML 3.3 <ul style="list-style-type: none"> ◦ Linear spatial referencing (extended in the GML 3.3 standard by 3D requirements from DIGGS) ◦ Compact Multipoint encoding
Enrich/refine domain model	<p>Restructured geotechnical and geoenvironmental elements and types:</p> <ul style="list-style-type: none"> • Made the Project feature mandatory in all DIGGS data and restricted the occurrence so that only one is permitted. All other features in the DIGGS file reference the single project • Test data structure align more closely with the OGC Observations and

	<p>Measurement model</p> <ul style="list-style-type: none"> Separated Sampling and Sampling Activity features to distinguish the physical sample from the activity that produces it Implemented Layer Systems using codelists to define constituents Implemented the GML coverage model for encoding CPT, geophysical, and similar data types
DIGGS implementability	<ul style="list-style-type: none"> Created two software implementations that support DIGGS 2.0a (Excel tool and KML tool). See user guides in Section 7. Created DIGGS CRS dictionary and coordinate transformation tool Created web sharable and machine readable codelists Created globally unique identification scheme registered with the Internet Assigned Numbers Authority (IANA)

After the conformance to the latest version of GML was completed, the complexity issue was solved by restructuring the schema and making use of profiles (subsets with restrictions) of both GML and WITSML. The use of the GML profile in V1.1 and further profile refinement and schema restructuring in V2.0a, resulted in a schema model of bounded complexity (with maximum possible data size less than 1 MB) as illustrated in Figure 1-1.

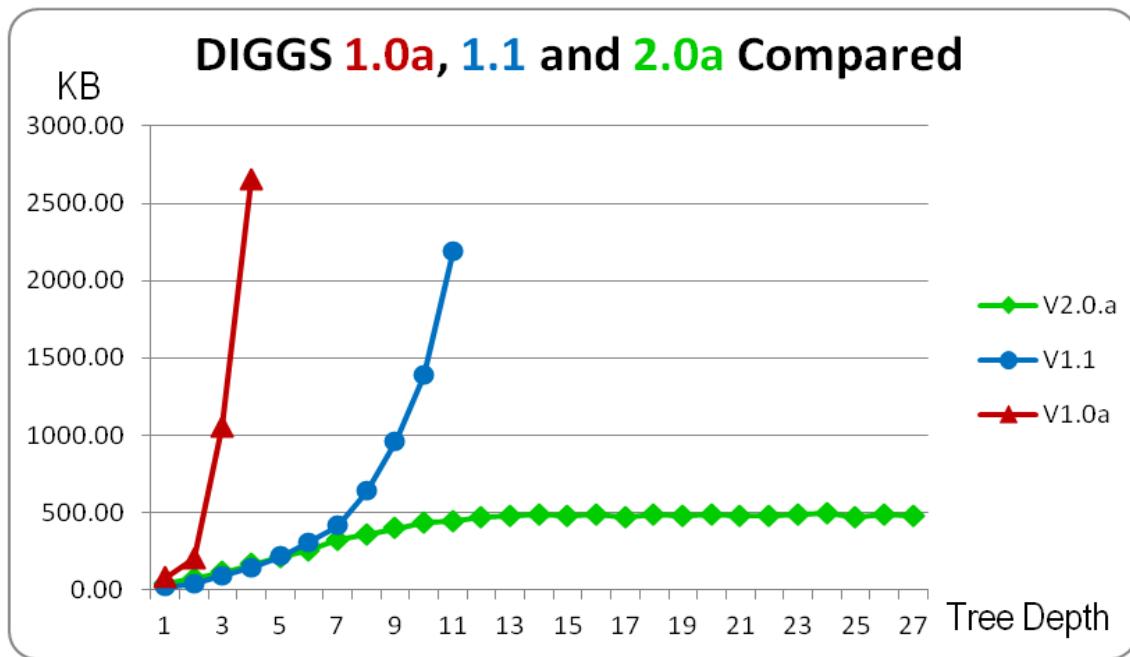


Figure 1-1: DIGGS dataset size (KB) over 27 different maximum tree depth settings

The complexity was analyzed through ‘stress-testing’ of the DIGGS schemas by generating random values for all possible elements/attributes, creating the most complicated DIGGS datasets that the schemas would allow (see Section 6.2 for the detailed methodology). The DIGGS dataset size would then grow as a function of recursion and nesting levels (tree depth) as tested over 27 increasing tree depth levels. Figure 1-1 shows V1.0a data increasing exponentially

without bound in size as compared with the V1.1 and V2.0a data using the same tests and software environment. The V1.1 data complexity was lower than V1.0a, but recursive loops still remained in the DIGGS schemas and hence the V1.1 data also grew exponentially without bound. In V2.0a, all recursion was removed from the schemas and the data size now has an upper bound (< 800 KB) with the average file size for trials approaching about 500KB. Note that the maximum recorded file size was 737KB out of several thousand randomly generated files.

The DIGGS V2.0a schema performance was also noticeably improved with an average schema loading time of 0.9 seconds (from 156 for V1.0a) and schema validation time of 0.5 seconds (from 87 seconds for V1.0a).

All of the DIGGS feature data, code lists and dictionaries were designed to be identified and retrieved over the web. Standard web practices were followed to define globally unique DIGGS identifiers called Uniform Resource Names (URNs), which were submitted for approval by the Internet Assigned Numbers Authority (IANA) and summarized in Table 1-2.

DIGGS URN Identification Structure				
urn:diggs:def:RESOURCE_CLASSIFICATION:AUTHORITY[:VERSION_NUMBER][:CODE][:SUB_CODE]				
Field Name	Description	Mandatory /Optional Indicator	Field Values	Comments
urn	Required prefix for all URNs	Mandatory	This field is fixed	
Diggs	Namespace Identifier	Mandatory	Diggs	Assigned by IANA and requested to be 'diggs'
Def	Indicates that the resource is a definition	Mandatory	Def	There are no other branches of this field to date, but new branches could be added (e.g. doc or spec)
RESOURCE_CLASSIFICATION	The type or classification of the resource.	Mandatory	codelist crs dictionary feature uom	'feature' is used to identify a feature instance Other RESOURCE_CLASSIFICATION values may be added as required. For example, if CRS components such as datums are added to the DIGGS CRS Dictionary, this scheme will be updated accordingly.
AUTHORITY	The maintaining authority of the resource	Mandatory	AGS Caltrans DIGGS EPSG OGC POSC SI USGS	Other AUTHORITY values may be added as required. Note: the concept of AUTHORITY is extended to cover dictionaries. For example, the EPSG (CRS dictionary) is considered an acceptable AUTHORITY for the CRS resource type (even though the Oil and Gas Producers (OGP) is the organization that approves and authorizes the EPSG CRS dictionary releases)
VERSION_NUMBER	Indicates the release or version number	Optional	String characters representing the version (e.g. 2.0a)	If VERSION_NUMBER is omitted but CODE is present, :: should be used between AUTHORITY and CODE. The VERSION_NUMBER shall be omitted when used as an identifier of resource types: dictionary, cnode (including featureType, attributeType, listedValue) or slot, in the versionless FDDs such as the NFCD.
CODE	The code or local identifier of the resource (e.g. gml:id value)	Optional	String characters representing the code (e.g. CRS20a)	For some codespace values the AUTHORITY alone is sufficient, in which case CODE can be omitted and VERSION_NUMBER may be omitted
SUB_CODE	The sub-code or local identifier of the resource (e.g. gml:id value)	Optional	String characters representing the subcode (e.g. 246tcp)	The sub-code is populated e.g. when the code corresponds to a dictionary or codelist that has sub-entries

Table 1-2: Tabular Summary of DIGGS URN Identification Structure

DIGGS feature instances carry references to other: features, objects, metadata, code lists, units, and CRS dictionaries, which can be retrieved, read and understood by software tools over the web.

The DIGGS Excel Tool was designed to present the text of DIGGSML data in a human readable spreadsheet format (Microsoft Excel). Note that the references to DIGGS identifiers are represented as linked cells in the spreadsheets. An example instance of a DIGGS data file is shown in spreadsheet format in Figure 1-3.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	bcd
5	Business Associate		#a23
6	Business Associate		#a24
7	Equipment		#equip-1
8	Equipment		#cone-1
9	Document Information		#d1
10	Ground		#g1
11	Project		#p1
12	Borehole		#LB_Webster
13	Trench Wall		#a22
14	Borehole		#cpt-1
15	Measurements		
16	Test		#d123
17	Test		#ps1
18	Test		#cpttest-1
19	Test		#gl1
20	Test		#spt-1
21	Test		#mc-1
22	Sampling Activity		#xyz
23	Sampling Activity		#zyx
24	Sampling Activity		#pointSample
25	Sample		#s321
26	Sample		#s123
27	Sample		#sampt
28	Layer System		#ls-1
29	Layer System		#ls2
30	Layer System		#ls1
31	Layer System		#fs1
32	Layer System		#lst_usc
33	Layer System		#lsp
34	Layer System		#lso1
35	Layer System		#sl-1
36	Sampling Feature Group		#lg1
37			

Diggs
BusinessAssociate
Address
Equipment

Figure 1-3: Example DIGGS Data Viewed in DIGGS Excel Tool

Both human readable (spreadsheets and web page tables) and machine readable (XML) encodings of CRS dictionaries and codelists were also created to support DIGGS data

documents. Such dictionaries and code lists can easily be made available in open standard formats over the web and can be viewed in web pages or retrieved programmatically. Example representations of such DIGGS resources are provided in the following paragraphs.

The DIGGS codelists can be viewed in spreadsheet format as shown in Figure 1-4.

1	Name of List	List URN	Code
7	Compaction Mould Type	urn:diggs:def:codelist:AGS:compaction_mould_type	Standard
8	Compaction Mould Type	urn:diggs:def:codelist:AGS:compaction_mould_type	CBR
9	Abundance	urn:diggs:def:codelist:BS5930:abundance	Little
10	Abundance	urn:diggs:def:codelist:BS5930:abundance	Some
11	Abundance	urn:diggs:def:codelist:BS5930:abundance	Trace
12	Abundance	urn:diggs:def:codelist:BS5930:abundance	Rare
13	Abundance	urn:diggs:def:codelist:BS5930:abundance	Occasional
14	Abundance	urn:diggs:def:codelist:BS5930:abundance	Much
15	Abundance	urn:diggs:def:codelist:BS5930:abundance	And
16	Driven Penetration Test Type	urn:diggs:def:codelist:AGS:driven_penetration_test_type	S
17	Driven Penetration Test Type	urn:diggs:def:codelist:AGS:driven_penetration_test_type	C
18	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Unknown
19	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Tangent method
20	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Secant method
21	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	246TCP
22	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	AG
23	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	DST
24	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	2MNAP

Figure 1-4: Example Code Lists Viewed as a Spreadsheet

Each of the DIGGS codelists can also be viewed as a web page, e.g. the chemical determinand code list excerpt is shown in Figure 1-5.

Chemical Determinand (ClassificationScheme)

AGS

id	urn:x-diggs:def:code-list:chemical_determinand
objectType	urn:oasis:names:tc:ebxml-regrep:ObjectTypeRegistryObject:ClassificationScheme
status	Submitted
isInternal	true
nodeType	urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode
xml view	Brief , Summary , Full
tree view	Children , Full Tree
associations	Associations To , Associations From
audit trail	All Audit Events

246TCP (ClassificationNode)

2,4,6 - Trichlorophenol

id	urn:x-diggs:def:code-list:chemical_determinand:id_246tcp
objectType	urn:oasis:names:tc:ebxml-regrep:ObjectTypeRegistryObject:ClassificationNode
status	Submitted
parent	urn:x-diggs:def:code-list:chemical_determinand
code	id_246TCP
path	/urn:x-diggs:def:code-list:chemical_determinand/id_246TCP
xml view	Brief , Summary , Full
tree view	Children , Full Tree
associations	Associations To , Associations From
audit trail	All Audit Events

Figure 1-5: Example 246TCP Code in Chemical Determinand Code List Viewed in a Web Browser

The DIGGS 3D Coordinate Reference System (CRS) dictionary also has human readable visualizations (spreadsheet and web page) and in addition was deployed as an example of an open standard web service in the DIGGS Demo 3D Coordinate Reference System Registry, shown in Figure 1-6.

<http://epsg.wrs.galdosinc.com/egpd/>

query by filter retrieve by code

Name: Click to choose
 Type: Compound CRS BBox: North Latitude West Longitude
 Name of the area of use (deg. deg.) South Latitude East Longitude Search
 Area: united kingdom Reset ?

DIGGS Geodetic Parameter Registry Version: DIGGS_7.11
 Welcome guest! | [login or register](#) | [help](#)

Search Results (1 - 50 of 146 possible results)
[Report all results](#) ? | [Report selected results](#) ? | Entities per page: 50 ▾
 <<first <prev | page 1 of 3 | [next](#) > last>

Report	Name	Code	Type	Status	Area Description	Remarks / Description
<input type="checkbox"/>	ED50 / TM 0 N + Belfast	DIGGS::23090_5732	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Douglas	DIGGS::23090_5750	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Fair Isle	DIGGS::23090_5741	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Flannan Isles	DIGGS::23090_5748	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Foula	DIGGS::23090_5743	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Lerwick	DIGGS::23090_5742	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Newlyn	DIGGS::23090_5701	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + Newlyn (Orkney Isles)	DIGGS::23090_5740	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + North Rona	DIGGS::23090_5745	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + St. Kilda	DIGGS::23090_5747	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view
<input type="checkbox"/>	ED50 / TM 0 N + St.	DIGGS::23090_5749	CompoundCRS	Valid	United Kingdom (UK) - offshore - North Sea.	view

Visit the [OGP Geomatics Committee](#) home page
 Developed by: [Galdos Systems Inc.](#)
 Version: 2.1.1

Figure 1-6: DIGGS 3D Compound CRS Definitions Retrieved from Open Standard Web Registry Service

All of the DIGGS codelists can be hosted by an open standard web registry service in a similar way as was done for the DIGGS CRS components illustrated in Figure 1-6.

The DIGGS KML Tool was designed to visualize 3D DIGGS data (primarily location, shape and identification information) in the popular and freely available earth browser (Google Earth). The DIGGS KML Tool supports coordinate transformations from DIGGS data to the OGC KML standard CRS (longitude, latitude, elevation). The DIGGS KML Tool was built upon the open source GeoTools spatial engine that has been extended to support all of the DIGGS 3D Compound CRS definitions. Linear referenced and planar referenced geometries are also supported in the DIGGS KML Tool. A sample screen shot of a planar referenced TrenchWall polygon is shown in Figure 1-7.

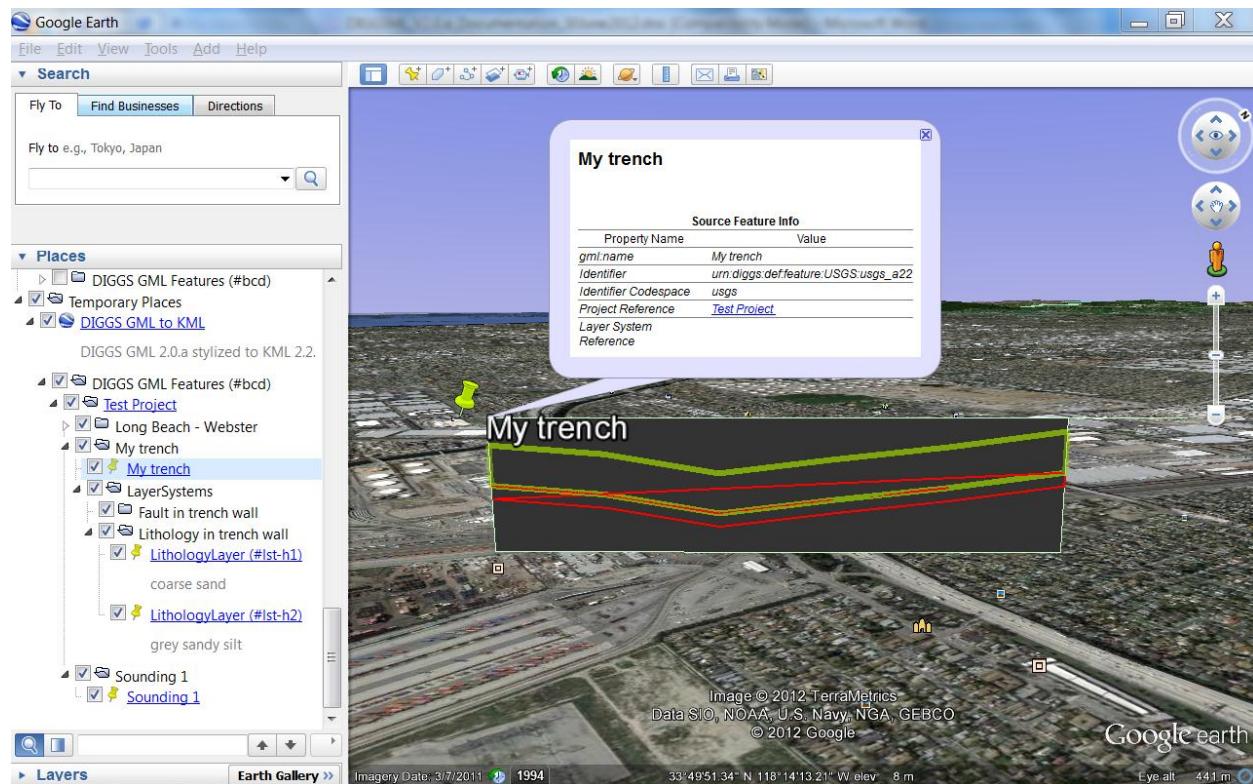


Figure 1-7: Representation of a Planar Referenced TrenchWall Polygon in the DIGGS KML Tool

2 Introduction

Data Interchange for Geotechnical and GeoEnvironmental Specialists (DIGGS) is a coalition of government agencies, universities and industry partners whose focus is on the creation and maintenance of an international data transfer standard for transportation related data. The coalition came into existence through coordination from the US Federal Highway Administration sponsoring meetings and eventually forming the pooled fund study project.

2.1 Document Purpose

This document contains user and developer guidance notes and accompanies the alpha release of the DIGGS Version 2.0a standard (DIGGS2.0a). The guidance notes focus on the changes and extensions made to the DIGGSML standard since the alpha release of the V1.0a schemas were published for public comment in 2008. In particular, the user guidance in this report describes what the DIGGS standard is and how it is used, including the applications available to process DIGGS. Several examples of DIGGS data instances are shown using screen shots of the DIGGS Excel Tool and DIGGS KML Tool throughout the document. XML editor tools (e.g. by Oxygen and Altova) have also been used to display schema diagrams and generate random test data. The development guidance herein describes the DIGGS schema model, the supporting data resources (e.g. dictionaries/codelists) and technical structure of the encodings including an assessment of the DIGGS schema parsing performance and complexity when represented as an object model.

2.2 DIGGS Scope

DIGGS 2.0a consists of a set of XML schemas, providing a common vocabulary and exchange model for geotechnical and geoenvironmental data constructs including boreholes, soil testing, site information and more. The DIGGS2.0a standard includes supporting information resources such as dictionaries, codelists and identifier names all encoded in machine readable XML that make use of IANA¹ registered DIGGS identifiers. For example the following information resources have been created and made available in the release of DIGGS 2.0a:

¹The Internet Assigned Numbers Authority (IANA) is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources

- 1) Coordinate Reference System (CRS) dictionaries containing Compound 3D CRSs that support DIGGS 3D data worldwide.
- 2) Units of Measurement (UoM) definitions that support typical measurements recorded by equipment used to capture DIGGS data.
- 3) Codelists that specify controlled vocabularies for test parameters, results, measurement phenomena, and other classifications typically recorded in DIGGS data.
- 4) Uniform Resource Name (URN) Structure and Governance Policies.

2.3 *DIGGSML Overview*

The DIGGS schemas are Open Geospatial Consortium (OGC) *Geography Markup Language (GML) application schemas* meaning that all schema constructs must derive from GML elements and types and follow GML's Object/property model, which govern how schema elements and XML instance documents are constructed. GML is an XML application that provides a grammar and base vocabulary for describing geographic data. GML was developed in order to provide a standard means of representing information about geospatial features – their properties, interrelationships, and so on. *Features* describe real world entities and are the fundamental objects in GML. Features can be concrete and tangible, such as boreholes and trench walls, or abstract and conceptual, such as projects and jurisdictional boundaries.

DIGGS/GML features are described in terms of their properties, which can represent spatial and temporal characteristics or associations with other features. For instance, DIGGS/GML can describe the location, shape, and extent of geographic objects as well as properties such as colour, speed, and density, some of which may depend on time. As it is impossible to describe all features for all application domains and predict their usage a priori, the GML core schemas do not fix definitions of specific instantiable feature types such as a trial pits or layer systems. Rather, specific features are defined in GML Application Schemas, which are created by user communities such as DIGGS.

GML provides a base of common geographic and geometric constructs (e.g. the AbstractFeature model, Points, LineStrings, and Polygons) that can be shared and reused by GML Application Schemas. In turn, the GML constructs are built upon XML constructs such as elements, attributes, types, datatypes (e.g. integers, strings, dates), international language support, etc. By building on upon successful existing web technologies, the DIGGS GML Application Schemas can leverage a whole world of GML and XML Tools.

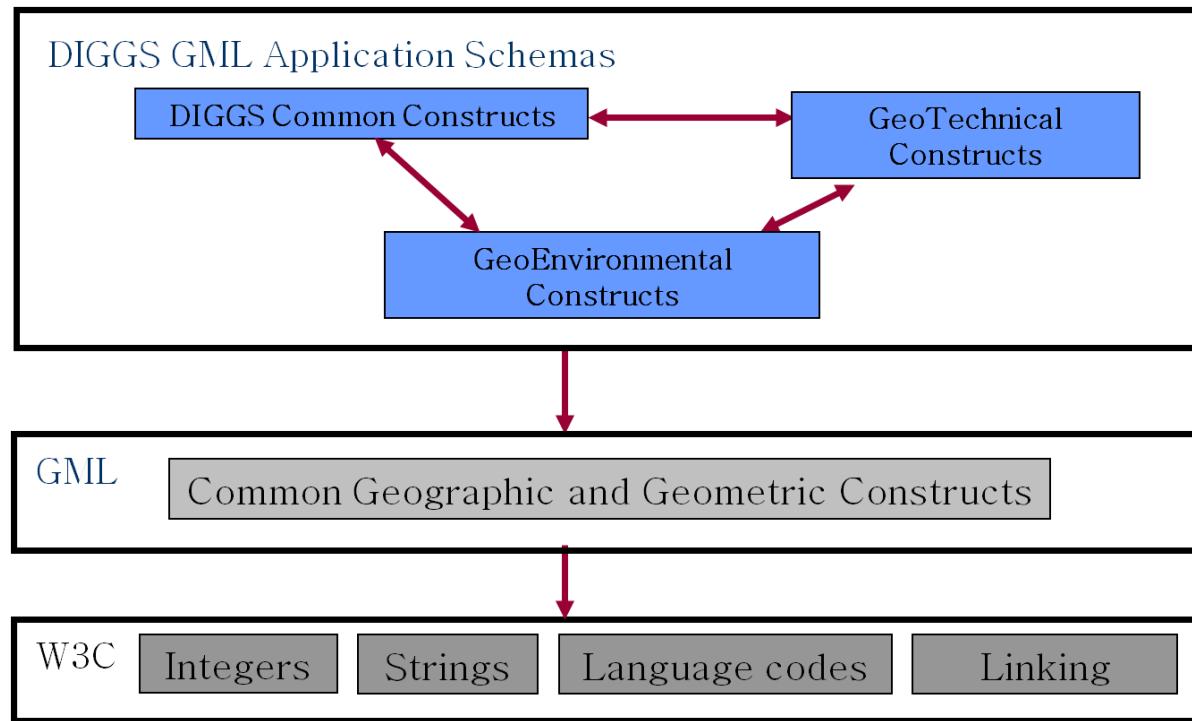


Figure 2-1: GML Application Schemas Build on GML and XML

2.3.1 Feature Model Overview

DIGGSML describes the world in terms of *features*, which represent physical or abstract objects representing geotechnical and geoenvironmental entities (eg. boreholes, samples, etc.) or processes (projects, tests, etc) that can be transferred as atomic units of information.

2.3.1.1 *DIGGS Objects*

Features are the primary *objects* in DIGGS, which are named entities comprised of descriptive properties. Non-feature objects also exist and are structurally the same as features - but typically are not shared out of context with their associated features. In DIGGS, objects appear as nested complex property values of features (a complex property element is one that contains child elements), e.g. a polygon representation of a trench wall's surface extent. A layer system defining soil descriptions is an example of a DIGGS feature, whereas the individual layers contained within a layer system are just objects that wouldn't be shared outside of the context of the layer system. *Metadata objects* are specially typed objects in GML, which describe contextual information about features or other objects.

2.3.1.2 **DIGGS Properties**

Properties are simply child elements of a feature or object. For example, a numeric result of a test is a property of the test feature. Figure 2-2 illustrates properties as direct children of a Borehole feature.

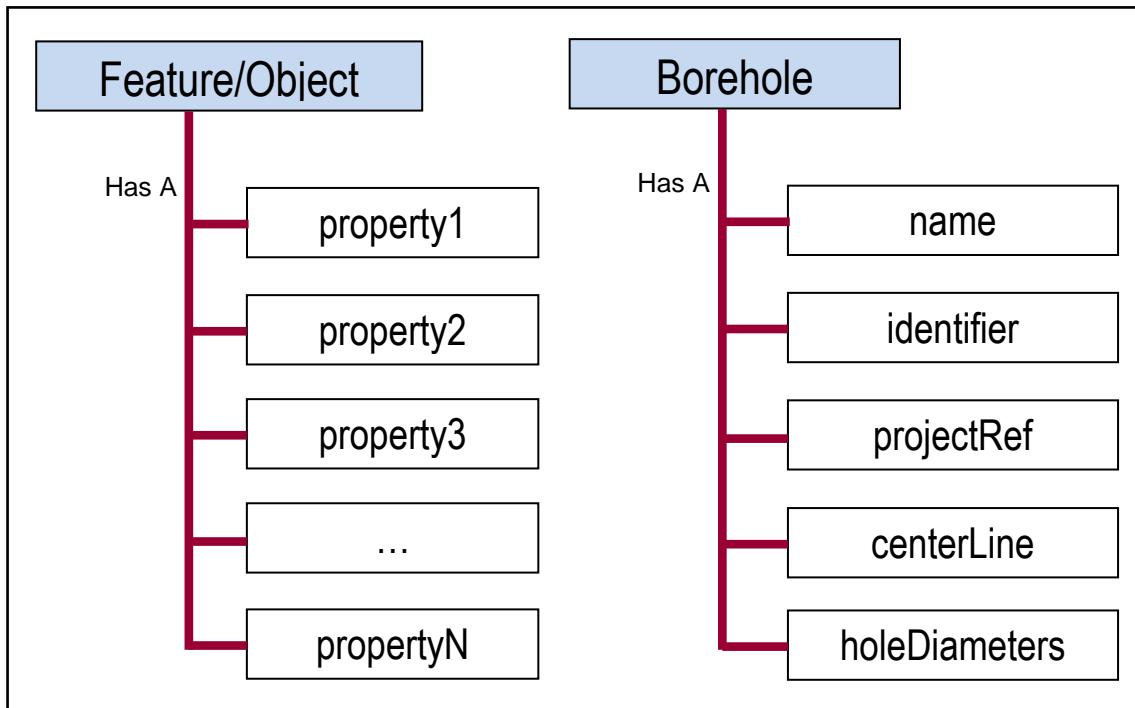


Figure 2-2: A DIGGS Feature or Object is described by its property children

Figure 2-2 also reveals a GML syntactic convention used to distinguish between Objects and properties; element and type names representing *Objects* are written in *UpperCamelCase* and the *property* names are written in *lowerCamelCase*.

2.3.1.3 **The GML Object/property/Value Model**

The GML *Object/property/Value* model is partially based on the Subject/predicate/Object model of W3C's Resource Description Framework (RDF), which is similar to the OMG's Class/association/Class model of UML. The Object/property/Value triple pattern as implemented in GML facilitates an easy transition and mapping from UML or RDF to GML.

In contrast to legacy GIS approaches, a feature is not defined primarily as a geometric object, but as a meaningful real world object having spatial and non-spatial properties. For example, Figure 2-2 shows that a Borehole feature has three simple properties that provide its name, identifier and reference to the Project the Borehole is associated with. Figure 2-2 also shows two complex properties that have object values describing the Borehole's physical characteristics

(centerLine and holeDiameters). Complex properties are used to describe relationships (associations) between two Objects, where the property name provides the name of the relationship or possibly the name of the role that the target (or source) value plays in the relationship –this is illustrated in Figure 2-3.



Figure 2-3: Property Names Capture the Relationship with or Role of the Target Value

Figure 2-2 shows that the Borehole feature has a centerLine property. The value of the centerline property is a geometry object LinearExtent (a one dimensional curve). The property name centerLine in this case represents the role that the LinearExtent target value plays with respect to the Borehole source value in the relationship – this is illustrated in Figure 2-4. Note that it is possible to describe several different relationships between the Borehole and geometry objects. For example a LinearExtent geometry might also represent the edge or circumference of the top of the Borehole. The Borehole also has another geometric property named referencePoint that is Point valued, which represents the location of the center of the base of the Borehole.

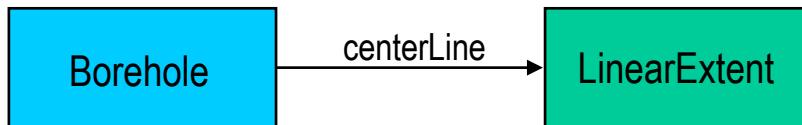


Figure 2-4: The centerLine Property Captures the Relationship between Feature and Geometry

An object cannot directly have a second object as a child element, only a property element can be a child of an object, but the property's value can be an object. The following example shows a correct example instance illustrating the parent-child element relationships in XML of the Borehole feature in GML:

```

<Borehole gml:id="LB_Webster">
  <gml:name>Long Beach - Webster</gml:name>
  <gml:identifier>urn:diggs:def:feature:USGS:LB_Webster</gml:identifier>
  ...
  <centerLine>
    <LinearExtent srsName="urn:diggs:def:crs:DIGGS:26911_5703" srsDimension="3" gml:id="LS0001">
      <gml:posList>387316.665116977 3742645.12297961 7.81507 387316.665116977 3742645.12297961 -
        420.124129847717</gml:posList>
    </LinearExtent>
  </centerLine>
  ...
  <holeDiameters>
    <BoreholeDiameter gml:id="bhd1">
  
```

```
<diameter uom="in">6</diameter>
</BoreholeDiameter>
</holeDiameters>
...
</Borehole>
```

Note that the properties name and identifier are simple property elements in the sense of XML Schema because their types have simple content (e.g. string, integer, and URI values), possibly with simple child attributes (but no child elements). The property centerLine on the other hand is a complex property because its type has complex content because it has a child object element value (the LinearExtent geometry).

2.3.2 DIGGS Applications

Both Custom of The Shelf (COTS) software (e.g. Saxon, GeoTools, Oxygen, Altova, Galdos GML SDK, Snowflake GML Viewer, OGC Web Feature/Map/Portrayal/Registry Service implementations) and specialized DIGGS software (e.g. DIGGS KML and Excel tools) can process DIGGS data structures for various purposes in varying degrees. For example, some GML aware COTS applications can detect and extract metadata or geometry types from GML instances and are designed to handle such typed information for specific purposes. Visualization applications (e.g. OGC Feature Portrayal Service) will detect and extract geometry properties to display on a map or earth browser and Registry applications (e.g. OGC Web Registry Service) can harvest metadata for discovery and archival purposes. Section 7 (Specialized DIGGS Tools) describe the software support created for DIGGS V2.0.a, namely the DIGGS Excel Tool and the DIGGS KML Tool.

2.4 DIGGSML Repository Location

The official DIGGS2.0a standard is available to the public from the DIGGSML web home page managed at <http://diggsml.org/> In particular the schemas can be accessed at <http://diggsml.org/2.0a/schemas>.

3 DIGGS Revision History from V1.0a to V2.0a

Several changes were made to the DIGGS 1.0a schemas following the submission of the V1.0a schema evaluation reports (see [1], [6], [9]). The release of DIGGS v1.1 applied the most straightforward fixes as recommended in [1], [6], and [9], i.e. the domain-independent schema issues, to quickly bring the DIGGS schemas into conformance with GML. The DIGGS V1.1 release (see [2]) also migrated the version of GML from V3.1 (see [3]) to the latest OGC version at that time, GML 3.2, mainly to benefit from the stability of the ISO TC211 (geographic information) adoption in that release of the GML Standard, published as ISO 19136 (see [7]).

The development of DIGGS v1.2 applied additional changes, including DIGGS domain-dependent issues, which required further discussion with the DIGGS core SIG and other stakeholders. During the development of DIGGS 1.2, new incremental extensions to GML 3.3 were being developed at OGC in parallel with the DIGGS development. The GML 3.3 extensions started in 2011 and were officially released in January 2012 (see [8]). The DIGGS schemas made use of some of the GML 3.3 extensions, namely linear referencing and the simplified multi-point expression, which started to be applied in draft form at DIGGS V1.2.4 and continued through to the development of the 2.0a release. Examples of the use of the GML 3.3 extensions into DIGGS 2.0a are included in this documentation (e.g. see Figure 5-10, Figure 5-15, Figure 5-26 followed by example instances).

3.1 DIGGS 1.0a

DIGGS 1.0a was finalized (see documentation [9], [10]) and released (see [5]) as an alpha version in 2008 to the DIGGS vendor community for testing and comment during a three month period starting in October 2008. An Invitational Meeting was later scheduled for DIGGS stakeholders, including members from the Geotechnical Management System (GMS), Geotechnical Data Coalition (GDC), Special Interest Group (SIG) Chairs, and selected industry partners in March 2009, to report on the status of working with DIGGS 1.0a. The presentation results of the Invitational Meeting were summarized in the Final Report (see [11]), which also identified and began the documentation of a set of core issues with the DIGGS 1.0a schemas. A few months later, the DIGGS Project Team announced a call for a formal independent review of the DIGGS 1.0a schemas (see [12]).

3.1.1 Independent Review of DIGGS 1.0a Schemas

The DIGGS Project Team contracted with Galdos Systems Inc. and Compusult Ltd. to carry out independent reviews of DIGGS version 1.0a. The objectives of this work were to:

1. Assess the current implementation of DIGGS with GML. (Were GML standards, conventions, best practices, and patterns implemented correctly? Was the appropriate version and/or variation of GML used based upon the requirements of DIGGS?)
2. Assess if the goals of the DIGGS standard is best implemented as a GML application schema, or if other encoding standards (both GML and non-GML) should be considered.
3. Assess the strategy that was used in the organization and composition of schema objects and files in the context of the physical observations and data that DIGGS is trying to capture. (Is the schema too de-composed, or appropriate given the requirements of DIGGS?)
4. Assess the core issues recently documented by the DIGGS Project Team on the discussion forums and at the March 2009 DIGGS meeting in Orlando.
5. Consider other issues, as appropriate, that may not have identified by the DIGGS Project Team, that exist with the schema design that may impact its application.

Final reports from Galdos (see [1]), Compusult (see [6]) and a synthesis of the findings (see [12]) were later published at diggsml.com with results and recommendations that DIGGS 1.0a was not implementable without some further development.

3.2 **DIGGS 1.1**

The list of issues identified in the DIGGS 1.0a Schema Evaluation reports [1], [6], and [12] were fixed as recommended and agreed with the DIGGS team leading to the release of DIGGSML V1.1. One of the main concerns of V1.0a to be addressed in V1.1 was usability and performance of the schemas, so the GML conformance fixes and GML profile creation were of the highest priority for V1.1. A performance assessment was carried out upon completion of the V1.1 schemas as summarized in the following Sections.

3.2.1 **Validation Performance Assessment**

The changes to DIGGS made to V1.1 resulted in a 60x validation speed-up from V1.0a. The Oxygen 10 Integrated Development Environment (IDE) was used to validate the complete set of DIGGS schemas (using the built in Xerces J parser) with the most exhaustive validation settings enabled including: ‘schema-full-checking’ and ‘honour all imports’.

The V1.1 schemas took 1.5 seconds to validate compared to 90 seconds for V1.0a with same validator, settings, and software environment.

3.2.2 Complexity Assessment

The metric used to measure the complexity of the DIGGS schemas was to average the file sizes of randomly generated instances from the DIGGS schemas. The random file size is proportional to the number of potential choices that can be made in populating DIGGS data and represents the complexity of transforming the DIGGS model to other data models/formats e.g. a relational database table structure or visualization formats such as KML.

The Diggs root element was generated with random values, using Oxygen 10, with maximum recursivity level set to 3 (the highest level at that time that could be carried out on the DIGGS schemas without memory overflow issues). Note that there was no user control in Oxygen 10 at that time, to set the maximum tree depth (nesting level) as was done for v2.0a with Oxygen 14. Several instance files were then generated automatically and the file sizes were averaged, with the following results

- V1.0a
 - Average file size = 10397.65 MB
- V1.0a with GML Profile
 - Average file size = 355.74 MB (~30x decrease from V1.0a)
- V1.1
 - Average file size = 14.2 KB (~750,000x decrease from V1.0a)

3.2.3 Overview of Changes in V1.1

The summary of the changes made to V1.1 are summarized as follows.

- Fixed GML Object-Property rule
- Fixed import/include statements – no longer need OASIS XML catalog
- Migrated DIGGS to GML 3.2
- Implemented a GML profile for DIGGS
- Reorganized DIGGS schemas to 5 namespace with one file per namespace (reduced further to 3 namespaces in V2.0a)
- Implemented `gml:identifier` and `gml:id`
 - `gml:identifier` – for globally unique id (use URN); only applies to gml features (locations, projects, samples, layer systems, tests)
 - `gml:id` – for all features and objects for referencing (database handle); objects are complex types that are “recognized” as properties in gml.

- Removed unnecessary abstract types
- Implemented a Registry Information Model (RIM) for codelists in lieu of GML dictionaries.
 - Allows for single vocabulary with language translations.
- Tabular data
 - retained existing structure, with the intention of implementing code lists (in v1.2) to restrict column types
 - removed generic table property
 - tables included under specific test features only
- Geometry
 - Projects – reference point, linear extent, areal extent
 - Linear referencing – identified the gml 3.3 method to linearly reference positions in a borehole (developed in V1.2 in parallel with gml 3.3 deveopment at OGC).
- Metadata
 - AssociatedFile, Role, Remark, Specification, Equipment, BusinessAssociate, and Contract were cast as GML metadata so that GML aware applications (e.g. Metadata Registry service) will recognize those objects as metadata for discovery and archiving purposes.
 - No longer assigned metadata properties at the base level to prevent metadata recursion.
 - Added associatedFile, roles, and remarks metadata properties to all features.
 - Added remarks metadata properties to all objects.
 - Tests defined as features and added specifications and equipment metadata properties as default.

It was a relatively straightforward set of tasks to apply the validation and conformance fixes to DIGGS 1.0a using GML (3.1), and then to migrate DIGGS to GML 3.2. The following sections provide more in-depth technical details of the changes made in V1.1 under the categories: XML Validation, GML Conformance, and Migration from GML 3.1 to GML 3.2.

3.2.4 Specific XML Changes in V1.1

3.2.4.1 XML Validation

Although the DIGGS 1.0a schemas validated with the Xerces J parser/validator, there was a minor issue detected by the Altova XML Spy validator (versions 2005 r3 and 2009 sp1). Other XML validators, including Xerces J, did not complain about this issue as it is commonly encountered and instead the fix suggested below was automatically and gracefully applied by these validators upon loading of the schema files.

Altova reported that the following 4 XML namespace declarations were missing in the complete.xsd schema:

```
xmlns:diggs_geo=http://schemas.diggsml.com/1.0a/geotechnical
xmlns:diggs_env=http://schemas.diggsml.com/1.0a/environmental
xmlns:diggs_mon=http://schemas.diggsml.com/1.0a/monitoring
xmlns:diggs_pil=http://schemas.diggsml.com/1.0a/piling
```

The issue was easily fixed by completing the following namespace declarations on the **schema** element as shown in bold below.

```
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:witsml="http://www.witsml.org/schemas/131"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:diggs="http://schemas.diggsml.com/1.0a"
  xmlns:diggs_geo="http://schemas.diggsml.com/1.0a/geotechnical"
  xmlns:diggs_env="http://schemas.diggsml.com/1.0a/environmental"
  xmlns:diggs_mon="http://schemas.diggsml.com/1.0a/monitoring"
  xmlns:diggs_pil="http://schemas.diggsml.com/1.0a/piling"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  targetNamespace="http://schemas.diggsml.com/1.0a"
  elementFormDefault="qualified">
  ...
</schema>
```

3.2.4.2 GML Conformance

Several design patterns were discovered in the DIGGS 1.0a schemas that did not conform to the GML 3.1 Object-Property rules (see [1] for details and references to the violated GML clauses). The fixes applied to v1.1 are organized according to the following sub-sections.

Property Types that Extended a GML Object Type

There were several Property types whose content model extends that of a GML Object type (e.g. gml:AbstractFeatureCollectionType), which renders the corresponding property element simultaneously as a GML Object, breaking the GML Object-property rule. The exhaustive list of such property types in the DIGGS 1.0a schemas are listed in [1]. For example, a schema fragment in AssociatedFile.xsd that illustrates an incorrect property type extension, which is non-conformant with the GML Object-property rule, is illustrated in **bold** below:

```
<complexType name="AssociatedFileType">
  <annotation>...</annotation>
  <complexContent>
    <b><extension base="gml:AbstractFeatureCollectionType"></extension></b>
```

```

<sequence>
  <element ref="diggs:_AssociatedFile" minOccurs="0" maxOccurs="unbounded"/>
  <element ref="diggs:Ref" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</extension>
</complexContent>
</complexType>

```

This problem pattern was fixed in V1.1 by removing the offending complexContent and extension tags from the property type definition. For example, the following commented tags were removed in the schema definition as shown in **bold**:

```

<complexType name="AssociatedFilePropertyType">
  <annotation>...</annotation>
  <!--remove <complexContent>
    <extension base="gml:AbstractFeatureCollectionType">-->
    <sequence>
      <element ref="diggs:_AssociatedFile" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="diggs:Ref" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  <!--remove </extension>
  </complexContent>-->
</complexType>

```

Property Elements Typed as a GML Object Type

There were several Property elements whose type is that of a GML Object type (e.g. diggs_geo:HoleType), which renders the corresponding property element simultaneously as a GML Object, breaking the GML Object-property rule. The exhaustive list of such property types in the DIGGS 1.0a schemas are listed in [1]. For example, a schema fragment in AssociatedFile.xsd that illustrates an incorrect property element declaration is illustrated in **bold** below:

```

<complexType name="PileConstructionType" mixed="false">
  <annotation>...</annotation>
  <complexContent>
    <extension base="diggs:IdentifiedFeatureType">
      <sequence>
        ...
        <element name="hole" type="diggs_geo:HoleType" minOccurs="0" maxOccurs="1">
          <annotation>...</annotation>
        </element>
        ...
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

The problem pattern was fixed in V1.1 by changing the type reference in the property element declaration from the object type to the corresponding property type, which contains the object as a child. For example, the property declaration was changed as shown in **bold** below:

```
<complexType name="PileConstructionType" mixed="false">
  <annotation>...</annotation>
  <complexContent>
    <extension base="diggs:IdentifiedFeatureType">
      <sequence>
        ...
        <element name="hole" type="diggs_geo:HolePropertyType" minOccurs="0" maxOccurs="1">
          <annotation>...</annotation>
        </element>
        <!--was <element name="hole" type="diggs_geo:HoleType" minOccurs="0" maxOccurs="1">
          <annotation>...</annotation>
        </element>-->
        ...
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Property Types with Non-homogeneous Content

Every property type definition in the DIGGS 1.0a schemas (i.e. all types with name suffix “.PropertyType”) had a content model, which contained two or more elements that are non-homogeneous, i.e. that do not derive from a common base type, which violated the GML property type pattern. For example, a non-conforming schema fragment in DiggsObject.xsd, is illustrated in **bold** below:

```
<complexType name="Role.PropertyType">
  <annotation>...</annotation>
  <complexContent>
    <extension base="gml:AbstractFeatureCollectionType">
      <sequence>
        <element ref="diggs:_Role" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="diggs:Ref" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

This issue was addressed in V1.1 by removing the diggs:Ref element and moving the diggs:Ref attributes: index and percentage, to a property element diggs:role in addition to the xlink association attributes used for GML remote references. The new declaration of diggs:role (renamed from diggs:roles) and the definition of Role.PropertyType became:

```

<complexType name="DiggsObjectType" mixed="false">
  <annotation>...</annotation>
  <complexContent>
    <extension base="diggs:Diggs BaseType">
      <sequence>
        ...
        <element name="role" type="diggs:RolePropertyType" minOccurs="0" maxOccurs="unbounded">
          <annotation>...</annotation>
        </element>
        ...
      </sequence>
      ...
    </complexContent>
  </complexType>

  <complexType name="RolePropertyType">
    <annotation>...</annotation>
    <!-- remove as recommended in 4.1.1.1.5 -->
    <complexContent>
      <extension base="gml:AbstractFeatureCollectionType">-->
        <sequence minOccurs="0">
          <element ref="diggs:_Role" minOccurs="0"/>
          <!--remove <element ref="diggs:Ref" minOccurs="0" maxOccurs="unbounded"/>-->
        </sequence>
        <!-- remove as recommended in 4.1.1.1.5 -->
      </complexContent>-->
      <attribute group ref="gml:AssociationAttributeGroup"/>
      <attribute name="index" type="integer"/>
      <attribute name="percentage" type="double"/>
    </complexType>

```

Elements that Substituted for “gml:_Object”

There were several elements listed in [1] whose types derived by extension from `gml:AbstractGMLType`, which did not substitute for the corresponding substitution group `gml:_GML`. For example, an element declaration and type definition that illustrates an inappropriate `substitutionGroup` value, is illustrated in **bold** below:

```

<element name="_Database" type="diggs:DatabaseType" substitutionGroup="gml:_Object"
abstract="true">
  <annotation>...</annotation>
</element>

<complexType name="DatabaseType" mixed="false">
  <annotation>...</annotation>
  <complexContent>
    <extension base="gml:AbstractGMLType">
      <sequence>
        ...
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

```
</extension>
</complexContent>
</complexType>
```

This issue was fixed by modifying the substitution group value (example in **bold** below) on all the element declarations exhaustively summarized in [1]:

```
<element name="_Database" type="diggs:DatabaseType" substitutionGroup="gml:_GML"
abstract="true">
<annotation>...</annotation>
</element>
```

Geometry Type Declaration

The diggs:GeometryType definition in Geometry.xsd did not conformant to the rules for user defined geometry objects. The diggs:GeometryType definition is illustrated below:

```
<complexType name="GeometryType" mixed="false">
<annotation>...</annotation>
<complexContent>
<extension base="gml:AbstractGMLType">
<sequence>
<element name="points" type="gml:MultiPointType" minOccurs="0" maxOccurs="1">
<annotation>...</annotation>
</element>
<element name="lines" type="gml:MultiLineStringType" minOccurs="0" maxOccurs="1">
<annotation>...</annotation>
</element>
<element name="polygons" type="gml:MultiPolygonType" minOccurs="0" maxOccurs="1">
<annotation>...</annotation>
</element>
</sequence>
</extension>
</complexContent>
</complexType>
```

This issue was addressed by redefining the point, line, and polygon geometry types conforming to GML and declaring the appropriate property elements directly on the parent features that use them (in place of the single generic geometry property). For example, the following Borehole has multiple geometric representations, each one distinguished by the appropriate geometry property names: referencePoint, centerLine, and totalMeasuredDepth, as illustrated by example below in **bold**:

```
<Borehole gml:id="LB_Webster">
...
<referencePoint>
<PointLocation srsName="urn:diggs:def:crs:DIGGS:0.1:26911_5703" srsDimension="3" gml:id="a33">
```

```

<gml:pos>387316.665116977 3742645.12297961 7.81507</gml:pos>
</PointLocation>
</referencePoint>
...
<centerLine>
<LinearExtent srsName="urn:diggs:def:crs:DIGGS:0.1:26911_5703" srsDimension="3"
  gml:id="ls">
  <gml:posList>387316.665116977 3742645.12297961 7.81507 387316.665116977 3742645.12297961 -
420.124129847717</gml:posList>
</LinearExtent>
</centerLine>
...
<totalMeasuredDepth>
<PointLocation gml:id="lb_web_td" srsDimension="1" srsName="#sr123">
  <gml:pos>427.9392</gml:pos>
</PointLocation>
</totalMeasuredDepth>
...
</Borehole>

```

Metadata

There were several property elements in DIGGS that carry feature metadata but are not strongly typed using GML metadata. For example: AssociatedFile, BusinessAssociate, Contract, Equipment, and Specification are such metadata elements but GML aware software such as a metadata harvester cannot detect this through the standard GML typing mechanism. In V1.1, all features carry associatedFile, roles, and remarks metadata properties and all objects carry remarks metadata properties. In addition, all tests are features and carry specifications and equipment metadata properties by default.

The metadata non-conformance issue was fixed in V1.1 by typing the metadata elements appropriately in the DIGGS schema Kernel.xsd according to the GML rules. An example of the corrected AssociatedFile declaration is shown below in **bold**:

```

<element name="AssociatedFile" type="diggs:AssociatedFileType"
  substitutionGroup="diggs:AbstractMetadata" abstract="false"/>

<complexType name="AssociatedFileType">
  <complexContent>
    <extension base="diggs:AbstractMetaDataType">
      <sequence>
        <element name="fileType" type="diggs:DiggsStringType" minOccurs="0" maxOccurs="1"/>
        <element name="creatingApplication" type="diggs:SoftwareApplicationPropertyType"
          minOccurs="0" maxOccurs="1"/>
        <element name="documentType" type="gml:CodeType" minOccurs="0" maxOccurs="1"/>
        <element name="fileDate" type="diggs:UnifiedDateTimeType" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

```
</extension>
</complexContent>
</complexType>
```

3.2.4.3 *Migration from GML 3.1 to GML 3.2*

Several straightforward changes were made to the DIGGS 1.1 schemas to migrate from GML 3.1 to GML 3.2. Such changes included: updating the namespace/version, minor changes to some element and type names (e.g. replace ‘gml:_Feature’ with ‘gml:AbstractFeature’), restricting cardinality of some attributes (e.g. gml:id becomes mandatory instead of optional), and incorporating some new elements (e.g. gml:identifier, see Section 4.1 Naming and Identification). An XSLT transformation was used to convert the DIGGS schema definitions to bring them into compliance with GML 3.2.

3.3 *DIGGS 1.2*

The developments made in V1.2 were closely coordinated with the DIGGS team as the changes were mainly domain dependent requiring DIGGS subject matter expertise. The changes range from V1.2.1 to V1.2.4.k, much of which reflect modifications based on DIGGS stakeholder comments.

3.3.1 *Summary of Changes in V1.2*

A bullet list summary of the changes made in V1.2 is as follows (for the fully detailed description of the changes, see Appendix D - DIGGS Change Release Log):

- Implementation of formal codelists and creation of a supplementary XML representation external to the schemas.
- Modifications to the test data structure to align more closely with the OGC Observations and Measurement model.
- Implementation of 4 types of sampling features (formerly named location features): borehole, trial pit, trench wall, and station that make use of some of the new GML 3.3 extensions including linear referencing
- Development of dictionaries and codelists.
- Implementation of Sampling and SamplingActivity as separate features to distinguish the physical sample from the activity that produces it.
- Implementation of layer systems using codelists to define constituents

- Implementation of the GML coverage model for encoding CPT, geophysical, and similar data types.

Some other structural changes include making the Project feature mandatory in all DIGGS instances and restrict the occurrence so that only one Project is permitted in a DIGGS file. All other features in the DIGGS file now must reference the single project.

3.4 DIGGS 2.0a

The changes made to V2.0a included schema optimization (to confirm usability and performance), reorganization (e.g. removed Piling.xsd) and other minor modifications due to namespace updates and alignment with OGC corrigenda changes (e.g. XLink updates, GML 3.3 finalization).

3.4.1 Summary of Changes in V2.0a

A bullet list summary of the changes made in V2.0a is as follows (for the fully detailed description of the changes, see Appendix D - DIGGS Change Release Log):

- Updated XLink schema and corresponding schema dependencies in DIGGS schema files from OGC xlink.xsd to W3C xlink.xsd following the OGC XLink corrigendum taking effect July 21, 2012
- Namespace and version changes to reflect V2.0a
 - <http://schemas.diggsml.com/2.0.a>
 - <http://schemas.diggsml.com/2.0.a/geotechnical>
 - <http://schemas.diggsml.com/2.0.a/environmental>
- Restricted DIGGS dependencies on WITSML schemas
- Changed order of DIGGSType properties so that all metadata properties are together at the end
- Changes to schema organization (e.g. drop GeoPhysical.xsd, Monitoring.xsd, Piling.xsd as separate schema files and integrate schema definitions into Kernel.xsd and Geotechnical.xsd)
- Schema optimization – tests for recursivity (looping) proved positive in Kernel.xsd and gml3.2Profile_diggs.xsd. All recursivity was removed from these schemas.
- Minor modifications to DIGGS URN Identifier structure and corresponding changes to gml3.2Profile_diggs.xsd, Kernel.xsd, and testInstance.xml
- Updated DIGGS KML and Excel tools to support V2.0a

- Changed GML 3.3 Linear Referencing schemas as final adjustments were made to accomodate GML 3.3.1 corrigendum
- Updated namespace value for `gml3.3Profile_diggs.xsd` to the recently adopted OGC official namespace (<http://www.opengis.net/gml/3.3/ce>)

4 DIGGS 2.0a Structures and Organization

The DIGGS 2.0a data structures consist of schemas, code lists, dictionaries, and feature data and metadata instances. This section summarizes how the data structures are identified and organized in the online DIGGSML repository at www.diggsml.org.

4.1 Naming and Identification

All of the DIGGS data structures are named (e.g. by controlled string patterns) and identified (by URIs) using formal internet mechanisms that enable referencing, linking and resolution over the web. This section summarizes the naming conventions, identification schemes and the corresponding authorities responsible for naming and identification of the DIGGS related resources used by the DIGGS data structures. For example, DIGGS data instances are identified using an OGC GML identification scheme (`gml:id` and `gml:identifier`), reference schemas by an HTTP namespace URI (e.g. <http://schemas.diggsml.com/2.0a>) and schema location URL (e.g. <http://schemas.diggsml.com/2.0/>) for validation purposes, have property element associations to other data instances using W3C XLink, and have attribute values that reference codes (e.g. AGS codes) and dictionary entries (e.g. EPSG CRS and SI Units) by URN that resolve to code lists and dictionaries maintained by different authorities.

4.1.1 Authorities and Namespaces

A namespace is an identifier maintained by an authority whose structure (string, URI) is defined by that authority to identify data structures of interest (e.g. schemas, data, codelists, dictionaries) to that authority. The following subsections provide examples of different namespaces and authorities.

4.1.1.1 Example 1

DIGGS is the authority for the core schema namespace (<http://schemas.diggsml.com/2.0a>), which is an HTTP URI identifier for the set of grouped schema definitions (found in `Kernel.xsd` and `Complete.xsd`). The DIGGS core schema namespace URI value is treated as a unique string defined by the DIGGS authority and is used to distinguish the schema definitions from others, e.g. the schema definitions in the GML namespace (<http://www.opengis.net/gml/3.2>) defined by the OGC authority. Note that both of these sets of schema definitions contain some elements with the same (local) name but with different definition, e.g. `<AbstractFeature/>`. These elements with the same local name are distinguished by their fully qualified names: `<diggs:AbstractFeature/>` and `<gml:AbstractFeature>`, where the ‘diggs’ namespace prefix is bound to the DIGGS core schema

namespace (<http://schemas.diggsml.com/2.0a>) and the ‘gml.’ namespace prefix is bound to the GML 3.2 namespace (<http://www.opengis.net/gml/3.2>) by the following namespace declarations, respectively:

```
<schema
  xmlns:diggs="http://schemas.diggsml.com/2.0.a"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  ...
</schema>
```

4.1.1.2 Example 2

DIGGS is also the authority for its own URN namespace identifier, which is registered with the Internet Assigned Numbers Authority (IANA, <http://www.iana.org/>). The DIGGS URN identifiers are used to identify resources such as codelists, dictionaries, feature instances, etc. The DIGGS URN structure is normatively defined in DIGGS_URN_Identifier_Scheme.xls, informatively described in Section 4.1.2, and the URN registration RFC as submitted to IANA is contained in Appendix A.

4.1.1.3 Example 3

The OGC is the authority for several schema namespaces, a URN namespace identifier structure, and other dictionary namespace definitions (e.g. the (lon, lat) ordered geodetic CRS definition made up of EPSG components). A select few samples of OGC schema namespaces are as follows:

```
http://www.opengis.net/gml
http://www.opengis.net/gml/3.2/
http://www.opengis.net/gml/3.3/lr
http://www.opengis.net/gml/3.3/lrov
http://www.opengis.net/kml/2.2
http://www.opengis.net/om/2.0
http://www.opengis.net/fes/2.0
http://www.opengis.net/wfs/2.0
```

The OGC is also the authority for a URN identification namespace, a scheme which the DIGGS CRS dictionary makes use of to reference some CRS components and DIGGS feature data makes use of to reference EPSG components. An example DIGGS CRS dictionary entry illustrates the area and CRS resources that are referenced by OGC URNs.

```
<dictionaryEntry>
  <CompoundCRS gml:id="diggs-crs-63226405_5713">
    <identifier codeSpace="urn:diggs:def:authority:DIGGS">
      >urn:diggs:def:crs:DIGGS:0.1:63226405_5713</identifier>
    <name>WGS 72 (deg) + Canadian Vertical Datum of 1928</name>
```

```
<domainOfValidity xlink:href="urn:ogc:def:area:EPSG::1289"/>
<scope>Geodetic and engineering surveying.</scope>
<componentReferenceSystem xlink:href="urn:ogc:def:crs:EPSG::63226405"/>
<componentReferenceSystem xlink:href="urn:ogc:def:crs:EPSG::5713"/>
</CompoundCRS>
</dictionaryEntry>
```

4.1.2 DIGGS URN Identifier Structure

The DIGGS URN structure is registered with IANA and is normatively defined in the accompanying *DIGGS_URN_Identifier_Structure.xlsx* document, which is expected to change over time as modifications and new additions are made. A summary snapshot is provided here for convenience.

The IANA requires that all URN structures have the following basic form:

```
urn:{Namespace ID}:{Namespace Specific String}
```

The Namespace ID (NID) for the URN is assigned by IANA and was requested to be “diggs” in the RFC registration submission by DIGGS (see Appendix A). The Namespace Specific String (NSS) of all URNs that use the “diggs” NID will have the following structure:

```
urn:diggs:{ResourceType}:{ResourceSpecificString}
```

The {ResourceType} part of the URN has the form def:RESOURCE_CLASSIFICATION, where ‘def’ indicates that the resource is a definition and RESOURCE_CLASSIFICATION is constrained to an enumerated list of strings specified in the *DIGGS_URN_Identifier_Structure.xlsx* document. The {ResourceSpecificString} part of the URN includes authority, version, code, and sub-code fields resulting in the following form of the DIGGS URN:

```
urn:diggs:def:RESOURCE_CLASSIFICATION:AUTHORITY[:VERSION_NUMBER][:CODE][:SUB_CODE]
```

Note that the VERSION_NUMBER, CODE, and SUB_CODE fields are optional as indicated by the square brackets. The allowed values of RESOURCE_CLASSIFICATION, AUTHORITY, VERSION_NUMBER, CODE, and SUB_CODE are specified in the *DIGGS_URN_Identifier_Structure.xlsx* document, which at the time of writing are summarized in Table 4-1.

DIGGS URN Structure				
urn:diggs:def:RESOURCE_CLASSIFICATION:AUTHORITY[:VERSION_NUMBER][:CODE][:SUB_CODE]				
Field Name	Description	Mandatory /Optional Indicator	Field Values	Comments
urn	Required prefix for all URNs	Mandatory	This field is fixed	
diggs	Namespace Identifier	Mandatory	diggs	Assigned by IANA and requested to be 'diggs'
def	Indicates that the resource is a definition	Mandatory	def	There are no other branches of this field to date, but new branches could be added (e.g. doc or spec)
RESOURCE_CLASSIFICATION	The type or classification of the resource.	Mandatory	codelist crs dictionary feature uom	'feature' is used to identify a feature instance Other RESOURCE_CLASSIFICATION values may be added as required. For example, if CRS components such as datums are added to the DIGGS CRS Dictionary, this scheme will be updated accordingly.
AUTHORITY	The maintaining authority of the resource	Mandatory	AGS Caltrans DIGGS EPSG OGC POSC SI USGS	Other AUTHORITY values may be added as required. Note: the concept of AUTHORITY is extended to cover dictionaries. For example, the EPSG (CRS dictionary) is considered an acceptable AUTHORITY for the CRS resource type (even though the Oil and Gas Producers (OGP) is the organization that approves and authorizes the EPSG CRS dictionary releases)
VERSION_NUMBER	Indicates the release or version number	Optional	String characters representing the version (e.g. 2.0a)	If VERSION_NUMBER is omitted but CODE is present, :: should be used between AUTHORITY and CODE. The VERSION_NUMBER shall be omitted when used as an identifier of resource types: dictionary, cnode (including featureType, attributeType, listedValue) or slot, in the versionless FDDs such as the NFCD.
CODE	The code or local identifier of the resource (e.g. gml:id value)	Optional	String characters representing the code (e.g. CRS20a)	For some codespace values the AUTHORITY alone is sufficient, in which case CODE can be omitted and VERSION_NUMBER may be omitted
SUB_CODE	The sub-code or local identifier of the resource (e.g. gml:id value)	Optional	String characters representing the subcode (e.g. 246tcp)	The sub-code is populated e.g. when the code corresponds to a dictionary or codelist that has sub-entries

Table 4-1: Tabular Summary of DIGGS URN Identification Structure

The DIGGS Naming Authority (DIGGSNA) will manage and document resources using the "diggs" NID and will be the authority for managing the assignment of the {ResourceType} and {ResourceSpecificString} fields for each resource class. The DIGGSNA will ensure the uniqueness of the strings themselves or will delegate secondary responsibility for the management of well-defined subfields.

The DIGGSNA may also permit the use of unregistered experimental type values of the following form (i.e. preceded by urn:x-diggs).

urn:x-diggs:{ResourceType}:{ResourceSpecificString}

Such experimentation is intended for testing purposes only and would be the only case where multiple users may end up using the same value for separate uses.

The following examples are representative of URNs that could be assigned

```
urn:diggs:def:dictionary:DIGGS:CompoundCRS20a
urn:diggs:def:dictionary:DIGGS:Units20a
urn:diggs:def:codelist:DIGGS:mv_calculation_method
urn:diggs:def:codelist:DIGGS:modulus_calculation_method
urn:diggs:def:codelist:AGS:compaction_mould_type
urn:diggs:def:codelist:AGS:chemical_determinand
urn:diggs:def:crs:DIGGS:63226405_5713
urn:diggs:def:uom:SI:m
urn:diggs:def:uom:DIGGS:kdyne
urn:diggs:def:uom:DIGGS:ozf
urn:diggs:def:uom:DIGGS:tonfUS
urn:diggs:def:codelist:DIGGS:mv_calculation_method:secant_method
urn:diggs:def:codelist:DIGGS:modulus_calculation_method:tangent_method
urn:diggs:def:codelist:AGS:compaction_mould_type:cbr
urn:diggs:def:codelist:AGS:chemical_determinand:246tcp
urn:diggs:def:feature:DIGGS:Proj0001
urn:diggs:def:feature:USGS:LB_Webster
```

4.1.3 DIGGS Feature Identifiers and References

All features and objects in DIGGS carry a mandatory id attribute (gml:id), which is a standard XML ID type and used as a database handle for referencing and linking to and from other features and objects. The gml:id must be unique within a DIGGS data instance document and has some lexical restrictions (such as no colons ':' and other special characters). DIGGS features and objects also carry a optional identifier (gml:identifier), which is a patterned globally unique URI, encoded by the DIGGS URN, with additional pattern restrictions as used by the V1.0a diggs:IdentifierType, which now takes the form: urn:diggs:def:feature:[A-Z]{1,8}:[A-z0-9_\.-]{1,200}) following DIGGS identifier structure described in Table 4-1. The exampleBorehole instance in Figure 4-2, shows examples of gml:id values on features and objects (highlighted in blue) and an example of the Borehole's gml:identifier value (highlighted in green).

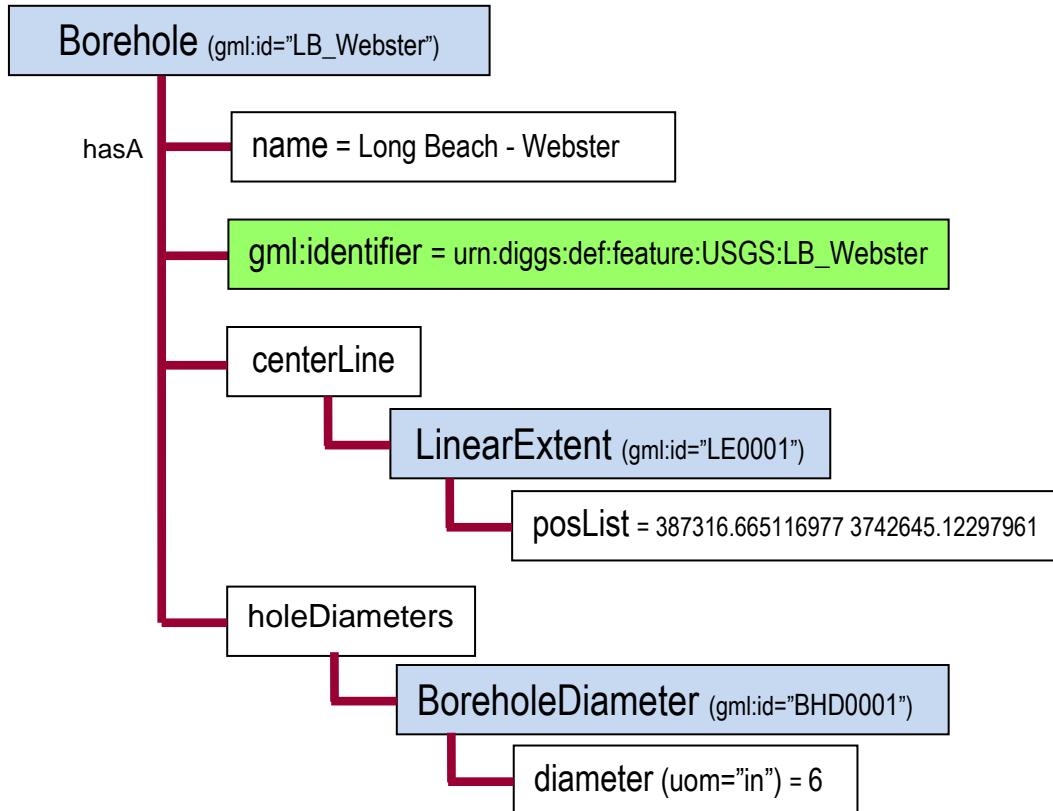


Figure 4-2: ExampleTruncated Borehole Instance

All DIGGS features carry property references/associations that can refer to either the gml:id attribute or the globally unique gml:identifier of the target feature. This reason for organizing features in this manner is that two-way references can be maintained between features, while allowing individual feature instances to be transmitted in XML documents without having to also transmit information about the objects they are associated with, except for their id's and identifiers. The reason that the referencing properties carry both the gml:id and gml:identifier is that an xlink:href may not actually resolve to a real object in an XML file somewhere (it could reside in a database instead), but the global identifier (the key) is still maintained in the referencing feature, so that references can be resolved later, in a database or processing software. This allows flexibility in business practice and reduces redundancy and requirements to update records in databases when there is no need to do so. When a hole is drilled, the driller can transmit the hole information in one instance file, next the geologist can independently transmit the layer descriptions with a reference to the hole, and finally, the logging company can deliver the geophysical log with a hole reference – all in separate instance documents but the information can be compiled together (and transmitted later together in a single instance document as part of a final report) because they will all carry references to the hole feature that they associate with. This can be done with or without a requirement to send along project information.

4.2 DIGGS Repository Organization

The DIGGS2.0a online repository includes the DIGGSML Schemas and the supporting Codelists, Dictionaries, Documentation and ExampleInstance directories as shown in Figure 4-3.

Name	Date modified	Type
CodeLists	23/04/2012 2:31 PM	File folder
Dictionaries	23/04/2012 8:32 PM	File folder
Documentation	23/04/2012 8:41 PM	File folder
Sample Instances	29/04/2012 9:19 PM	File folder
Schemas	30/04/2012 4:33 PM	File folder

Figure 4-3: DIGGS 2.0a Root Level Directory Structure

4.2.1 Official Schemas

The online DIGGSML Schema repository contains 9 XML Schema Definition (XSD) files as shown in Figure 4-4.

Name	Type	Size
Complete.xsd	XML Schema	2 KB
Environmental.xsd	XML Schema	9 KB
Geotechnical.xsd	XML Schema	97 KB
glrovProfile_diggs.xsd	XML Schema	3 KB
glrProfile_diggs.xsd	XML Schema	19 KB
gml3.2Profile_diggs.xsd	XML Schema	111 KB
gml3.3Profile_diggs.xsd	XML Schema	2 KB
Kernel.xsd	XML Schema	204 KB
xlink.xsd	XML Schema	10 KB

Figure 4-4: DIGGS 2.0a Schema File Directory

The DIGGSML top-level “wrapper” schema file is aptly named *Complete.xsd* as it aggregates all of the other dependent schemas. The creation of the *Complete.xsd* wrapper schema file makes it convenient for developers to validate/import all of the DIGGS schemas in a single execution/declaration. The main DIGGSML GML Application Schema definitions are contained in: *Kernel.xsd*, *Environmental.xsd*, and *Geotechnical.xsd*. The GML profiles (subsets of the core GML Schemas) that are used by DIGGSML are:

- ***gml3.2Profile_diggs.xsd*** – a subset of elements and types appropriately restricted from GML 3.2 for use by DIGGS
- ***gml3.3Profile_diggs.xsd*** – a single element (SimpleMultiPoint) and type (SimpleMultiPointType) from the GML 3.3 core schema used by DIGGS
- ***glrProfile_diggs.xsd*** – a subset of elements and types from GML 3.3 Linear Referencing used by DIGGS
- ***glrovProfile_diggs.xsd*** – a subset of elements and types from GML 3.3 Linear Referencing with Offset Vectors used by DIGGS

Both the DIGGS application schemas and the core GML schemas make use of the following W3C schemas:

- ***xlink.xsd*** – provides the simpleLink attribute group primarily used to locate a remote resource (using the xlink:href attribute) and defines supplementary behaviour, semantics and traversal attributes
- ***xml.xsd*** – (this file is imported by *xlink.xsd*) defines standard attributes such as *xml:lang* (to specify an international language code) and *xml:base* (for relative referencing flexibility)

4.2.1.1 **Schema Namespaces**

The DIGGS2.0a schemas are divided into different namespaces, where each namespace is essentially an identifier for the set of grouped schema definitions, whose purpose is to:

1. Prevent name collisions (e.g. distinguish an element with the same name in two different schemas)
2. Capture the schema version (at least in the case of the DIGGS identification scheme)
3. Distinguish schemas (e.g. core vs extensions) which have different constructs and conceptual models

The DIGGS2.0a schema namespaces are as follows:

<http://schemas.diggsml.com/2.0.a>
<http://schemas.diggsml.com/2.0.a/geotechnical>
<http://schemas.diggsml.com/2.0.a/environmental>

The first namespace (<http://schemas.diggsml.com/2.0a>) is the core namespace, which contains the common constructs (in *Kernel.xsd*) shared by the other DIGGS schemas. The core namespace also contains a wrapper schema (*Complete.xsd*) which declares the necessary references to aggregate all of the other schemas in a single package. The second and third namespaces listed above contain extensions to the core constructs specific to the Geotechnical and GeoEnvironmental domains, respectively. The main reason for the division of the schemas into these separate namespaces is so that the schema definitions in each of these namespaces can be revised independently by different subject matter experts. The schema file dependencies organized by namespace are illustrated in Figure 4-5, where a solid arrow represents inclusion of the target schema file (which is in the same namespace) into the schema at the base of the arrow and a dashed arrow represents an import of the target schema from another namespace. There are a total of 9 namespaces used in DIGGS2.0a, 3 of which are defined by DIGGS as described above, 4 defined by OGC (the GML schemas), and another 2 defined by W3C (XML and XLink).

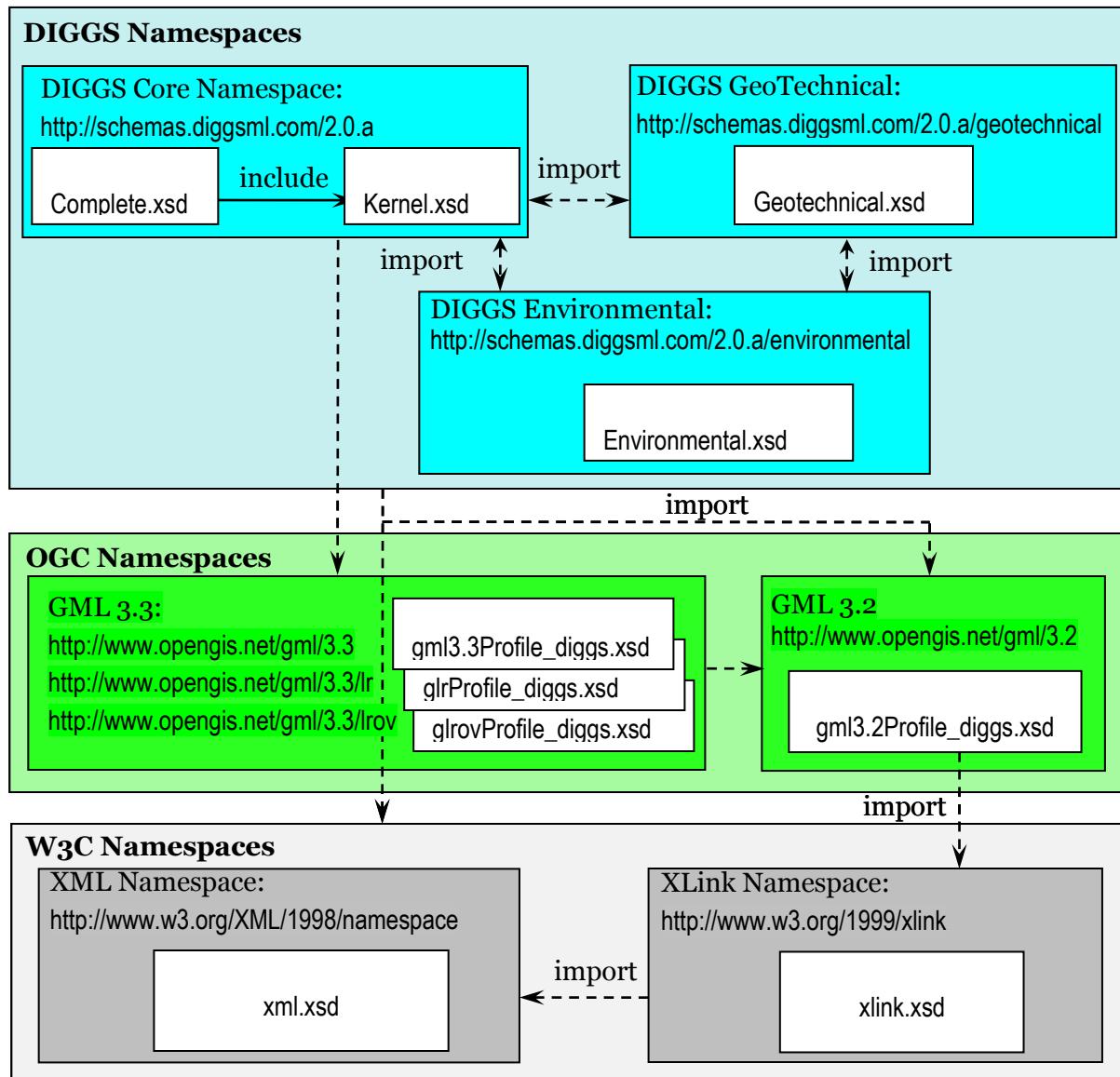


Figure 4-5: DIGGS 2.0a Schema Hierarchy Organized by Namespace

4.2.2 Data

Instances can be created and stored anywhere, online or offline, but were designed for sharing over the web. Data repositories are maintained by DIGGS users and can be read by applications on mobile devices, desktop workstations, or computer servers from various datastores:

- File directories – accessible online as public or private web pages or offline in local file directories (e.g. for field work without internet access).

- Spatial Databases – accessible online through public or secure web interfaces or offline using a standalone client interface

Data instances can be validated against the official DIGGS schemas online or can be validated by a locally saved/cached copy of the DIGGS schemas.

4.2.3 Dictionaries

GML provides an XML encoding to define both CRS and Units dictionaries specifically designed to conform to the international standard models for CRS (ISO TC211 19111 Spatial Refencing by Coordinates) and Units (SI), respectively. Such GML CRS and Units dictionaries can be defined and extended for custom use in specific application domains and was done for DIGGS.

4.2.3.1 **DIGGS Compound Coordinate Reference Systems (CRS)**

DIGGSML data is generally 3D, consisting of a pair of ‘horizontal’ coordinates followed by a third ‘vertical’ coordinate. Since many of the Earth’s commonly used horizontal and vertical CRS components are available in the Oil and Gas Producers (OGP) CRS dictionary (formerly known as the EPSG CRS database), it was natural to define the DIGGS CRS dictionary as a derived extension of OGP’s. The OGP CRS dictionary is managed online in a deployed web registry service and each of its CRS dictionary entries are stored as GML and can be retrieved as GML or WKT by online request. Each 3D CRS in DIGGS is defined as composition of the horizontal and vertical CRS components in the OGP CRS dictionary and the collection such compound CRSs is encoded as GML in the DIGGS CRS Dictionary. The DIGGS CRSs have been documented in all of the following ways:

- GML CRS Dictionary
- WKT CRS Dictionary
- CRS Component summary spreadsheets
- Registry Information Model (ebRIM XML encoding, standardized by OASIS and adopted by the OGC standard CSW-ebRIM, the geospatial Catalogue Service for the Web)

The GML and WKT CRS dictionaries were generated automatically from the CRS Component spreadsheets: *North American CRS.xls*, *UK CRS.xls*, and *World CRS.xls*.

DIGGS GML CRS Dictionary

The normative DIGGS CRS Dictionary is defined in the standard GML encoding designed for referencing and resolving over the web. The DIGGS CRS Dictionary contains a list of Compound

3D CRS definitions generated from components based on the EPSG (European Petroleum Survey Group) Database V7.5.8.

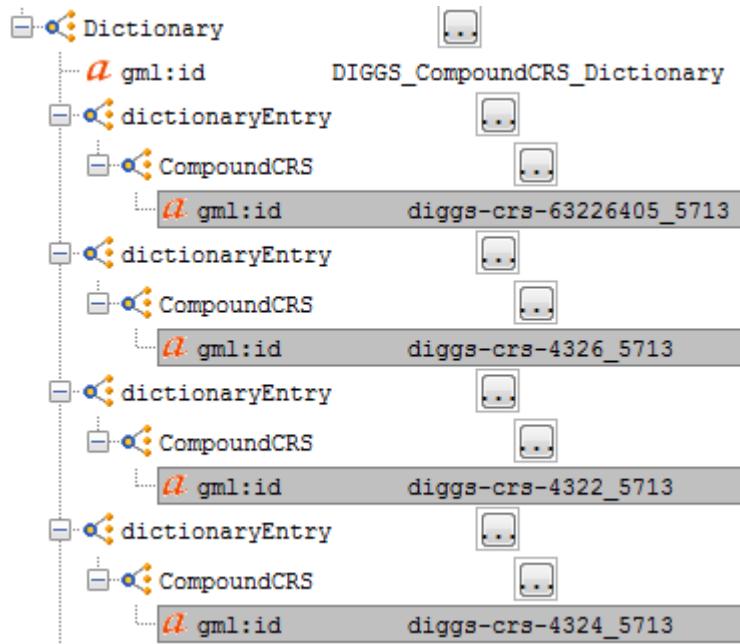


Figure 4-6: Excerpt of DIGGS CRS Dictionary of Compound 3D CRSS

The gml:id and gml:identifier values for each of the DIGGS Compound CRSSs were created by concatenating the EPSG codes (of the form [EPSG code]_[EPSG code]) from the component horizontal and vertical CRSSs.

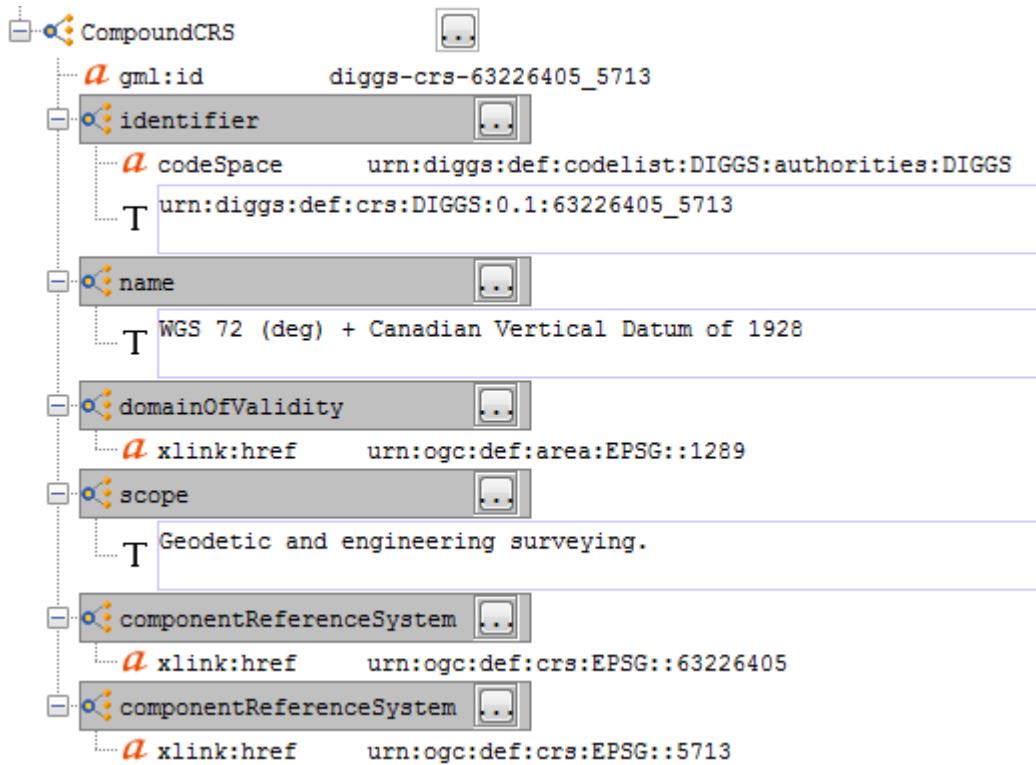


Figure 4-7: ExampleDIGGS Compound CRS Definition

Since the Compund CRSs were selected for use by DIGGS data, the URN identifiers are in the DIGGS namespace.

DIGGS WKT Dictionary

A text based version of the DIGGS CRS dictionary is also maintained as a list of Well-Known Text (WKT) definitions. This WKT dictionary is used existing extend spatial engines and transformation tools such as Oracle Spatial, Bentley Microstation and GeoTools. An exampleWKT encoding of the first compound CRS in the DIGGS dictionary is as follows.

```

63226405_5713=COMPD_CS["WGS 72 (deg) + Canadian Vertical Datum of 1928",
  GEOGCS["WGS 72 (deg)", DATUM["World Geodetic System 1972",
    SPHEROID["WGS 72", 6378135.0, 298.26, AUTHORITY["EPSG", "7043"]],
    TOWGS84[0.0, 0.0, 4.5, 0.0, 0.0, 0.554, 0.045171992568114105], AUTHORITY["EPSG", "6322"]],
    PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG", "8901"]],
    UNIT["degree", 0.017453292519943295],
    AXIS["Geodetic longitude", EAST],
    AXIS["Geodetic latitude", NORTH], AUTHORITY["EPSG", "63226405"]],
  VERT_CS["CGVD28 height",
    VERT_DATUM["Canadian Geodetic Vertical Datum of 1928", 2005, AUTHORITY["EPSG", "5114"]],
    UNIT["m", 1.0], AXIS["Gravity-related height", UP], AUTHORITY["EPSG", "5713"]],
  AUTHORITY["DIGGS", "63226405_5713"]]

```

The DIGGS KML Visualization Tool (see Section 7.4) makes use of the free, open source GeoTools geospatial toolkit and was extended by the DIGGS WKT Compound CRS definitions to support the transformation of any of the DIGGS Compound CRSs to the KML CRS.

CRS Component Summary Spreadsheet

The CRS component spreadsheets are human readable lists of horizontal and vertical CRS components extracted from the EPSG database, which are used by DIGGS data in North America (US and Canada), the UK, and the World. These spreadsheets are used as inputs to an XSLT script that automatically generates the GML and WKT CRS dictionaries for DIGGS and are located at the following paths in the DIGGS2.0a directory.

```
\Dictionaries\CRS\input\North American CRS.xls
\Dictionaries\CRS\input\UK CRS.xls
\Dictionaries\CRS\input\World CRS.xls
```

The first worksheet in each of the spreadsheets listed above itemizes the horizontal (2D), vertical (1d) and Compound (3D) CRS's that are used by DIGGS in the OGP/EPSG database as shown in Figure 4-8, Figure 4-9 and Figure 4-10.

coord_ref_sys_code	coord_ref_sys_name	coord_ref_sys_kind	area_name
2038	NAD83(CRSRS98) / UTM zone 20N	projected	Canada - 66 to 60 deg W
2961	NAD83(CRSRS) / UTM zone 20N	projected	Canada - 66 to 60 deg W
2037	NAD83(CRSRS98) / UTM zone 19N	projected	Canada - 72 to 66 deg W
2960	NAD83(CRSRS) / UTM zone 19N	projected	Canada - 72 to 66 deg W
2955	NAD83(CRSRS) / UTM zone 11N	projected	Canada - Alberta W of 114 deg W
2153	NAD83(CRSRS98) / UTM zone 11N	projected	Canada - Alberta W of 114 deg W

Figure 4-8: Excerpt of Horizontal CRS Components used by DIGGS in North America

They are ordered by area name for convenient browsing (as opposed by sorting by the EPSG code). The second worksheet lists the 4 vertical coordinate systems to combine with each of the horizontal ones.

coord_ref_sys_code	coord_ref_sys_name	coord_ref_sys_kind	area_of_use_code	area_name
5713	Canadian Vertical Datum of 1928	vertical	1289	Canada - CVD28
5703	North American Vertical Datum of 1988	vertical	1730	North America - NAVD88
5702	National Geodetic Vertical Datum of 1929	vertical	1323	USA - conus
5714	mean sea level height	vertical	1262	World

Figure 4-9: Excerpt of Vertical CRS Components used by DIGGS in North America

The third worksheet lists all of the compound CRS's that were defined in the EPSG database.

coord_ref_sys_code	coord_ref_sys_name	coord_ref_sys_kind	area_of_use_code	area_name
7407	NAD27 / Texas North + NGVD29	compound	2253	USA - Texas - SPCS - N
7406	NAD27 + NGVD29	compound	1323	USA - conus

Figure 4-10: Excerpt of Compound CRS Components used by DIGGS in North America

Each resultant compound CRS is a combination of a horizontal CRS with each of the vertical CRSs (with intersecting domains of validity) yielding a total of 2781 DIGGS Compound CRSs. An example instance of such a compound CRS (26911_5703) has horizontal component EPSG:26911 (Easting, Northing) in meters (UTM zone 11N) and vertical component EPSG:5703 (Height) in meters (NAVD 88). When queried from the DIGGS Demo CRS Registry Service the ebRIM response can be viewed as a web page as shown in Figure 4-11.

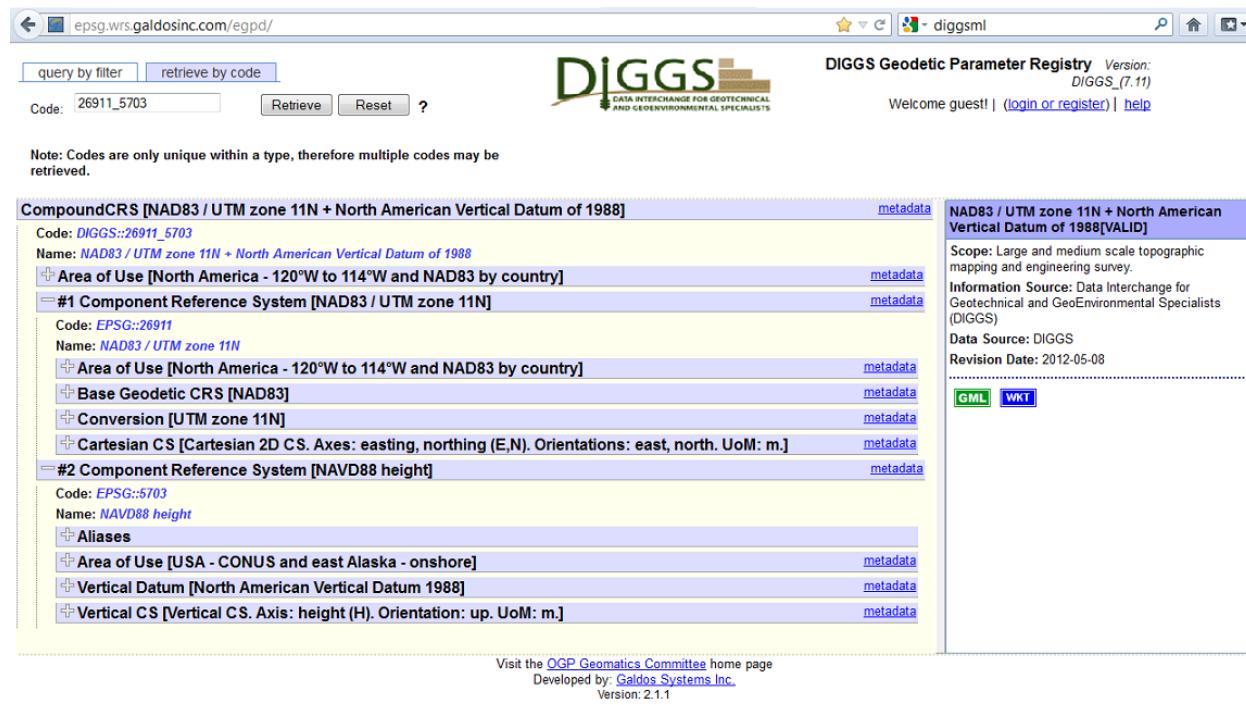


Figure 4-11: DIGGS 3D CRS Components Viewed by DIGGS Demo CRS Registry Service

The machine readable GML and WKT encodings for the selected CRS definition can be retrieved over the web by URL (e.g. the ‘GML’ and ‘WKT’ buttons on the lower right of Figure 4-11) or programmatically via a simple HTTP GET or POST query.

4.2.3.2 Units

The units of measure used in DIGGS can include any of the standard SI units (see <http://www.bipm.org/en/si/>), additional derived units listed in the online Unified Code for Units of Measure (UCUM) dictionary (see <http://unitsofmeasure.org/>), or the more geotechnical domain specific unit definitions imported by DIGGSML in WITSML V.1.3.1.1. Such domain specific unit definitions are located in the following WITSML schema files and can be made retrievable over the web as was done for the DIGGS CRS components (see Section 4.2.3.1).

`typ_baseType.xsd`
`typ_catalog.xsd`

typ_dataTypes.xsd
typ_measureType.xsd
typ_quantityClass.xsd

The WITSML units are used to quantify measures such as the following:

accelerationLinear
anglePerLength
anglePerTime
area
areaPerArea
density
dimensionless
dynamicViscosity
electricCurrent
electricPotential
energyPerArea
equivalentPerMass
force
forcePerLength
forcePerVolume
frequency
illuminance
length
lengthPerLength
magneticFieldStrength
magneticInduction
massConcentration
mass
massPerLength
momentOfForce
perLength
planeAngle
power
pressure
relativePower
specificVolume
thermodynamicTemperature
time
velocity
volume
volumeFlowRate
volumePerVolume

For example, the force measure can be specified by units with the following abbreviations:

N, daN, dyne, gf, kdyne, kgf, klbf, kN, lbf, Mgf, mN, MN, ozf, pdl, tonfUK, tonfUS, uN

4.2.4 CodeLists

Code lists are controlled vocabularies used by DIGGS property values. Such controlled vocabularies are used to avoid errors and ambiguities often found in data that make use free text values. An example of such a code list would be all the types of chemical determinands that can be observed from sample test readings.

The code lists were generated as an XML encoding automatically from a summary spreadsheet maintained in Excel (*DIGGSCodeTypes.xlsx*) including the code lists originally derived from the DIGGS V1.0a dictionaries.

1	Name of List	List URN	Code
7	Compaction Mould Type	urn:diggs:def:codelist:AGS:compaction_mould_type	Standard
8	Compaction Mould Type	urn:diggs:def:codelist:AGS:compaction_mould_type	CBR
9	Abundance	urn:diggs:def:codelist:BS5930:abundance	Little
10	Abundance	urn:diggs:def:codelist:BS5930:abundance	Some
11	Abundance	urn:diggs:def:codelist:BS5930:abundance	Trace
12	Abundance	urn:diggs:def:codelist:BS5930:abundance	Rare
13	Abundance	urn:diggs:def:codelist:BS5930:abundance	Occasional
14	Abundance	urn:diggs:def:codelist:BS5930:abundance	Much
15	Abundance	urn:diggs:def:codelist:BS5930:abundance	And
16	Driven Penetration Test Type	urn:diggs:def:codelist:AGS:driven_penetration_test_type	S
17	Driven Penetration Test Type	urn:diggs:def:codelist:AGS:driven_penetration_test_type	C
18	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Unknown
19	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Tangent method
20	Modulus Calculation Method	urn:diggs:def:codelist:DIGGS:modulus_calculation_method	Secant method
21	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	246TCP
22	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	AG
23	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	DST
24	Chemical Determinand	urn:diggs:def:codelist:AGS:chemical_determinand	2MNP

Figure 4-12: Example Code Lists Viewed as Spreadsheet

The spreadsheet (*DIGGSCodeTypes.xlsx*) shown in Figure 4-12 contains all codes from the DIGGS1.0a code lists plus additional enumerations and codes added in v2.0a, which are categorized into three types A, B, and C as summarized in the following table.

Type	Description	Proposed DIGGS Implementation
A	Codes to describe in more detail a specific data element, where the data element cannot be controlled or validated by the schema alone (e.g. table data and CPT parameter names).	If the code is absolutely necessary for DIGGS to function and be unambiguous for source and target data interchange, then these codes should be implemented into enumerated lists. Enumerated lists are part of the schema and are validated by schema alone.
B	Codes created, maintained, and published by recognized standards organizations, used in practice, and commonly referenced with or without software (e.g. USCS Group Symbols for soil classification, Munsell color codes, EPSG spatial reference codes).	For codes that are commonly referenced, nomenclature and abbreviations well documented, and maintained by a standards body, these should be implemented in DIGGS using codetype and codespace attributes. DIGGS might require that some codetype and codespace attributes be mandatory. Although the codespace would reference the standards organization (e.g. USCS, AASHTO), the full list of codes (e.g. SP, SW) would not be in the codelist, since the standards organization maintains this list, and it would be left to the users to comply with the standards published by that standards organization.

C	<p>Codes created by an organization, government agency, trade group, or company to standardize nomenclature and terms across a specific user base (e.g. roles, titles, equipment names, test names).</p>	<p>Codes that are used in localized practice should be made available for integration into DIGGS as needed. Codespace and codetype attributes would be optional. This would be applicable, for example, for codes such as “roles” where the value itself likely carries meaning without other external references. However, specific user groups may want to standardize the possible values being used. Three possibilities:</p> <ul style="list-style-type: none">• DIGGS file authors could simply use codes (uncontrolled) without any reference to a codetype or codespace. However, the recipient of the DIGGS file would not know what standards are being referenced.• The DIGGS author could populate the codetype and codespace attributes. Since these are optional and the format uncontrolled, the recipient may still be unable to resolve the references in a systematic manner.• The DIGGS author could reference a published codespace that can be validated with schematron.
----------	--	--

In DIGGS2.0a a new XML encoding was used. The DIGGS1.0a code lists (e.g. *agsCodeList_V1.xml*) were converted from a GML Dictionary encoding to the XML encoding called ebRIM, which has international language support standardized by the international OASIS standards body and adopted by the OGC as a Registry Information Model (RIM). The ebRIM encoding is a machine readable XML encoding that was designed for publishing and sharing common information resources such as code lists and dictionaries over the web. The advantages of using ebRIM are: that it is an XML encoding; includes support for international languages, discovery and life cycle management of the information; and can easily be viewed in human readable formats, such as spreadsheets, or on a web page as shown in Figure 4-13.

Chemical Determinand (ClassificationScheme)

AGS

id	urn:x-diggs:def:code-list:chemical_determinand
objectType	urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObjectClassificationScheme
status	Submitted
isInternal	true
nodeType	urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode
xml view	Brief , Summary , Full
tree view	Children , Full Tree
associations	Associations To , Associations From
audit trail	All Audit Events

246TCP (ClassificationNode)

2,4,6 - Trichlorophenol

id	urn:x-diggs:def:code-list:chemical_determinandid_246tcp
objectType	urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObjectClassificationNode
status	Submitted
parent	urn:x-diggs:def:code-list:chemical_determinand
code	id_246TCP
path	/urn:x-diggs:def:code-list:chemical_determinand/id_246TCP
xml view	Brief , Summary , Full
tree view	Children , Full Tree
associations	Associations To , Associations From
audit trail	All Audit Events

Figure 4-13: 246TCP Code in Chemical Determinand Code List. Encoded as ebRIM and Viewed as HTML in a Web Browser

All of the DIGGS codelists are encoded in ebRIM XML, have human readable visualizations (e.g. Excel spreadsheet in Figure 4-12 and web page HTML in Figure 4-13) and can be made retrievable programatically over the web as was done for the DIGGS CRS components (see Section 4.2.3.1).

5 DIGGS 2.0a Feature Model

DIGGS2.0a defines 8 base classes of features (as shown in Figure 5-1 below) that can be contained as a child under the root DIGGS element. This classification is formalized in the schemas so that all existing features in DIGGS are categorized by derivation from these base classes. The existing features in DIGGS2.0a are the commonly used and requested features by the geotechnical and geoenvironmental communities, e.g. Project, Borehole, Sample, SamplingActivity etc, and are discussed in detail in Section 5.2-DIGGS Feature Classes and Objects.

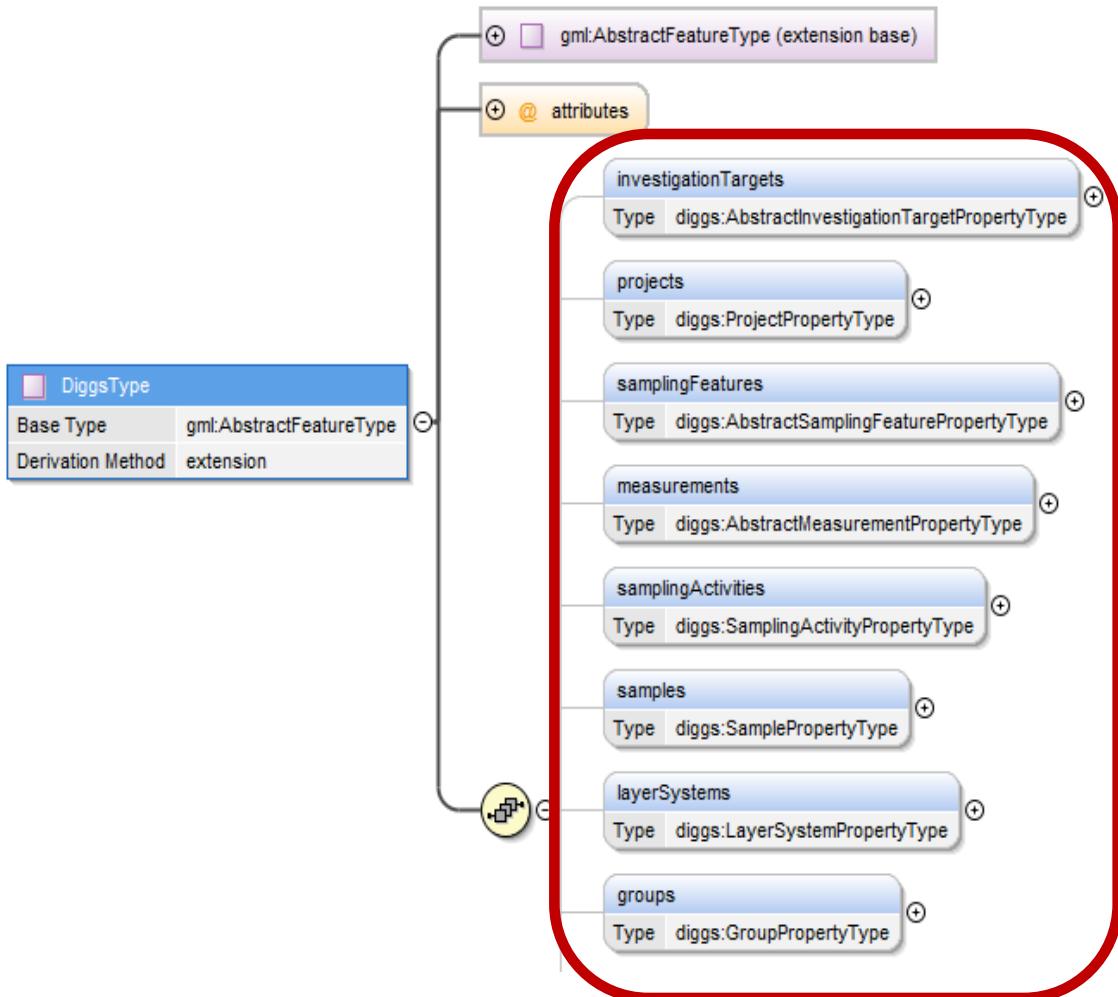


Figure 5-1: The Diggs Element and its Properties DIGGS 2.0a

The 8 base feature classes are classified by Processes, Entities, and Groups as follows:

- 1) **InvestigationTarget** –target features of interest being sampled/measured [Entity]
- 2) **Project** - business activities that collect, compile, and process information from locations [Process]

- 3) **SamplingFeature** - real world places and constructions (e.g. Boreholes) from which observations are made, samples are collected, or tests are run. [Entity]
- 4) **Measurement** – test readings (in-situ or not) taken from samples collected from sampling features, or created via a sampling activity [Process]
- 5) **SamplingActivity** - the process of sample creation or collection [Process]
- 6) **Sample** - earth material, fluids, or gases collected or created for observation and testing [Entity]
- 7) **LayerSystem** - ordered interval observations or interpretations of earth materials, properties or features at a location [Entity]
- 8) **Group** - collections of projects, locations, samples or groups of these, for the purpose of providing meaningful context to observations and measurements.

5.1 Feature Properties and Attributes

Optional properties of all objects include status, description, and remarks metadata; and all features include additional optional properties including associated file and role metadata objects. Projects, Sampling Features, Samples, Layer Systems, Sensors, and Groups are "named" features – in addition to the identifiers and other properties, they also carry a mandatory name property.

Some DIGGS objects are also named (i.e. carry a mandatory name property) including some of the Layers and all of the Metadata objects.

5.1.1 Named Features and Inherited Properties

The Named Features class, represented by the diggs:AbstractNamedFeature element, is an abstract class of features with common properties and attributes that all named features inherit including: gml:id, gml:description, gml:name, gml:identifier, status and a group of metadata properties: associatedFiles, roles, and remarks, as shown in Figure 5-2.

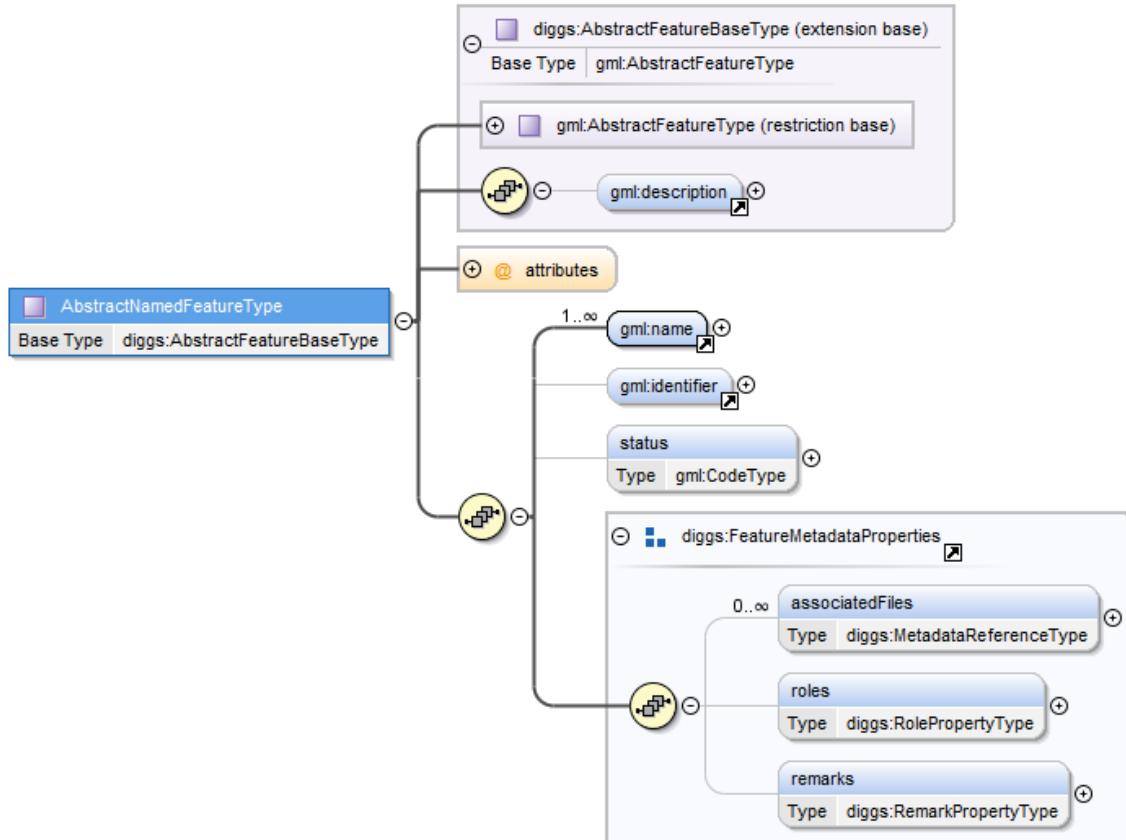


Figure 5-2: The Named Feature Class and its Properties

The **gml:id** attribute and **gml:description** property are inherited from a higher abstract feature class called **diggs:AbstractFeatureBase** as shown in Figure 5-3.

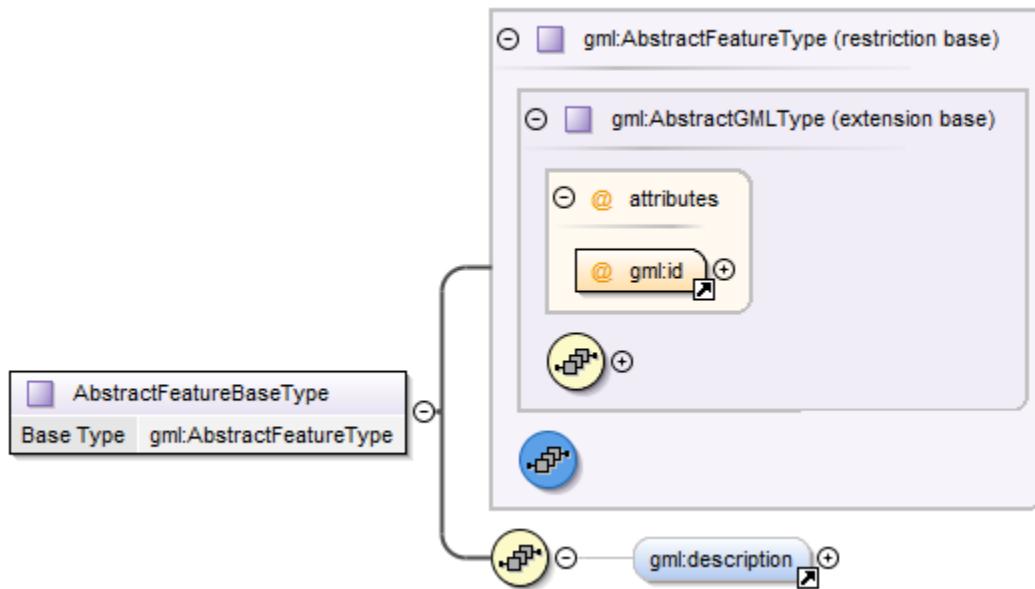


Figure 5-3: The Abstract Feature Base Class and its Properties

5.2 DIGGS Feature Classes and Objects

The special root level Diggs feature element is at the top of the feature instance hierarchy and its properties organize DIGGS data into logical components: projects, samplingFeatures, measurements, samplingActivities, samples, and layerSystems. All other features such as Property, Borehole, etc. are at the same level in the hierarchy (siblings) and instantiated as the descendent property children of the Diggs feature element. New to DIGGS 2.0a, features are no longer nested inline inside of a parent Project element as in some previous versions of DIGGS.

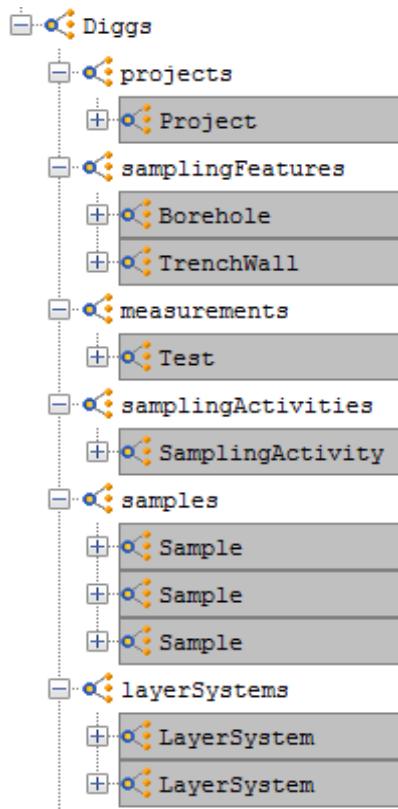


Figure 5-4: Organization of DIGGS Feature Data into Logical Components: projects, samplingFeatures, measurements, samplingActivities, samples, and layerSystems

Instead, a typical feature in DIGGS is now associated in V2.0a to one or more Project features and possibly to other features by reference. For example, consider a single Project that has one Borehole as in the following XML example instance. Note that the Project feature references the associated Borehole sampling feature and vice-versa as illustrated in Figure 5-5.

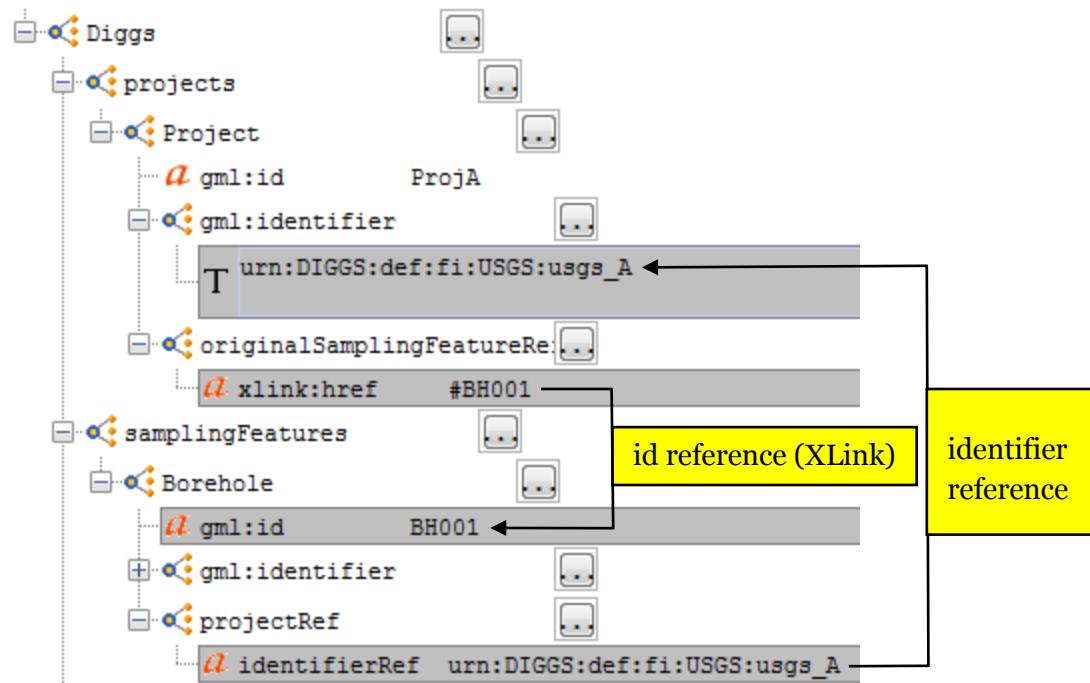


Figure 5-5: Example Feature Associations in DIGGS 2.0a

Some illustrative remarks about the example instance of Figure 5-5:

- The Project and the Borehole are at the same level in the feature hierarchy.
- The Project (gml:id="ProjA") has a property element called originalSamplingFeatureRef, which points to the Borehole feature instance with gml:id="BH001" (in the short XLink/XPointer notation). Such a "simple" link is similar to the common link on an HTML page that points to another Web page or to a resource to download
- The Borehole sampling feature "BH001" has a property element reference called projectRef to Project "ProjA".

5.2.1 Projects

The Project feature is a concrete (instantiable) feature with no subclasses. The Project feature must be included in every Diggs document and all other features (samples, holes, etc.) must belong to a single project in a Diggs document. Therefore, a single Project feature is mandatory in all DIGGS instances and all other features in the DIGGS file have a projectRef property and must reference the single Project, as illustrated as a feature association in the previous Figure 5-5.

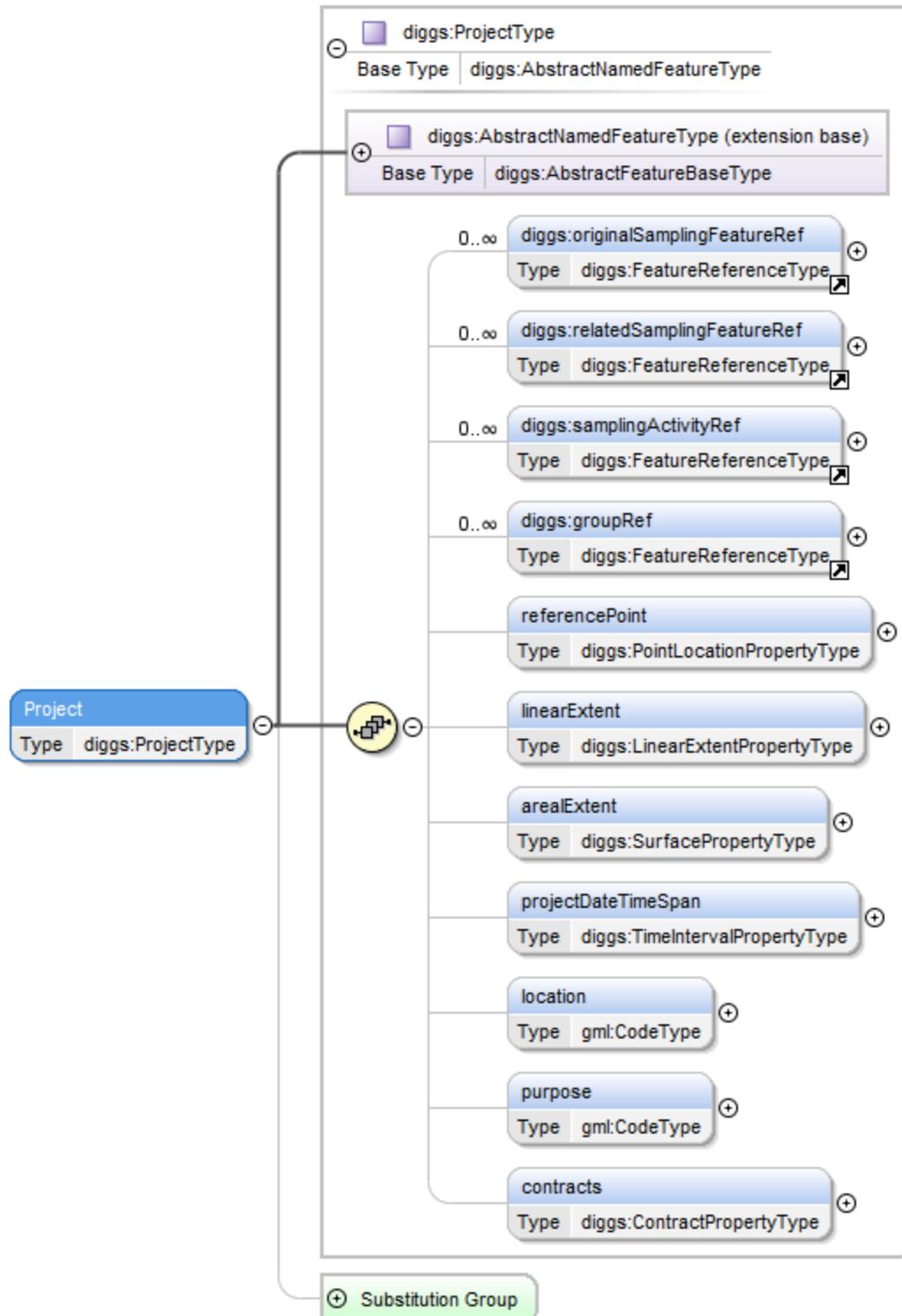


Figure 5-6: The Project Feature and its Properties

The Project feature has the properties: originalSamplingFeatureRef, relatedSamplingFeatureRef, samplingActivityRef, groupRef, referencePoint, linearExtent, areaExtent, projectDateTimeSpan, location, purpose, and contracts, as shown in Figure 5-6.

The first 4 properties with the ‘Ref’ suffix are used to reference, as indicated by the property name, the corresponding features (either by `gml:identifier` or `gml:id`): SamplingFeature, SamplingActivity, and Group. Each of the referenced features are described in following sections of the document.

The referencePoint property contains a PointLocation geometry with coordinates that could approximate the location of the Project on the earth/globe. The PointLocation for a project is typically defined by 3D coordinates in an earth-based Coordinate Reference System (CRS) (e.g. a map projection with height/depth, a Geodetic CRS (geodetic latitude, longitude, elevation), or a local Engineering CRS). Alternatively, but less typically for projects, the PointLocation can be referenced as a distance along another well-known piece-wise linear curve (see the linear referencing discussion in Section 5.2.2.1 and Figure 5-10), such as the linear extent geometry of the project.

The linearExtent property contains a linear curve geometry encoded by the `diggs:LinearExtent` that represents the shape and location of the linear extent of the project site. The LinearExtent geometry records control point positions (joined together by linear interpolation), which are each encoded a 3D earth based CRS.

The areaExtent property contains a surface polygon geometry typically represented by a `gml:Polygon` that represents the shape and location of the area extent of the project site. The Polygon geometry records control point positions (each typically encoded a 3D earth based CRS) forming a closed linear ring representing the exterior boundary of the area extent (joined together by linear interpolation). Note that the polygon geometry contains all the points lying within the exterior boundary, so it makes sense to calculate the area (e.g. in square meters) of the polygon and hence the area extent of the project site. Alternatively, but less typically for projects, the Polygon can be referenced along a planar surface formed by the extrusion of the project’s linearExtent (see the planar referencing discussion in Section 5.2.2.3 and Figure 5-26).

The projectDateTimeSpan records the (estimated) active time interval of the project.

The location property provides a location keyword for the project using a text value from a controlled list.

The purpose property provides a text keyword description of the purpose for the project that is intended to come from a controlled list.

The contracts property contains details about the contract(s) associated with the project, including the type of contract, references to files/documents, clients, and contractors.

An example instance of a Project is shown in Figure 5-7.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	p1
5	Name		Test Project
6	Remark		#temp-id-1
7	Related Sampling Feature Ref		#LB_Webster
8	Related Sampling Feature Ref		#a22
9	Related Sampling Feature Ref		#cpt-1
10	Sampling Activity Ref		#zyx
11	Group Ref		#g1
12			

Navigation buttons: << < > >> Project Remark Borehole Role PointLocation Location

Figure 5-7: Example Project Instance Viewed with DIGGS Excel Tool

5.2.2 Sampling Features

Sampling features are in an abstract (non-instantiable) class of features with common properties and attributes that all sampling features inherit including: investigativeTargetRef, projectRef, originalProjectRef, associatedProjectRef, groupRef, measurementRef, layerSystemRef, samplingActivityRef and sensorRef. All these inherited properties with the ‘Ref’ suffix are used to reference the associated features (either by gml:identifier or gml:id) as indicated by the property name.

Sampling features are further divided into 3 abstract subclasses based on their geometry type: Point, Linear and Planar as shown in Figure 5-8.

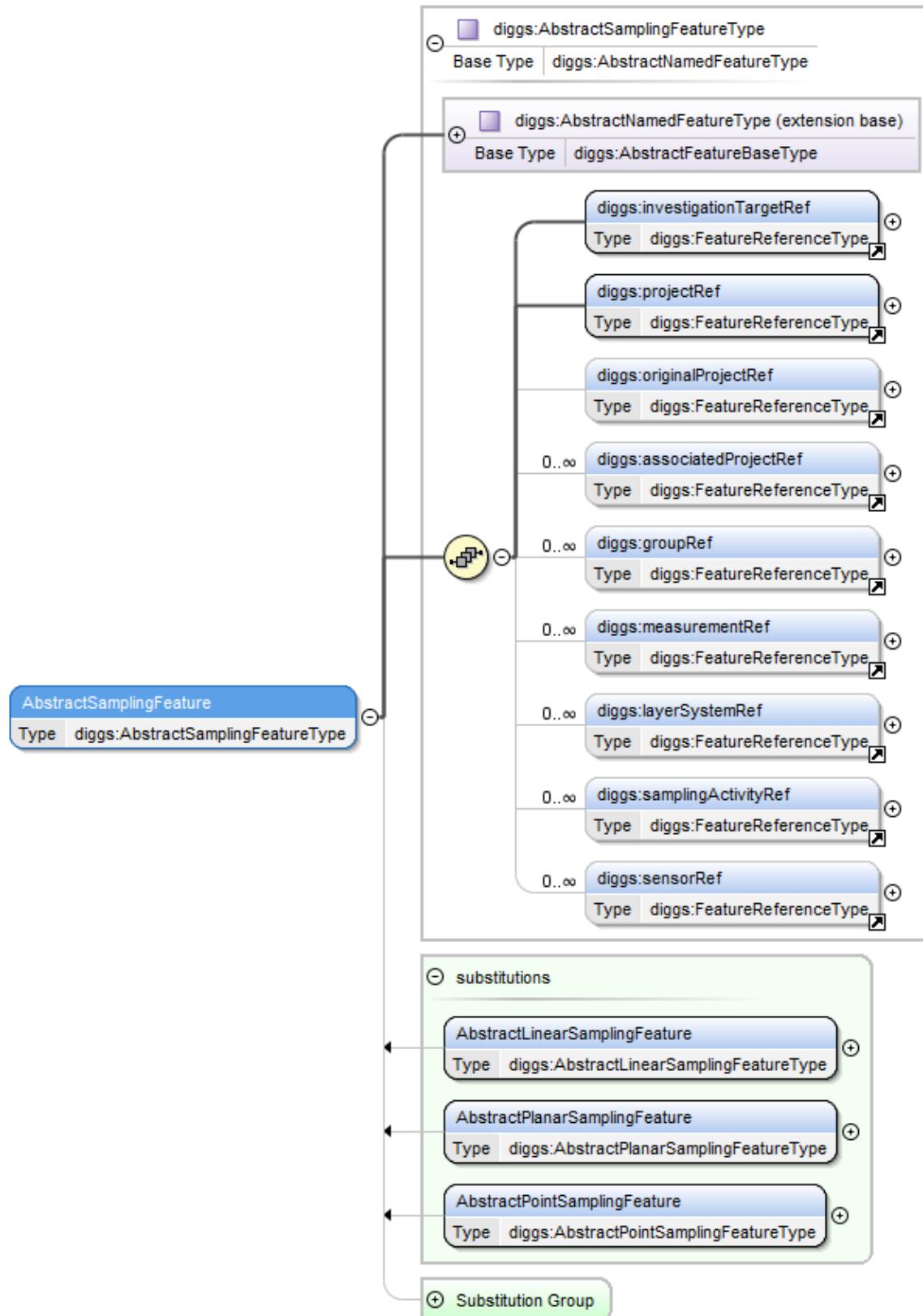


Figure 5-8: The Abstract Sampling Feature Class and its Properties and Subclasses

Each subclass of the sampling feature (Point, Linear, and Planar) is described separately in the following subsections.

5.2.2.1 Point Sampling Features

The Point Sampling feature class is an abstract class represented by the AbstractPointSamplingFeature element that inherits all the properties from the Sampling feature class. The Point Sampling feature class defines common properties and attributes that all point sampling features will inherit including: referencePoint, referencePointAccuracy, and linearReferencing as shown in Figure 5-9.

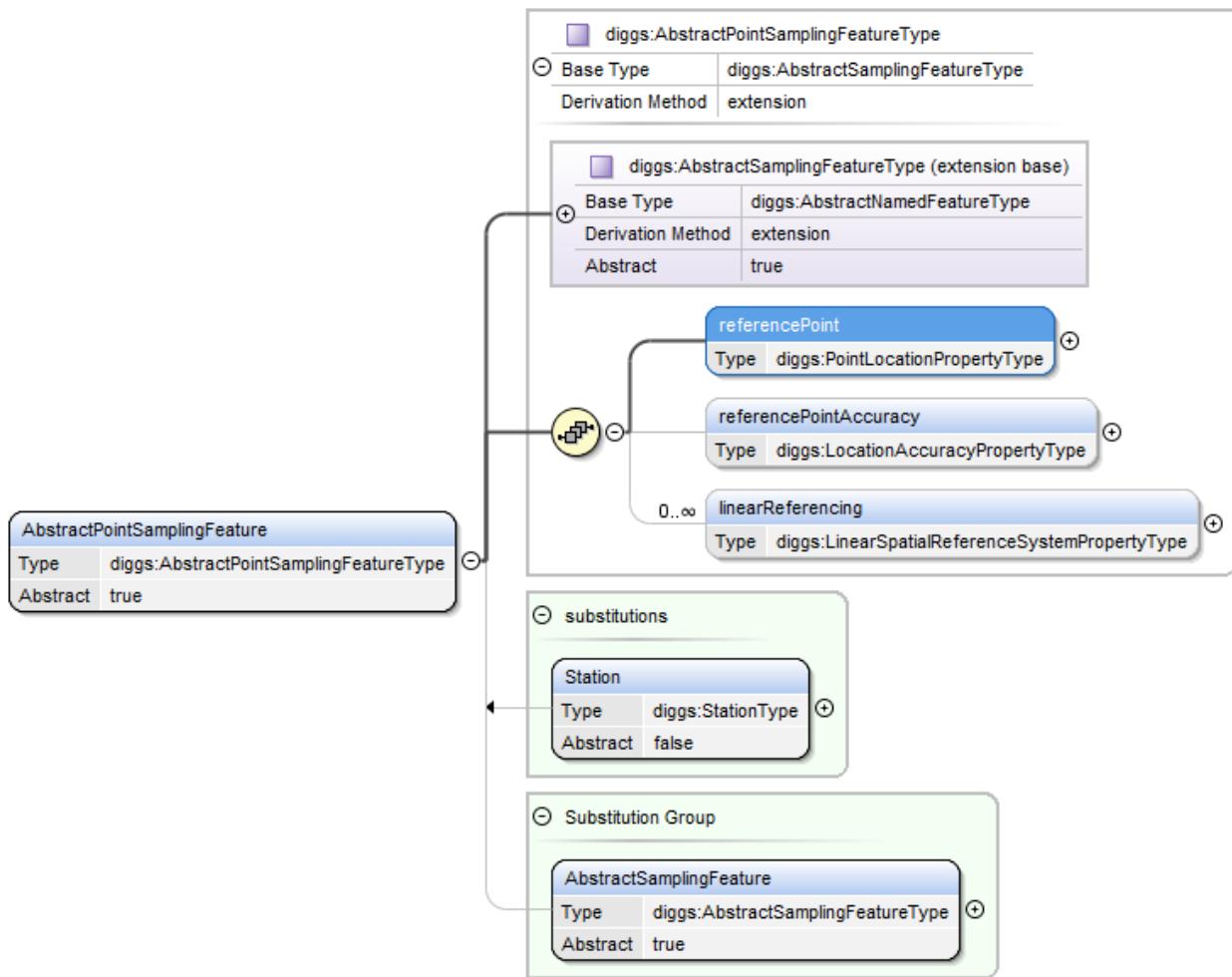


Figure 5-9: The Point Sampling Feature Class and its Properties and Subclass (Station)

The **referencePoint** property contains a **PointLocation** geometry that represents the location of the sampling feature. The **PointLocation** can be defined as a list of coordinates or can be referenced as a distance along a piece-wise linear curve (**gml:LineString**). In either case, the positional accuracy is

recorded by the `referencePointAccuracy` property. In the case where the position is referenced along an existing well-known curve, the `linearReferencing` property describes the reference curve, the starting point for measuring along the curve, and the units of measure (see Section **Error!**

Reference source not found. for the details of the GML encoding of reference positions along a curve and the following examples) as illustrated in Figure 5-10.

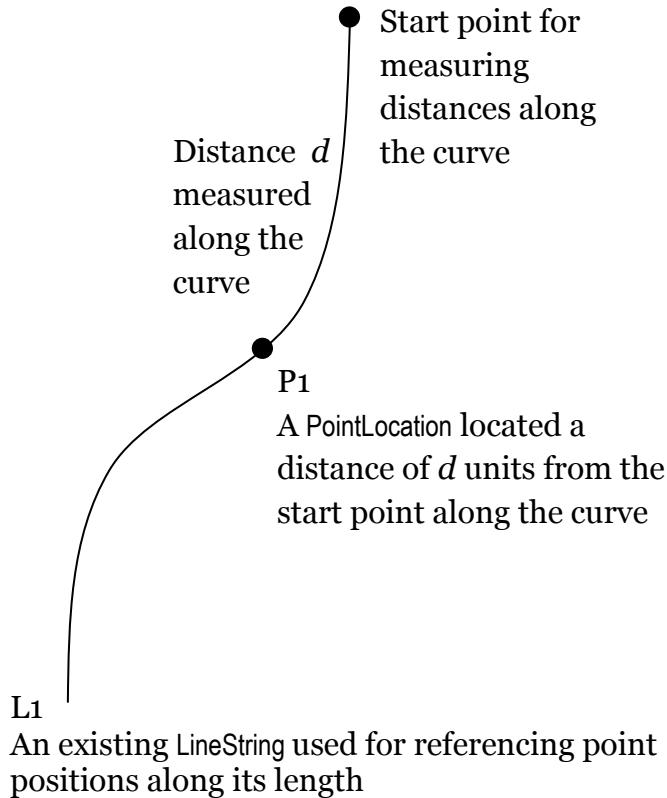


Figure 5-10: A Linear Referenced PointLocation (P1) at a Distance *d* from the Start Along a Curve

An example excerpt of the PointLocation P1 could be as follows:

```
<diggs:PointLocation gml:id="P1" srsDimension="1" srsName="#LinearReferencingSRS1">
  <gml:pos>d</gml:pos>
</diggs:PointLocation>
```

Note that the PointLocation P1 references a Linear Reference System (defined by the `LinearSRS` element) in its `srsName` attribute (`LinearReferencingSRS1`), which can be used to represent a sample or activity location, a hole depth, or a general reference point for a feature. Note also that *any* point geometry in GML, besides the `diggs:PointLocation`, can make use of linear referencing along a curve by using the `srsName` attribute to reference an appropriate `LinearSRS`.

The Linear Referencing system in turn specifies the `linearElement`, which is the reference curve L1 in this case as shown in Figure 5-10, and the linear referencing method, which is typically

measured from the start of the curve in specified units (meters in the example). The Linear Referencing system can be represented as shown in the following example instance.

```
<gmllr:LinearSRS gml:id="LinearReferencingSRS1">
  <gmllr:linearElement>
    <gml:LineString srsName="urn:def:crs:EPSG:4326" srsDimension="3" gml:id="L1">
      <gml:posList>x1 y1 z1 x2 y2 z2 ... x50 y50 z50</gml:posList>
    </gml:LineString>
  </gmllr:linearElement>
  <gmllr:lrm>
    <gmllr:LinearReferencingMethod gml:id="LRM001">
      <gmllr:name>chainage</gmllr:name>
      <!--chainage = measurement along curve in metres -->
      <gmllr:type>absolute</gmllr:type>
      <!--absolute = measure from start of linear element -->
      <gmllr:units uom="m"/>
    </gmllr:LinearReferencingMethod>
  </gmllr:lrm>
</gmllr:LinearSRS>
```

Note that the LineString L1 was specified by absolute geographic 3D coordinates (x1, y1, z1), (x2, y2, z2), ..., (x50, y50, z50) and the PointLocation P1 was specified as a relative position measured a distance of d units (meters) along the curve L1 from the start point (which is the beginning of the curve by default). Example instances of PointLocation elements are shown in Figure 5-11.

	A	B	C
3	Property Name	Attribute Name	Property Value
20		ID:	a34
21		SRS Name:	urn:diggs:def:crs:DIGGS:0.1:26911_5703
22		SRS Dimension:	3
23	Pos		387516.665116977 3742645.12297961 500
24		ID:	cpt1
25		SRS Name:	urn:diggs:def:crs:DIGGS:0.1:26911_5703
26		SRS Dimension:	3
27	Pos		387416.665116977 3742645.12297961 6
28		ID:	cptd
29	Pos		20
30		SRS Name:	#cptsr1
31		SRS Dimension:	1
32		ID:	pt1-1
33	Pos		30
34		SRS Name:	#sr123
35		SRS Dimension:	1
36		ID:	pt1-2
37	Pos		30
38		SRS Name:	#sr123

H
◀
▶
M
PointLocation
LocationAccuracy
LinearExtent
LinearSpatialReferenceSy

Figure 5-11: Example PointLocation Instances Viewed by DIGGS Excel Tool

The first two PointLocation instances (with gml:id's: a34 and cpt1) in Figure 4-11 are defined in absolute 3D coordinates (Northing, Easting, Height) in meters (and were used as the CRS example of Figure 4-11). The latter three PointLocation instances shown in Figure 5-11 (with gml:id's: cptd, pt1-1 and pt1-2) are defined as distances along two different LineString curves as defined in the Linear Spatial Reference System instances shown in Figure 5-12.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	sr123
5	Identifier		urn:diggs:def:feature:DIGGS:lsr123
6		Code Space:	urn:diggs:def:codelist:DIGGS:featureIDs
7	Linear Element		#ls
8	Linear Referencing Method		#lr123
9	Location Accuracy		#temp-id-5
10	Location Accuracy		#temp-id-6
11		ID:	cptsr1
12	Identifier		urn:diggs:def:feature:DIGGS:cptsr1
13		Code Space:	urn:diggs:def:authority:DIGGS
14	Linear Element		#ls1
15	Linear Referencing Method		#lrcpt1
16			

LinearSpatialReferenceSystem
LinearReferencingMethod
Backfill
TimeI

Figure 5-12: Example LinearSpatialReferenceSystem Instances Viewed by DIGGS Excel Tool

Station

The Station feature is a concrete point sampling feature with no subclasses. The Station represents a point on the earth's surface where observations or tests are performed, where samples are collected, or where monitoring devices are installed.

The Station feature has the type property, as shown in Figure 5-17, to specify the kind of station feature it is (eg. outcrop, survey marker, etc.), which is intended to come from a list of codes by a controlling authority.

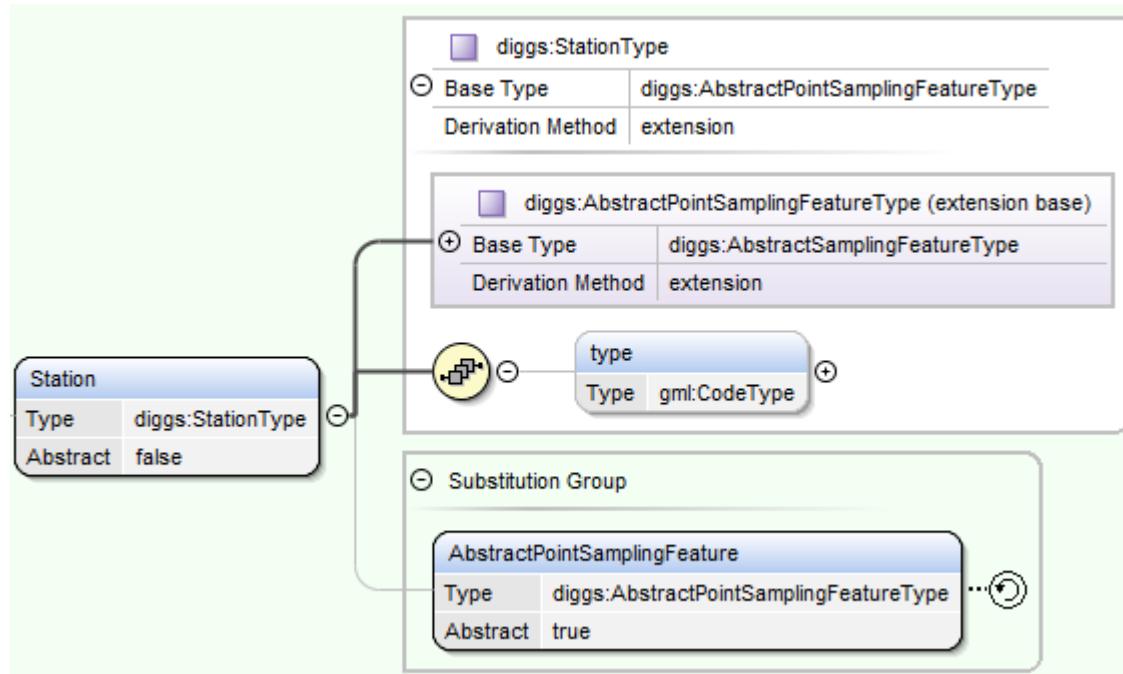


Figure 5-13: The Station Feature and its Properties

Putting together the linear reference examples in the context of a Station, with a reference point and linear reference system encoding could be as follows:

```
<diggs:Station gml:id="Station001">
...
<diggs:referencePoint>
  <diggs:PointLocation gml:id="P1" srsDimension="1" srsName="#LinearReferncingSRS1">
    <gml:pos>d</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>
<diggs:referencePointAccuracy>
  <diggs:LocationAccuracy>
    <diggs:measurementMethod
      codeSpace="http://www.oxygenxml.com/">measurementMethod0</diggs:measurementMethod>
    <diggs:result uom="uom1">0</diggs:result>
  </diggs:LocationAccuracy>
</diggs:referencePointAccuracy>
<diggs:linearReferencing>
  <gmllr:LinearSRS gml:id="LinearReferncingSRS1">
    <gmllr:linearElement>
      <gml:LineString srsName="urn:def:crs:EPSG:4326" srsDimension="2" gml:id="L1">
        <gml:posList>x1 y1 x2 y2 x3 y3 ... x50 y50</gml:posList>
      </gml:LineString>
    </gmllr:linearElement>
  <gmllr:lrn>
    <gmllr:LinearReferencingMethod gml:id="LRM001">
      <gmllr:name>chainage</gmllr:name>
```

```

<!--chainage = measurement along curve in metres -->
<gmllr:type>absolute</gmllr:type>
<!--absolute = measure from start of linear element -->
<gmllr:units uom="m"/>
</gmllr:LinearReferencingMethod>
</gmllr:lrm>
</gmllr:LinearSRS>
</diggs:linearReferencing>
<diggs:type>outcrop</diggs:type>
</diggs:Station>

```

5.2.2.2 Linear Sampling Features

The Linear Sampling feature class is an abstract class represented by the AbstractLinearSamplingFeature element that inherits all the properties from the Sampling feature class. The Linear Sampling feature class defines common properties and attributes that all point sampling features will inherit including: referencePoint, referencePointAccuracy, centerLine, and linearReferencing, as shown in Figure 5-14.

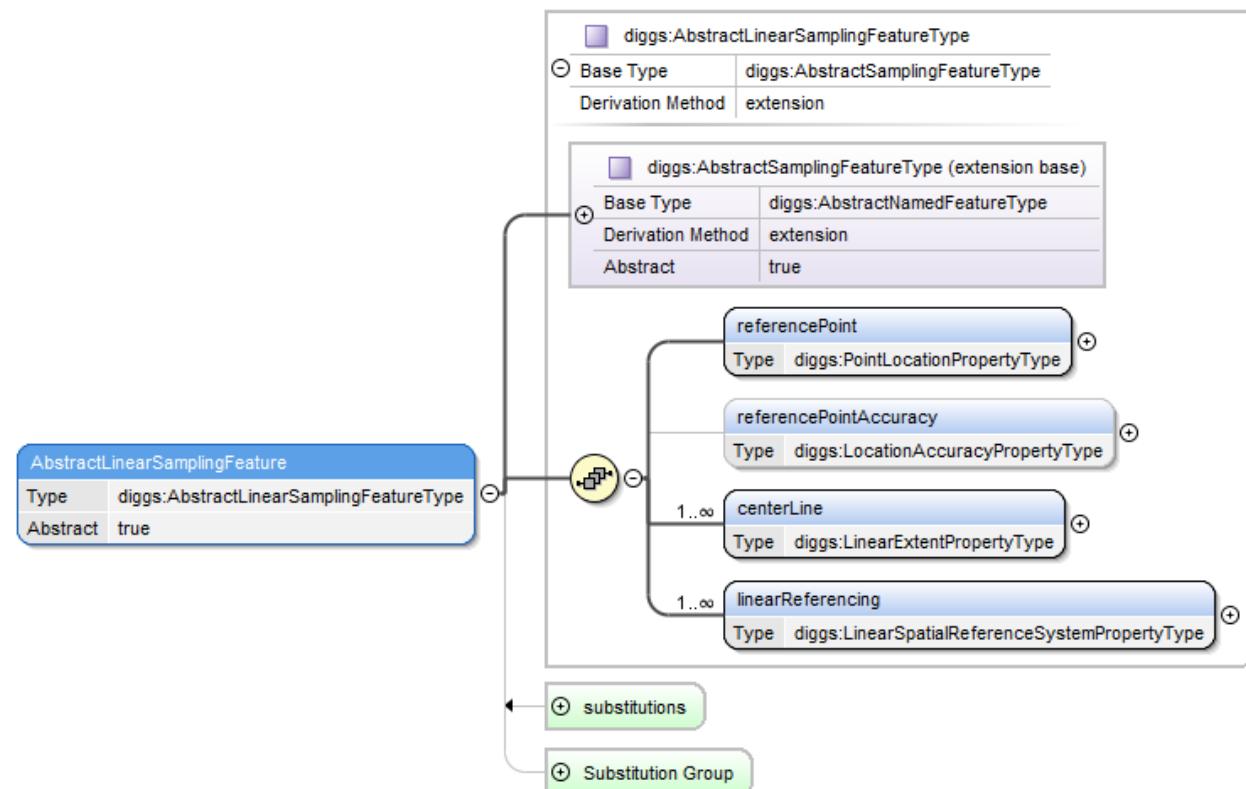


Figure 5-14: The Linear Sampling Feature Class and its Properties

The referencePoint, referencePointAccuracy and linearReferencing properties serve the same purpose as for Point sampling features (see Section 5.2.2.1).

The centerLine property contains a diggs:LinearExtent, which is a linear curve geometry that represents the shape and location of the sampling feature. The LinearExtent geometry can be defined by a list of direct positions in a coordinate reference system or it can be referenced (e.g. in the case of a LithologyLayer or a Well feature) along another existing, well-known curve (e.g. the parent Borehole's center line geometry), where the direct positions of the LinearExtent are just linear referenced in the same way as for Point Sample features. An example of such a LinearExtent referenced along another curve is illustrated in Figure 5-15.

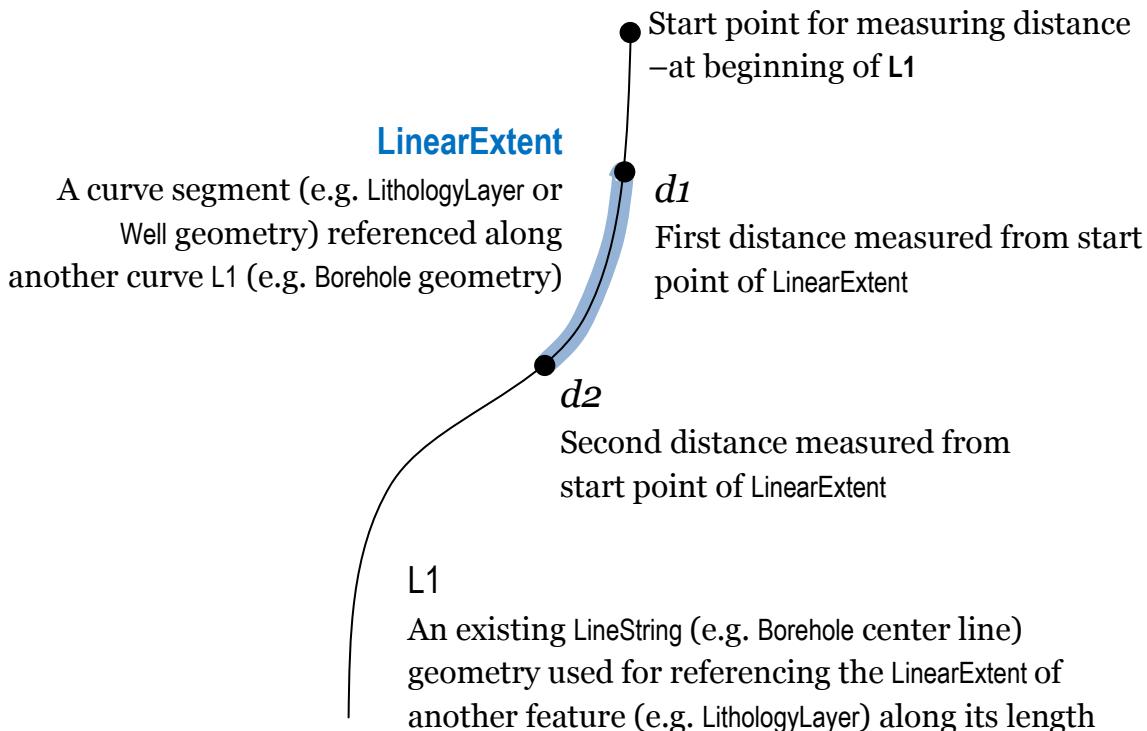


Figure 5-15: The **LinearExtent Segment can be Referenced Along Another Curve **L1****

An example instance of the **LinearExtent** could be as follows:

```
<diggs:LinearExtent id="LinearExtent001" srsDimension="1" srsName="#LinearReferencingSRS1">
  <gml:posList>d1 d2</gml:posList>
</diggs:LinearExtent>
```

The **LinearExtent** references a linear reference system (**LinearSRS**) in its **srsName** attribute (**LinearReferencingSRS1**), which in this case is the same as for Figure 5-10.

The concrete subclasses of the linear sampling features are: Borehole, Transect, TrialPit, and Well, as shown in Figure 5-16.

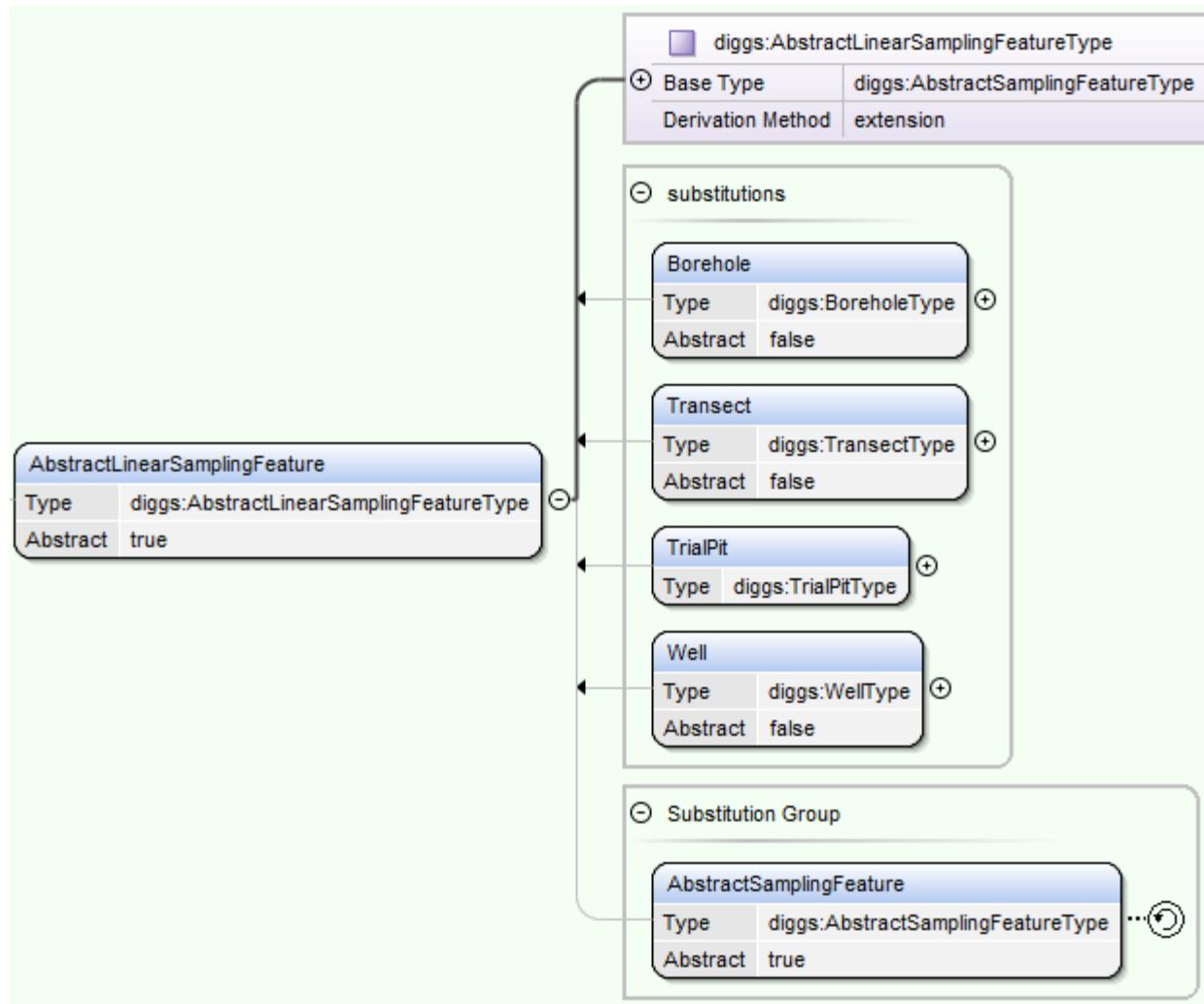


Figure 5-16: The Linear Sampling Feature Class and its Subclasses: Borehole, Transect, TrialPit, Well

Borehole

The Borehole feature is a concrete linear sampling feature with no subclasses. The Borehole represents a deep, narrow excavation made by drilling or insertion of a probe. Boreholes are constructed typically for the purpose of investigating subsurface geologic or geotechnical conditions, exploring for water or oil, for installation of wells or other downhole monitoring installations.

The Borehole feature has the properties: totalMeasuredDepth, boreholePurpose, backfill, casings, constructionMethods, constructionEvents, chiselings, flushes, holeDiameters, and waterStrikes as shown in Figure 5-17.

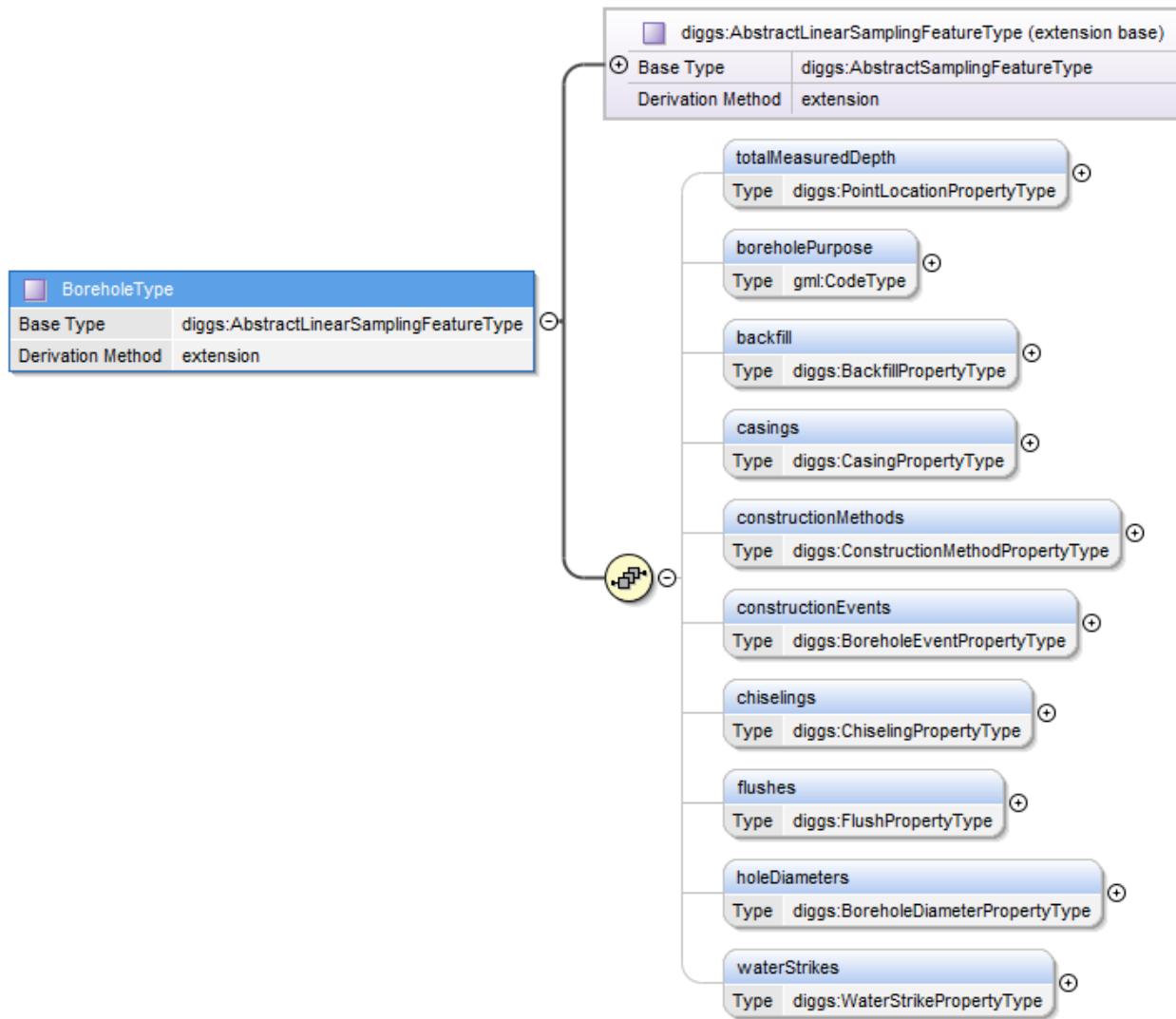


Figure 5-17: The Borehole Feature and its Properties

The **totalMeasuredDepth** property records the depth to the bottom of the hole, measured along a Borehole's linear referencing system.

The **boreholePurpose** property provides a description of the purpose for drilling the hole (eg. Exploratory boring or monitoring well) and is intended to come from a controlled list.

The **backfill** property records relevant information on the construction of the Borehole backfill after drilling.

The **backfill** property contains information on the Borehole casing installed while drilling.

The **constructionMethods** property provides information regarding construction of the hole over different depth intervals. Some example **ConstructionMethod** instances are shown in Figure 5-18.

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4		ID:	cm1
5	Description		mud rotary
6	Equipment Used		CM-1 Shaker Table
7	Equipment Used Ref		#equip-1
8		ID:	cm22
9	Description		backhoe
10	Remark		#temp-id-10
11		ID:	cmcpt
12			
us  ConstructionMethod BoreholeEvent Parameter			

Figure 5-18: Example ConstructionMethod Instance Viewed with DIGGS Excel Tool

Note that the reference “#equip-1” to the equipment used contains a link from the ConstructionMethod worksheet to the corresponding Equipment object worksheet - example instances the Equipment object worksheet are shown in Figure 5-19.

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4		ID:	equip-1
5	Name		Fielding BH-1 Rig
6	Model Number		100A
7		ID:	cone-1
8	Name		Hogentogler PiezoCone
9	Model Number		1234
10	Detector		#det-1
11			
 Equipment Detector DocumentInformation			

Figure 5-19: Example Equipment Instance Viewed with DIGGS Excel Tool

In turn, the Equipment instance (cone-1) contains a link to a Detector instance (det-1) and is shown in Figure 5-20.

A	B	C	
3	Property Name	Attribute Name	Property Value
4		ID:	det-1
5	Description		Tip resistance transducer
6	Measurand		tip_resistance
7		Code Space:	urn:diggs:def:codelist:DIGGS:properties
8			

Navigation buttons: Detector DocumentInformation SoftwareApplication Ground

Figure 5-20: Example Detector Instance Viewed with DIGGS Excel Tool

The constructionEvents property is meant to provide information on borehole construction events with time.

The chiselings property describes chiseling activity in the Borehole.

The flushes property provides information about events when the Borehole is flushed with fluid.

The holeDiameters property records various Borehole diameters with depth.

The waterStrikes property provides information on when ground water is struck during drilling of the Borehole.

An example instance of a Borehole is shown in Figure 5-21.

A	B	C
Property Name	Attribute Name	Property Value
4	ID:	LB_Webster
5 Name		Long Beach - Webster
6 Name		LB_Webster
7	Code Space:	urn:diggs:def:codelist:USGS:names
8 Name		334904118130301
9	Code Space:	urn:diggs:def:codelist:USGS:identifiers
10 Name		004S013W23D003S
11	Code Space:	urn:diggs:def:codelist:Caltrans:well-codes
12 Name		004S013W23D004S
13	Code Space:	urn:diggs:def:codelist:Caltrans:well-codes
14 Name		334904118130303
15	Code Space:	urn:diggs:def:codelist:Caltrans:well-codes
16 Name		334904118130304
17	Code Space:	urn:diggs:def:codelist:USGS:identifiers
18 Name		004S013W23D007S
19	Code Space:	urn:diggs:def:codelist:USGS:abbreviations
20 Identifier		urn:diggs:def:feature:USGS:LB_Webster
21	Code Space:	urn:diggs:def:codelist:USGS:featureIDs
22 Role		#temp-id-2
23 Role		#temp-id-3
24 Investigation Target Ref		#q1
25 Project Ref		#p1
26 Associated Project Ref		#p3
27	Identifier Ref:	urn:diggs:def:feature:USGS:usgs_p3
28 Group Ref		#q1
29 Layer System Ref		#ls-1
30 Layer System Ref		#ls2
31 Sampling Activity Ref		#xyz
32 Sampling Activity Ref		#pointSample
33 Point Location		#a33
34 Location Accuracy		#temp-id-4
35 Linear Extent		#ls
36 Linear Spatial Reference System		#sr123
37 Point Location		#lb_web_td
38 Borehole Purpose		Multi-port Monitoring Well
39 Backfill		#bf1
40 Construction Method		#cm1
41 Borehole Event		#bhc-1
42 Borehole Diameter		#bhd1
43 Water Strike		#ws1
44	ID:	est.1
	Borehole	Role
	PointLocation	LocationAccuracy
		LinearExtent
		LinearSp

Figure 5-21: Example Borehole Instance Viewed with DIGGS Excel Tool

Transect

The Transect feature is a concrete, generic linear sampling feature with no subclasses. The Transect extends along the surface of an investigation target; used for such features as transects, measured sections, etc.

The Transect feature inherits the properties and attributes from the Linear Sampling Features class and does not add any new ones.

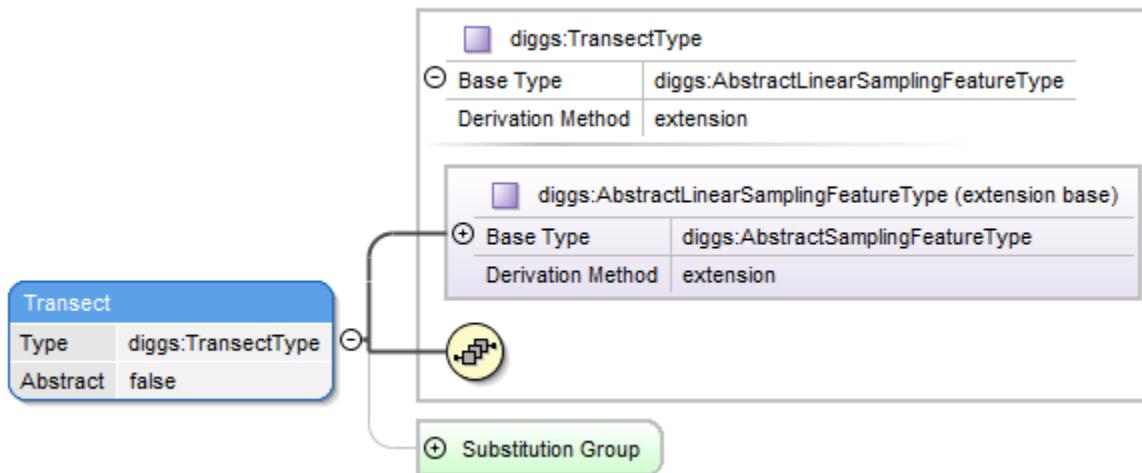


Figure 5-22: The Generic Transect Feature is an Empty Extension of the Linear Sampling Feature

TrialPit

The TrialPit feature is a concrete linear sampling feature with no subclasses. The TrialPit is a relatively shallow excavation into the earth's surface, dug either manually or by a mechanical excavator. Samples, observations and tests in the trial pit are referenced in a linear referencing system only (1D). This is a legacy sampling feature to support AGS trial pit constructs. The trench wall sampling feature should be used to represent more detail on walls of pits or trenches in 2D.

The TrialPit feature has the properties: totalMeasuredDepth, pitLength, backfill, casings, constructionMethods, constructionEvents, chiselings, flushes, holeDiameters, and waterStrikes as shown in Figure 5-23.

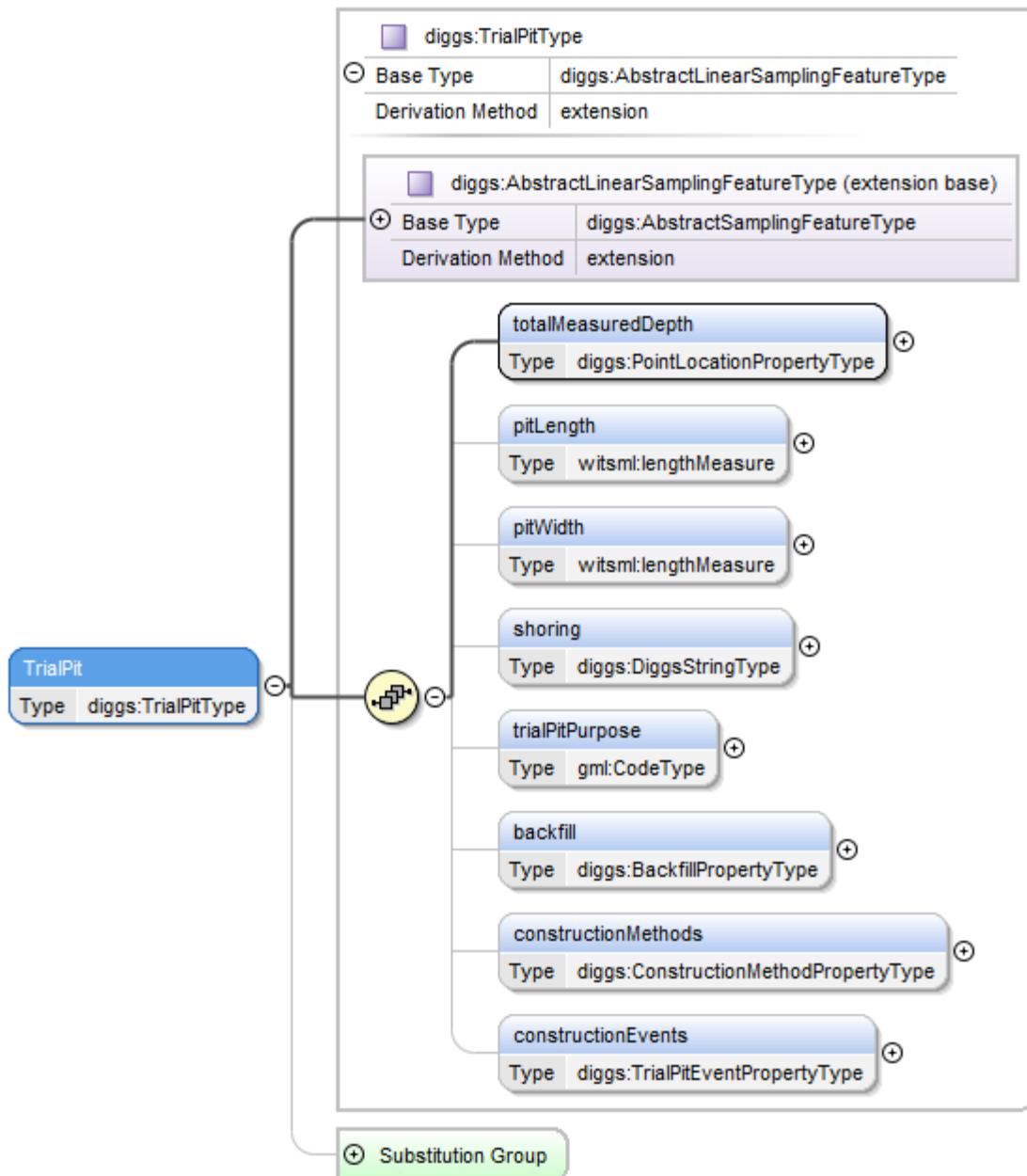


Figure 5-23: The Trialpit Feature and its Properties

The **totalMeasuredDepth** property records the maximum depth of the pit, along a linear reference system.

The **pitLength** property contains the length of the long horizontal dimension of the pit.

The **pitWidth** property provides the width of the short horizontal dimension of the pit.

The **shoring** property provides a description of shoring equipment and method.

The trialPitPurpose property describes the purpose for excavating the pit (eg. exploratory) and is intended to come from a controlled list.

The backfill property provides information on the construction of the TrialPit backfill.

The constructionMethods property contains information regarding the construction of the pit over different depth intervals.

The constructionEvents property records information on the construction of the trial pit with time.

Well

The Well feature is a linear sampling feature that is an installation within a borehole, used for observing, withdrawing, or injecting fluids. Although wells are discrete sampling features in their own right, they are installed within boreholes, and should reference a borehole feature within the parentBorehole property, if the associated borehole feature is instantiated. Multiple Wells within a borehole should be contained within a group feature and referenced in the groups property.

The Well feature has the properties: parentBorehole, wellDepth, wellPurpose, fluidPurpose, sanitarySealType, sanitarySealDepth, wellFinishType, initialDevelopmentMethod, initialDevelopmentTime, wellSpecialTreatment, maintenanceEvents, wellAbandonment, casings, and openings as shown in Figure 5-24.

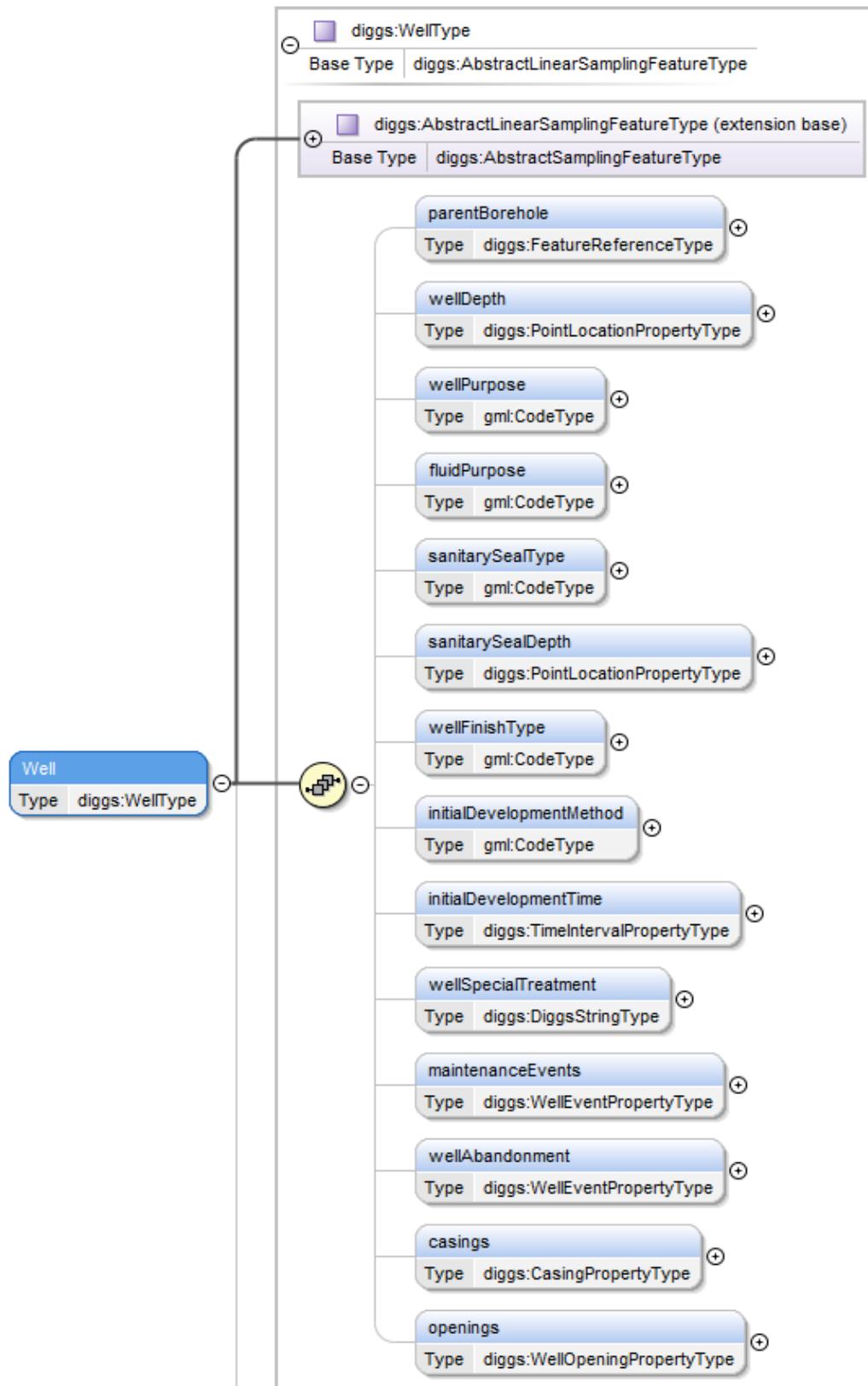


Figure 5-24: The Well Feature and its Properties

The parentBorehole property provides reference to the Borehole feature within which the Well is installed.

The wellDepth property records the measured depth to the bottom of the Well, measured in the Well's linear referencing system.

The wellPurpose property describes the purpose for installing the hole (eg. ground water, withdrawal, observation, etc.) and is intended to come from a controlled list.

The fluidPurpose property describes whether the Well is used for production or extraction and a code or description of the purpose for the fluid (eg. public supply, municipal, etc.)

The sanitarySealType property indicates the type of material used for the sanitary or surface seal (eg. bentonite, grout, etc.)

The sanitarySealDepth property records the depth to the base of the sanitary seal.

The wellFinishType property contains a description of how the Well is finished (eg. gravel pack with perforations, open end, sand point, etc.)

The initialDevelopmentMethod property describes the method used to develop the Well after initial construction (eg. compressed air, surged, etc.)

The initialDevelopmentTime property records the time required for the initial development of the Well.

The wellSpecialTreatment property describes any special procedures employed upon initial development of the Well.

The maintenanceEvents property provides information on the occurrence and activities involved in Well maintenance and servicing.

The wellAbandonment property contains information on the occurrence and procedures performed upon abandoning the Well.

The casings property provides information on the well casing installed.

The openings property provides information on the location and type of the Well openings (or perforations) installed to allow fluid communication between the Well and adjacent earth material.

5.2.2.3 Planar Sampling Features

The Planar Sampling feature class is an abstract feature class represented by the AbstractPlanarSamplingFeature element that inherits all the properties from the Sampling feature class. The Planar Sampling feature class defines common properties and attributes that all planar sampling features will inherit including: referencePoint, referencePointAccuracy, referenceEdge, relativeFeatureBoundary, and planarReferencing as shown in Figure 5-25.

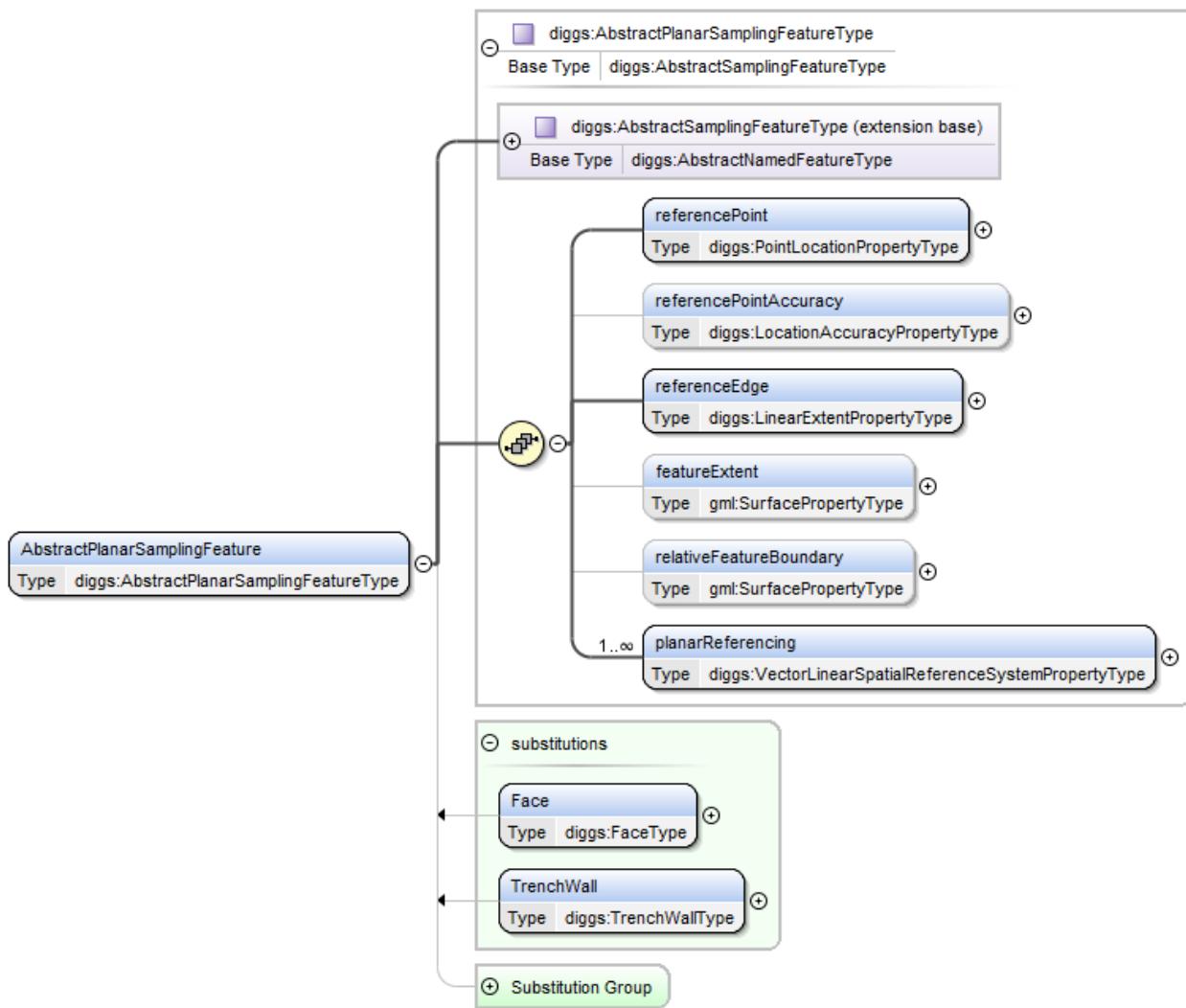


Figure 5-25: The Planar Sampling Feature Class, its Subclasses and Properties

The `referencePoint` and `referencePointAccuracy` serve the same purpose as for both the point and linear sampling features (see Section 5.2.2.1).

The `referenceEdge` property contains a `diggs:LinearExtent`, which is a linear curve geometry that represents the shape and location of the reference edge of a surface. The `LinearExtent` geometry is

typically defined by a list of direct positions in an earth based coordinate reference system when representing a reference edge.

The `relativeFeatureBoundary` property defines the polygon extent of the planar feature using a absolute (non-relative) CRS. This element should be used for visual representations using mapping software that does not support planar referencing.

The `relativeFeatureBoundary` property contains a polygon extent of the planar feature using a relative planar referencing system (must use `planarReferencing` property contents). This element should be used for software that can handle planar referencing (e.g. `LinearSpatialReferenceSystem` in GML 3.3)

The `planarReferencing` property describes the reference curve, the starting point for measuring along the curve, one or more offset vectors the units of measure (see Section **Error! Reference source not found.** for the details of the GML encoding of reference positions along a curve and the following examples) as illustrated in Figure 5-10.

An example of referencing with an offset vector (planar referencing) is illustrated in Figure 5-26.

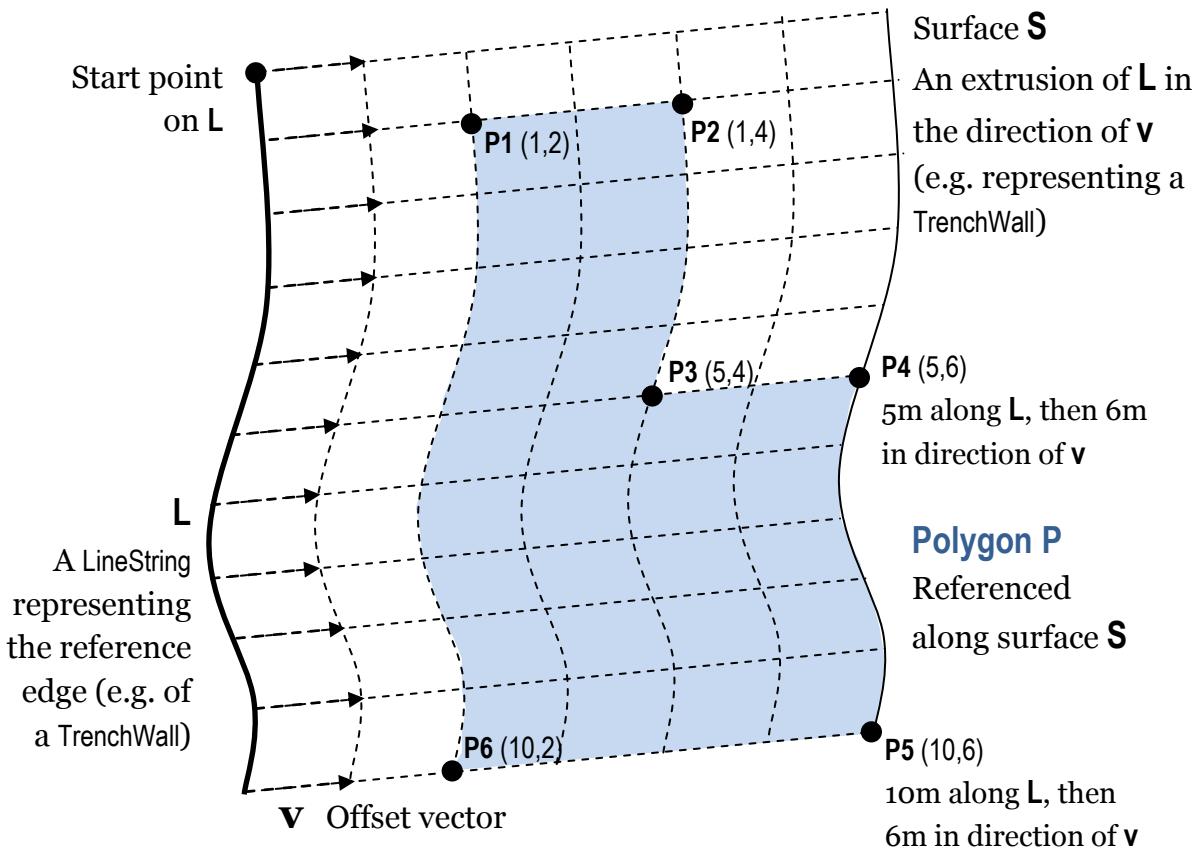


Figure 5-26: Points and Polygons Referenced Along the Extrusion of L in the direction of v (Surface S)

Any one of the PointLocations: P1, P2, P3, P4, P5, P6, can serve as the reference point for the planar sampling feature (but only 1 referencePoint property is allowed), e.g. as follows:

```

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P1" srsDimension="2" srsName="#LROV001">
    <gml:pos>1 2</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P2" srsDimension="2" srsName="#LROV001">
    <gml:pos>1 4</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P3" srsDimension="2" srsName="#LROV001">

```

```

<gml:pos>5 4</gml:pos>
</diggs:PointLocation>
</diggs:referencePoint>

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P4" srsDimension="2" srsName="#LROV001">
    <gml:pos>5 6</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P5" srsDimension="2" srsName="#LROV001">
    <gml:pos>10 6</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>

<diggs:referencePoint>
  <diggs:PointLocation gml:id="P6" srsDimension="2" srsName="#LROV001">
    <gml:pos>10 2</gml:pos>
  </diggs:PointLocation>
</diggs:referencePoint>

```

An example instance of a relative feature boundary that uses the Polygon P could be as follows. Note that the previous PointLocations: P1, P2, P3, P4, P5, P6 are reused in this example and the first point P1 is repeated at the end to close up the linear ring.

```

<diggs:relativeFeatureBoundary>
  <gml:Polygon gml:id="P" srsDimension="2" srsName="#LROV001">
    <gml:exterior>
      <gml:LinearRing>
        <gml:pointProperty xlink:href="#P1"/>
        <gml:pointProperty xlink:href="#P2"/>
        <gml:pointProperty xlink:href="#P3"/>
        <gml:pointProperty xlink:href="#P4"/>
        <gml:pointProperty xlink:href="#P5"/>
        <gml:pointProperty xlink:href="#P6"/>
        <gml:pointProperty xlink:href="#P1"/>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</diggs:relativeFeatureBoundary>

```

An equivalent example instance of a relative feature boundary instance that uses the Polygon P with all the referenced coordinates (distance along L1, distance along v) encoded inline (no point locations referenced) is as follows:

```

<diggs:relativeFeatureBoundary>
  <gml:Polygon gml:id="P" srsDimension="2" srsName="#LROV001">
    <gml:exterior>
      <gml:LinearRing>

```

```
<gml:posList>1 2 1 4 5 4 5 6 10 6 10 2 1 2</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Polygon>
</diggs:relativeFeatureBoundary>
```

In all the encodings of the PointLocation elements and Polygon P above, a Vector Linear Spatial Reference System (with gml:id="LROV001") is referenced from the srsName attribute, which is contained in the planarReferencing property and can be defined as follows:

```
<diggs:planarReferencing>
<diggs:VectorLinearSpatialReferenceSystem gml:id="LROV001">
<gml:identifier codeSpace="..."><!-->
```

```
<glr:linearElement xlink:href="#L"/>
<gmllr:lrm>
<gmllr:LinearReferencingMethod gml:id="LRM001">
<gmllr:name>chainage</gmllr:name>
<!--chainage = measurement along curve in metres -->
<gmllr:type>absolute</gmllr:type>
<!--absolute = measure from start of linear element -->
<gmllr:units uom="m"/>
</gmllr:LinearReferencingMethod>
</gmllr:lrm>
<glrov:offsetVector srsName="urn:ogc:def:crs:EPSG::7405">1 0 0</glrov:offsetVector>
<!--this is the offset vector v in the Figure above -->
</diggs:VectorLinearSpatialReferenceSystem>
</diggs:planarReferencing>
```

An example instance of a Vector Linear Spatial Reference System from the DIGGS test data is shown in Figure 5-27.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	lsrs002
5	Identifier		urn:diggs:def:feature:DIGGS:lsrs002
6		Code Space:	urn:diggs:def:codelist:DIGGS:featureIDs
7	Linear Element		#ply1_top
8	Lrm		#lr123
9	Offset Vector		1 0 0
10		SRS Name:	urn:ogc:def:crs:EPSG::7405
11	Location Accuracy		#temp-id-8
12	Location Accuracy		#temp-id-9
13			

 **VectorLinearSpatialReferenceSys** Test TestResult ResultSet

Figure 5-27: Example VectorLinearSpatialReferenceSystem Instance Viewed by DIGGS Excel Tool

The concrete subclasses of the Planar Sampling feature are: TrenchWall and Face, which are described in the following subsections.

Trench Wall

The TrenchWall feature is a concrete planar sampling feature with no subclasses. The Trenchwall typically represents the vertical face of a trench or pit dug into the ground and is spatially approximated by a surface.

The TrenchWall feature has the properties: shoring, trenchPurpose, backfill, constructionMethods, and constructionEvents, as shown in Figure 5-28.

The shoring property provides a text description of the shoring equipment and method used for the trench.

The trenchPurpose property specifies the purpose for excavating the trench (e.g. exploratory) and is intended to come from a controlled list.

The backfill property provides information on the construction of the trench backfill covering the trench wall exposure.

The constructionMethods property provides information regarding the construction of the pit over different depth intervals.

The constructionEvents property records information on the construction of the trench with time.

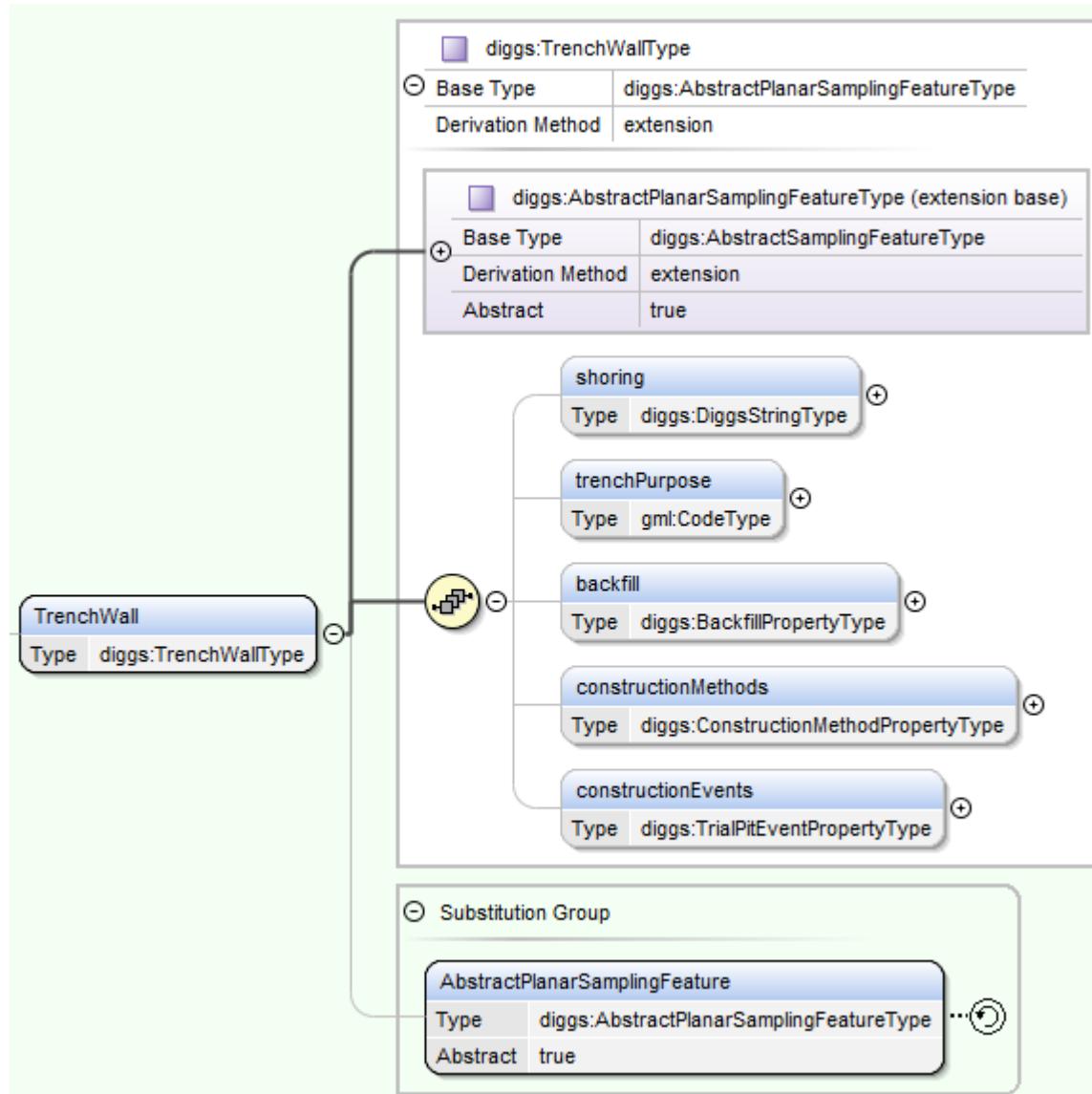


Figure 5-28: The TrenchWall Feature and its Properties

The following example instance illustrates the use of planar referencing (using the previous examples in Section 5.2.2.3) in the context of a TrenchWall feature.

```
<TrenchWall gml:id="MyTrench">
  <gml:name>My trench</gml:name>
  <gml:identifier codeSpace="..." ...></gml:identifier>
  <projectRef xlink:href="#p1"/>
  <groupRef xlink:href="#g1" identifierRef="urn:diggs:def:fi:USGS:usgs_g1"/>
  <referencePoint xlink:href="#P"/>
  <referenceEdge>
    <gml:LineString srsName="urn:diggs:def:crs:DIGGS:0.1:26911_5703" gml:id="RefEdge">
      <gml:posList>387516.665116977 3742645.12297961 500 387516.665116977 3742655.12297961 400
      387516.665116977 3742635.12297961 300 387516.665116977 3742655.12297961 200
    </gml:posList>
  </referenceEdge>
</TrenchWall>
```

```
387516.665116977 3742645.12297961 100</gml:posList>
</gml:LineString>
</referenceEdge>
<relativeFeatureBoundary xlink:href="#P"/>
<planarReferencing xlink:href="#LROV001"/>
<constructionMethods>
  <ConstructionMethod gml:id="cm22">
    <gml:description>backhoe</gml:description>
    <remarks>
      <Remark>
        <content>Backhoe was brand new and worked like a charm</content>
      </Remark>
    </remarks>
  </ConstructionMethod>
</constructionMethods>
</TrenchWall>
```

An example planar referenced TrenchWall polygon viewed as KML (generated by the DIGGS KML Tool) in Google Earth is shown in Figure 7-2.

Face

The Face feature is a concrete, generic planar sampling feature with no subclasses. A Face extends across the surface of an investigation target (e.g. the ground) and is used to represent features such as an outcrop or embankment.

The Face feature inherits the properties and attributes from the Planar Sampling Features class and does not add any new ones.

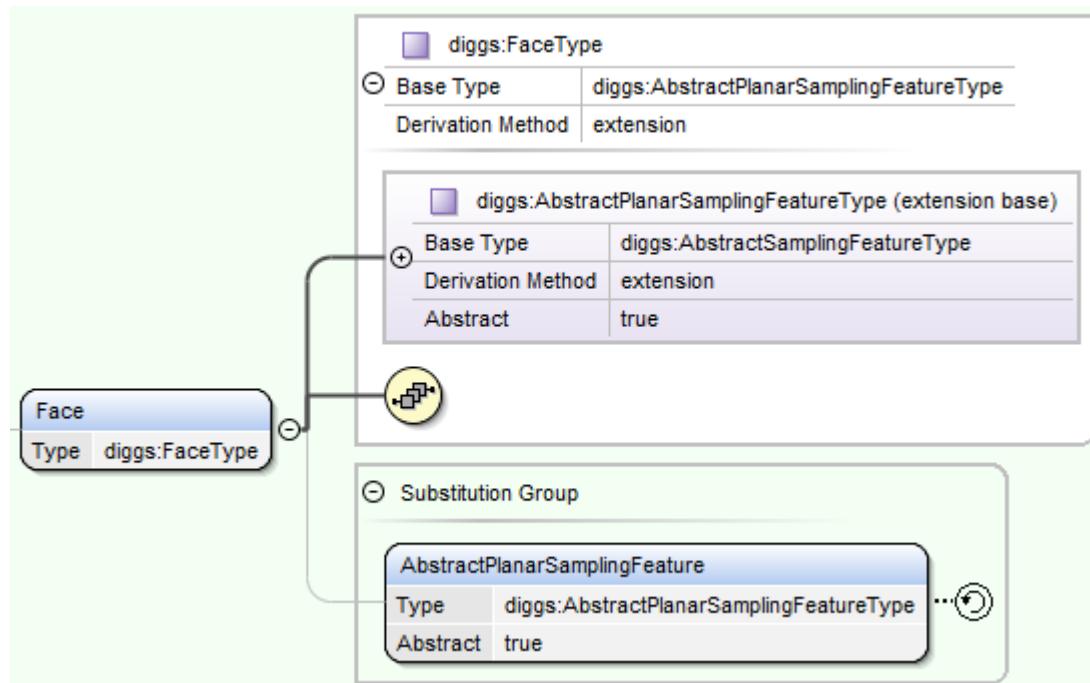


Figure 5-29: The Generic Face Feature is an Empty Extension of the Planar Sampling Feature

5.2.3 Measurement

The Measurement feature class is an abstract feature class represented by the AbstractMeasurement element that inherits all the properties from the diggs:AbstractFeature class. The Measurement feature class defines common properties and attributes that all measurement features (e.g. Monitoring and Test) will inherit including: investigationTargetRef, projectRef, relatedSamplingFeatureRef, sampleRef, samplingActivityRef, and parameters, as shown in Figure 5-30.

The first 5 properties with the ‘Ref’ suffix are used to reference, as indicated by the property name, the corresponding features (either by gml:identifier or gml:id): AbstractInvestigationTarget, Project, AbstractSamplingFeature, Sample, and SamplingActivity, respectively.

The parameters property is used to record environmental parameters, or event-specific parameters that are not tightly bound to either the earth materials for which properties are being estimated, or the procedure. Parameters that are tightly bound to the procedure should be encoded within the procedure object.

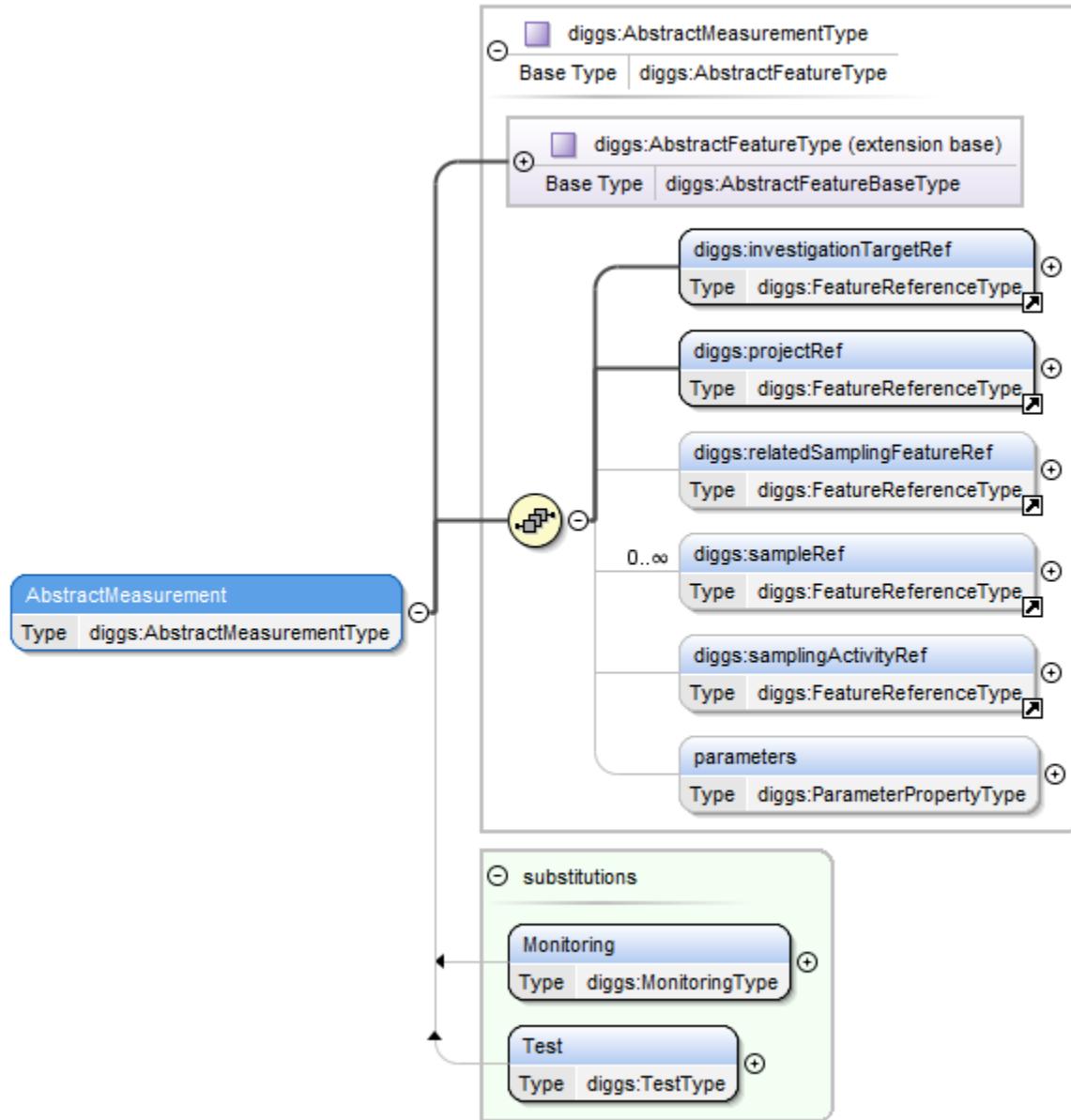


Figure 5-30: The Measurement Feature Class, its Subclasses and its Properties

5.2.3.1 Monitoring

The Monitoring feature is a concrete linear sampling feature with no subclasses. The Monitoring feature represents the act of observing/monitoring, whose results are estimates of the value of properties of the investigation target, measured at a time instant or within a time interval, and at a location or series of locations, derived from a test procedure.

The Monitoring feature has the properties: monitoringLocation, outcome, and process, as shown in Figure 5-31.

The monitoringLocation property contains a geometry (point, curve, polygon) that represents the monitoring location.

The outcome property contains the information about what properties are being measured, the results of the measurement, and the associated locations that the measurement results relate to.

The process property contains metadata related to the monitoring/observation process, such as equipment and specifications used.

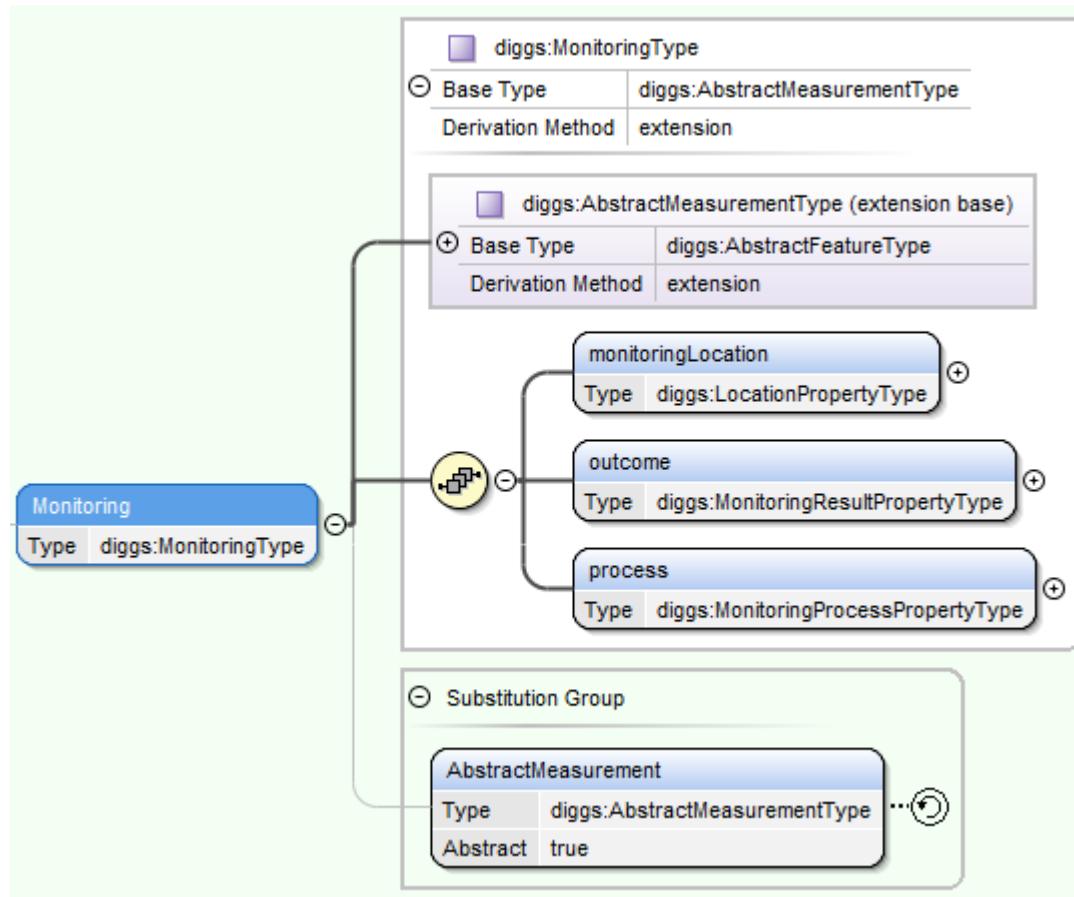


Figure 5-31: The Monitoring Feature and its Properties

5.2.3.2 Test

The Test feature is a concrete linear sampling feature with no subclasses. The Test represents the act testing/observing, whose results are estimates of the values of properties of the investigation target, measured at a time instant or within a time interval, and at a location or series of locations, derived from a test procedure.

The Test feature has the properties: samplingTime, resultTime, validTime, outcome, and procedure, as shown in Figure 5-34.

The samplingTime property provides the time interval that the result applies to. This is often the time of interaction by a sampling activity or measurement procedure with a real-world feature.

The resultTime property describes the time when the result became available, typically when the procedure associated with the observation was completed. For some observations, this is identical to the sampling time. However, there are important cases where they differ. The result time is the time when the procedure associated with the measurement act was applied. For some measurements this is identical to sampling time, in which case the result time may be omitted. For example, where a measurement is made on a specimen in a laboratory, the sampling time should record the time the specimen was retrieved from its host, while the result time should record the time the laboratory procedure was applied. As another example, where monitoring observation results are post-processed, the resultTime is the post-processing time, while the samplingTime preserves the time of initial interaction with the world.

The validTime property describes the time period during which the result is intended to be used.

The outcome property contains test result information (encoded in the TestResult object), i.e. what properties are being measured, the results of the measurement, and the associated locations that the measurement results relate to. Some example TestResult instances with various measurements at different point locations linearly referenced along a Borehole centerline is shown in Figure 5-32 – the TestResult is encoded using a GML MultiPoint Coverage and is viewed in tabular spreadsheet form using the DIGGS Excel Tool.

A	B	C	D	E	F	G
3	Property Name	Attribute Name	Property Value			
4		ID:	m100			
5		Location (#sr123)		Bulk Density (g/cm3)		
6		5 6	2.07			
7		ID:	m102			
8		Location (#sr123)		Percent Fines (%)	Coef. of Uniformity	Median Grainsize (mm)
9		5 6	18.5	1.39	2.5	
10		ID:	m103			
11		Location (#cptsr1)		Qc (kN/m2)	Fs (kN/m2)	Friction Ratio
12		0.010	0.1300	0.40	0.0000	0.0013
13		0.020	0.2400	0.40	0.1000	0.0078
14		0.030	0.5500	0.40	0.0040	0.0126
15		0.040	0.6800	0.40	0.0070	-0.0017
16		0.050	0.7800	0.30	0.0120	-0.0121
17		0.060	0.9000	0.30	0.0150	-0.0161
18		0.070	0.9600	0.40	0.0200	0.0191
19		0.080	1.0400	0.40	0.0240	-0.0120
20		0.090	1.0700	0.30	0.0270	-0.0129
21		0.100	1.1000	0.30	0.1000	-0.0123
22		0.110	1.1300	0.40	0.0350	-0.0176
23		0.120	1.1800	0.30	0.0400	-0.0234
24		0.130	1.2400	0.40	0.0430	-0.0206
25		0.140	1.2600	0.40	0.0460	-0.0277

Figure 5-32: Example TestResult Instances Viewed in DIGGS Excel Tool

Note that for the TestResult with gml:id="m103", the measurements such as Qc (tip resistance), Fs (sleeve friction), Friction Ratio, and u1 (pore water pressure) are recorded for each borehole depth location. The definition of each property measurement is linked from the spreadsheet to the corresponding Property object worksheet - example instances of the related Property objects are shown in Figure 5-33 as viewed with the DIGGS Excel Tool.

A	B	C	
3	Property Name	Attribute Name	Property Value
32		ID:	Ddle267
33		Index:	1
34	Property Code		Qc
35	Type Data		double
36	Property Class		tip_resistance
37		Code Space:	urn:x-diggs:def:code-list:property
38	UOM		kN/m ²
39	Null Value		9999
40		Reason:	missing
41		ID:	Dd1e284
42		Index:	2
43	Property Code		Fs
44	Type Data		double
45	Property Class		sleeve_friction
46		Code Space:	urn:x-diggs:def:code-list:property
47	UOM		kN/m ²
48	Null Value		9999
49		Reason:	missing
50		ID:	Dd1e301
51		Index:	3
52	Property Code		Friction Ratio
53	Type Data		double

◀ ◀ ▶ ▶ | TestResult | ResultSet | PropertyParameters | **Property** | DensityTest | ▶

Figure 5-33: Example Property Descriptions Viewed in DIGGS Excel Tool

The procedure property contains the metadata about the testing procedure, including the equipment and specifications used.

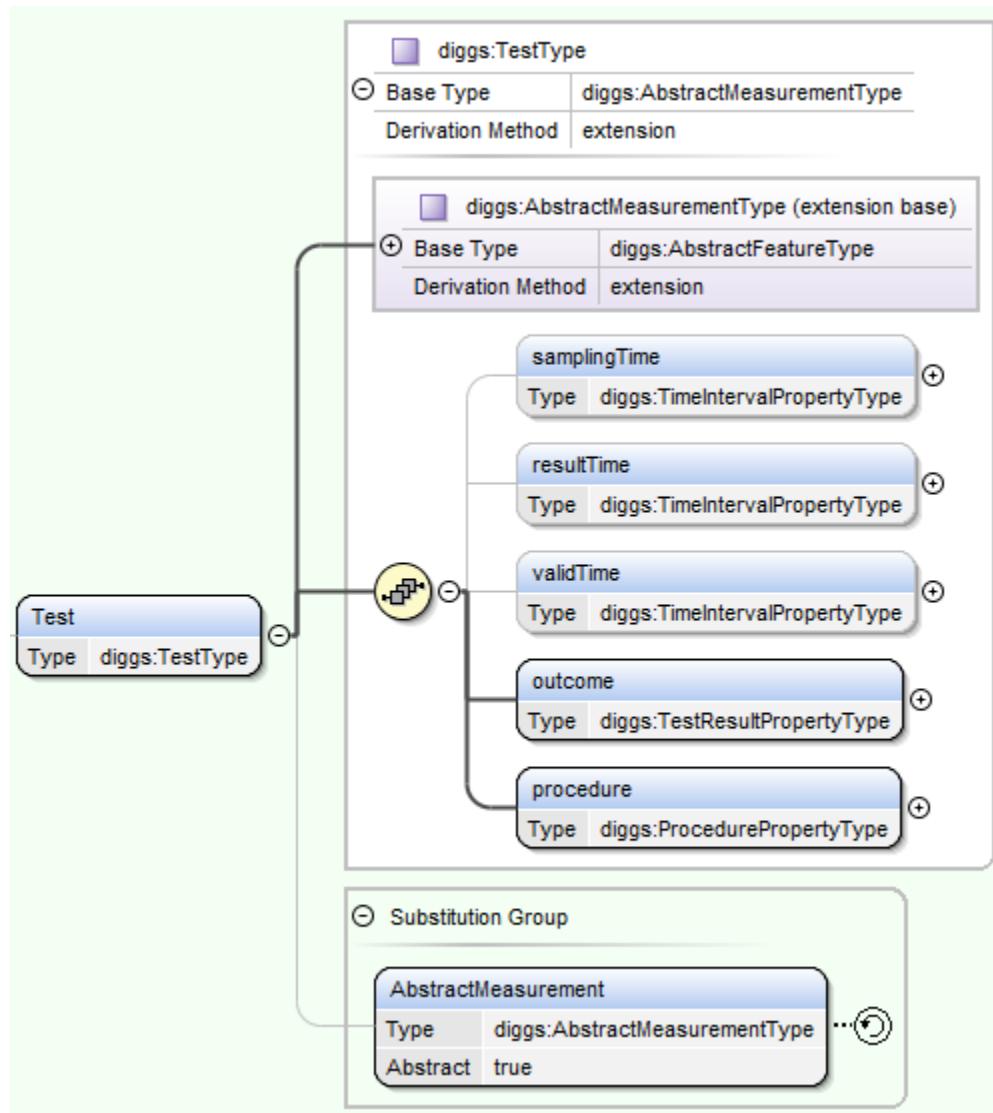


Figure 5-34: The Test Feature and its Properties

5.2.4 SamplingActivity

The SamplingActivity feature is a concrete feature with no subclasses. The SamplingActivity represents the action taken to obtain or produce a physical sample, although the activity may not produce a sample (e.g. a core run that produces no recovery). This activity typically occurs at a location on a sampling feature, or could occur elsewhere (eg. a laboratory) in the case of test or blank samples, or can produce aggregate samples where the location of the samples produced have no meaning. Note that all Sample features must refer to a SamplingActivity feature.

The SamplingActivity feature has the properties: investigationTargetRef, projectRef, relatedSamplingFeatureRef, measurementRef, sourceSample, activityLocation, samplesProduced, derivedSampleType, samplingDate, samplingEnvironment, samplingEquipmentRef, samplingEquipment, samplingProcedureRef, and samplingProcedure, as shown in Figure 5-35.

All of the properties with the 'Ref' suffix are used to reference the associated features (either by gml:identifier or gml:id) as indicated by the property name.

The sourceSample property contains a reference to a sample or samples that are used to create the sample (identified by the sampleRef attribute) produced by this activity. This element is only used for activities that sub-sample or aggregate samples from other samples. For aggregate samples, the percentage attribute optionally defines how much of the total new sample is composed from the source sample.

The activityLocation property contains a geometry representing the location of the activity.

The samplesProduced property contains a geometry location of the sample and a reference to the Sample feature.

The derivedSampleType property indicates the type of sample created by this activity using a controlled vocabulary, intended be used by applications to validate other activity information:

- collected** –sample was created by collection at a sampling feature;
- aggregate** – activity created a sample by aggregating existing samples; sourceSamples should have more than one subelement;
- subsample** – activity created a sample by subsampling an existing sample. Only one sourceSampleRef should be specified.
- test** – activity produced a test, standard or blank sample that does not relate to any field sample; activity should be related to a project and no sourceSampleRef should be specified.
- none** – the sample activity failed to produce a physical sample.

The samplingDate property records the date and time for the activity.

The samplingEnvironment property records barometric pressure, gas flow, gas pressure and temperature measurements.

The samplingEquipment property contains details, such as model, serial number, and calibration information of the equipment used.

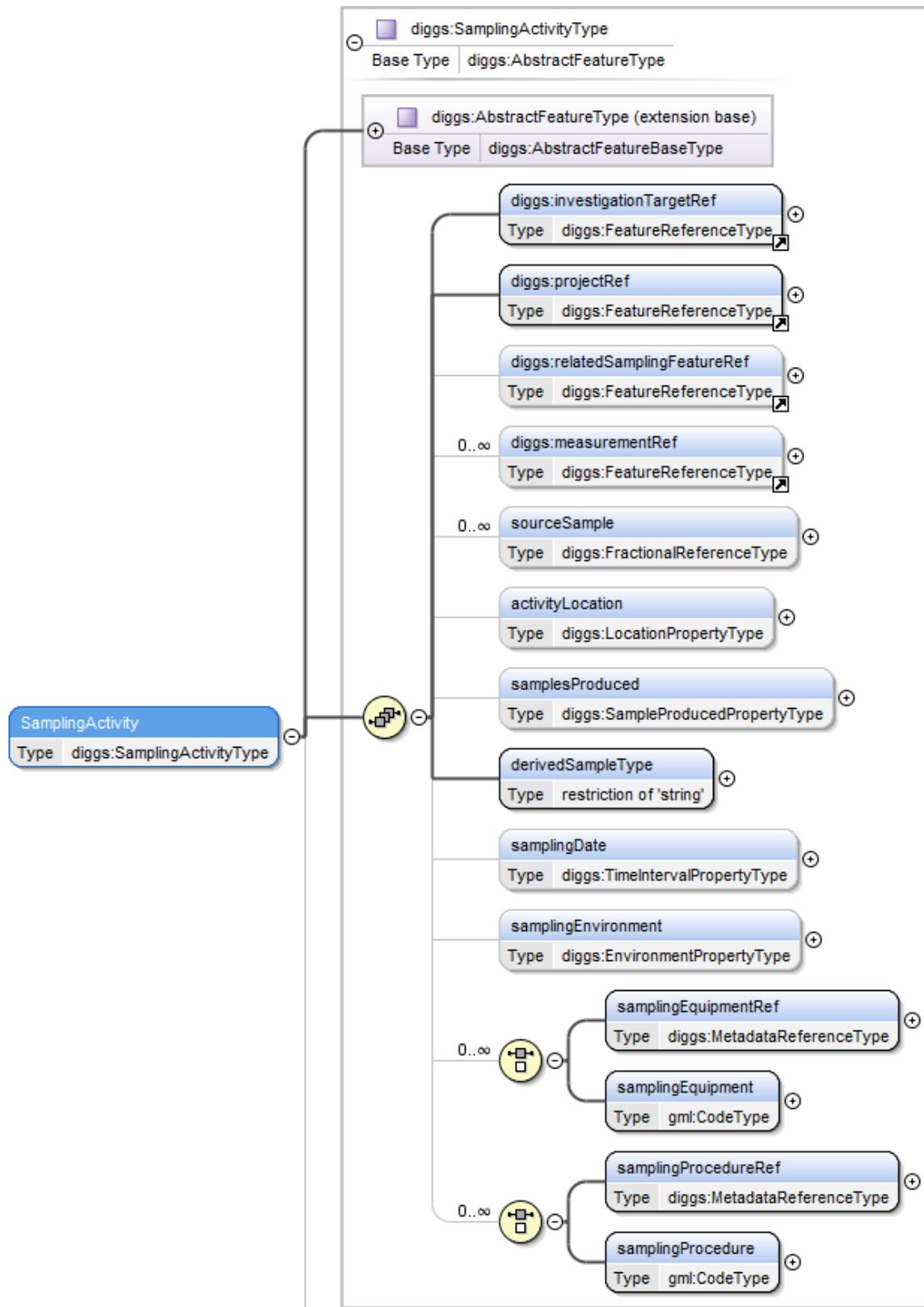


Figure 5-35: The Sampling Activity Feature and its Properties

5.2.5 Sample

The Sample feature is a concrete linear sampling feature with no subclasses. The Sample feature represents a specimen of earth material, liquid or gas that is obtained as a result of a sampling activity, for the purpose of testing and/or observation.

The Sample feature has the properties: projectRef, samplingActivityRef, groupRef, classification, purpose, condition, matrix, medium, cylindricalSampleDetails, blockSampleDetails, primaryLithology, componentLithologies and chainOfCustodyEvents as shown in Figure 5-39.

All of the properties with the 'Ref' suffix are used to reference the associated features (either by gml:identifier or gml:id) as indicated by the property name.

The classification property specifies the class of the sample collected. This is intended to come from a controlled list of values.

The purpose property provides a text description of the purpose of taking this sample. This is a free text string (not a controlled list) that may describe the test or tests that are intended to be run on this sample.

The condition property provides a text description of the sample condition.

The matrix property provides description of the sample matrix, if applicable. This is intended to come from a controlled list.

The medium property describes the medium of the Sample (e.g. gas, liquid, solid). This is intended to come from a controlled list.

The cylindricalSampleDetails property provides a more detailed description of the Sample in the case that the Sample is a core (i.e. cylindrical with integrity such that the location of the ends of the sample can be defined in space). An example instance of the CylindricalSampleDetail value object of this property is shown in Figure 5-36.

A	B	C	D
1 Home Page	Previous Page	Next Page	
2			
3 Property Name	Attribute Name	Property Value	
4	ID:	cs123	
5 Description		A really nice core sample	
6 Diameter		9	
7	UOM:	cm	
8 Length		156	
9	UOM:	cm	
10 Total Core Recovery		102	
11	UOM:	%	
◀◀ ◀ ▶ ▶▶ / SampleProduced / Environment / Sample / CylindricalSampleDetail			

Figure 5-36: Example of CylindricalSampleDetail Viewed with the DIGGS Excel Tool

The blockSampleDetails property provides a more detailed description of the Sample in the case of a block Sample with integrity and measurable width, height, and depth.

The primaryLithology property describes the lithology of the sample (if derived from earth materials). For a core sample consisting of multiple lithologies, this element should generally be left blank and a lithologyLayerSystem should be created to carry the lithologic descriptions related to core samples.

The componentLithologies property describes the lithologies that make up a minor portion of the sample. An example instance of a ComponentLithology value object of this property is shown in Figure 5-37.

A	B	C
3 Property Name	Attribute Name	Property Value
4	ID:	sl1
5	Association:	interbedded with the primary lithology
6 Lithology		#11
7 Abundance Percent		20
8	UOM:	%
9		
◀◀ ◀ ▶ ▶▶ / ComponentLithology / ColorLayer / Color / DiscontinuityLayer		

Figure 5-37: Example of ComponentLithology Viewed with the DIGGS Excel Tool

The chainOfCustodyEvents property contains information on the chain-of-custody for this sample. An example instance of the ChainOfCustodyEvent value object of this property is shown in Figure 5-38.

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4		ID:	coc1
5	Description		Transferred to laboratory
6	Source Sampling Feature Ref		#a24
7	Destination Sampling Feature		ReallyGood Laboratories
8	Date Sent		2008-09-09
9	Date Received		2008-09-10
10	Container Type		cooler
11	Preservative Added		nitrogen
12			

◀ ◀ ▶ ▶ ChainOfCustodyEvent LayerSystem LithologyLayer Component

Figure 5-38: Example of ChainOfCustodyEvent Viewed with the DIGGS Excel Tool

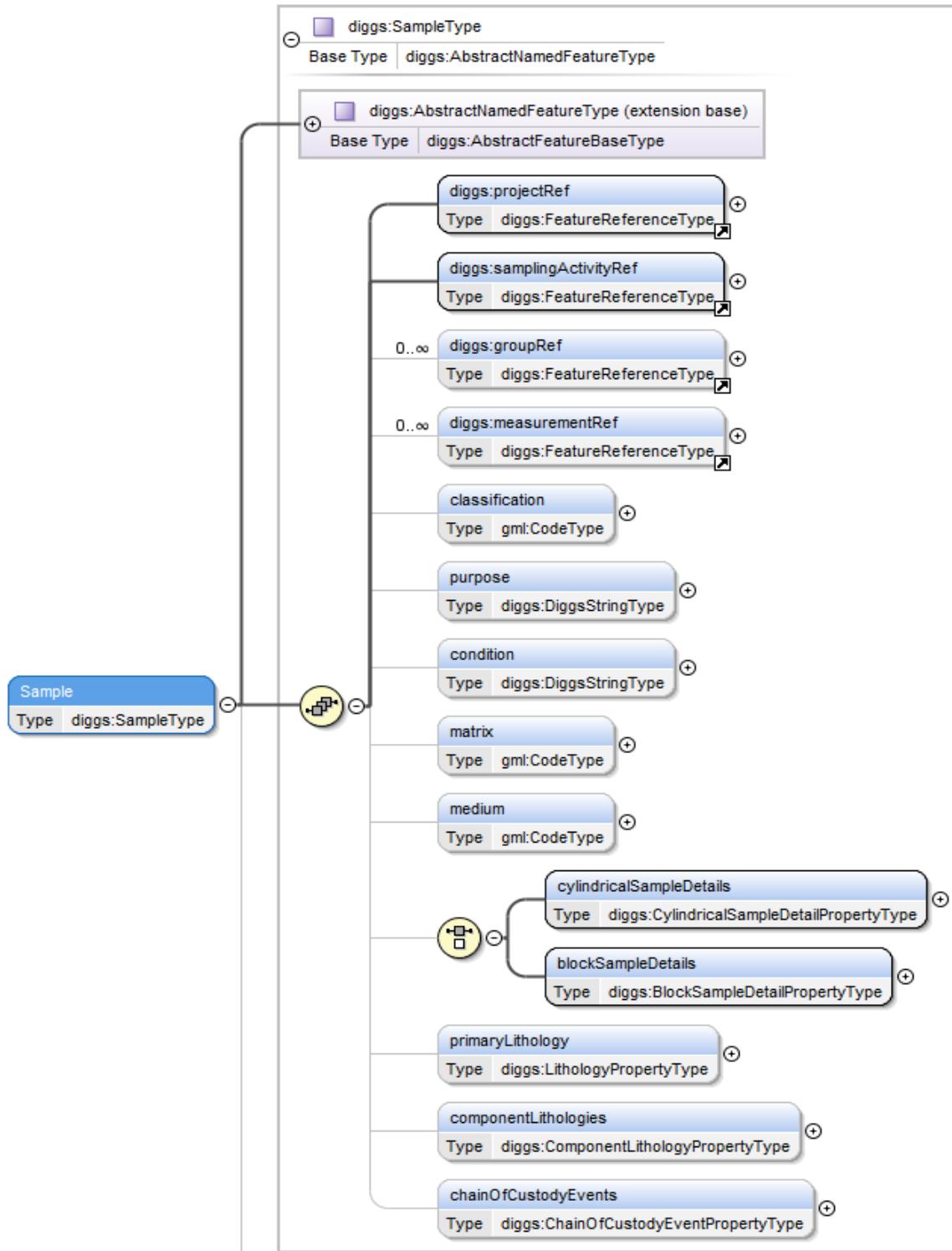


Figure 5-39: The Sample Feature and its Properties

5.2.6 LayerSystem

The LayerSystem feature is a concrete named feature with no subclasses. The LayerSystem represents a collection of layers that contain observations or interpretations made over a region or interval a sampling feature.

The LayerSystem feature has the properties: projectRef, relatedSamplingFeatureRef and layers as shown in Figure 5-40.

The projectRef and relatedSamplingFeatureRef are used to reference (either by gml:identifier or gml:id) a Project or Sampling Feature, respectively.

The layers property contains a sequence of layer features, including:

ColorLayer – A color layer describes the color of materials encountered. All layers within a color layer system must not overlap, although multiple colors can be described within a single zone.

ConstituentLayer – Constituent layers describe details of earth materials encountered within a sampling feature. Constituent layers restricted to a specific lithologic layer are encoded as constituents in the Lithology layers themselves. Layers within a constituent layer system need not be continuous and may overlap, but layers with the same named constituents cannot overlap.

DiscontinuityLayer – Describes fractures and joints and their spacing. Individual discontinuities or zone of discontinuities within a discontinuity layer system may overlap.

LithologyLayer – Lithology layers that describe the earth materials encountered at a sampling feature defined by the layer's location. Layers within a lithology layer system must not overlap.

OrientationLayer – Orientation layers describe the geometry of vectors or planar surfaces encountered at a sampling feature, such as bedding, joints, cross-beds, etc. These layers are designed to characterize regions with generalized geometries. Individual measurements of planar geometries of boundaries or faults are recorded in Discontinuity or Lithology layers. Layers within an orientation zone system must not overlap.

PropertyLayer – A property layer system contains layer that are defined by simple text or numeric values - usually interpreted as a result of some lab or in-situ test. Layers within a property layer system must not overlap and the value of the gml:name of each property layer must be the same among all layers in the system.

StratigraphyLayer – Stratigraphy layers are ordered bodies of rock or sediment, such as formations, biostratigraphic units or aquifers. Layers within a stratigraphy layer system must not overlap, and

stratUnit values within a layer system cannot repeat (eg. they must be unique within the layer system).

OtherLayer – A structure for defining a layer system of unknown type. Layers consist of name-value pairs, where these names should reference code lists.

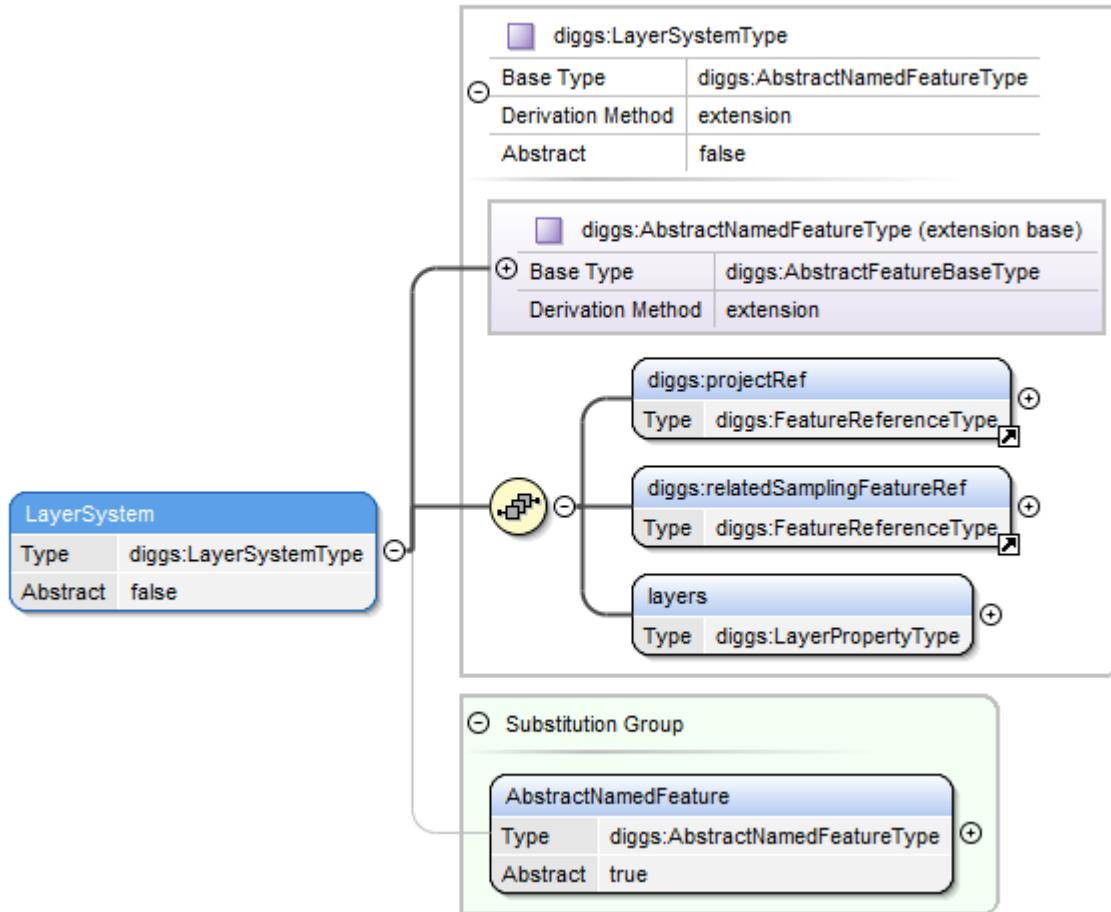


Figure 5-40: The LayerSystem Feature and its Properties

5.2.7 Group

The Group feature class is an abstract feature class represented by the AbstractGroup element that inherits all the properties from the diggs:AbstractNamedFeature class and does not define any new properties. The Group feature class has four concrete subclasses: GroupGroup, ProjectGroup, SampleGroup and SamplingFeatureGroup, as shown in Figure 5-41.

The ProjectGroup contains a list of references (via the associatedProjectRef property) to Project features. Likewise, the SampleGroup contains a list of references to Sample features, the

SamplingFeaturesGroup contains a list of references to sampling features, and the GroupGroup contains a list of references to other Group features.

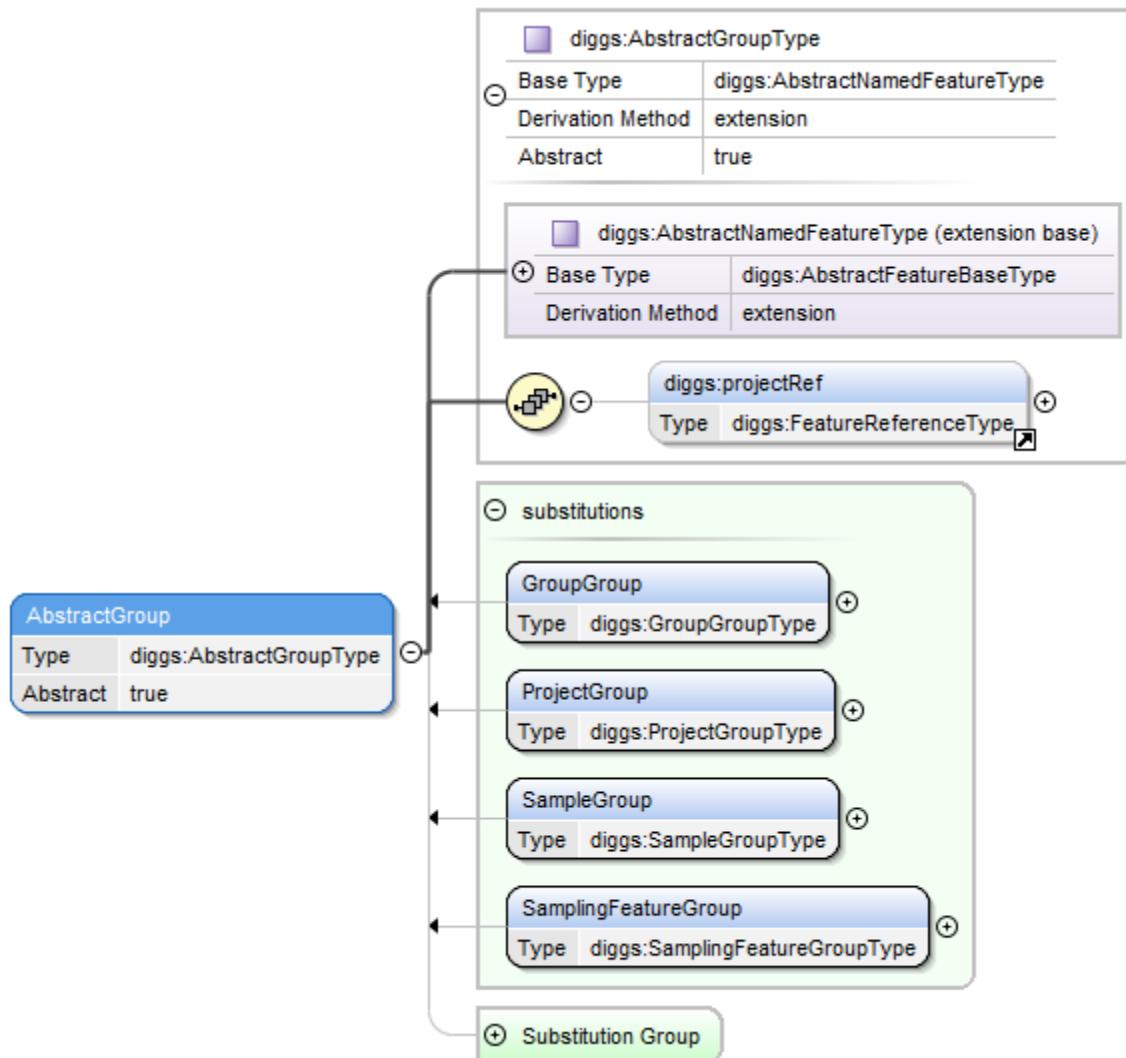


Figure 5-41: The Group Feature and its Subclasses

5.3 Feature Metadata

The Metadata object class is an abstract object class represented by the `diggs:AbstractMetadata` element that inherits all the properties from the DIGGS named object class and does not add

any new ones. The Metadata object class inherits the following properties: gml:name, status and remarks, as shown in Figure 5-47.

Metadata objects provide contextual information (e.g. associated files) about the parent feature as opposed to defining characteristics of the feature. The concrete subclasses of the metadata object class are shown in Figure 5-48 and summarized as follows:

- **Associated Files** - references to non-XML documents or records outside of the XML instance
- **Business Associates** - provides contact information of individuals or institutions, such as address, phone number, email address, etc.
- **Contracts** – references to legal documents concerning sales, employment, or tenancy
- **Document Information** - information about the specific XML instance document
- **Equipment** – necessary tools to carry out procedures
- **Specifications** - a detailed description of the requirements, design, and/or materials for procedures

Example instances of BusinessAssociate metadata are shown in Figure 5-42.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	a23
5	Name		Joe Blow
6	Address		#Adr001
7	Email Address		joe.blow@google.com
8	Phone Number		314-1592
9	Associated With		#a24
10		ID:	a24
11	Name		Joe Geologist
12	Address		#Adr002
13	Email Address		joe.geo@google.com
14	Phone Number		271-8128
15	Associated With		#a23
16			

◀ ◀ ▶ ▶ BusinessAssociate Address Equipment Detect

Figure 5-42: Example BusinessAssociate Instances Viewed in DIGGS Excel Tool

Note that Figure 5-42 shows that each BusinessAssociate metadata object can contain an Address object, which are shown in Figure 5-43.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	Adr001
5	Number		1234
6	Street		Main Street
7	City		Gotham City
8	State		TN
9	County		Humphreys
10	Country		USA
11	Postal Code		12345
12		ID:	Adr002
13	Number		5678
14	Street		Yellow Brick Road
15	City		Emerald City
16	State		Kansas
17	Country		USA
18	Postal Code		45678
19			

◀◀
◀
▶
▶▶
Address
Equipment
Detector
DocumentI

Figure 5-43: Example Address Instances Viewed in DIGGS Excel Tool

Examples of Equipment are shown in the context of construction methods of boreholes in Figure 5-19.

DIGGS also defines a Simple Metadata object class which is an abstract object class represented by the diggs:AbstractSimpleMetadata element that carries just the gml:id attribute for identification and linking. The concrete subclasses of the simple metadata object class are shown in Figure 5-49 and summarized as follows:

- **Document Information** –information about the author, date, language, etc. of documents.
- **Location Accuracy** – provides a measurement of accuracy and the measurement method.
- **Remark**– provides a text remark along with the author and date/time of the remark
- **Role** – describes the role performed by a business associate, time the role was performed and related remarks about the role.

An example instance of DocumentInformation metadata is shown in Figure 5-44.

	A	B	C
1	Home Page	Previous Page	Next Page
3	Property Name	Attribute Name	Property Value
4		ID:	d1
5	Creation Date		2010-01-01
6	Effective Date		2009-01-01
7	Author Ref		#a23
8	Software Application		#sap1
9			
◀◀ ▶▶		DocumentInformation	SoftwareApplication

Figure 5-44: Example DocumentInformation Instances Viewed in DIGGS Excel Tool

Example instances of Remark metadata are shown in Figure 5-45.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	temp-id-1
5	Content		This could be a virtual project that has no relevance outside of this instance
6		ID:	temp-id-10
7	Content		Backhoe was brand new and worked like a charm
8			
◀◀ ▶▶		Remark	Borehole Role PointLocation LocationAccuracy LinearExtent LinearSpatialReference

Figure 5-45: Example Remark Instances Viewed in DIGGS Excel Tool

Example instances of Role metadata are shown in Figure 5-46.

	A	B	C
3	Property Name	Attribute Name	Property Value
4		ID:	temp-id-2
5	Role Performed		drilling_company
6		Code Space:	urn:diggs:def:codelist:DIGGS:roles
7	Business Associate		A-1 Drillers
8		ID:	temp-id-3
9	Role Performed		geologist
10		Code Space:	urn:diggs:def:codelist:DIGGS:roles
11	Business Associate Ref		#a24
12		ID:	temp-id-14
13	Role Performed		service company
14	Business Associate		Schlumberger
15			
◀◀ ▶▶		Role	PointLocation LocationAccuracy LinearExtent LinearSpatialReference

Figure 5-46: Example Role Instances Viewed in DIGGS Excel Tool

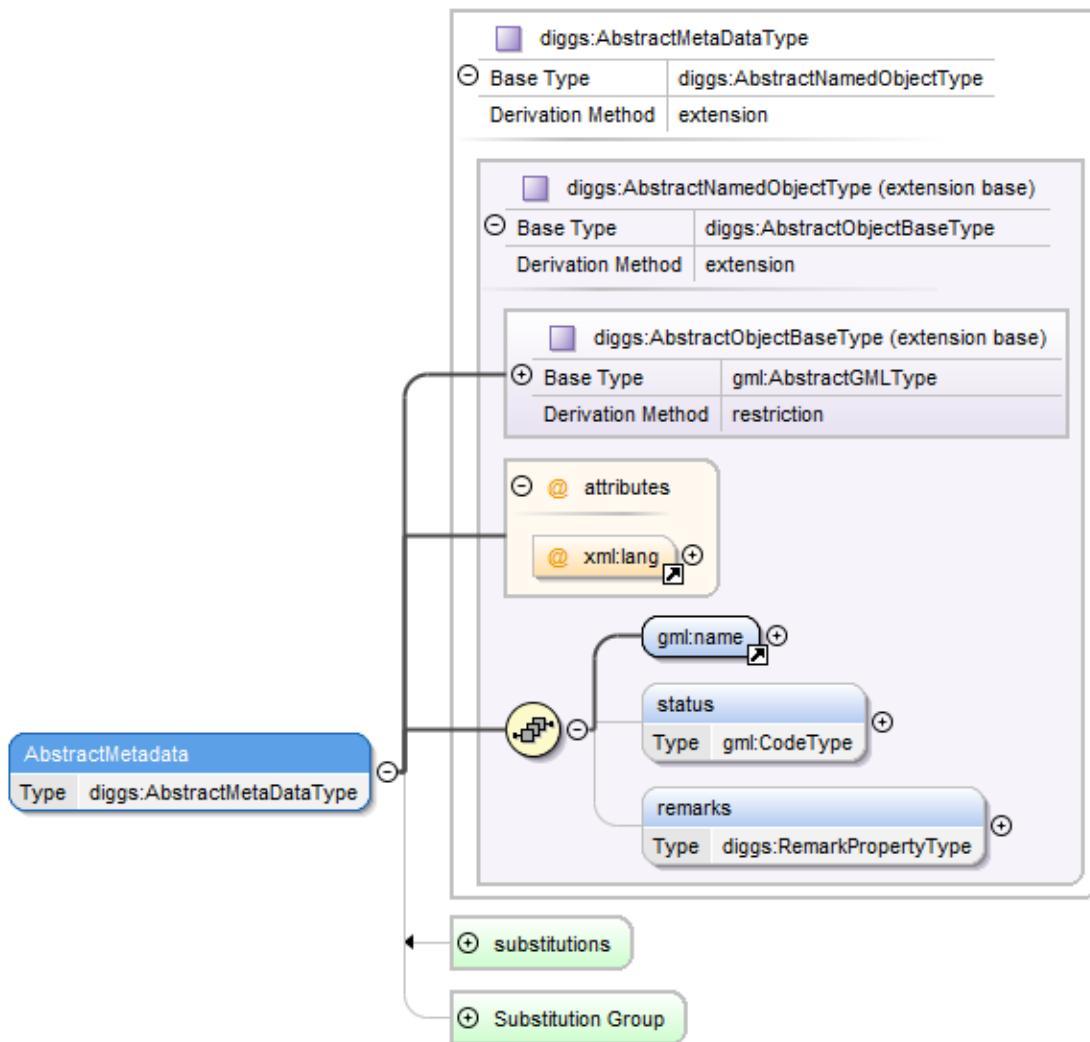


Figure 5-47: Metadata Object Class and its Properties

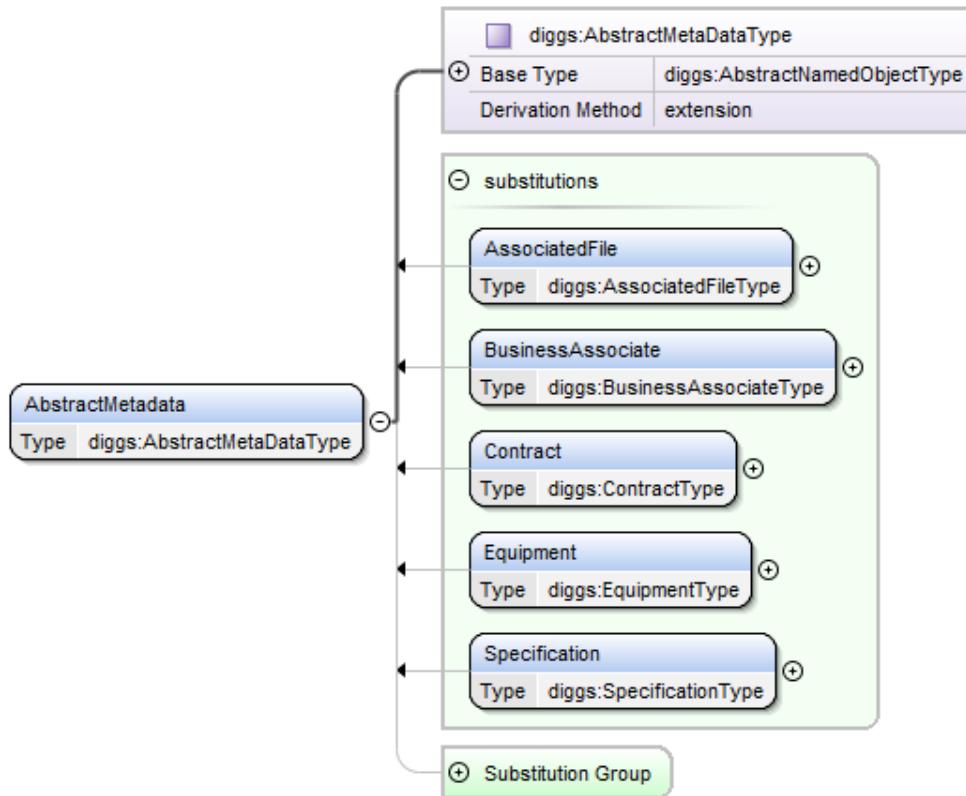


Figure 5-48: Metadata Object Class and its Subclasses

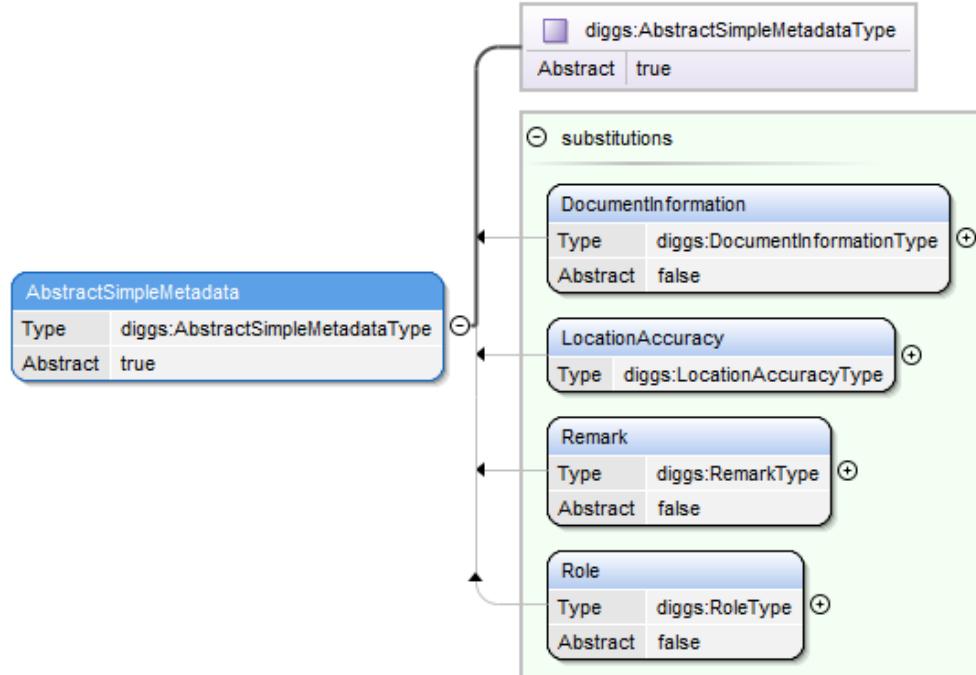


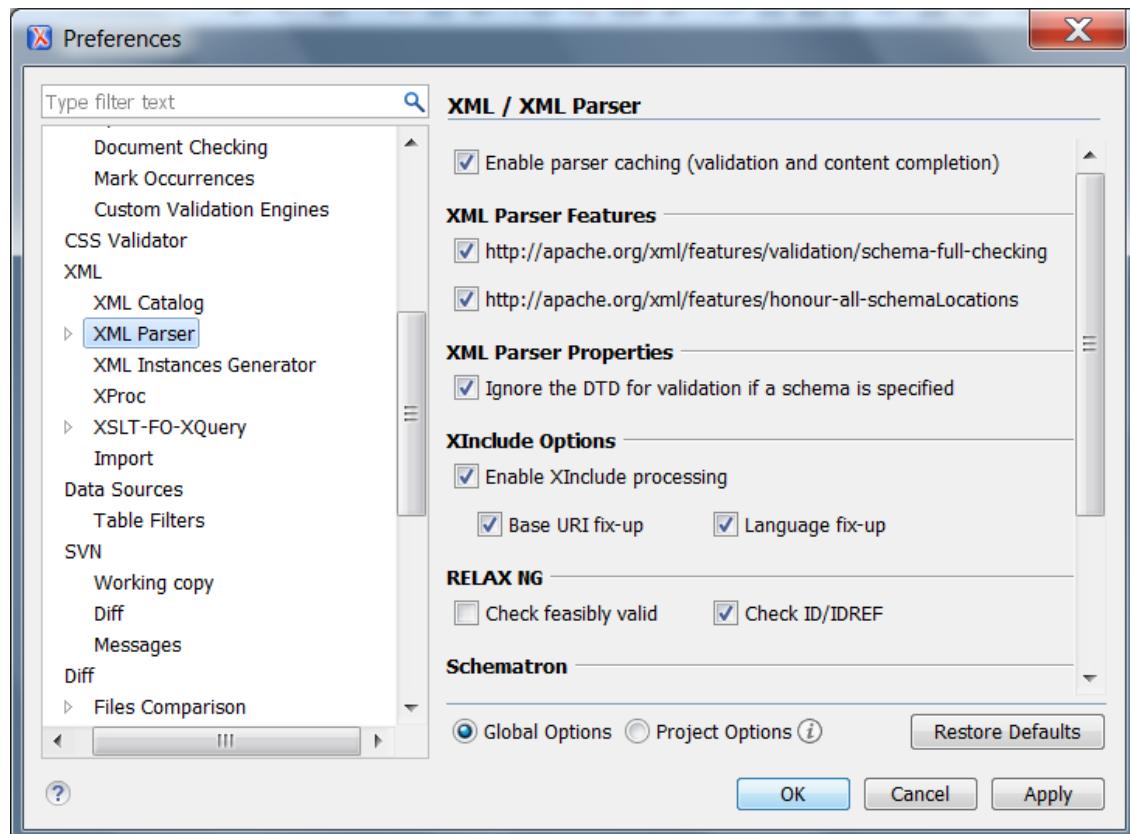
Figure 5-49: Simple Metadata Class and its Subclasses

6 DIGGS 2.0a Schema Complexity Evaluation

This report also includes a measure of the XML complexity based on ‘stress-testing’ of the 2.0a schemas. The complexity is measured by creating the most hypothetically complicated datasets that the schemas will allow, i.e. by constructing a Document Object Model (DOM) from the schemas and populating all possible elements/attributes with valid random values, so that the data remains valid against the schemas. The creation of a DOM and data instances are representative XML processing tasks that are typically required of XML software to perform common tasks including conversion of queries and data to a relational database. The assessment of DIGGS complexity was measured in a similar way for DIGGS V1.1 as shown in Section 3.2.2, the only difference being an updated software environment (Oxygen Enterprise V14.0 running on typical 2012 commodity hardware) and larger numbers of trials that the results were averaged over.

6.1 *Schema Load and Validate Performance*

In the Oxygen Enterprise V14.0 Integrated Development Environment (IDE), the Complete.xsd schema was loaded and validated (using the default Xerces J parser) with the most exhaustive settings enabled including: ‘schema-full-checking’ and ‘honour all schema locations’.



6.1.1 Results

6.1.1.1 Load Test

The DIGGS 2.0a Complete.xsd schema took 0.9 seconds to load (uncached), compared with 6.5 seconds for V1.1, and 156 seconds for V1.0a.

6.1.1.2 Validation Test

The DIGGS 2.0a Complete.xsd schema took 0.5 seconds to validate after it was loaded upon first validation (subsequent validation is faster), compared with 1.4 seconds for V1.1, and 87 seconds for V1.0a.

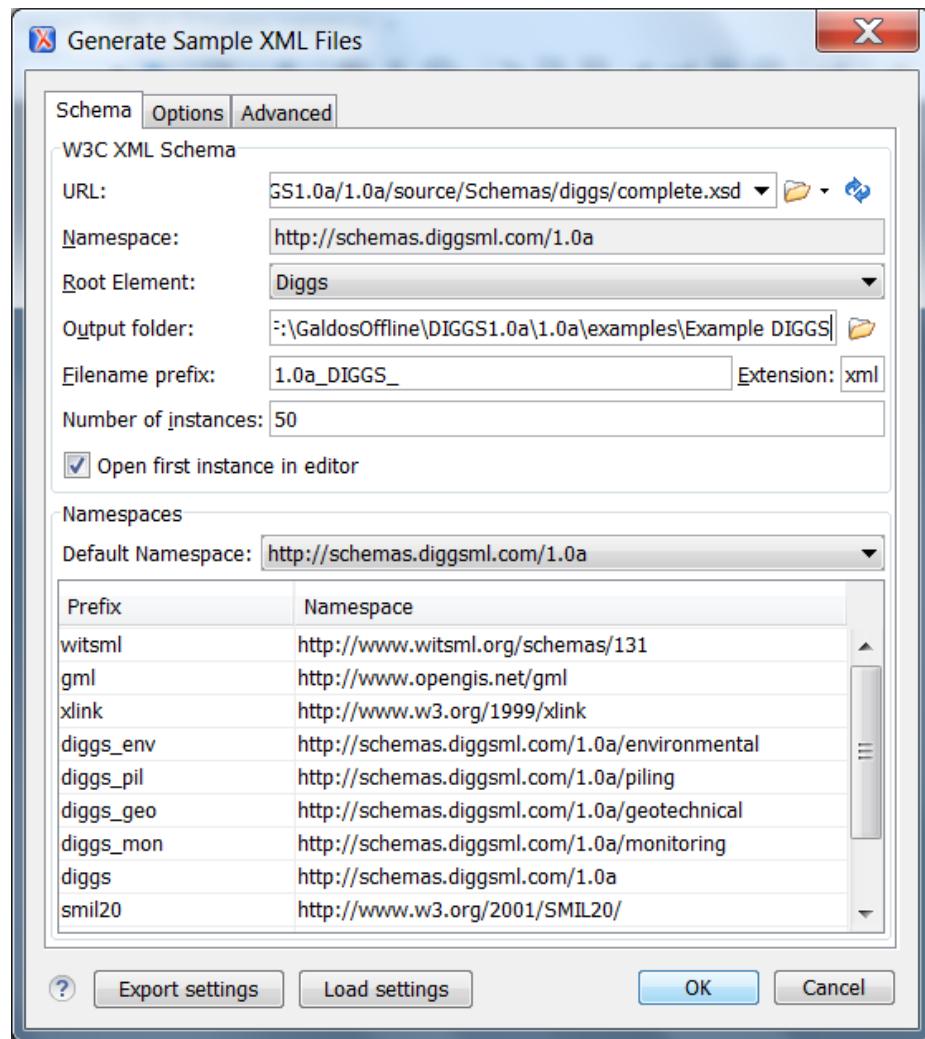
Therefore the changes made to DIGGS 2.0a, since V1.0a resulted in a 173x speed-up to load and 174x speed-up to validate the Complete.xsd schema file using the same validator, settings, and software environment.

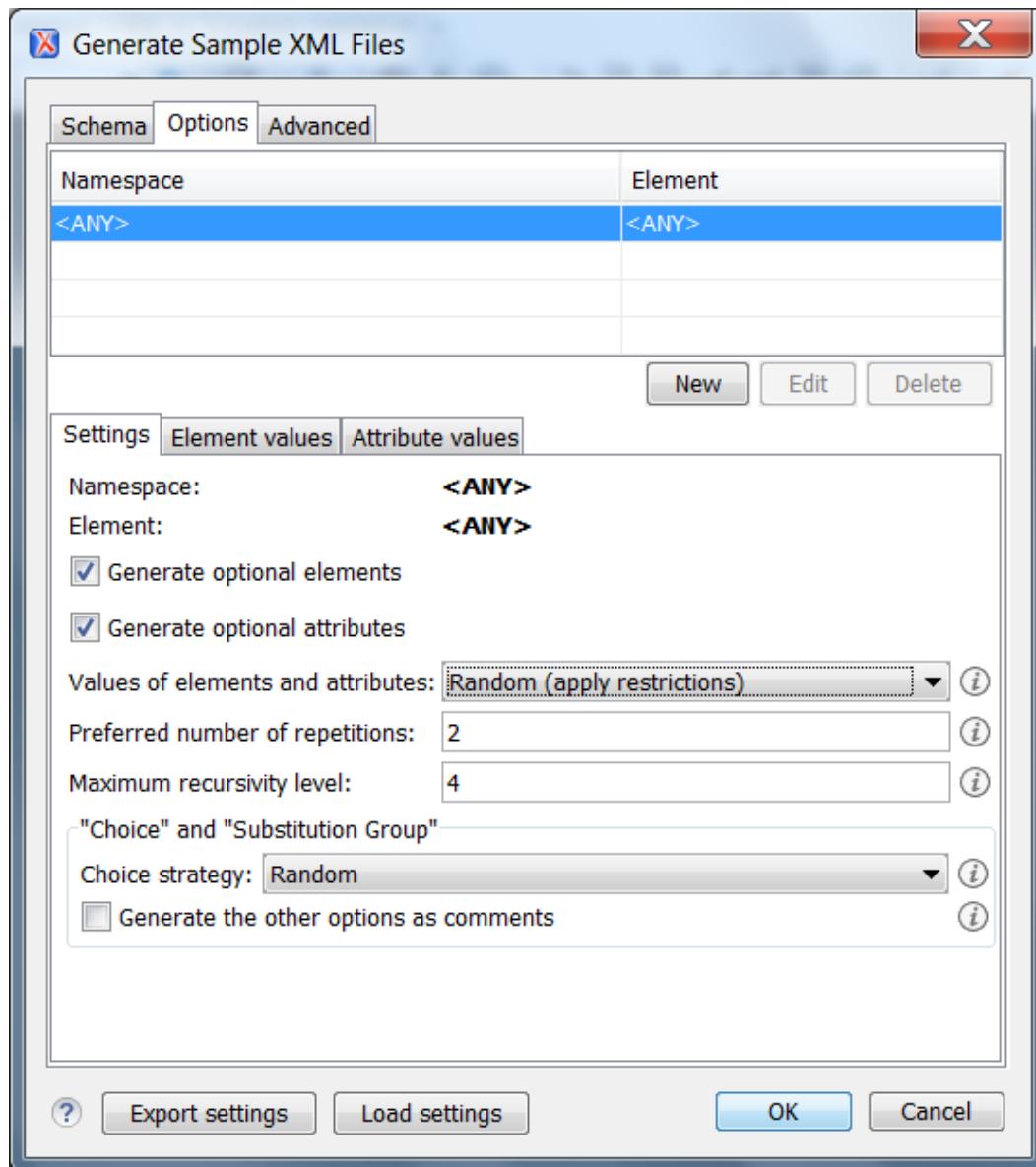
6.2 Model Complexity assessment

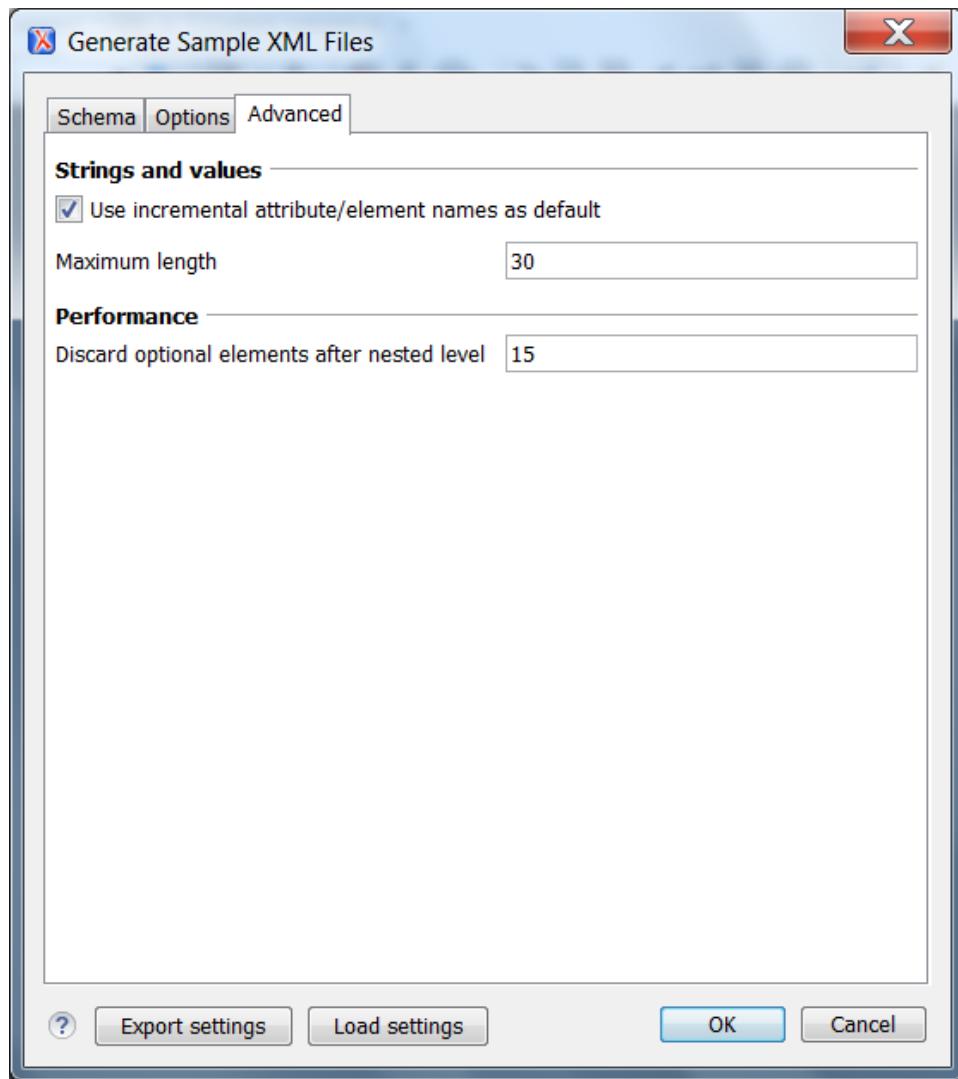
Several options were set such as number of repetitions, maximum recursivity level, maximum text string length, and nesting levels, prior to the automatic generation of DIGGS instances from the schema using Oxygen 14. Several of these options were fixed for all of the data generation, but two were varied as parameters: maximum recursivity and nest level (i.e. tree depth). These two parameters were gradually increased across 27 test levels of maximum recursivity and tree depth to observe the corresponding growth in data set size for each of the DIGGS schema versions tested (1.0a, 1.1, and 2.0a). For V1.0a, the file size and execution time was prohibitively exceeded after test level 8, and likewise for V1.1 after test level 17. On the other hand, results were easily obtained and recorded for all 27 test levels for V2.0a. Figure 1-1 in Section 1 (Executive Summary) provides a comparison of the data size growth for each version across the 27 test levels. The following Subsections 6.2.1 and 6.2.2 provide details on the testing environment and additional tables/graphs of the results.

6.2.1 Options Used for Oxygen V14.0 Random Instance Generation

The following screenshots provide a sample of the selected options for the example data generation.







Note that the data values were selected randomly by the instance generation tool according to the datatypes specified in the schemas. Fifty example data files were created for each of the parameter settings and for each schema version. The file sizes were then averaged over the 50 trials.

6.2.2 Results

The dataset size grew as a function of the increasing recursion and nesting levels (tree depth) as tested over 27 increasing tree depth levels. Table 6-1 summarizes the results across all 27 test levels, including the specific Oxygen data generations settings for each level, and the resulting file size averages. Figure 6-2 shows the V1.0a data increasing exponentially without bound up to test level 8, until computer memory limits were exceeded. Figure 6-3 shows the V1.1 data growth rate lower than that of V1.0a, but since recursive loops still remained in the DIGGS

schemas, the V1.1 data size also increased exponentially without bound. By V2.0a, all recursion was removed from the schemas and the data size was shown to have an upper bound (< 1 MB) as illustrated in Figure 6-4, with the average file size approaching 500KB. Note that the maximum recorded file size was 737KB out of several thousands of generated files.

The comparison of data size growth with test levels for each pair of schema versions: V1.0a vs 1.1, V1.0a vs 2.0a, and V1.1 vs V2.0a, are shown in Figure 6-5, Figure 6-6, and Figure 6-7, respectively.

The fully detailed tables of the file data statistics captured for each version are contained in the accompanying spreadsheet file DIGGS2.0aSchemaComplexity.xlsx.

Test #	Maximum Repetitions	Maximum Recursivity	Nest level/tree depth	# of files	V2.0a avg file size	V1.1 avg file size	V1.0a avg file size
1	2	1	4	50	42.69	19.71	79.84
2	2	2	5	50	73.96	37.71	200.68
3	2	2	6	50	113.50	94.13	1057.42
4	2	2	7	50	165.47	141.88	2655.65
5	2	2	8	50	212.25	217.48	6124.04
6	2	2	9	50	255.82	305.96	14474.30
7	2	2	10	50	320.87	415.18	42344.12
8	2	3	11	50	356.73	637.98	137141.19
9	2	3	12	50	397.60	960.40	
10	2	4	13	50	435.26	1386.21	
11	2	5	14	50	443.70	2185.61	
12	2	5	15	50	471.58	3329.01	
13	2	6	16	50	480.71	5414.70	
14	2	6	17	50	484.36	9900.67	
15	2	6	18	50	477.50	17060.59	
16	2	6	19	50	487.07	25387.20	
17	2	6	20	50	471.45	54042.73	
18	2	7	21	50	485.99		
19	2	8	23	50	475.69		
20	2	9	25	50	487.83		
21	2	10	27	50	481.41		
22	2	10	30	50	481.43		
23	2	20	50	50	484.32		
24	2	50	100	50	493.09		
25	2	60	120	50	473.00		
26	2	70	140	50	485.81		
27	2	80	160	50	478.99		

Table 6-1: DIGGS Data Test Levels, Settings, and Average File Size Results for V1.0a, 1.1 and 2.0a

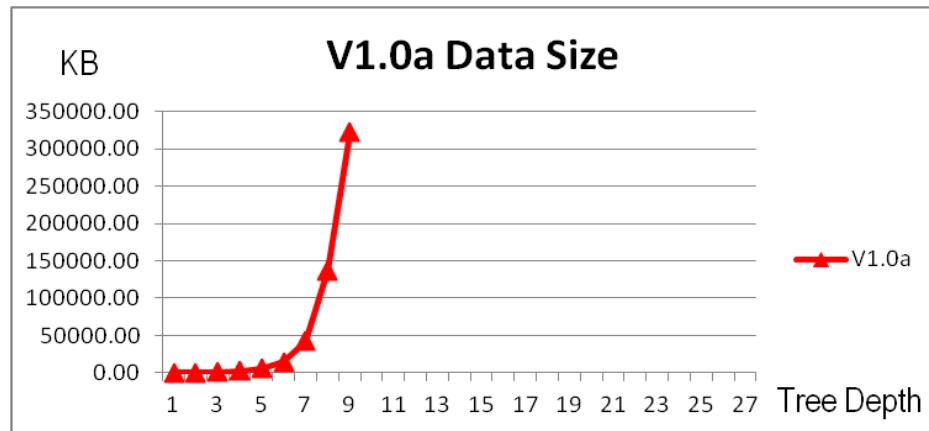


Figure 6-2: DIGGS V1.0a Data Size (KB) up to Test Level 8

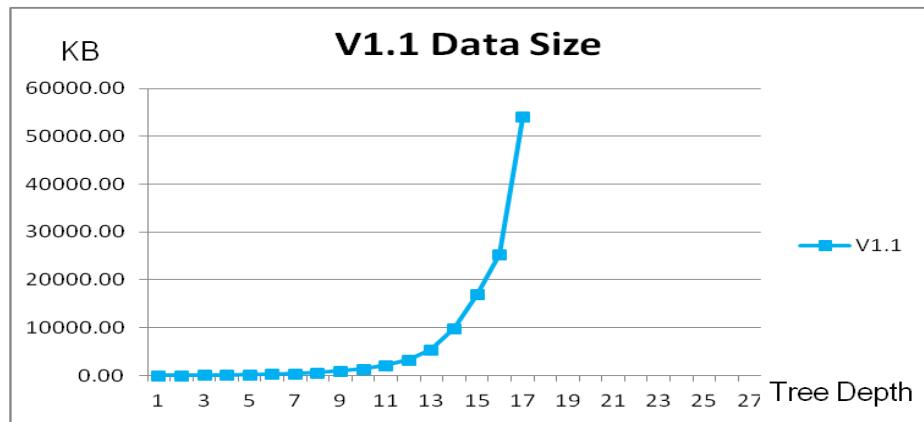


Figure 6-3: DIGGS V1.1 Data Size (KB) up to Test Level 17

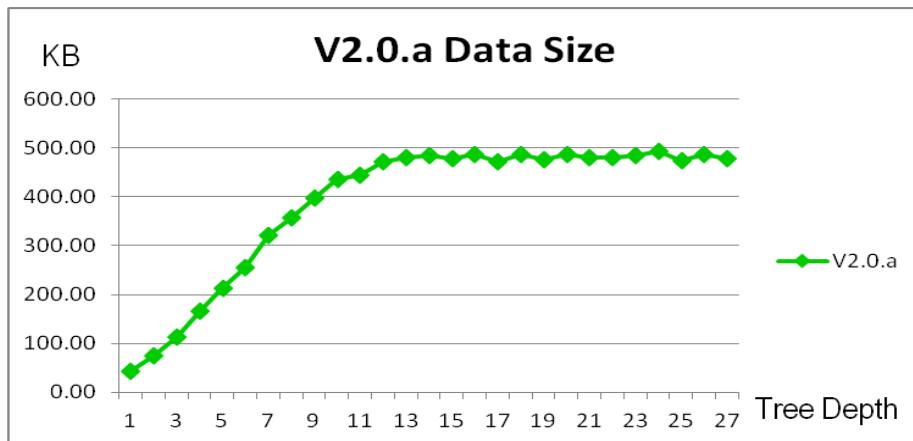


Figure 6-4: DIGGS V2.0a Data Size (KB) up to Test Level 27

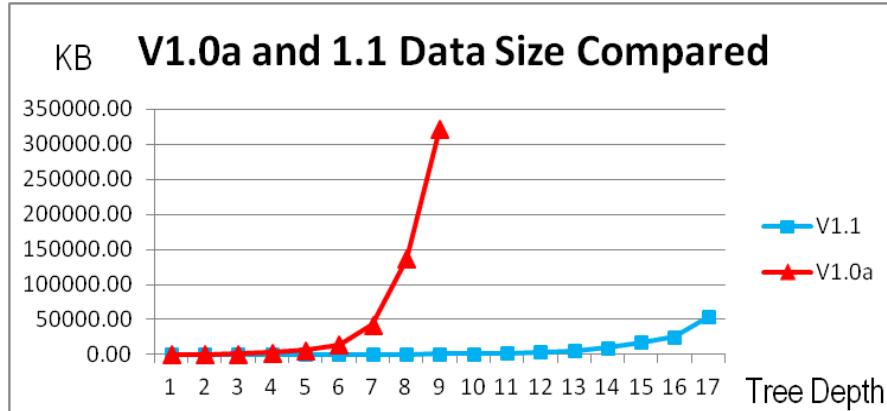


Figure 6-5: DIGGS V1.0a and V1.1 Data Size (KB) Comparison

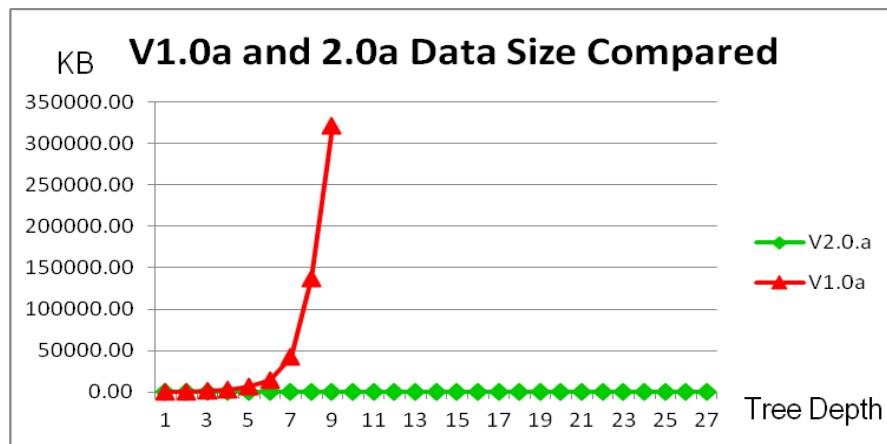


Figure 6-6: DIGGS V1.0a and V2.0a Data Size (KB) Comparison

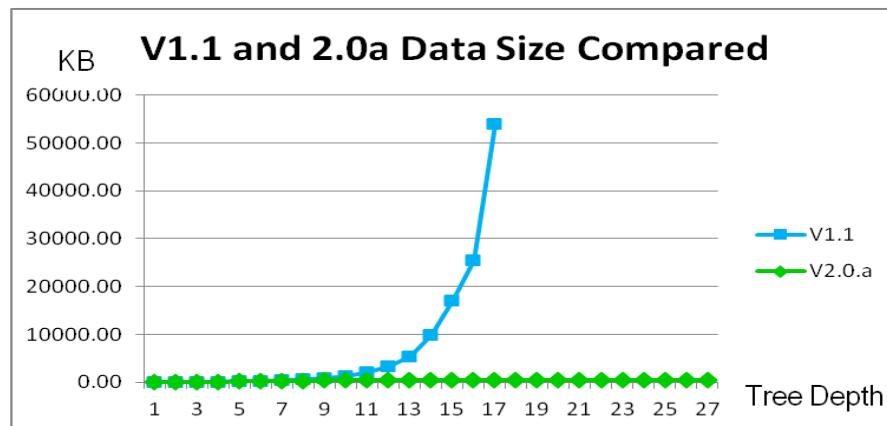


Figure 6-7: DIGGS V1.1 and V2.0a Data Size (KB) Comparison

7 Specialized DIGGS Tools

Two software tools (DIGGS Excel Tool and DIGGS KML Tool) were developed and released with DIGGS 2.0a to test the usability of DIGGSML 2.0 and facilitate its adoption. These tools are convenient software utilities that were developed to process DIGGSML data and to demonstrate that DIGGS V2.0a is implementable and can be supported by custom and third party software tools (e.g. GeoTools) using common web technology standards, best practices and techniques.

The following sub-section 7.1 provides a brief overview of each of these tools.

7.1 Overview

7.1.1 DIGGS Excel Tool

The DIGGS Excel Tool was designed to present the text of DIGGSML data in a human readable spreadsheet format (Microsoft Excel XLS). The tabular representation of DIGGS data in Excel follows the standard XML mapping best practice as summarized in the following table and illustrated with a screen shot from the DIGGS Excel Tool in Figure 7-1.

General DIGGSML to Excel Table Mapping Rules	
DIGGSML	Excel
Feature/Object Name (element name)	Worksheet Name
Feature/Object Content (complexType contents)	Table in Worksheet
Property element name	Table row under 'Property Name' column
Property value (simple)	Value entered in cell under 'Value' column
Property value (complex/Object)	Linked cell to new worksheet under 'Value' column
Attribute name	Table row under 'Attribute Name' column
Attribute value (always simple)	Value entered in cell under 'Value' column

A	B	C	D
1 Home Page	Previous Page	Next Page	
2			
3 Property Name	Attribute Name	Value	
4	ID:	p1	
5 Name		Test Project	
6 Remark		#temp-id-1	
7 Related Sampling Feature Ref		#LB_Webster	
8 Related Sampling Feature Ref		#a22	
9 Related Sampling Feature Ref		#cpt-1	
10 Sampling Activity Ref		#zyx	
11 Group Ref		#g1	
12			
13			
14			
15			
Diggs Project Ground Remark Borehole Role PointLocation LocationAccuracy LinearExtent			

Figure 7-1: DIGGS Data Layout in Project Worksheet of DIGGS Excel Tool

7.1.2 DIGGS KML Tool

The DIGGS KML Tool was designed to visualize 3D DIGGS data (primarily location, shape and identification information) in a popular and freely available earth browser (as KML in Google Earth). The DIGGS KML Tool supports coordinate transformations from DIGGS data to the standard KML CRS (longitude, latitude, elevation), as it is built upon the open source GeoTools spatial engine that has been extended to support all of the DIGGS 3D Compound CRS definitions. Linear referenced and planar referenced geometries are also supported in the DIGGS KML Tool. A sample screen shot of a planar referenced TrenchWall polygon is shown in Figure 7-2.

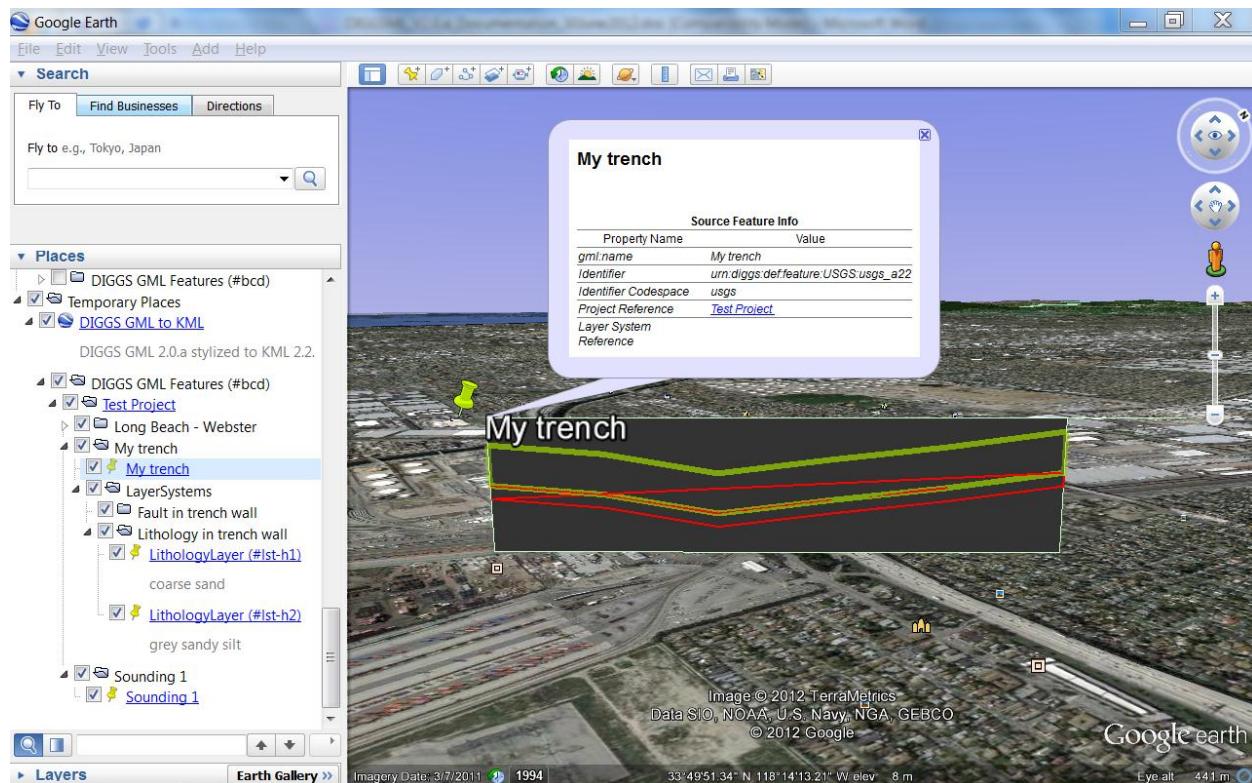


Figure 7-2: Representation of a Planar Referenced TrenchWall Polygon in the DIGGS KML Tool

7.2 DIGGS Excel Tool Installation

7.2.1 DIGGS Excel Tool Operating Environment

Microsoft Office	MS Office version 2003 or later
------------------	---------------------------------

7.2.1.1 *Supported Versions of Office*

Only Microsoft Office is supported at this time.

Other office products, such as OpenOffice, may not support all of the VBA code and have not been tested.

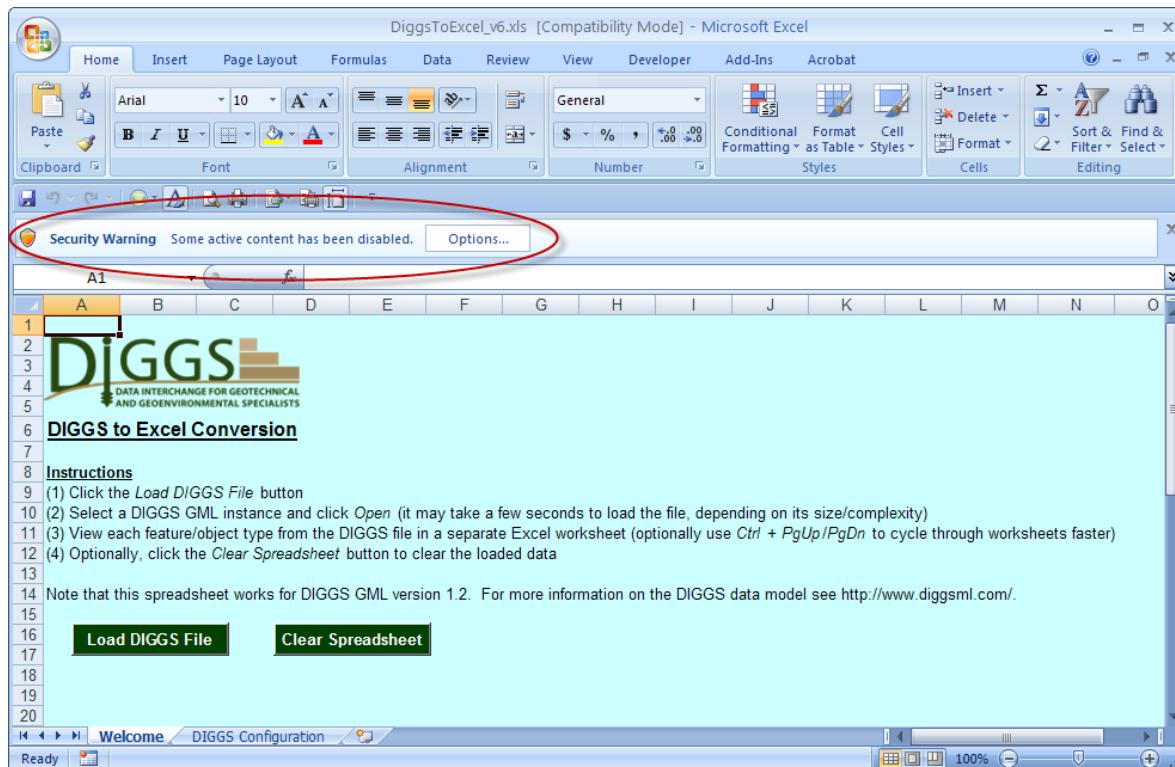
Note that the spreadsheet uses embedded Visual Basic for Applications (VBA) to implement the DIGGS to Excel conversion.

7.2.1.2 *Enable Macros in Excel*

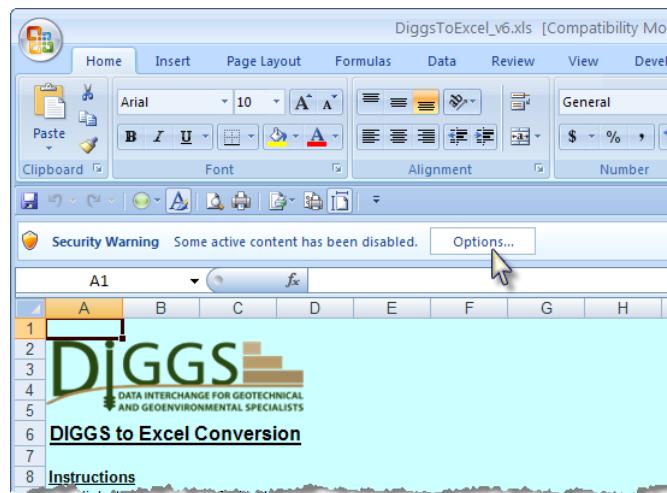
Macros must be enabled in Excel otherwise the conversion functions will not work. This requires the user to explicitly enable macros as they can pose a security threat from untrusted sources.

To enable macros in Excel:

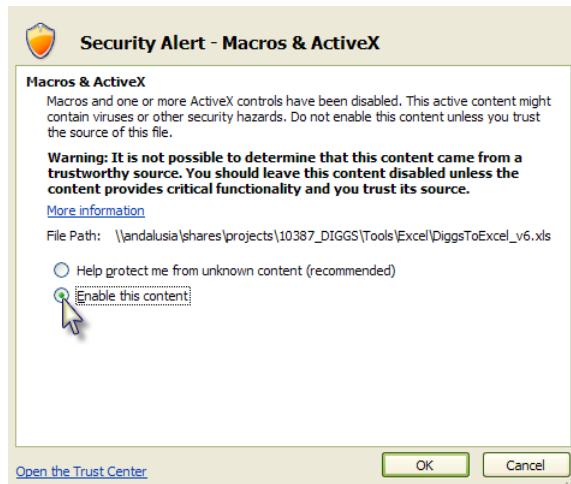
1. Open the DiggstoExcel_v7.xls file;
2. If macros are not already enabled, a Security Warning will be displayed:



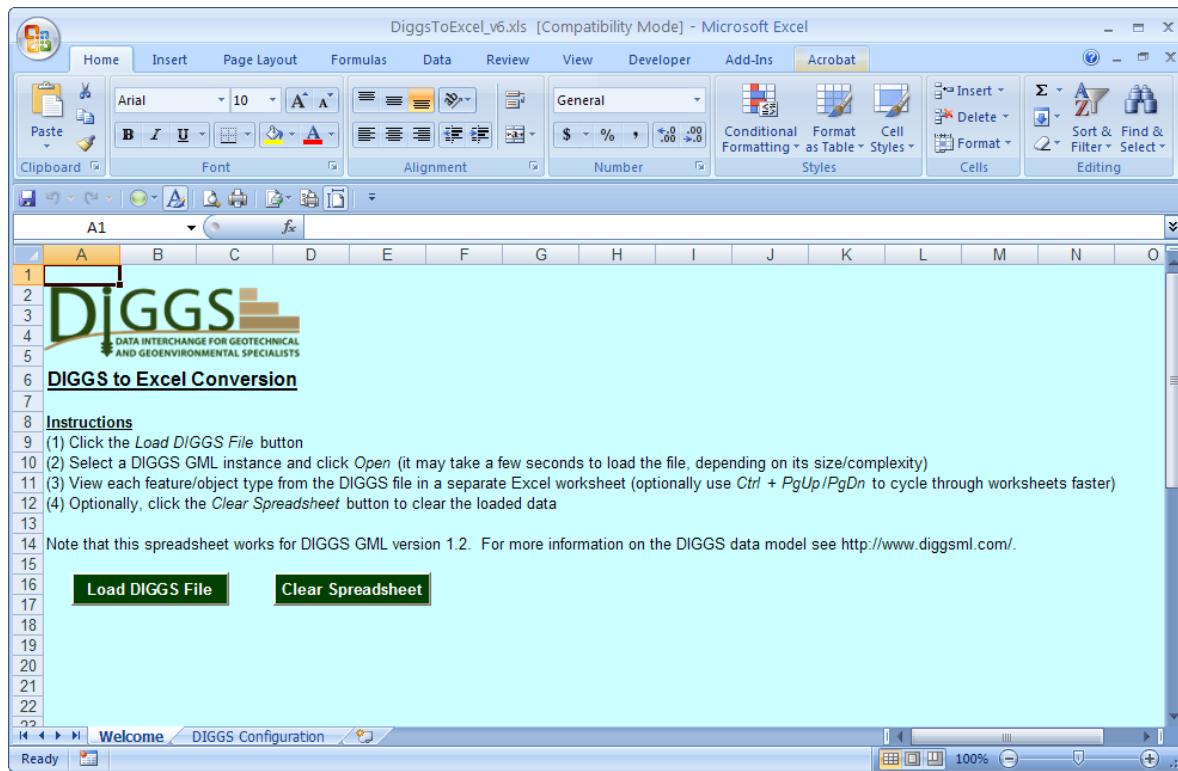
- Click on the [Options] button to open the Security Alert dialogue box:



- In the Security Alert dialogue box, click on the option to enable the macro content:



5. Click [OK] to save the changes;
6. The Security Warning will no longer be displayed:



DiggsToExcel_v6.xls [Compatibility Mode] - Microsoft Excel

DIGGS to Excel Conversion

Instructions

- (1) Click the Load DIGGS File button
- (2) Select a DIGGS GML instance and click Open (it may take a few seconds to load the file, depending on its size/complexity)
- (3) View each feature/object type from the DIGGS file in a separate Excel worksheet (optionally use *Ctrl + PgUp/PgDn* to cycle through worksheets faster)
- (4) Optionally, click the Clear Spreadsheet button to clear the loaded data

Note that this spreadsheet works for DIGGS GML version 1.2. For more information on the DIGGS data model see <http://www.diggsml.com/>.

Load DIGGS File **Clear Spreadsheet**

7. The DIGGS Excel Tool is now ready to be used.

7.3 Using the DIGGS Excel Tool

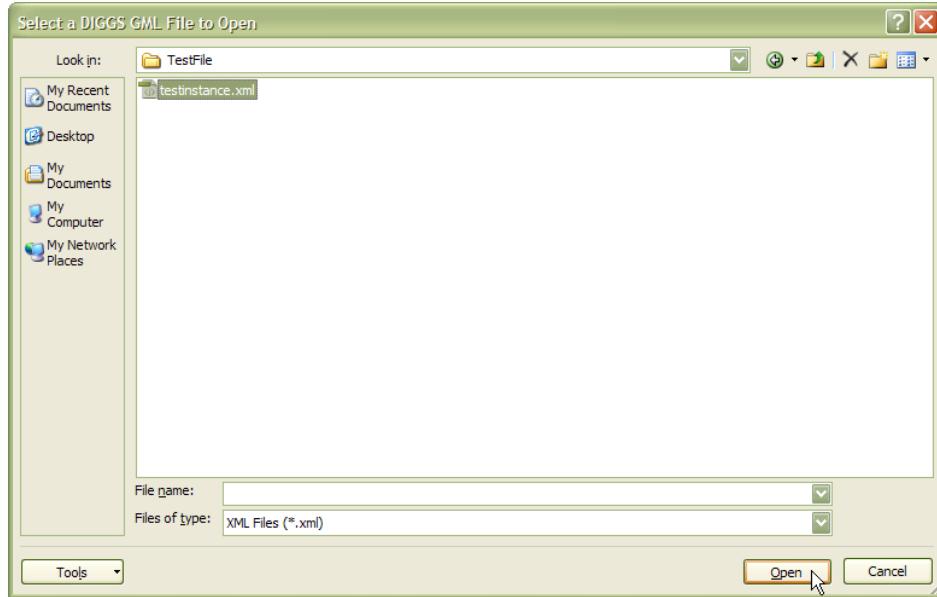
The DIGGS Excel Tool is used to convert DIGGS v2.0a data to an Excel spreadsheet.

- ! The tool is designed to support DIGGS v2.0a. Other versions of DIGGS may not be converted properly.

7.3.1 Convert a DIGGS GML File to an Excel Spreadsheet

To use the DIGGS GML to Excel conversion tool:

1. Open the spreadsheet in Microsoft Excel (2003 or later);
2. Enable Macros in Excel (if not already done);
3. Click the [Load DIGGS File] button;
4. In the dialogue box, navigate to the folder containing the file to be converted;
5. Select the DIGGS file:



6. Click [Open] to convert the file or [Cancel] to abort the action.

Depending on the size and complexity of the selected file, it may take a few seconds to load.

It is recommended that the spreadsheet always be cleared before loading another DIGGS file, otherwise data from the second file will be appended to the data from the first file.

7.3.2 View the DIGGS Data in Excel

After a DIGGS GML instance has been loaded into Excel, the spreadsheet will open the tab containing the root DIGGS element:

A	B	C	D	E	F
1	Home Page	Previous Page	Next Page		
2					
3	Property Name	Attribute Name	Property Value		
4	Diggs	ID:	bcd		
5	Document Information		#d1		
6	Project		#p1		
7	Borehole		#LB_Webster		
8	Trench Wall		#a22		
9	Borehole		#cpt-1		
10	Density Test		#d123		
11	Static Cone Test		#d1e242		
12	Geophysical Log Test		#gl1		
13	Sampling Activity		#xyz		
14	Sampling Activity		#zyx		
15	Sampling Activity		#pointSample		
16	Sample		#s321		
17	Sample		#s123		
18	Sample		#sampt		
19	Layer System		#ls-1		
20	Layer System		#ls2		
21	Layer System		#lst1		
22	Layer System		#fs1		
23	Layer System		#lst_usc		
	Diggs	DocumentInformation	SoftwareApplication	Project	Borehole
					Poi

Figure 3: DIGGS File Overview Worksheet

Multiple additional tabs will have been added to the spreadsheet; each tab accesses a worksheet containing one of the DIGGS feature/object types from the converted file:

	A	B	C	D	E	F
1	Home Page	Previous Page	Next Page			
2						
3	Property Name	Attribute Name	Property Value			
4	Document Information		ID: d1			
5	Creation Date		2010-01-01			
6	Effective Date		2009-01-01			
7	Author Ref		#a23			
8	Software Application		#sap1			
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

DocumentInformation SoftwareApplication Project Borehole Point Line

Figure 4: Document Information Worksheet

	A	B	C	D	E	F
1	Home Page	Previous Page	Next Page			
2						
3	Property Name	Attribute Name	Property Value			
4	Software Application		ID: sap1			
5	Name		GINT v.8			
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

Navigation icons: back, forward, search, etc.

Software Application Project Borehole Point LineString LinearSpatialRef

Figure 5: Software Application Worksheet

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4	Project	ID:	p1
5	Name		Test Project
6	Remarks		
7	Remark		
8	Content		This could be a virtual project that has no relevance
9	Associated Location Ref		#LB_Webster
10	Associated Location Ref		#a22
11	Associated Location Ref		#cpt-1
12	Sampling Activity Ref		#zyx
13	Group Ref		#q1
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

Navigation icons: back, forward, search, etc.

Category tabs: Project, Borehole, Point, LineString, LinearSpatialReferenceSystem, Linea

Figure 6: Project Worksheet

A	B	C	D
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4	Borehole	ID:	LB_Webster
5	Name	Code Space:	Long Beach - Webster
6		Code Space:	USGS Common Name
7	Name	Code Space:	LB_Webster
8		Code Space:	USGS_ID
9	Name		334904118130301
10		Code Space:	USGS Site ID
11	Name		004S013W23D003S
12		Code Space:	Calif. State Well No.
13	Name		004S013W23D004S
14		Code Space:	Calif. State Well No.
15	Name		334904118130302
16		Code Space:	USGS Site ID
17	Name		004S013W23D005S
18		Code Space:	Calif. State Well No.
19	Name		334904118130303
20		Code Space:	USGS Site ID
21	Name		004S013W23D006S
22		Code Space:	Calif. State Well No.
23	Name		334904118130304
 Borehole Point LineString LinearSpatialReferenceSystem LinearReferencin			

Figure 7: Borehole Worksheet

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4	Point	ID:	a33
5		SRS Name:	urn:diggs:def:crs:DIGGS:0.1:26911_5703
6		SRS Dimension:	3
7	Pos		387316.665116977 3742645.12297961 7.81507
8	Reference Point Accuracy		
9	Positional Accuracy		
10	Measurement Method		GPS
11	Result		5
12		UOM:	m
13	Point	ID:	lb_web_td
14		SRS Dimension:	1
15		SRS Name:	#sr123
16	Pos		427.9392
17	Borehole Purpose		Multi-port Monitoring Well
18	Point	ID:	hd1
19	Pos		15
20		SRS Dimension:	1
21		SRS Name:	#sr123
22	Point	ID:	wsp1
23		SRS Name:	#sr123
		Point	LineString
		LinearSpatialReferenceSystem	LinearReferencingMethod

Figure 8: Point Worksheet

	A	B	
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4	Line String	ID: ls	
5		SRS Name: urn: diggs: def: crs: DIGGS: 0.1: 26911_5703	
6		SRS Dimension: 3	
7	Pos List		387316.665116977 3742645.12297961 7
8	Line String	ID: bfls1	
9		SRS Dimension: 1	
10		SRS Name: #sr123	
11	Pos List		5 10
12	Back Fill Material		bentonite
13		Code Space: USGS	
14	Line String	ID: ply1_top	
15		SRS Name: urn: diggs: def: crs: DIGGS: 0.1: 26911_5703	
16	Pos List		387516.665116977 3742645.12297961 5
17	Line String	ID: ls1	
18		SRS Dimension: 3	
19		SRS Name: urn: diggs: def: crs: DIGGS: 0.1: 26911_5703	
20	Pos List		387416.665116977 3742645.12297961 6
21	Line String	ID: cptls-2	
22	Pos List		0 5.44
23		CPO Dimension: 1	

 **LineString** **LinearSpatialReferenceSystem** **LinearReferencingMethod** **Backfill**

Figure 9: LineString Worksheet

	A	B	C
1	Home Page	Previous Page	Next Page
2			
3	Property Name	Attribute Name	Property Value
4	Linear Spatial Reference System	ID:	sr123
5	Identifier		urn:diggs:def:fi:DIGGSINC:sr123
6		Code Space:	urn:x-diggs:def:authority:DIGGSINC
7	Linear Element		#ls
8	Linear Referencing Method		#lr123
9	Linear Spatial Reference System	ID:	cptsr1
10	Identifier		urn:diggs:def:fi:DIGGSINC:cptsr1
11		Code Space:	urn:x-diggs:def:authority:DIGGSINC
12	Linear Element		#ls1
13	Linear Referencing Method		#rcpt1
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

 [LinearSpatialReferenceSystem](#) [LinearReferencingMethod](#) [Backfill](#) [TimeInterv](#)

Figure 10: Linear Spatial Reference System Worksheet

	A	B	C	D	E
1	Home Page	Previous Page	Next Page		
2					
3	Property Name	Attribute Name	Property Value		
4	Linear Referencing Method	ID:	lr123		
5	Name		chainage		
6	Type		absolute		
7	Units		m		
8	Linear Element Accuracy				
9	Positional Accuracy				
10	Measurement Method		GPS		
11	Result		5		
12		UOM:	m		
13	Linear Referencing Method Accuracy				
14	Positional Accuracy				
15	Measurement Method		driller		
16	Result		1		
17		UOM:	ftUS		
18	Linear Referencing Method	ID:	lrcpt1		
19	Name		chainage		
20	Type		absolute		
21	Units		m		
22					
23					

Figure 11: Linear Referencing Method Worksheet

In some cases, the GML object is shown as a special table to better display complicated data structures; one such example is the Log object.

The Log object typically has a large amount of data, and that data is related to other elements within the object. In this case, a table is created for this specific object that helps the user better visualize the data; the `LogData` column headers are shown and the data for each column is taken from other parts of the GML object and displayed below:

A	B	C	D	E	F
1	Home Page	Previous Page	Next Page		
2					
3	Property Name	Attribute Name	Property Value		
4	Log	ID:	MPC001		
5		Log position (#cptsr1)	Tip Resistance (kN/m ²)	Sleeve Resistance (kN/m ²)	Friction Ratio
6		0.010	0.1300	0.40	0.0000
7		0.020	0.2400	0.40	0.1000
8		0.030	0.5500	0.40	0.0040
9		0.040	0.6800	0.40	0.0070
10		0.050	0.7800	0.30	0.0120
11		0.060	0.9000	0.30	0.0150
12		0.070	0.9600	0.40	0.0200
13		0.080	1.0400	0.40	0.0240
14		0.090	1.0700	0.30	0.0270
15		0.100	1.1000	0.30	0.1000
16		0.110	1.1300	0.40	0.0350
17		0.120	1.1800	0.30	0.0400
18		0.130	1.2400	0.40	0.0430
19		0.140	1.2600	0.40	0.0460
20		0.150	1.2600	0.40	0.0480
21		0.160	1.2800	0.40	0.0490
22		0.170	1.2900	0.40	0.0500
23		0.180	1.2600	0.30	0.0500

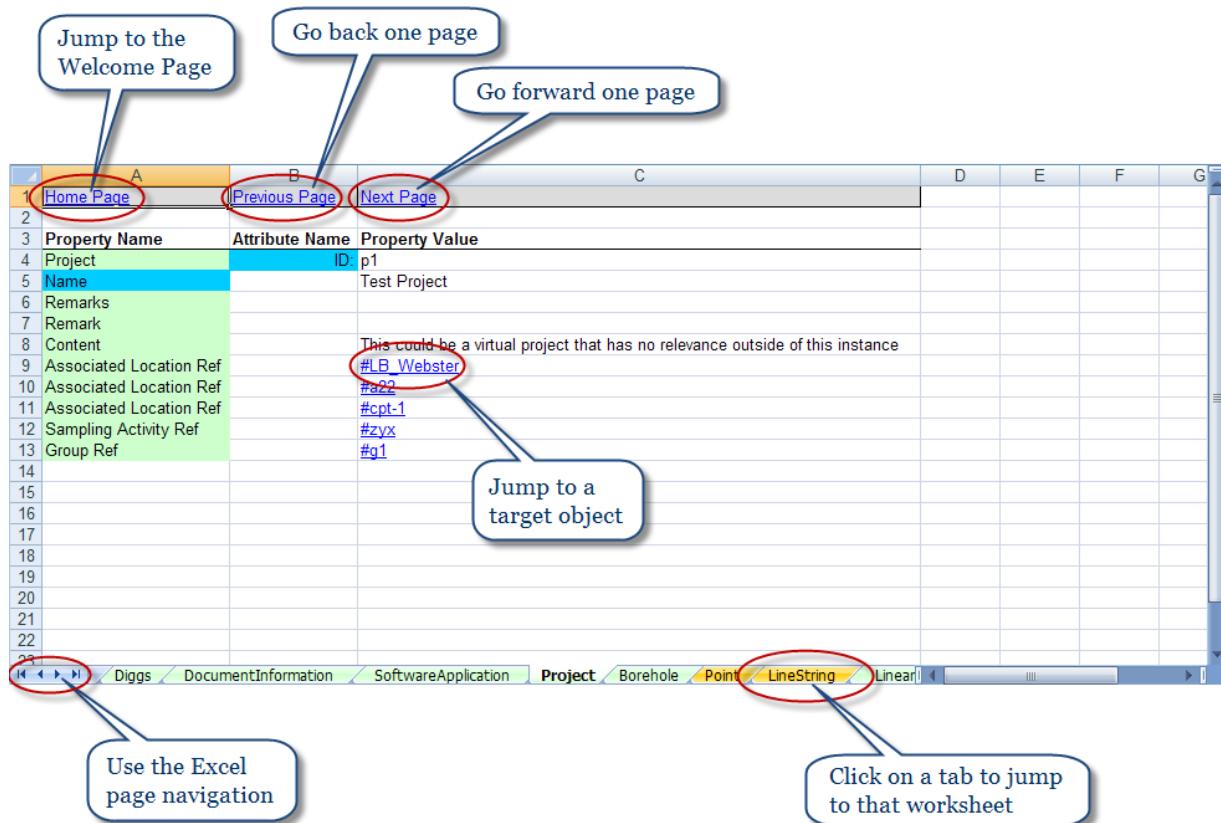
Figure 12: The LogData Worksheet

7.3.3 Navigating the DIGGS Data Worksheets

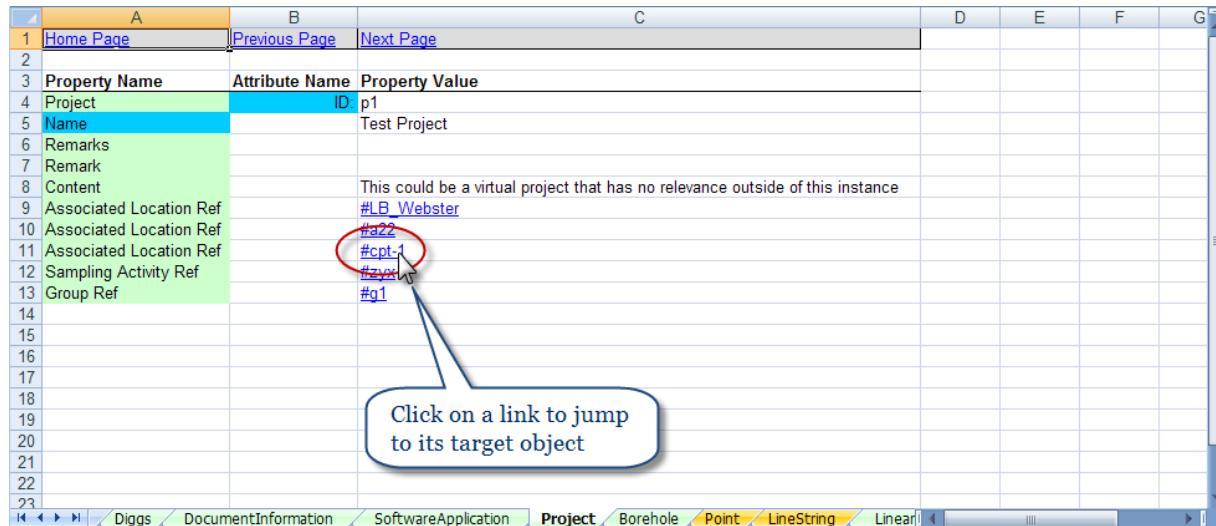
To manually navigate between the pages:

1. Use the **Previous Page** and **Next Page** links at the top of each worksheet to page through the worksheets or to return to the **Welcome Page**;
2. Use the Excel page navigation tools in the bottom left corner of the spreadsheet;
3. Use the **Ctrl + PageUp/PageDown** keyboard shortcuts.

The various worksheets containing objects from the converted DIGGS GML file can also be manually browsed by clicking on the tabs along the bottom of the spreadsheet:

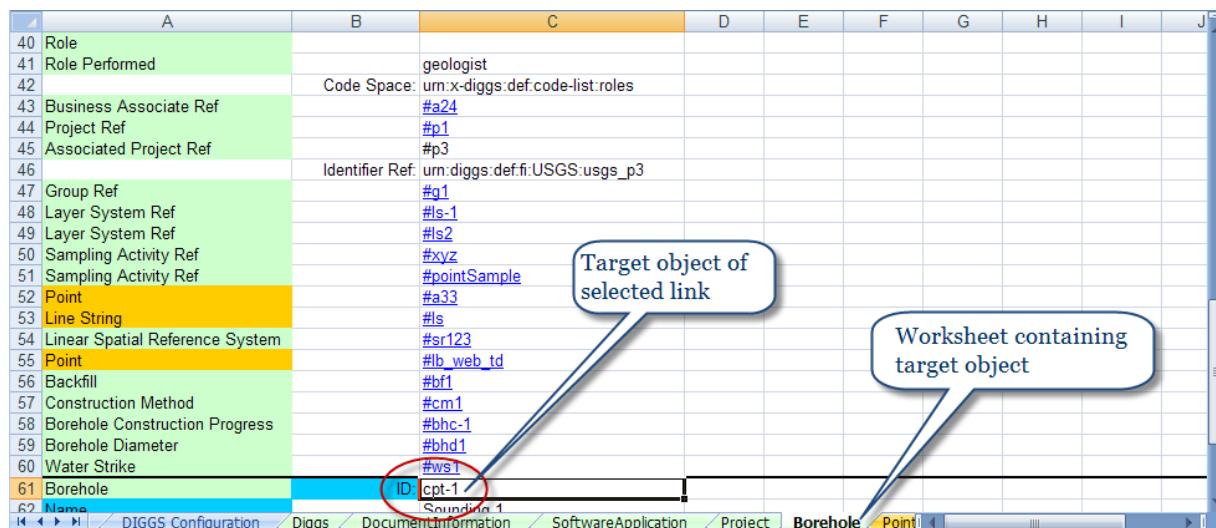


Features/objects can be accessed directly by clicking on the links in the spreadsheet:



A	B	C	D	E	F	G
1 Home Page	Previous Page	Next Page				
2						
3 Property Name	Attribute Name	Property Value				
4 Project	ID	p1				
5 Name		Test Project				
6 Remarks						
7 Remark						
8 Content		This could be a virtual project that has no relevance outside of this instance				
9 Associated Location Ref		#LB_Webster				
10 Associated Location Ref		#a22				
11 Associated Location Ref		#cpt-1				
12 Sampling Activity Ref		#zwp				
13 Group Ref		#g1				
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

Clicking on a link redirects to the object identified by the link:



A	B	C	D	E	F	G	H	I	J
40 Role									
41 Role Performed		geologist							
42		Code Space:	urn:x-diggs:def:code-list:roles						
43 Business Associate Ref		#a24							
44 Project Ref		#p1							
45 Associated Project Ref		#p3							
46		Identifier Ref:	urn:diggs:def:fi:USGS:usgs_p3						
47 Group Ref		#g1							
48 Layer System Ref		#ls-1							
49 Layer System Ref		#ls2							
50 Sampling Activity Ref		#xyz							
51 Sampling Activity Ref		#pointSample							
52 Point		#a33							
53 Line String		#ls							
54 Linear Spatial Reference System		#sr123							
55 Point		#lb_web_td							
56 Backfill		#bf1							
57 Construction Method		#cm1							
58 Borehole Construction Progress		#bhc-1							
59 Borehole Diameter		#bhd1							
60 Water Strike		#ws1							
61 Borehole	ID	cpt-1							
62 Name		Sounding_1							

7.3.4 Save the Converted Excel Spreadsheet

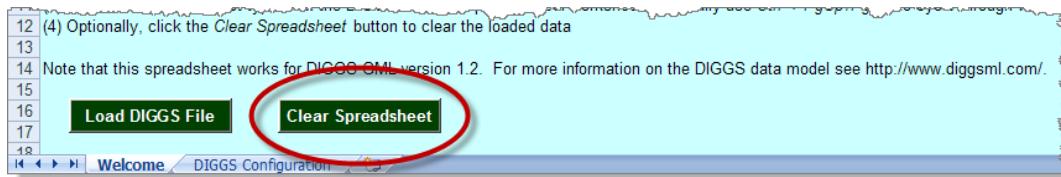
To save the converted file:

1. From the [Office] button or the **File** menu, select **Save As...**;
2. In the dialog box, navigate to the folder where the converted file will be saved;

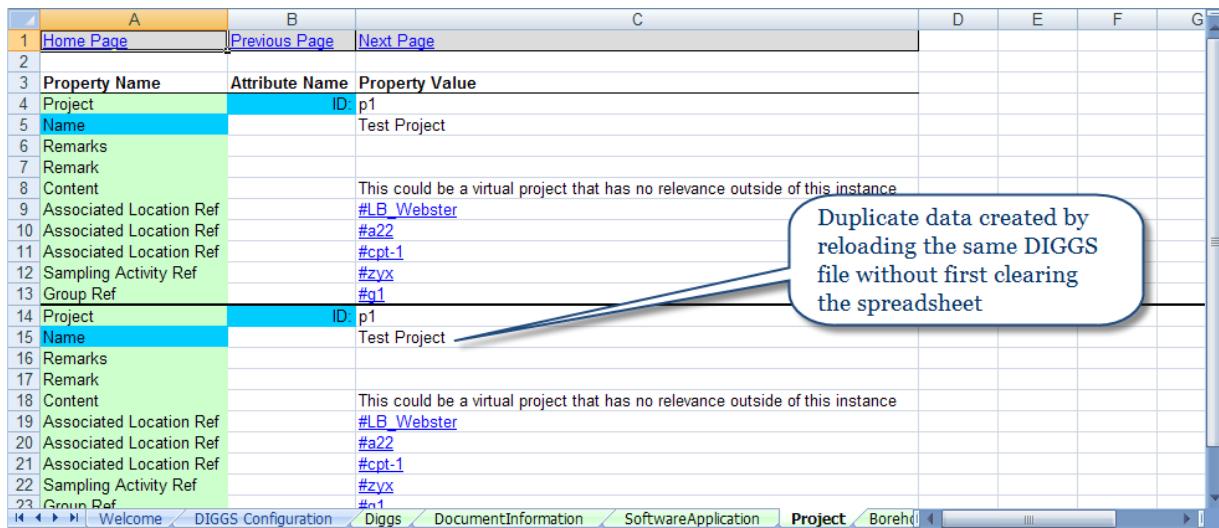
3. Enter an appropriate filename;
4. Click [Save] to save a copy of the file or [Cancel] to abort the action.

7.3.5 Clear the Excel Spreadsheet

Clear the loaded DIGGS instance by clicking the [Clear Spreadsheet] button in the Welcome worksheet:



If the spreadsheet is not cleared before another file is loaded, the data from the second file will be processed into the same worksheet, and it will be appended to the data from the first file:



The screenshot shows a spreadsheet with two rows of data. The first row contains columns for 'Property Name', 'Attribute Name', and 'Property Value'. The second row contains similar columns. A callout bubble points to the second row with the text: 'Duplicate data created by reloading the same DIGGS file without first clearing the spreadsheet'.

A	B	C	D	E	F	G
1 Home Page	2 Previous Page	3 Next Page				
4 Project	5 Name	6 Remarks	7 Remark	8 Content	9 Associated Location Ref	10 Associated Location Ref
					#LB_Webster	#a22
					#cpt-1	
					#zyx	
					#q1	
14 Project	15 Name	16 Remarks	17 Remark	18 Content	19 Associated Location Ref	20 Associated Location Ref
					#LB_Webster	#a22
					#cpt-1	
					#zyx	
					#q1	

7.3.6 Configuration Options

Various configuration options, for name mapping and color coding, are available in the spreadsheet under the **DIGGS Configuration** tab. This supports better readability of the converted data.

The configuration options allow for mapping the feature and property names in the application schema to a more human-readable format.

The values listed in the worksheet present a mapping between the DIGGS GML schema and the readable names presented in the Excel spreadsheet.

A	B	C	D	E	F	G	H
1 Notes							
2 The values listed here present a mapping between the DIGGS GML schema and the readable names presented in the Excel spreadsheet.							
3 Feel free to add additional values in each section, but do not change the heading names or the order of the sections (or the worksheet names).							
4 Make entries for features or properties according to their fully-qualified name (with namespace prefix).							
5 The colour code priority is as follows (first match is the colour that is used): 1) feature name, 2) feature property, 3) namespace.							
6							
7 [Namespace Prefix]	[Namespace URI]	[Colour Code]					
8 gml	http://www.opengis.net/gml/3.2						
9 g3.3	http://www.opengis.net/gml/3.3						
10 g3.3	http://www.forward_compatible_definitions.net/gml/3.3						
11 glr	http://www.opengis.net/gml/3.3/lr						
12 glrov	http://www.opengis.net/gml/3.3/lrov						
13 xlink	http://www.w3.org/1999/xlink						
14 xsi	http://www.w3.org/2001/XMLSchema-instance						
15 witsml	http://www.witsml.org/schemas/131						
16 diggs	http://schemas.diggsml.com/1.2a						
17 diggs_geo	http://schemas.diggsml.com/1.2a/geotechnical						
18 diggs_env	http://schemas.diggsml.com/1.2a/environmental						
19 diggs_mon	http://schemas.diggsml.com/1.2a/monitoring						
20 diggs_pil	http://schemas.diggsml.com/1.2a/piling						
21							
22 [Schema Feature Name]	[Readable Feature Name]	[Colour Code]					
23 CPTCone	CPT Cone						
24							
25 [Schema Feature Property Name]	[Readable Feature Property Name]	[Colour Code]					
26 srsName	SRS Name						
27 srsDimension	SRS Dimension						
28 gml:name	Name						
29 gml:id	ID						
30 xlink:href	XLink HREF						
31 uom	UOM						
32							
33							
34							
35							
36							
37							

The colour code priority is:

1. specified colour
2. feature name
3. feature property
4. namespace

A namespace and colour code has been added for older data:

A	B	C	D	E	F	G
Notes						
1 The values listed here present a mapping between the DIGGS GML schema and the readable names presented in the Excel spreadsheet.						
2 Feel free to add additional values in each section, but do not change the heading names or the order of the sections (or the worksheet names).						
3 Make entries for features or properties according to their fully-qualified name (with namespace prefix).						
4 The colour code priority is as follows (first match is the colour that is used): 1) feature name, 2) feature property, 3) namespace.						
5						
6						
[Namespace Prefix]	[Namespace URI]	[Colour Code]				
8 gm1	http://www.opengis.net/gml/3.2					
9 g3.3	http://www.opengis.net/gml/3.3					
10 g3.3	http://www.forward-compatible-definitions.net/gml/3.3	Legacy namespace for older instances.				
11 glr	http://www.opengis.net/gml/3.3/lr					
12 glrov	http://www.opengis.net/gml/3.3/irov					
13 xlink	http://www.w3.org/1999/xlink					
14 xsi	http://www.w3.org/2001/XMLSchema-instance					
15 witsml	http://www.witsml.org/schemas/131					
16 diggs	http://schemas.diggsml.com/1.2a					
17 diggs_geo	http://schemas.diggsml.com/1.2a/geotechnical					
18 diggs_env	http://schemas.diggsml.com/1.2a/environmental					
19 diggs_mon	http://schemas.diggsml.com/1.2a/monitoring					

7.3.6.1 *Changing Configuration Values*

To change existing configuration values:

1. Right-click in the cell containing the Colour Code to be changed;
2. From the shortcut menu, select **Format Cells...**;
3. In the dialog box, click on the **Fill** tab;
4. Select one of the displayed colour swatches, or click on **More Colors...** and select a colour using the Color Picker;
5. Click [OK] to accept the new colour or [Cancel] to abort the change.

7.3.6.2 ***Adding New Configuration Values***

Additional values can be added in each section.

- ! If additional configuration values are added, DO NOT change:
- the column heading names,
 - the order of the sections, or
 - the worksheet names.

Create entries for features or properties according to their fully-qualified name, and make sure to include the namespace prefix.

To add new configuration values:

1. Click in the blank row immediately after the last configuration item of the type to be added;
If the feature or property in the row above has a specified colour, this will be copied into the Colour Code cell of the new row;
2. Enter the feature or property name and its readable name in the appropriate columns;
3. Change the Colour Code to the desired colour for the feature or property;
4. Save a new copy of the spreadsheet to make the new configuration a permanent part of the DIGGS-to-Excel Converter.

7.4 ***DIGGS KML Tool Installation***

7.4.1 ***DIGGS KML Tool Operating Environment***

JDK	Java JDK 6 or higher
-----	----------------------

7.4.1.1 *Check for Existing Java Installation*

To check whether Java has been installed, and to see which version is running, use the tool provided on the Java website:

- <http://www.java.com/en/download/installed.jsp>

7.4.1.2 *Install Java*

If Java is not installed, or if an earlier version is running, download the appropriate package and follow the instructions to install the latest Java JDK:

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

7.4.1.3 *Set the JAVA_HOME Environment Variable*

When the installation is complete, add or update your JAVA_HOME environment variable to the JDK installation path:

1. In Windows, from the **Start** menu select **Control Panel**;
2. Open the **System** dialogue box;
3. Click on the tab to access the **Advanced** settings;
4. Click on the [Environment Variables] button at the bottom;
5. In the list of System Variables, create or update the JAVA_HOME system variable:
for example:

```
JAVA_HOME=C:\Program Files\Java\jdk1.6.0_10
```

7.4.1.4 *Install the Application Files*

To install the application files:

1. Download and save the package containing the application files; the application files are packaged in the diggs-kml-2.0.zip file;
2. Create a folder to install the application files into, for example: **C:\diggstools**;
3. Extract the contents of the diggs-kml-2.0.zip file into the application folder just created.

7.4.1.5 *Manifest for the Application Package*

The following is a list of the folders and files contained in the diggs-kml-2.0.zip file. This shows the directory structure and contents that will be extracted to the application folder:

```
📁 Excel
    📄 DiggsToExcel_v7.xls

📁 KML
    📄 diggs-kml-2.0
        📄 README.txt
    📁 bin
        📄 configuration.props
        📄 launch.bat
        📄 launch.sh
    📁 conf
        📄 configuration.properties
        📄 crsAuthorityCodeEPSGMapping.xml
        📄 DIGGS_KML_STYLING.xml
        📄 DIGGS_WKT_CRS_DICTIONARY.txt
    📁 xsl
        📄 diggs-gml2kml.xsl
        📄 srsConverter.xsl
    📁 lib
        📄 appframework-1.0.3.jar
        📄 commons-pool-1.5.4.jar
        📄 diggs-0.2.jar
        📄 geoapi-2.3-M1.jar
        📄 geoapi-pending-2.3-M1.jar
        📄 gt-api-2.7-M4.jar
        📄 gt-epsg-hsql-2.7-m4.jar
        📄 gt-main-2.7-M4.jar
        📄 gt-metadata-2.7-M4.jar
        📄 gt-referencing3D-2.7-M4.jar
        📄 gt-referencing-2.7-M4.jar
        📄 hsqldb-1.8.0.7.jar
        📄 jai_core.jar
        📄 jsr-275-1.0-beta-2.jar
        📄 junit-4.4.jar
        📄 saxon9-9.1.0.7.jar
        📄 saxon9-dom-9.1.0.7.jar
        📄 swing-worker-1.1.jar
        📄 vecmath-1.3.2.jar
        📄 xercesImpl-2.9.0.jar
        📄 xml-apis-2.9.0.jar
    📁 sample
        📄 testinstance.xml
```

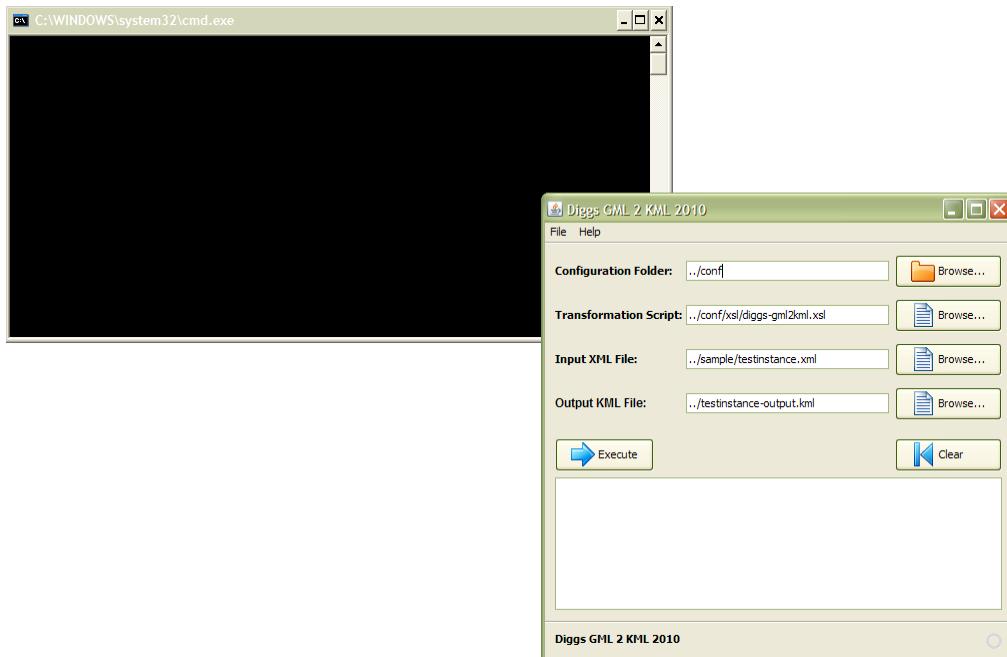
7.5 Using the DIGGS KML Tool

The DIGGS KML Tool is used to convert DIGGS v2.0a data to KML that can be displayed on Google Earth.

7.5.1 Open the DIGGS KML Tool

To start the DIGGS KML Tool:

1. Navigate to the **../bin** folder inside the folder where the application was installed;
2. Run the *launch.bat* file (or *launch.sh* file in Linux) to start the application;
3. The application window will open with an empty **Cmd** window behind it:

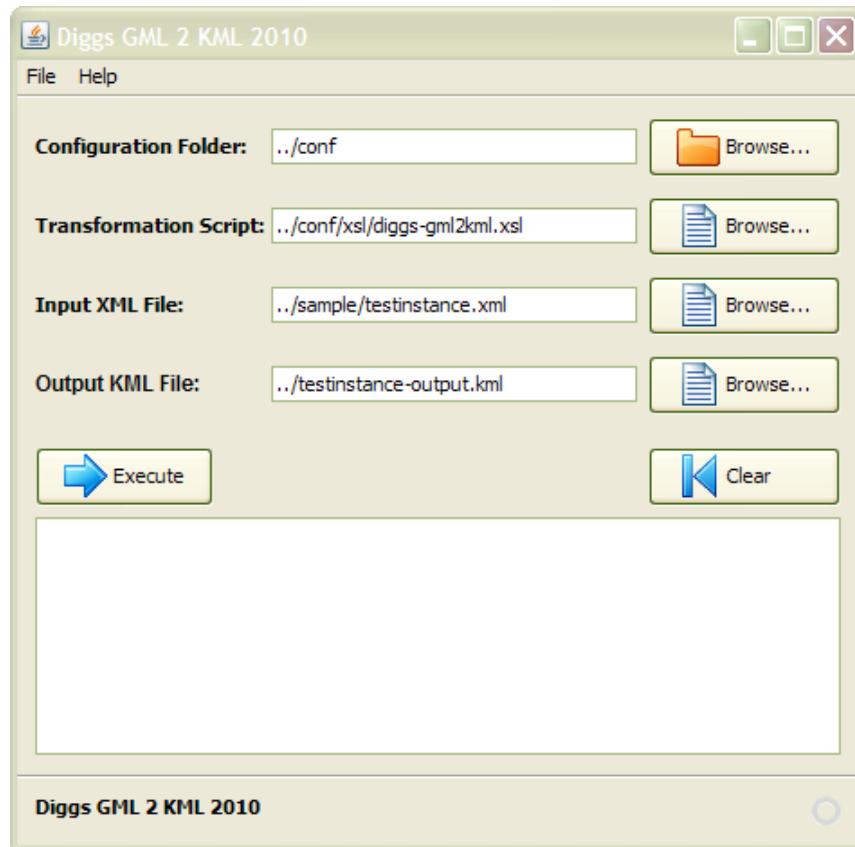


Do not close the **Cmd** window while the DIGGS KML Tool is running. The **Cmd** window runs in the background, and closing the **Cmd** window will close the application.

7.5.2 Process a DIGGS File

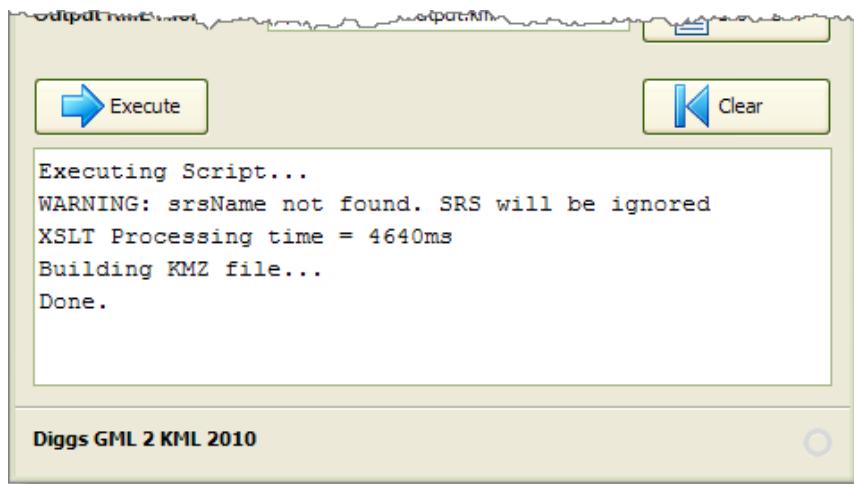
To convert a file:

1. The fields in the application window are prepopulated with the default values in the *configuration.props* file:



2. The configuration property fields specify:
 - a. the **Configuration Folder** where the *configuration.props* file is located;
 - b. the location and filename of the **Transformation Script**;
 - c. the location and filename of the **Input XML File** to be converted;
 - d. the location and filename of the **Output KML File** containing the data converted from the input file;
3. To use different configuration values for any of the fields, use the [Browse...] button for that field to navigate to and select the desired input or output, or enter the correct information into the field manually;
4. Click the [→Execute] button to convert the specified DIGGS file to KML;

5. The text area at the bottom of the window will display messages such as any errors encountered as the file is being processed, the time taken to run the XSLT script, etc.;
6. When the processing is complete, a message will be displayed that the conversion is done:



7. The converted KML file will be saved with the filename and to the location specified in the **Output KML File** field;
8. Process another file [**<Clear**] button.

7.5.3 View the KML File

To view the output file:

1. When the application window opens, select the appropriate configuration directory, XSLT, and input and output files, depending on what conversion is desired;
2. Click the [Execute] button;
3. Open the resulting KMZ file in Google Earth.

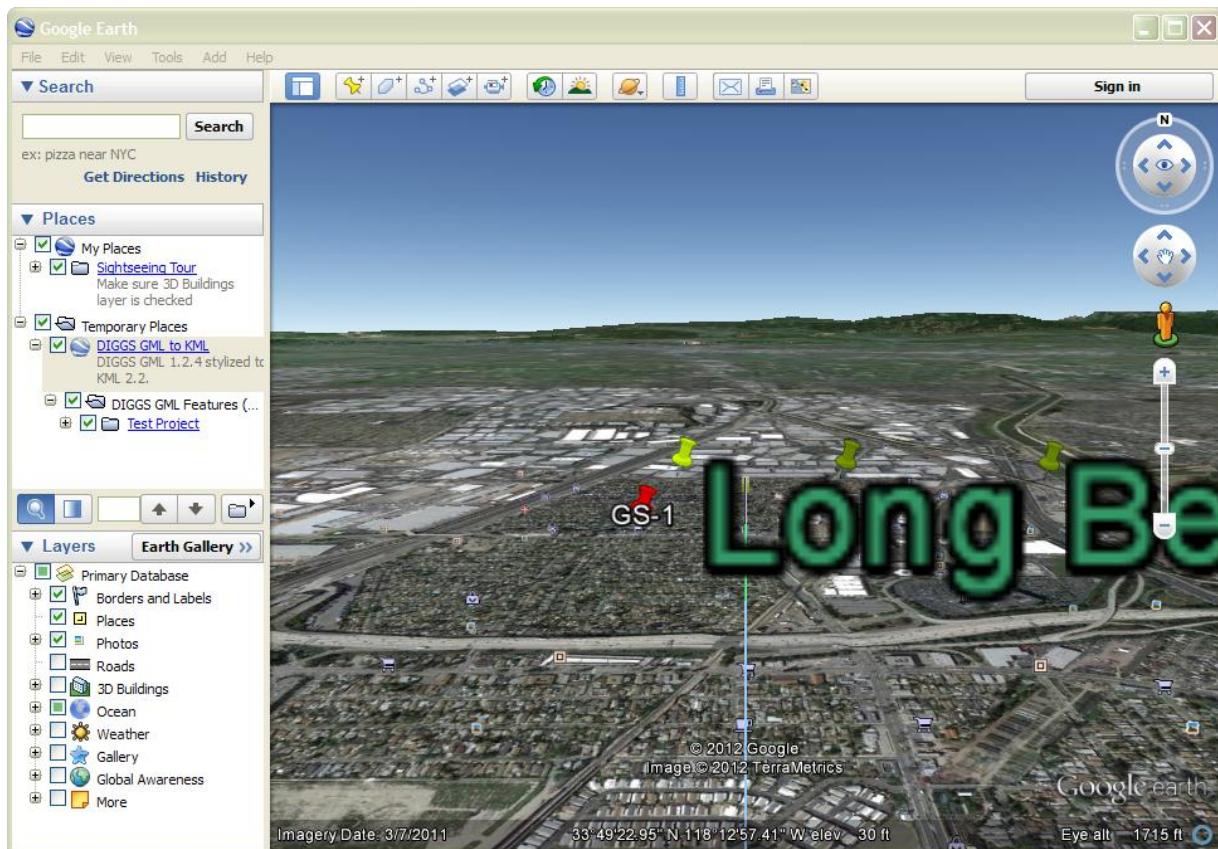


Figure 13: Converted KML displayed in Google Earth

7.5.4 Exit the Converter

To exit the converter and close both the application and the **Cmd** window:

1. Open the **File** menu then click on **Exit**; or

2. Press **Ctrl+Q** on the keyboard.

7.5.5 Configuration Options

There are a number of configuration options for the DIGGS KML Tool.

The DIGGS KML Tool will run without any changes being needed, but the options allow changes to be made to the default settings, and for additional parameters to be specified.

7.5.5.1 Default User Properties

The default user properties are stored in the *configuration.props* file located in the **..../bin** folder.

The default properties are:

Property	PropertyName	Default Value
Configuration Directory	configFolder	../conf
Transformation Script	xsltFolder	../conf/xsl/diggs-gml2kml.xsl
Input XML File	inputFile	../sample/testinstance.xml
Output KML File	outputFile	../testinstance-output.kml

The DIGGS KML Tool remembers the last values used. The values in the configuration fields are automatically updated in the *configuration.props* file on exiting the application so any changes will be saved and used to populate the fields the next time that the application is opened.

7.5.5.2 Additional Parameters

Additional configuration parameters can be found in the set of files in the **..../conf** folder:

- *configuration.properties*
Global application configuration properties (default CRS values, tile server, temporary file management, etc.);
- *crsAuthorityCodeEPSGMapping.xml*
Maps custom CRS definitions to well-known text CRS definitions (CRS to WKT);

- *DIGGS_KML_STYLING.xml*
Styling rules that are applied to the conversion (KML styling rules);
- *DIGGS_WKT CRS DICTIONARY.txt*
List of well-known text (WKT) CRS definitions.

7.5.5.3 ***Changing the Configuration Values***

If desired, the properties in the configuration files can be modified.

Note that when a configuration property is changed the application needs to be restarted.

To modify the additional configuration properties:

1. Open the configuration file containing the value to be changed using an appropriate authoring tool;
2. Locate the property to change;
3. Replace the existing value of the property with the desired new value;
4. Save and close the file;
5. Restart the application for the change to take effect.

! A backup should always be created before any files are modified.

Specific details about the configuration file and how to use it are in the following section.

7.5.6 **Defining Styling for the Output KML**

The *DIGGS_KML_STYLING.xml* file is used as an input to the transformation script that is used to transform a DIGGS file to KML. For more information on the KML 2.2 styling properties see <http://code.google.com/apis/kml/documentation/kmlreference.html>.

7.5.6.1 Opening the KML Styling File

The *DIGGS_KML_STYLING.xml* file is designed to be opened and modified using Microsoft Excel.

To open the *DIGGS_KML_STYLING.xml* file:

1. In Excel, select **Open** from the **File** menu;
2. Navigate to the ..\conf folder in the directory where the DIGGS tools are installed for example: **C:\diggestools\KML\diggs-kml-0.2\conf**
3. Click on the *DIGGS_KML_STYLING.xml* file and open it;
4. If any changes are to be made to the file, immediately save a backup copy with a different filename, then re-open the original file to make modifications.

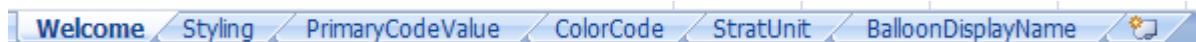
The *DIGGS_KML_STYLING.xml* file will open as an Excel spreadsheet containing a number of worksheets.

7.5.6.2 Saving Changes to the KML Styling File

After changes have been made, save the file with the original filename.

7.5.6.3 Worksheets in the KML Styling File

The *DIGGS_KML_STYLING.xml* file initially contains the following worksheets:



Worksheet Name	Description
Welcome	Worksheet containing information about the spreadsheet and the rules for styling elements and properties.
Styling	Worksheet for specifying the default styling properties for DIGGS elements.
PrimaryCodeValue	Customized worksheet for properties of the DIGGS LithologyLayer element.
ColorCode	Customized worksheet for properties of the DIGGS ColorLayer element.

Worksheet Name	Description
StratUnit	Customized worksheet for properties of the DIGGS StratigraphyLayer element.
BalloonDisplayName	Worksheet for mapping XPath expressions to what will actually be displayed in a balloon

Additional worksheets can be added to define customized styling for properties of specific DIGGS elements.

The Welcome worksheet contains some notes about DIGGS GML, the Rules for how to use the file to specify KML styling, a list of the properties that can be styled, and a list of the supported namespaces:

	A	B	C	D	E	F	G	H	I	J
1										
2										
3	<u>DIGGS to KML Styling</u>									
4										
5	This spreadsheet is used as an input to the DIGGS to KML XSLT.									
6										
7	Note that this spreadsheet works for DIGGS GML version 1.2. For more information on the DIGGS data model see http://www.diggsml.com/ .									
8	For more information on the KML 2.2 styling properties see http://code.google.com/apis/kml/documentation/kmlreference.html									
9										
10										
11	Rules									
12	(1) Styling rules for features are defined in the Styling worksheet									
13	(2) Styling rules based on a feature property are defined in worksheet defined in the 'Property Based Styling - property mapping worksheet' column of the Styling worksheet									
14	(3) Each row in the worksheet represents a unique DIGGS element									
15	(4) Only color, scale, href and the hotSpot KML properties are supported for Point geometries									
16	(5) Only color and width KML properties are supported for LineString geometries									
17	(6) Only color and fill KML properties are supported for Polygon geometries									
18	(7) Colors can be specified by filling a cell using Excel, by specifying a RGB color as a string prefixed by the number sign (e.g. '#7c006f) or by specifying a value									
19	(8) Readable name for Xpath can be specified in the BalloonDisplayName worksheet									
20	(9) Values specified in the property specific mapping worksheets will override any values specified at the feature level.									
21	(10) The following columns are used in the Style worksheet									
22	(11) For point features, the origin of the coordinate system is in the lower left corner of the icon.									
23										
24	DIGGS Element Name	Element name in the DIGGS schema (e.g. 'Borehole')								
25	Point - color	Color to be used for Point features - See Rule #7 above for specifying the color								
26	Point - alpha	An hexadecimal opacity value to apply to the Point color (00 is fully transparent and FF is fully opaque)								
27	Point - scale	A double value used to resize the icon. Default KML value is 1.0								
28	Point - href	An HTTP address or a local file specification used to load an icon								
29	Point - hotSpot x	Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the x position								
30	Point - hotSpot y	Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the y position								
31	Point - hotSpot xunits	Units in which the x value is specified. A value of fraction indicates the x value is a fraction of the icon width								
32	Point - hotSpot yunits	Units in which the y value is specified. A value of fraction indicates the y value is a fraction of the icon height								
33	LineString - color	Color to be used for LineString features - See Rule #7 above for specifying the color								
34	LineString - alpha	An hexadecimal opacity value to apply to the LineString color (00 is fully transparent and FF is fully opaque)								
35	LineString - width	Width of the line in pixels. Default KML value is 1.0								
36	Polygon - color	Color to be used for Polygon features - See Rule #7 above for specifying the color								
37	Polygon - alpha	An hexadecimal opacity value to apply to the Polygon color (00 is fully transparent and FF is fully opaque)								
38	Polygon - fill	Boolean value (0 or 1) to specify whether to fill the polygon. Default KML value is 1, the polygon is filled								

7.5.6.4 KML Styling Rules

The following rules apply to all the worksheets in the *DIGGS_KML_STYLING.xml* file:

Rule	Description
1	Styling rules for features are defined in the Styling worksheet.
2	Styling rules based on a feature property are defined in the ' Property Based Styling - property mapping worksheet ' column of the Styling worksheet.
3	Each row in the worksheet represents a unique DIGGS element.
4	Only color, scale, href, and the hotSpot KML properties are supported for Point geometries.
5	Only color and width KML properties are supported for LineString geometries.
6	Only color and fill KML properties are supported for Polygon geometries.
7	Colors can be specified in one of the following ways: <ul style="list-style-type: none"> • by filling a cell using Excel; • by specifying a RGB color as a string prefixed by the number sign (e.g. '#7c006f'); • by specifying a valid BGR color as required by KML (e.g. '6foo7c'), default KML color will be used for empty cell (e.g. white).
8	Readable name for XPath can be specified in the BalloonDisplayName worksheet.
9	Values specified in a property specific mapping worksheet will override any values specified at the feature level.
10	The following columns are used in the Style worksheet (<i>see the tables in 7.5.6.5</i>).
11	For point features, the origin of the coordinate system is in the lower left corner of the icon.

7.5.6.5 *Supported Namespaces*

[Namespace Prefix]	[Namespace URI]
gml	http://www.opengis.net/gml/3.2
g3.3	http://www.forward_compatible_definitions.net/gml/3.3
xlink	http://www.w3.org/1999/xlink
xsi	http://www.w3.org/2001/XMLSchema-instance
witsml	http://www.witsml.org/schemas/131
diggs	http://schemas.diggsml.com/1.2a
diggs_geo	http://schemas.diggsml.com/1.2a/geotechnical
diggs_env	http://schemas.diggsml.com/1.2a/environmental
diggs_mon	http://schemas.diggsml.com/1.2a/monitoring
diggs_pil	http://schemas.diggsml.com/1.2a/piling

7.5.6.6 *KML Properties for Styling*

7.5.6.6.1 Point

Term	Description
Point - color	Color to be used for Point features. See Rule #7 for specifying the color.
Point - alpha	A hexadecimal opacity value to apply to the Point color (00 is fully transparent and FF is fully opaque). Default alpha color is FF.
Point - scale	A double value used to resize the icon. Default KML value is 1.0.
Point - href	An HTTP address or a local file specification used to load an icon.
Point - hotSpot x	Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the x component of a point on the icon. Default value is 1.0.
Point - hotSpot y	Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the y component of a point on the icon. Default value is 1.0.
Point - hotSpot xunits	Units in which the x value is specified. A value of fraction indicates the x value is a fraction of the icon. A value of pixels indicates the x value in pixels. A value of insetPixels indicates the indent from the right edge of the icon. Default value is fraction.
Point - hotSpot yunits	Units in which the y value is specified. A value of fraction indicates the y value is a fraction of the icon. A value of pixels indicates the y value in pixels. A value of insetPixels indicates the indent from the top edge of the icon. Default value is fraction.

7.5.6.6.2 LineString

Term	Description
LineString - color	Color to be used for LineString features. See Rule #7 for specifying the color.
LineString - alpha	A hexadecimal opacity value to apply to the LineString color (00 is fully transparent and FF is fully opaque). Default alpha color is FF.
LineString - width	Width of the line in pixels. Default KML value is 1.0.

7.5.6.6.3 Polygon

Term	Description
Polygon - color	Color to be used for Polygon features. See Rule #7 for specifying the color.
Polygon - alpha	A hexadecimal opacity value to apply to the Polygon color (00 is fully transparent and FF is fully opaque). Default alpha color is FF.
Polygon - fill	Boolean value (0 or 1) to specify whether to fill the polygon. Default KML value is 1, specifying that the polygon will be filled.

7.5.6.6.4 Label

Term	Description
Label - color	Color to be used for labels. See Rule #7 for specifying the color
Label - alpha	A hexadecimal opacity value to apply to the label color (00 is fully transparent and FF is fully opaque). Default alpha color is FF
Label - scale	A double value used to resize the label. Default KML value is 1.0

7.5.6.6.5 Balloon

Term	Description
Balloon	Comma separated list of properties described as XPath to display in the pop up balloon (by default gml:name and gml:description will always be included). The namespace prefixes to use are declared below. XPath are relative to the Feature root (e.g. if a Borehole has a child element name 'property1' defined in the namespace 'ns1', than the 'Balloon' column can contain the following XPath 'ns1:property1').
Balloon - xpath	XPath of a property used in the pop up balloon.
Balloon - display name	A meaningful name to display in the pop up balloon for the XPath of a property.
Balloon - resolved reference xpath property	XPath of a property to use after resolving a feature referenced by an xlink:href. Values should only be specified for 'Balloon - xpath' ending with xlink:href and reference a unique property.

7.5.6.6.6 Property Based Styling

Term	Description
Property Based Styling - property	The name of the property to use for specific styling.
Property Based Styling - property mapping worksheet	The name of a worksheet in this workbook to use to resolve the specific property value.

7.5.6.7 *Defining the Custom KML Styling*

A number of DIGGS elements are included in the initial spreadsheet, but only a few of them have any specific styling defined. Styling can be defined for any of the other DIGGS Elements listed in the Styling worksheet, or new elements can be styled by adding their Element Name to the list.

Any elements that do not have specific styling rules in the spreadsheet will be processed during the conversion without custom styling.

Unstyled elements will use default styling, and there will be no content in the balloon.

If any property value is not explicitly defined for an element, the default value will be used. If a property of an element has been defined, the property value will override the default value.

7.5.6.7.1 Styling

The Styling worksheet contains default styling properties for the element named in the first column. Each column defines one style value. Columns are grouped together according to what property is being styled.

7.5.6.7.1.1 Geometry Styling

Styling for Point, LineString, and Polygon geometries are in the first columns of the worksheet, followed by styling for Label elements.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Point									LineString			Polygon			Label		
2	DIGGS Element Name	color	alpha	scale	href	hotSpot x	hotSpot y	hotSpot xunits	hotSpot yunits	color	alpha	width	color	alpha	fill	color	alpha	scale
3	Project																	
4	Borehole					2.0												
5	TrialPit																	
6	Sample						3.0											
7	Station																	
8	TrenchWall																	
9	ColorLayer																	
10	LithologyLayer																	
11	StratigraphyLayer																	
12																		
13																		
14																		
15																		
16																		
17																		
18																		

To style the geometry of an element:

1. Identify what type of geometry the feature has and fill in the desired values for the geometry properties.
2. If a feature has multiple geometries (i.e. point and polygon), any or all of the geometries may be styled.

To style the label (KML Placemark) for an element:

1. Enter the desired values for the Label properties.
2. Use the same rules for specifying the color and alpha properties of a Label as for the color and alpha properties of a geometry.
3. To specify the size of the label – enter a value > 1 for the scale property.

Some notes on specifying geometry values:

- To specify the colour for the fill property of a Polygon – follow rule 7 for how to enter the colour value.
- To specify the alpha value for transparency – use a hexadecimal value between 00 and FF, for example: 00 is fully transparent, FF is fully opaque, and 7F is 50%.
- To specify whether or not to display the fill colour of a polygon – use a Boolean value of 0 or 1. The fill value must be 1 if the polygon is to be displayed in Google Earth.

7.5.6.7.1.2 **Property Based Styling**

The Styling worksheet allows global styling to be defined for an element. Specific properties of an element can also be given specific styling.

If specific properties of an element are to be custom styled, the Property Based Styling worksheet allows those specific properties to be mapped to a customized worksheet that will contain the list of properties and their styling values.

	A	S	T
	DIGGS Element Name	Property Based Styling	property mapping worksheet
1			
2	DIGGS Element Name	property	property mapping worksheet
3	Project		dig
4	Borehole		gm
5	TrialPit		dig
6	Sample	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue	PrimaryCodeValue
7	Station		dig
8	TrenchWall	diggs:colors/diggs:Color/diggs:colorCode	ColorCode
9	ColorLayer	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue	PrimaryCodeValue
10	LithologyLayer	diggs:stratUnit	StratUnit
11	StratigraphyLayer		dig
12			dig
13			gm
14			dig
15			dig
16			dig
17			dig
18			dig

To define styling for properties of an element:

1. specific styling for element properties can be defined by filling in the Property Based Styling columns
2. select the property to use for styling
3. identify the worksheet
4. identify the name of the worksheet where the property styling is specified

5. create new worksheet by copying and renaming an existing one
6. property – path to property in the schema
7. properties can be at any level using an XPath expression to specify the property to use

7.5.6.7.1.3 *Balloon Styling*

The Balloon column contains a comma separated list of XPath expressions identifying each property to display in the balloon associated with the element.

For each XPath property to be displayed in the balloon, use the BalloonDisplayName worksheet (see section 7.5.6.7.5) to specify the display name to be used in the balloon.

A	U Balloon	V	W
1			
2 DIGGS Element Name			
3 Project	diggs:associatedLocationRef/@xlink:href		
4 Borehole	gml:name,gml:identifier,gml:identifier/@codeSpace,diggs:projectRef/@xlink:href,diggs:layerSystemRef/@xlink:href		
5 TrialPit			
6 Sample	diggs:sampleType,diggs:lithology/diggs:Lithology/diggs:primaryCodeValue,diggs:lithology/diggs:Lithology/diggs:primaryCodeValue/@codeSpace		
7 Station			
8 TrenchWall	gml:name,gml:identifier,gml:identifier/@codeSpace,diggs:projectRef/@xlink:href,diggs:layerSystemRef/@xlink:href		
9 ColorLayer	diggs:colors/diggs:Color/diggs:colorCode,diggs:colors/Color/diggs:colorCode/@codeSpace		
10 LithologyLayer	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue,diggs:lithology/diggs:Lithology/diggs:primaryCodeValue/@codeSpace		
11 StratigraphyLayer			
12			
13			
14			
15			
16			
17			
18			

7.5.6.7.2 PrimaryCodeValue

This is an element-specific worksheet that defines customized styling for properties of the DIGGS LithologyLayer element:

1	Property	Point								LineString			Polygon			Label				
		2	Value	color	alpha	scale	href	hotSpot x	hotSpot y	hotSpot xunits	hotSpot yunits	color	alpha	width	color	alpha	fill	color	alpha	scale
3	SP				7F									7F	6.0		7F			
4	SM				7F									7F	6.0		7F			
5	CL		#7c006f		7F									6f007c	7F					
6	ML			6f007c														7F	7F	
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				

7.5.6.7.3 ColorCode

This is an element-specific worksheet that defines customized styling for properties of the DIGGS ColorLayer element:

1	Property	Point								LineString			Polygon			Label				
		2	Value	color	alpha	scale	href	hotSpot x	hotSpot y	hotSpot xunits	hotSpot yunits	color	alpha	width	color	alpha	fill	color	alpha	scale
3	10YR4/3			7F										7F		7F		7F		
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				

7.5.6.7.4 StratUnit

This is an element-specific worksheet that defines customized styling for properties of the DIGGS StratigraphyLayer element:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Property	Point								LineString		Polygon		Label				
2	Value	color	alpha	scale	href	hotSpot x	hotSpot y	hotSpot xunits	hotSpot yunits	color	alpha	width	color	alpha	fill	color	alpha	scale
3	Holocene	7F								7F			3F			7F		
4	Pleistocene																	
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		

7.5.6.7.5 BalloonDisplayName

The information displayed in a balloon is defined by an XPath expression. The XPath expression is mapped to the name to display in the Balloon.

	A	B	C	D	
Balloon					
1	xpath	display name	resolved reference xpath property		
2	diggs:associatedLocationRef/@xlink:href	Associated Location Reference	gml:name		
3	diggs:layerSystemRef/@xlink:href	Layer System Reference	gml:name		
4	diggs:projectRef/@xlink:href	Project Reference	gml:name		
5	diggs:samplingActivityRef/@xlink:href	Sampling Activity Reference			
6	gml:identifier	Identifier			
7	gml:identifier/@codeSpace	Identifier Codespace			
8	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue	Lithology Code Value			
9	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue/@codeSpace	Lithology Code Value Codespace			
10	diggs:colors/diggs:Color/diggs:colorCode	Color Layer code			
11	diggs:colors/diggs:Color/diggs:colorCode/@codeSpace	Color Layer Codespace			
12					
13					
14					
15					
16					
17					
18					

The value of the XPath can reference the `id` of another element (that is, the XPath expression ends in syntax `@xlink:href`). If this is the case, the balloon can be configured to either display the `id` of the feature or resolve the `id` to display the element's name. To resolve the `id` and display the value of a property of the element, use the last column to identify the property whose value should be displayed instead.

8 Future Enhancements

The emphasis on future enhancements should be placed on software tool support for DIGGS, because it is software support that makes the wide adoption of any open standard possible. The complexity of a rich information model such as the DIGGS model is meant to be hidden from end users by software tools that take care of the 'heavy lifting'. Software developers should embrace the richness/complexity of the model (e.g. linear referencing, coverage encodings, observations, etc) and the guidance provided in this document to create tools that exploit the rich features of the model so that end users can benefit from the enhanced functionality (without requiring deep technical knowledge of the encodings, which are meant to be processed by machines). Initial suggestions for the types of software tool support include the following:

- DIGGS Validator – to enforce the validity of DIGGS data (beyond schema validation). Likely to involve formal assertions (e.g. Schematron) to define and enforce general DIGGS business rules.
- DIGGS Web authoring tool – to aid in the creation of DIGGS data and metadata.
- DIGGS Data and Map Server – to query, serve and visualize DIGGS data over the web.
- DIGGS Processing/Analysis applications – for technical and business analysis of geotechnical and geoenvironmental data.
- DIGGS Identifier Registry – for discovery and management of globally unique identifiers of DIGGS resources including of feature data, dictionaries (CRS and units), code lists, symbols/styles, services and feature catalogues.
- DIGGS CRS and Units Registry – for discovery, management and machine readable access to CRS components, transformations and units.
- DIGGS Data/Metadata Registry – for discovery of DIGGS services (e.g. data, analysis and map services) and data resources (datasets, symbology, dictionaries, codelists) and life cycle management of the services and data resources.

9 Bibliography

- [1] Burggraf, David (Galdos Systems Inc.) March 2009. DIGGS Technical Report *DIGGS 1.0a Schema Evaluation*, pp. 137.
- [2] Burggraf, David, Dan Ponti, Chris Bray, Loren Turner. April 2010. DIGGSML 1.1 Schema Release Notes. *Release of DIGGS V1.1*, pp. 14.
- [3] Cox, Simon, et. al. (2004). Geography Markup Language (GML) version 3.1, OGC Recommendation Paper (Doc. No. OGC 03-105r1), 595p.
- [4] Daigle, L., et. al. (2002). IETF RFC. Uniform Resource Names (URN) *Namespace Definition Mechanisms*. <http://tools.ietf.org/html/rfc3406>. Link verified 2012-04-24.
- [5] Hoit, Marc. DIGGS 1.0a Release Notes. *Data Interchange for Geotechnical and GeoEnvironmental Specialists (DIGGS)*, pp. 2.
- [6] Mitten, Paul. (Compusult Ltd.) August 2009. DIGGS Technical Report *DIGGS 1.0a Schema Evaluation*, pp. 62.
- [7] Portele, Clemens, et. al. (2007). Geography Markup Language (GML 3.2). Joint OGC/ISO Implementation Encoding Standard (OGC Doc. No. 07-036, ISO TC 211 19136), 437p.
- [8] Portele, Clemens, et. al. (2012). Geography Markup Language (GML 3.3). OGC Implementation Encoding Standard (OGC Doc. No. 10-129r1), 91p.
- [9] Power, Chris and Roger Chandler. July 2008. DIGGSML Documentation. *Introduction to DIGGS*, pp. 68.
- [10] Power, Chris. July 2008. DIGGS Engineering Advisory Document. *DIGGSML V1.0a Data Dictionary*, pp. 239.
- [11] Turner, Loren L. March 2009. Proceedings of DIGGS Invitational Meeting, Orlando Florida, March 25-26, 2009. *Report on Project Status and Development of a New Roadmap*, pp. 298.
- [12] Turner, Loren L. August 2009. DIGGSML Blog Announcement. *DIGGS Project Team contracts with Galdos Systems and Compusult Ltd. to carry out review of DIGGS version 1.0a*, <http://www.diggsml.com/diggs-project-team-contracts-galdos-systems-and-compusult-ltd-carry-out-review-diggs-version-1-0a>. Link verified 2012-03-05.
- [13] Turner, Loren L. October 2009. DIGGS Summary Report. *Synthesis of Findings and Recommendations from Evaluation of DIGGS 1.0a Schema*, pp. 8.

Appendix A. DIGGS URN Registration RFC to IANA

The following document is an RFC submission for registration of the DIGGS URN Identification Scheme to Internet Assigned Numbers Authority (IANA, <http://www.iana.org/>) following the standard URN definition template (Appendix A, [4])

Introduction

DIGGS is a coalition of government agencies, universities and industry partners whose focus is on the creation and maintenance of an international data transfer standard for transportation related data. The coalition came into existence through coordination from the US Federal Highway Administration sponsoring meetings and eventually forming the pooled fund study project. The initial base schema consists of geotechnical data including Borehole, soil testing, site information and more. The first SIG is extending the schema to include Geo-Environmental testing.

Namespace ID

Assigned by IANA. The NID string is requested to be “diggs”

Registration Information

Registration Version Number: 1.0

Registration Date: 2010-08-09

Declared registrant of the namespace

Registering organization Name: Data Interchange for Geotechnical and Geo-environmental Specialists (DIGGS)

Address: < Update this to reflect ASCE as the new DIGGSML custodian>

Designated contact person

Name: Loren Turner

Coordinates: loren_turner@dot.ca.gov

Declaration of syntactic structure

The Namespace Specific String (NSS) of all URNs that use the "diggs" NID will have the following structure:

urn:diggs:{DIGGS_resource}:{ResourceSpecificString}

where the "DIGGS_resource" is a US-ASCII string that conforms to the URN syntax requirements [RFC2141] and defines a specific class of resource type. Each resource type has a specific labeling scheme that is covered by "ResourceSpecificString", which also conforms to the naming requirements of [RFC2141].

DIGGS maintains a naming authority, the DIGGS Naming Authority (DIGGSNA), which will manage the assignment of the "DIGGS_resource" and "ResourceSpecificString" fields for each resource class.

Relevant ancillary documentation

The DIGGS Naming Authority (DIGGSNA) provides information on the registered resources and the registrations for each. More information about DIGGSNA and the registration activities and procedures to be followed are available at:

<http://www.diggsml.com/>

Identifier uniqueness considerations

The DIGGSNA will manage resources using the "diggs" NID and will be the authority for managing the resources and subsequent strings associated. In the associated procedures, DIGGSNA will ensure the uniqueness of the strings themselves or shall permit secondary

responsibility for management of well-defined sub-trees. DIGGS may permit use of experimental type values that will not be registered. As a consequence, multiple users may end up using the same value for separate uses. As experimental usage is only intended for testing purposes, this should not be an issue.

Identifier persistence considerations

DIGGSNA will provide clear documentation of the registered uses of the "diggs" NID. This will be structured such that each "DIGGS_resource" will have a separate description and registration table. The registration tables and information will be published and maintained by the DIGGSNA on the DIGGS web site.

Process of identifier assignment

DIGGSNA will use the DIGGS standards policies and procedures for discussion, approval and registration of each type of resource that it maintains. Each such resource may have three types of registration activities:

- 1) Registered values associated with DIGGS specs or services
- 2) Registration of values or sub-trees to other entities
- 3) Name models for use in experimental purposes

Process for identifier resolution

Not applicable – the namespace is not listed with a Resolution Discovery System.

Rules for Lexical Equivalence

No special considerations; the rules for lexical equivalence of [RFC2141] apply.

Conformance with URN Syntax

No special considerations.

Validation mechanism

None specified. URN assignment will be handled by procedures implemented in support of DIGGSNA activities.

URN Scope

Global.

Example

The following example is representative of a URN that could be assigned

urn:diggs:def:fi:DIGGS:SCT23423

The URN above identifies the definition of a feature instance with code SCT23423 that is maintained by the DIGGS authority.

Namespace Considerations

There is currently no available namespace that will allow the DIGGS to uniquely specify and access resources, such as codelists and feature instances that are required by organizations implementing DIGGS standards. There is also a need for other organizations, such as the Association of Geotechnical and Geoenvironmental Specialists (AGS) to be able to access and reference DIGGS specific resources.

Community Considerations

The current DIGGSML standard requires access to resources, such as schemas, codelists, and coordinate reference system dictionaries. In order for the larger geotechnical and geo-environmental communities to be able to effectively implement applications that access DIGGS resources, a unique namespace is required. These resources are intended to be freely and openly available as a set of community resources.

Security Considerations

There are no additional security considerations other than those normally associated with the use and resolution of URNs in general.

Informative References

[1] Moats, R., "URN Syntax", RFC 2141, May 1997. <<http://www.ietf.org/rfc/rfc2141.txt>>

[2] Daigle, L. et al., "Uniform Resource Names (URN) Namespace Definition Mechanisms", RFC 3406, October 2002. <<http://www.ietf.org/rfc/rfc3406.txt>>

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY

OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be

found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at

ietf-ipr@ietf.org.

Appendix B. GML 3.2 Practices Adopted by DIGGS

B.1 Coverage Encodings

The data encoding pattern of the table entries (e.g. test results) is typically captured in GML using a coverage encoding. The coverage model and corresponding best practices are provided in this section.

A GML coverage is essentially a distribution of data readings/measurements, defined over a geographic domain, which usually consists of a set of geometry elements. For example, the geometry elements may be a set of polygons in a surface tessellation, a set of curves segments along curve, a discrete set of points along a curve, or a rectified grid. A GML coverage can be thought of as a function that maps a spatial or temporal domain to data values in the range. The range of such a coverage function could take on any values, for example elevation, temperature, pressure, rock type, or reflectance as shown in Figure 1.

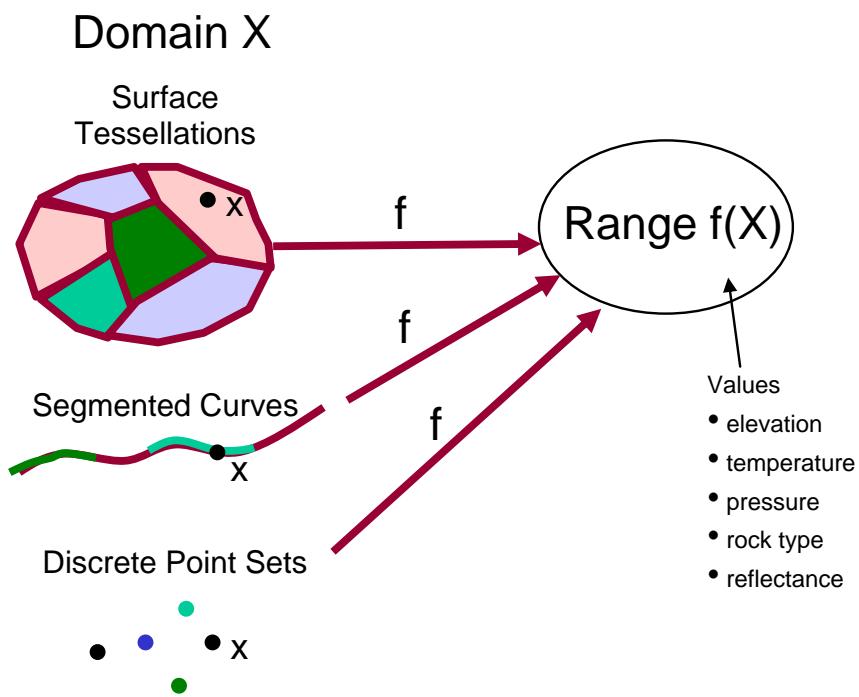


Figure 1 Coverages are distribution functions defined on some domain.

There are three components that define a GML coverage – the domain, range, and coverage function. These components contained by the three properties: `domainSet`, `rangeSet` and

coverageFunction, respectively. The GML spatial coverage model is represented by the Entity Relationship (ER) diagram in Figure 2.

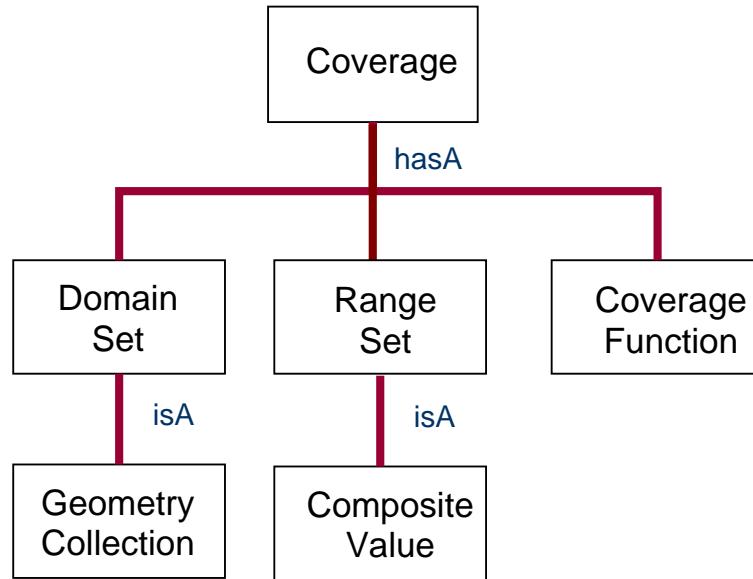


Figure 2. Entity Relationship (ER) View of a Coverage

Domain Set

In a GML instance, the value of the domainSet property will often be a GML geometry aggregate, such as MultiPoint, MultiCurve, MultiPolygon, or an even spaced Grid such as RectifiedGrid.

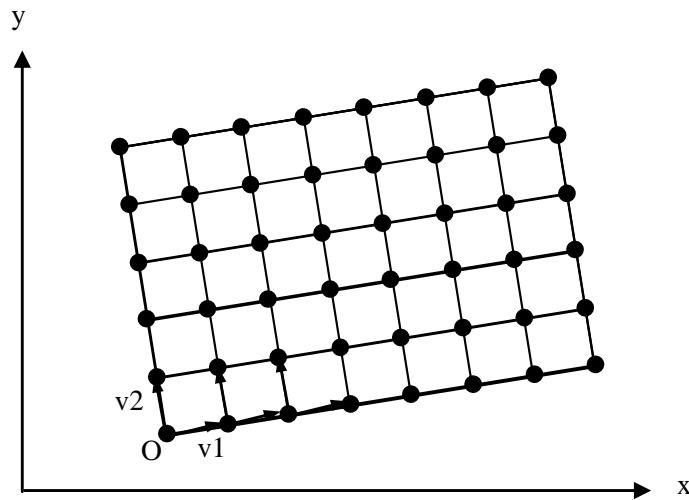


Figure 3. A Two-Dimensional Rectified Grid with Offset Vectors v_1 and v_2

In Figure 3, the origin is denoted as $\mathbf{0}$, and the offset vectors are v_1 and v_2 . The origin and offset vectors must all be of the same dimension – usually 1D, 2D or 3D. The GML RectifiedGrid has the properties: limits, axisName, origin and offsetVector. The value of limits is a GridEnvelope, which has two properties called low and high. The value of low is an integer list, for example [n1,n2], that represents the “lower left” corner $\mathbf{0}+n_1\mathbf{v}_1+n_2\mathbf{v}_2$ of the grid. Similarly, high represents the “upper right” corner of the grid.

Range Set

The range set in GML can be encoded as an aggregate value, a data block or a binary file. An aggregate value encoding is the most verbose; the binary encoding is the most efficient, and the data block encoding lies somewhere in between. In DIGGS it was agreed that only the DataBlock encoding would be appropriate. This section provides examples of the data block encoding with temperature and pressure values. Note that the temperature and pressure values in these examples are defined in a GML application schema.

Data Block

A gml:DataBlock consists of two properties, rangeParameters and tupleList, whose values describe the range of a coverage. The value of rangeParameters is of gml:_Value type from valueObjects.xsd, which describes the quantities in the tupleList, including the units of measure used.

The following example shows how temperature and pressure values can be encoded in a DataBlock:

```

<gml:rangeSet>
  <gml:DataBlock>
    <gml:rangeParameters>
      <gml:CompositeValue>
        <gml:valueComponents>
          <app:Temp uom="urn:ogc:def:uom:SI:1999:degreesC">template</app:Temp>
          <app:Pressure uom="urn:ogc:def:uom:SI:1999:kPa">template</app:Pressure>
        </gml:valueComponents>
      </gml:CompositeValue>
    </gml:rangeParameters>
    <gml:tupleList>0,101.1 24,101.2 17,101.3 19,101.4 ...</gml:tupleList>
  </gml:DataBlock>

```

```
</gml:rangeSet>
```

Note that the aggregate value object `CompositeValue` is the target of `rangeParameters` in the instance above, which has a `valueComponents` property. `Temp` and `Pressure` are value objects derived from `gml:MeasureType` that are encapsulated by the `valueComponents` property tags. The `tupleList` property contains a list of coordinate tuples, where the entries of the coordinate tuples provide the quantities of the range parameters.

Coverage Function

The value of `gml:coverageFunction` is a choice between `gml:MappingRule` or `gml:GridFunction`. The `MappingRule` is of type `gml:StringOrRefType`, which is a string or a URI reference to a mapping rule defined elsewhere. The mapping rule by default is linear if not specified, where linear mapping assigns the first geometry element (in document order) in the domain to the first value in the range, and so on.

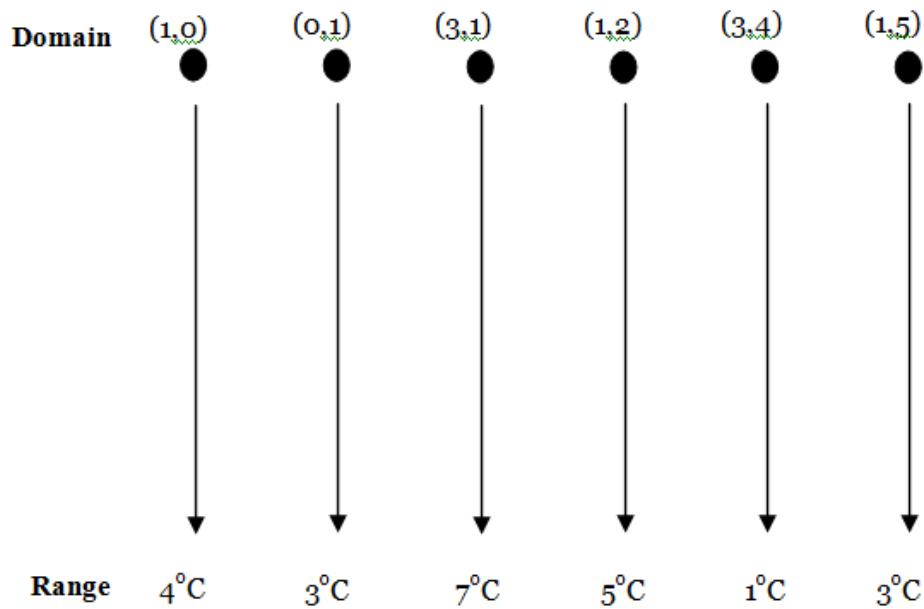


Figure 4. Linear Mapping Rule for a MultiPoint Temperature Coverage

Grid Functions

The `GridFunction` is used instead of `MappingRule` for grid coverages and has two properties, `sequenceRule` and `StartPoint`. The value of `startPoint` is an integer list that represents the first grid post in the grid to be traversed, the default start value is the value of `low` in `GridEnvelope`. The value

of sequenceRule is one of the strings in the enumerated list sequence rules: Linear, Boustrophedonic, Cantor-diagonal, Spiral, Morton, or Hilbert. The sequenceRule property also has an optional order attribute whose value specifies one of the increment orders in the enumerated list: "+x+y", "+y+x", "x-y", "-x-y", in the case where the grid is 2-dimensional. The increment order of "+x+y" means that the grid points are traversed from lower to higher on the x-axis and from lower to higher on the y-axis. For example, Figure 5 and Figure 6 both illustrate the linear sequence rule but with different increment orders.

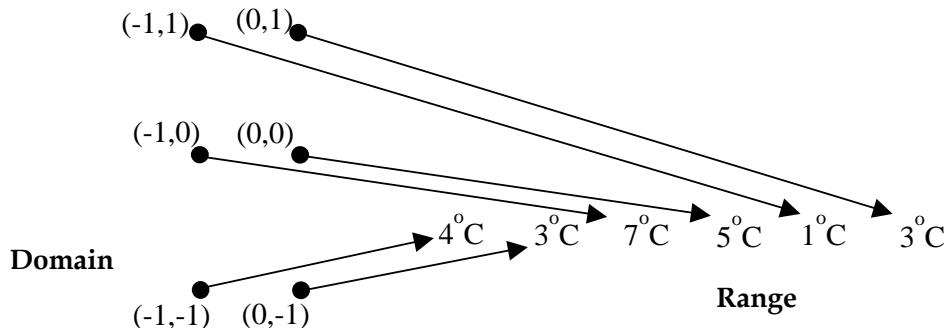


Figure 5. Linear Grid Function for a Grid Coverage with “+x+y” Increment Order

Note that in Figure 5, the starting point (-1,-1) is the same as the value of low in the corresponding GridEnvelope. The increment order "+x-y" means the grid points are traversed from lower to higher on the x-axis and from higher to lower on the y-axis as illustrated in Figure 6.

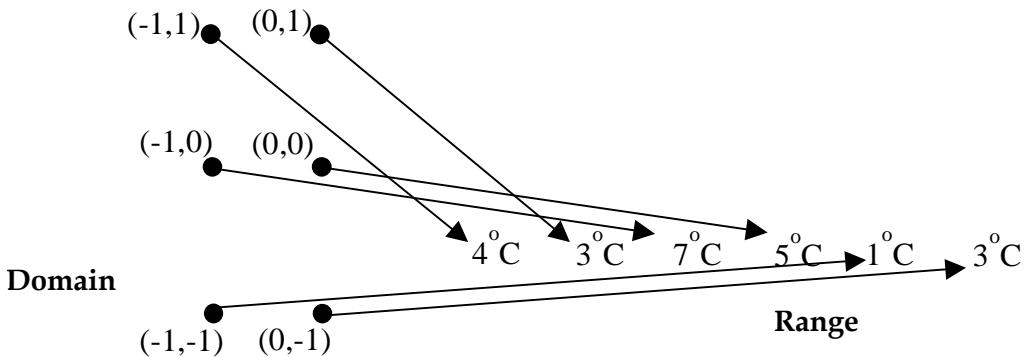


Figure 6. Linear Grid Function for a Grid Coverage with “+x-y” Increment Order

As shown in Figure 6, the starting point (-1,1) is not the same as the value of low in the corresponding GridEnvelope. In this case the startPoint property is required as shown in the following instance:

```

<gml:domainSet>
  <gml:Grid dimension="2">
    <gml:limits>
      <gml:GridEnvelope>
        <gml:low>-1 -1</gml:low>
        <gml:high>0 1</gml:high>
      </gml:GridEnvelope>
    </gml:limits>
    <gml:axisName>x</gml:axisName>
    <gml:axisName>y</gml:axisName>
  </gml:Grid>
</gml:domainSet>
<rangeSet>...</rangeSet>
<coverageFunction>
  <GridFunction>
    <sequenceRule order="+x-y">Linear</sequenceRule>
    <startPoint>0 1</startPoint>
  </GridFunction>
</coverageFunction>

```

Note that if the `startPoint` was not specified, the default starting point would have been the value of `low`, (-1,-1).

Encoding a Rectified Grid

The following example instance shows the entire temperature and pressure coverage, `TempPressure`, with range data encoded as a `DataBlock`:

```

<app:TempPressure>
  <gml:rectifiedGridDomain>
    <gml:RectifiedGrid dimension="2">
      <gml:limits>
        <gml:GridEnvelope>
          <gml:low>-1 -1</gml:low>
          <gml:high>2 2</gml:high>
        </gml:GridEnvelope>
      </gml:limits>
      <gml:axisName>u</gml:axisName>
      <gml:axisName>v</gml:axisName>
      <gml:origin>
        <gml:Point gml:id="O" srsName="urn:ogc:def:crs:EPSG::1234">
          <gml:coordinates>25,27</gml:coordinates>
        </gml:Point>
      </gml:origin>
      <gml:offsetVector>1, 0.2</gml:offsetVector>
      <gml:offsetVector>-0.2, 1</gml:offsetVector>
    </gml:RectifiedGrid>
  </gml:rectifiedGridDomain>

```

```
</gml:RectifiedGrid>
</gml:rectifiedGridDomain>
<gml:rangeSet>
  <gml>DataBlock>
    <gml:rangeParameters>
      <gml:CompositeValue>
        <gml:valueComponents>
          <app:Temp uom="urn:ogc:def:uom:SI:1999:degreesC">template</app:Temp>
          <app:Pressure uom="urn:ogc:def:uom:SI:1999:kPa">template</app:Pressure>
        </gml:valueComponents>
      </gml:CompositeValue>
    </gml:rangeParameters>
    <gml:tupleList>3,101.2 5,101.3 7,101.4 11,101.5 13,101.6 17,101.7 19,101.7 23,101.8 29,101.9
      31,102.0 37,102.1 41,102.2 43,102.3 47,102.0 53,102.5 59,102.6</gml:tupleList>
  </gml>DataBlock>
</gml:rangeSet>
</app:TempPressure>
```

Appendix C. GML 3.3 Extensions Adopted by DIGGS

C.1 Simple MultiPoint Encoding

`gmlce:SimpleMultiPoint` implements, and provides a simplified encoding for, ISO 19107 GM_MultiPoint (see ISO 19107:2003, 6.5.4). A `gmlce:SimpleMultiPoint` consists of a list of coordinates (`DirectPositions`).

```

<complexType name="SimpleMultiPointType">
  <complexContent>
    <extension base="gml:AbstractGeometricAggregateType">
      <sequence>
        <element ref="gml:posList"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="SimpleMultiPoint" type="gmlce:SimpleMultiPointType"
substitutionGroup="gml:AbstractGeometricAggregate" />

<complexType name="MultiPointPropertyType">
  <choice minOccurs="0">
    <element ref="gml:MultiPoint"/>
    <element ref="gmlce:SimpleMultiPoint"/>
  </choice>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

C.2 Linear Referencing

C.2.1 Linear Spatial Reference System

C.2.1.1 Introduction

A Linear Spatial Reference System enables the use of direct position (e.g. captured by `gml:pos` or `gml:posList`) to define a linearly referenced location. Such a direct position in the spatial context of a Linear Spatial Reference System implements the ISO 19148 LR_PositionExpression, with restrictions. The linear element and linear referencing method components of the position expression are specified by a Linear SRS. Because `gml:pos` is only appropriate for geometries, the

linear element shall be limited to curves. Because gml:pos only allows values of type double, relative Linear Referencing Methods would be precluded since the distance along would otherwise require inclusion of a gmllr:referent. The gml:pos specifies the distance expression and is limited to a single coordinate specifying the distanceAlong value.

C.2.1.2 LinearSRS

```

<complexType name="LinearSRSType">
  <complexContent>
    <extension base="gml:IdentifiedObjectType">
      <sequence>
        <element ref="gmllr:linearElement"/>
        <element name="lrm" type="gmllr:LinearReferencingMethodPropertyType"/>
        <element name="defaultLength" type="gml:LengthType" minOccurs="0"/>
        <element name="startValue" type="gmllr:StartValueType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="linearElement" type="gml:CurvePropertyType"/>

<element name="LinearSRS" type="gmllr:LinearSRSType" substitutionGroup="gml:Definition"/>
```

The gmllr:LinearSRS element is an identified object that specifies a Linear Spatial Reference System as a combination of a linear element and a Linear Referencing Method, the first two components of a gmllr:PositionExpression.

The value of the element gmllr:linearElement is a curve in the substitution group gml:AbstractCurve.

The element gmllr:lrm specifies the linear referencing method of measurement.

The gmllr:LinearSRS implements the measure and startValue operations of the LR_ILinearElement interface in ISO 19148 (clause 6.2.8) as the optional property elements defaultLength and startValue, since the linearElement value is a curve possibly referenced from an existing dataset. The operations: defaultLRM, translateToInstance, and translateToType are not implemented as properties by gmllr:LinearSRS. The defaultLRM operation is not implemented because gmllr:LinearSRS has a mandatory lrm property and the ‘translate’ operations are ignored because gmllr:LinearSRS is instantiated as an XML/GML element,

The optional property element gmllr:totalLength specifies the overall length of the linear element (curve). The gmllr:totalLength value can be used for all calculations requiring a total linear element length, unless it is derived directly from the source curve geometry (e.g. at runtime) or retrieved from source metadata (e.g captured in. gml:metaDataProperty).

gmllr:startValue provides the value at the start of the linear element for the specified Linear Referencing Method. This is usually 0 (zero).

The gmllr:LinearSRS is identifiable by a gml:id and can be referenced as a Spatial Reference System (SRS), by GML geometry elements using the srsName attribute. Each direct position (control point) captured by the pos or posList properties of the GML geometry element taken together with the gmllr:LinearSRS, implements the ISO 19148 LR_PositionExpression. For example:

```
<gml:Point gml:id="p1" srsName="#LSRS123">
  <gml:pos>15.5</gml:pos>
</gml:Point>
```

defines a Point geometry as a distance along (15.5) the gml:LineString linear element specified in LSRS123, measured in accordance with the Linear Referencing Method defined by LRM001:

```
<gmllr:LinearSRS gml:id="LSRS123">
  <gmllr:linearElement>
    <gml:LineString srsName="..." srsDimension="3" gml:id="LS_BH18">
      <gml:posList>407829 268621 23.93 407415 268600 8.43</gml:posList>
    </gml:LineString>
  </gmllr:linearElement>
  <gmllr:lmr>
    <gmllr:LinearReferencingMethod gml:id="LRM001">
      <gmllr:name>chainage</gmllr:name>
      <!--chainage = measurement in metres -->
      <gmllr:type>absolute</gmllr:type>
      <!--absolute = measure from start of linear element -->
      <gmllr:units uom="m"/>
    </gmllr:LinearReferencingMethod>
  </gmllr:lmr>
</gmllr:LinearSRS>
```

Additional examples are provided in Section C.3.3 (Vector Offset Linear Spatial Reference System).

C.2.1.3 *LinearSRSPropertyType*

```
<complexType name="LinearSRSPropertyType">
  <sequence minOccurs="0">
    <element ref="gmllr:LinearSRS"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

C.3 *Linear Referencing Offset Vectors*

C.3.1 Target namespace

All schema components specified in this subclause are in the target namespace:

<http://www.opengis.net/gml/3.3/lrov>

C.3.2 Introduction

Linear Referencing with Offset Vectors includes the specification of linearly referenced locations which can have vector offsets.

C.3.3 Vector Offset Linear Spatial Reference System

C.3.3.1 *Introduction*

A Vector Offset Linear Spatial Reference System enables the use of `gml:pos` to define a linearly referenced location which may include offset vectors. The linear element and Linear Referencing Method components of the position expression are specified by a Vector Offset Linear SRS. Because `gml:pos` is only appropriate for geometries, the linear element shall be limited to curves. Because `gml:pos` only allows values of type double, relative Linear Referencing Methods would be precluded since the distance along would otherwise require inclusion of a `gml:referent`. The `gml:pos` element specifies the distance expression and its coordinates specify the mandatory `distanceAlong` and optional vector offset values.

C.3.3.2 *VectorOffsetLinearSRS*

```

<complexType name="VectorOffsetLinearSRSType">
  <complexContent>
    <extension base="gml:LinearSRSType">
      <sequence>
        <element name="offsetVector" type="gml:VectorType" maxOccurs="3"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="VectorOffsetLinearSRS" type="gml:VectorOffsetLinearSRSType"
  substitutionGroup="gml:LinearSRS"/>

<complexType name="VectorType">
  <complexContent>
    <extension base="gml:VectorType">
      <attribute name="offsetUom" type="gml:UomIdentifier"/>
    </extension>
  </complexContent>
</complexType>

```

The `gml:VectorOffsetLinearSRS` element is an identified object that specifies a Multidimensional Spatial Reference System by extending `gml:LinearSRSType` with one or more offset vectors to define a reference frame relative to the linear element (examples are provided below).

The element `gml:rov:offsetVector` specifies an offset vector direction (vector length/magnitude is ignored), which is the direction that the corresponding offset distance will be measured. The attribute `srsName`, inherited from `gml:VectorType`, specifies the offset vector Coordinate Reference System.

The `gml:rov:VectorType` extends `gml:VectorType` with an optional attribute `gml:rov:offsetUom`, which specifies the units of measure of the offset distance. It is of type `gml:UomIdentifier`. If the `gml:rov:offsetUom` attribute is not provided, the uom value defaults to the `gml:lr:units` value of the Linear Referencing Method, otherwise `gml:rov:offsetUom` possesses the overriding value.

The `gml:rov:VectorOffsetLinearSRS` can be referenced as a Spatial Reference System (SRS), by GML geometry elements using the `srsName` attribute. Each direct position (control point) captured by the `pos` or `posList` properties of the GML geometry element taken together with the offset vectors and `gml:rov:VectorOffsetLinearSRS`, implements the ISO 19148 LR_PositionExpression, with restrictions. Vector offsets are accommodated in `gml:pos` or `gml:posList` expressions with the following assumptions:

- 1) the Linear Referencing Method type cannot be “relative”
- 2) the linear element must be of type “curve”
- 3) the first double value in `gml:pos` (or `gml:posList` tuple) is the distance along the linear element
- 4) subsequent double values in `gml:pos` (or `gml:posList` tuple), if present, correspond to the component distance along the offset vectors in document order as specified in the `gml:rov:VectorOffsetLinearSRSType`.

EXAMPLE 1 Single Offset Vector

A single offset vector can be used to describe the positions P1 to P11 relative to a linear element in the direction of the offset vector \mathbf{v} as illustrated in **Error! Reference source not found..**

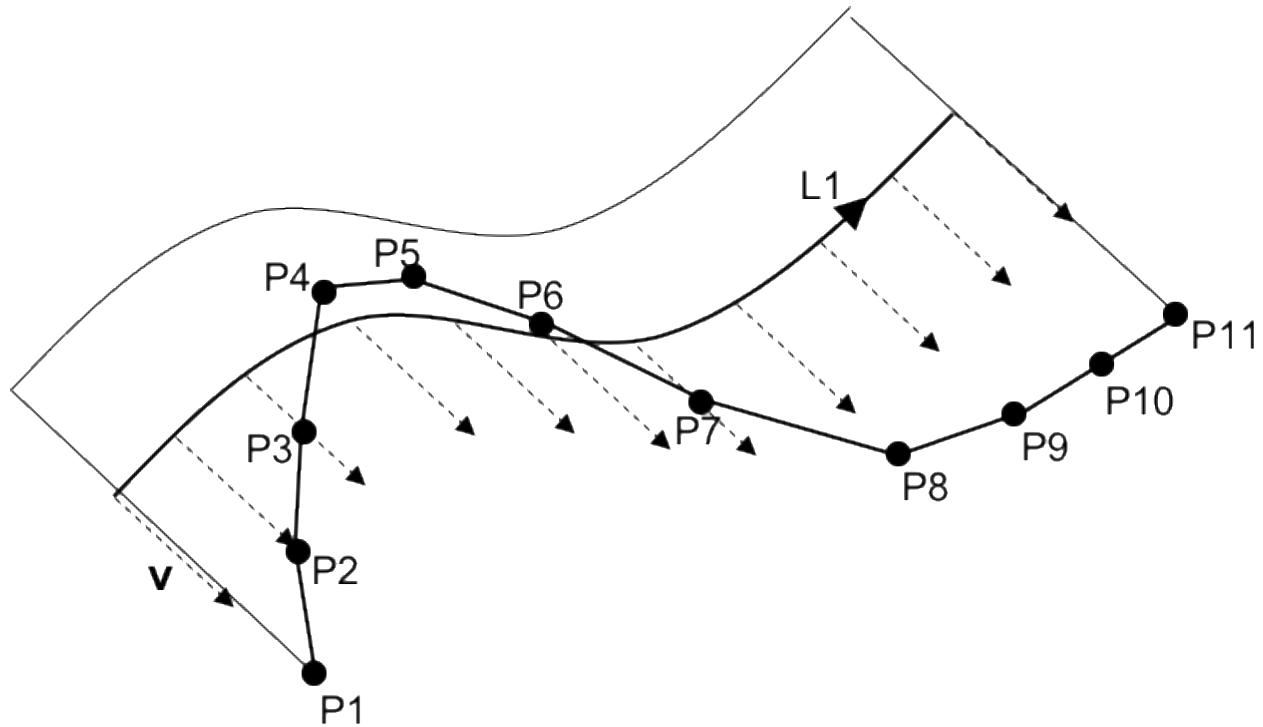


Figure 1 – Single offset vector

Sample gml:Point encodings of the positions P1 to P11 are shown as follows, where the first ordinate in each gml:pos element corresponds to distance along the linear element L_1 and the second ordinate in each gml:pos element corresponds to the distance (in units specified by offsetUom) in the direction of the offset vector v .

```

<gml:Point gml:id="P1" srsDimension="2" srsName="#volsrs001">
  <gml:pos>0 1.7</gml:pos>
</gml:Point>
<gml:Point gml:id="P2" srsDimension="2" srsName="#volsrs001">
  <gml:pos>1 1</gml:pos>
</gml:Point>
<gml:Point gml:id="P3" srsDimension="2" srsName="#volsrs001">
  <gml:pos>2 0.6</gml:pos>
</gml:Point>
<gml:Point gml:id="P4" srsDimension="2" srsName="#volsrs001">
  <gml:pos>3 -0.3</gml:pos>
</gml:Point>
<gml:Point gml:id="P5" srsDimension="2" srsName="#volsrs001">
  <gml:pos>4 -0.3</gml:pos>
</gml:Point>
<gml:Point gml:id="P6" srsDimension="2" srsName="#volsrs001">
  <gml:pos>5 -0.1</gml:pos>
</gml:Point>
```

```

<gml:Point gml:id="P7" srsDimension="2" srsName="#volsrs001">
  <gml:pos>6 0.7</gml:pos>
</gml:Point>
<gml:Point gml:id="P8" srsDimension="2" srsName="#volsrs001">
  <gml:pos>7 1.5</gml:pos>
</gml:Point>
<gml:Point gml:id="P9" srsDimension="2" srsName="#volsrs001">
  <gml:pos>8 1.7</gml:pos>
</gml:Point>
<gml:Point gml:id="P10" srsDimension="2" srsName="#volsrs001">
  <gml:pos>9 1.8</gml:pos>
</gml:Point>
<gml:Point gml:id="P11" srsDimension="2" srsName="#volsrs001">
  <gml:pos>10 1.9</gml:pos>
</gml:Point>

<gmllrov:VectorOffsetLinearSRS gml:id="volsrs001" xmlns:gmllr="http://www.opengis.net/gml/3.3/lr"
  xmlns:gmllrov="http://www.opengis.net/gml/3.3/lrov" xmlns:gml="http://www.opengis.net/gml/3.2">
  <gml:identifier codeSpace="..."><...</gml:identifier>
  <gmllr:linearElement xlink:href="#L1" xlink:title="LinearElement"/>
  <gmllr:lrm xlink:href="#lrm0001" xlink:title="LinearReferencingMethod"/>
  <gmllrov:offsetVector srsName="http://www.opengis.net/def/crs/EPSG/0/7405" offsetUom="m">0 1
    0</gmllrov:offsetVector>
</gmllrov:VectorOffsetLinearSRS>

<gmllr:LinearReferencingMethod gml:id="lrm0001" xmlns:gmllr="http://www.opengis.net/gml/3.3/lr">
  <gmllr:name>chainage</gmllr:name>
  <gmllr:type>absolute</gmllr:type>
  <gmllr:units>m</gmllr:units>
</gmllr:LinearReferencingMethod>

```

EXAMPLE 2 Two Offset Vectors

A basis of two offset vectors **v1** and **v2** can be used to describe the positions S1 to S11 relative to a linear element L2 in the offset reference frame as illustrated in **Error! Reference source not found..** Sample gml:Point encodings of the positions S1 to S11 are shown adjacent to the diagram below, where the first ordinate in each gml:pos element corresponds to distance along the linear element L2, the second ordinate in each gml:pos element corresponds to the component distance along the direction of offset vector **v1** and the third ordinate in each gml:pos element corresponds to the component distance along the direction of the offset vector **v2**.

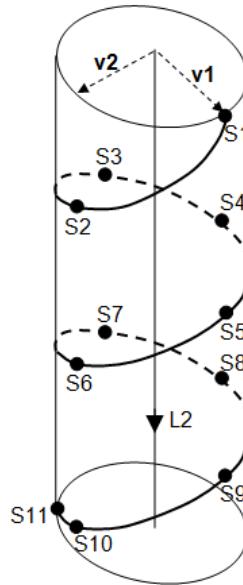


Figure 2 – Two offset vectors

```

<gml:Point gml:id="S1" srsDimension="3" srsName="#volsrs002">
  <gml:pos>0 1 0</gml:pos>
</gml:Point>
<gml:Point gml:id="S2" srsDimension="3" srsName="#volsrs002">
  <gml:pos>1 0 1</gml:pos>
</gml:Point>
<gml:Point gml:id="S3" srsDimension="3" srsName="#volsrs002">
  <gml:pos>2 -1 0</gml:pos>
</gml:Point>
<gml:Point gml:id="S4" srsDimension="3" srsName="#volsrs002">
  <gml:pos>3 0 -1</gml:pos>
</gml:Point>
<gml:Point gml:id="S5" srsDimension="3" srsName="#volsrs002">
  <gml:pos>4 1 0</gml:pos>
</gml:Point>
<gml:Point gml:id="S6" srsDimension="3" srsName="#volsrs002">
  <gml:pos>5 0 1</gml:pos>
</gml:Point>
<gml:Point gml:id="S7" srsDimension="3" srsName="#volsrs002">
  <gml:pos>6 -1 0</gml:pos>
</gml:Point>
<gml:Point gml:id="S8" srsDimension="3" srsName="#volsrs002">
  <gml:pos>7 0 -1</gml:pos>
</gml:Point>
<gml:Point gml:id="S9" srsDimension="3" srsName="#volsrs002">
  <gml:pos>8 1 0</gml:pos>
</gml:Point>
<gml:Point gml:id="S10" srsDimension="3" srsName="#volsrs002">
  <gml:pos>9 0 1</gml:pos>

```

```
</gml:Point>
<gml:Point gml:id="S11" srsDimension="3" srsName="#volsrs002">
  <gml:pos>9.3 -0.45 0.89</gml:pos>
</gml:Point>

<gmllrov:VectorOffsetLinearSRS gml:id="volsrs002">
  <gml:identifier codeSpace="...">>...</gml:identifier>
  <gmllr:linearElement xlink:href="#L2" xlink:title="Linear curve element"/>
  <gmllr:irm xlink:href="#irm0001" xlink:title="LinearReferencingMethod"/>
  <gmllrov:offsetVector srsName="http://www.opengis.net/def/crs/EPSG/0/7405" offsetUom="m">0 1
    0</gmllrov:offsetVector>
  <gmllrov:offsetVector srsName="http://www.opengis.net/def/crs/EPSG/0/7405" offsetUom="m">0 0
    1</gmllrov:offsetVector>
</gmllrov:VectorOffsetLinearSRS>
```

C.3.3.3 *VectorOffsetLinearSRSPROPERTYTYPE*

```
<complexType name="VectorOffsetLinearSRSPROPERTYTYPE">
  <sequence minOccurs="0">
    <element ref="gmllrov:VectorOffsetLinearSRS"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Appendix D. DIGGS Change Release Log

D.1 Version 1.1

D.1.1 Change Log 2010-05-18T14:30

Author: David Burggraf

Date/Time: May-18-2010 2:30 PM

Schema Version: 1.1

Topic: DIGGS brought to GML Conformance

Description:

The list of issues identified in the DIGGS 1.0a Schema Evaluation reports [1], [6], and [12] were fixed as agreed including:

- GML “Object-Property” Patterning
- Migration from GML 3.1 to GML 3.2
- Modified inheritance hierarchy of Objects
- Created GML 3.2 Profile
- Enabled relative referencing in the schemas
- Modified and validated the 20 example instances.
- Executed GML SDK to check for GML conformance violations – passed.
- Measured the reduction in complexity – reduction in schema size, number of schemas, includes/imports, schema load time, auto instance generation, etc.
- Verified that instances can open in XML editors including Altova, Oxygen, and Stylus with no need for special configuration.

D.2 Version 1.2a

D.2.1 Change Log 2010-05-20T12:25

Author: David Burggraf

Date/Time: May-20-2010 12:25 PM

Schema Version: 1.2a

Topic: Code lists**Description:**

- Identified all elements (except in Piling.xsd) that have codespace attributes
- Identified elements (except in Piling.xsd) that use enumerated lists. TBD if these are to be code lists instead (Loren to work with DIGGS core SIG on this)
- Generated Excel spreadsheet summary of candidate code lists.

D.3 Version 1.2.1

D.3.1 Change Log 2010-06-10

Author: Daniel Ponti

Date/Time: Jun-10-2010 2:30 PM

Schema Version: 1.2.1

Topic: Feature base classes formalized

Description:

In this version, we formalized the 9 feature base classes (see Figure below) so that all features in Kernel.xsd, Geotechnical.xsd, and Environmental.xsd fall within these classes and derive from the appropriate base types.

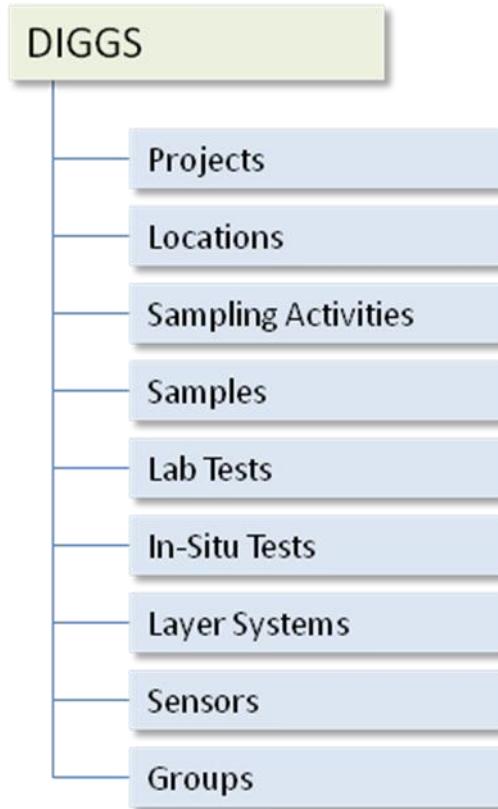


Figure 1 – There are 9 top-level feature classes in DIGGS v1.2a.

The nine feature classes are:

- **Projects** - business activities that collect, compile, and process information from locations [Process]
- **Locations** - real world places and constructions from which observations are made, samples are collected, or tests are run. [Entity]
- **Sampling Activities** - the process of sample creation or collection [Process]
- **Samples** - earth material, fluids, or gases collected or created for observation and testing [Entity]
- **Layer Systems** - ordered interval observations or interpretations of earth materials, properties or features at a location [Entity]
- **Laboratory Tests** - analyses performed on samples collected from locations, or created via a sampling activity [Process]
- **In-Situ Tests** - analyses or observations at a location [Process]
- **Sensors** (I'd suggest a nomenclature change to Installations or Monitoring Installations) - equipment or devices installed at locations that collect repeated measurements or observations [Entity]
- **Groups** - collections of projects, locations, samples or groups of these, for the purpose of providing meaningful context to observations and measurements.

The following Figure shows an illustration of how the feature classes are associated in DIGGS.

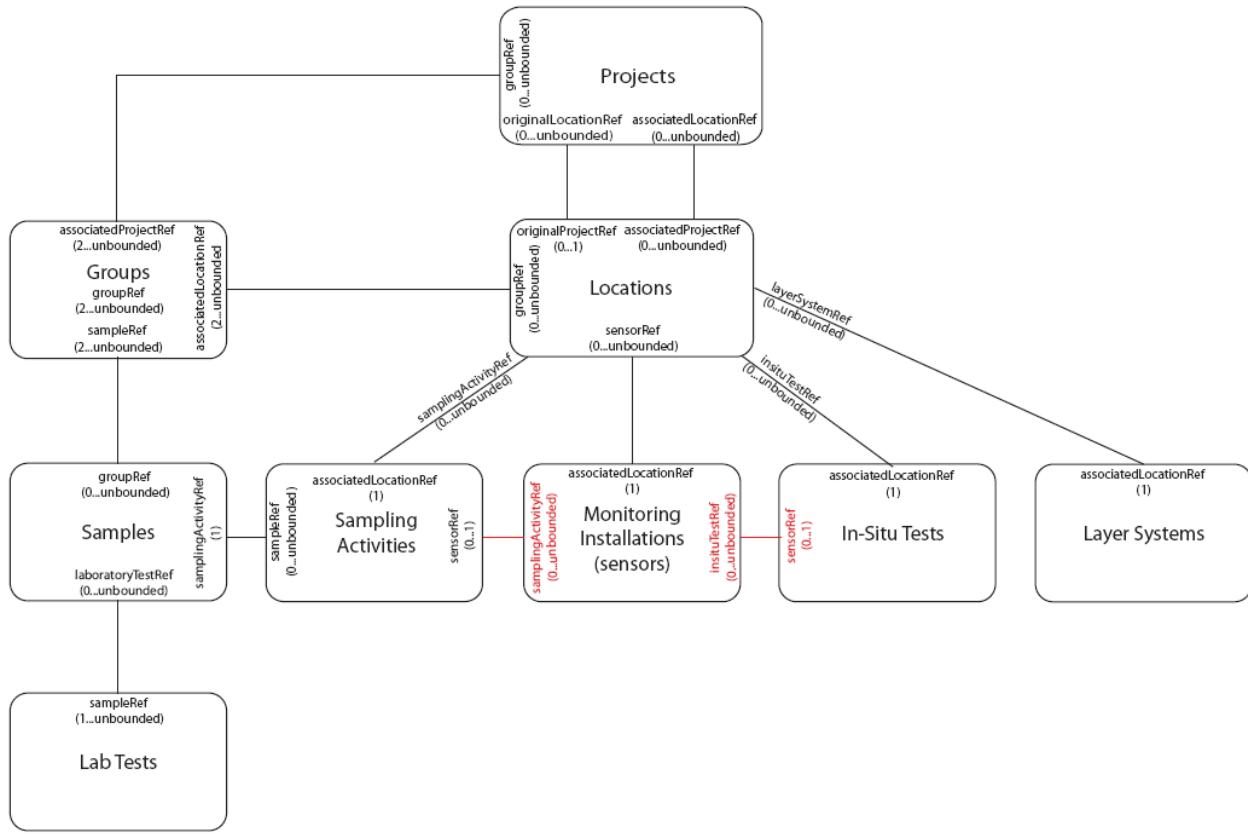


Figure 2 – Feature class associations in DIGGS

All features in DIGGS carry a mandatory id (*gml:id*), required by GML and used for referencing and linking with other features. All features also carry an identifier (*gml:identifier*), required by DIGGS, which is a globally unique key for the feature, and uses a URN pattern. Optional properties of all feature classes include status (needs clearer definition), description, and associated file, role, and remarks metadata objects.

Projects, Locations, Samples, Layer Systems, Sensors, and Groups are "named" features. In addition to the properties above, they also carry a mandatory name property.

All objects (complex properties of features) must carry a mandatory id; optional properties of all objects are description, status, and the remarks metadata objects. Some metadata objects are named (eg. equipment and specifications), and carry a mandatory name property.

Metadata objects currently defined are:

- **Document Information** - information about the specific XML instance document
- **Associated Files** - references to non-XML documents or records outside of the XML instance
- **Business Associates** - persons and institutions

- **Equipment** - well, you know, equipment :-)
- **Specifications** - test specifications or procedures

D.4 Version 1.2.2

D.4.1 Change Log 2010-07-01T13:30

Author: Dan Ponti

Date/Time: July-01-2010 1:30 PM

Schema Version: 1.2.2

Topic: Further feature class restructuring

Description:

- gml:identifier is now optional
- Project was moved to top hierarchy.
 - Retained the structure originally proposed where all features are at the root level.
 - Made the Project feature mandatory in all DIGGS instances.
 - Restricted Project feature so that only one Project is permitted in a DIGGS file.
 - All features in the DIGGS file now associated to the single project (via href).
- Added Location Features - In v 1.1, the Hole feature was somewhat generic, it was set up to handle properties of a geotechnical borehole, but could be other types that were of similar geometry but not really boreholes (eg. a transect or trial pit). In 1.2a, there are now specific Location features that derive from either AbstractPointLocation, AbstractLinearLocation, or AbstractPlanarLocation location feature types, each of which have properties specific to that type of feature. This way, we can model more types of features in a straightforward fashion in the future, such as embankments, tunnels, roads, etc. Diggs 1.2a now has defined in it the following location features:
 - Borehole - very similar to Hole in v 1.1, but modified to handle 1.2 constructs. (Linear Location)
 - Trial Pit - a shallow excavation - legacy to handle current AGS trial pit constructs. (Linear Location)
 - Trench Wall - designed to supplant Trial Pit in the future. A wall of a trench or pit represented by a vertical planar surface. (Planar Location); in 1.2a this is only partially built
 - Station - a point on the earth's surface (Point Location)

D.5 Version 1.2.3a

D.5.1 Change Log 2010-07-06T11:31

Author: Daniel Ponti

Date/Time: July-06-2012 11:31 AM

Schema Version: 1.2.3a

Topic: Layer Systems and Location Features

Description:

Layer Systems

Layer systems became more specific to make data mapping to DIGGS easier and to provide more specificity in coding. The Layer system object now carries mandatory properties that indicate which type of layer system it is, and that layer system's subtype. Layer system types come from an enumerated list hard-coded into the schema; sub-types are code types that would be defined in referenced code lists. The types of layer systems (enumerations) are:

- *Color* – describes the color of materials encountered
- *Component* – describe details of earth materials encountered
- *Discontinuity* – describes fractures and joints and their spacing
- *Lithology* – describe the earth materials encountered
- *Orientation* – describes the geometry of vectors or planar surfaces encountered at a location, such as bedding, joints, cross-beds, etc.
- *Other* – describes a layer system of unknown type, using name-value pairs.
- *Property* – describes a layer system where the results are simple text or numeric values - usually interpreted as a result of some lab or in-situ test (eg. porosity).
- *Stratigraphy* – describes ordered bodies of rock or soil, such as formations, biostratigraphic units or aquifers.

The layer objects that are properties of layer systems are defined separately for each type of layer system. For example, a layer system of type lithology has a layer of type LithologyLayerType, which has properties unique to lithology layer systems. Note - Discontinuity layer systems replace the discontinuity and fracture spacing properties of the Hole feature in v 1.1.

Positions of Sampling Activities, Layers, Tests and Sensors at Locations

Diggs v 1.1 uses point properties of top and base to describe the positions of observations (eg. sampling activities layers, tests, sensors) at locations. This position type is suitable only for holes and hole type features where the positions are described along a vertical (or nearly so) linear reference. This is constraining and does not allow for easy reuse of these observation features.

In 1.2a, all of these observations carry a position property type that contains a LocationPosition object that is defined by the specific type of Location feature where the observation occurs. So, position properties for a borehole feature are contained within a BoreholePosition position object. A BoreholePosition object consists of a choice of two properties:

- measuredDepth (a `gml:PointProperty` in 1D)
- depthInterval (`gml:CurveProperty` in 1D).

A `TrenchWallPosition` object consists of a choice between a `pointPosition` (2D Point property), `linearElementPosition` (2D Curve property), or a `surfacePosition` (2D surface property).

For example, a layer described in a hole would contain a `BoreHolePosition` object in its position property - typically with a `depthInterval` property that defines the top and base of the layer. If the layer were associated with a trench wall, its position property would contain a `TrenchWallPosition` object to define its position on a trench wall in a 2D reference system; this would most likely be a surface property (polygon) that defines the exposure of the layer in the trench wall. The layer feature's other properties remain exactly the same; there is no need to redefine a layer for different location features.

D.6 Version 1.2.3.b

D.6.1 Change Log 2010-07-05T12:59

Author: David Burggraf

Date/Time: July-05-10 12:59 PM

Schema Version: 1.2.3b

Topic: MultiPoint Coverage Implementation

Description:

Created the schema constructs for `MultiPointCoverage` and example instance – excerpt as follows.

```
<diggs_geo:coverageData>
  <gml:MultiPointCoverage gml:id="MPC001">
    <gml:domainSet>
      <g3.3:MultiPoint gml:id="MP001">
        <g3.3:position>0.010 0.020 0.030 0.040 0.050 0.060 0.070 0.080 0.090 0.100 0.110 0.120
          0.130 0.140 0.150 0.160 0.170 0.180 0.190 0.200 0.210 0.220 0.230 0.240 0.250 0.260
          0.270 0.280 0.290 0.300 0.310 0.320 0.330 0.340 0.350 0.360 0.370 0.380 0.390 0.400
          <!-- snip-->
          5.310 5.320 5.330 5.340 5.350 5.360 5.370 5.380 5.390 5.400 5.410 5.420 5.430 5.440
        </g3.3:position>
      </g3.3:MultiPoint>
    </gml:domainSet>
    <gml:rangeSet>
      <gml>DataBlock>
        <gml:rangeParameters>
```

```

<gml:CompositeValue gml:id="CV001">
  <gml:valueComponents>
    <diggs:Column gml:id="Dd1e266" index="1">
      <dataType>xs:double</dataType>
      <meaning>Measure</meaning>
      < uom>MN/m2</ uom>
      <source xlink:href="#DIGGS-CPT-CONE-1-RES"/>
    </diggs:Column>
    <diggs:Column gml:id="Dd1e283" index="2">
      <dataType>xs:double</dataType>
      <meaning>Measure</meaning>
      < uom>uS/cm</ uom>
      <source xlink:href="#DIGGS-CPT-CONE-1-COND"/>
    </diggs:Column>
    <diggs:Column gml:id="Dd1e300" index="3">
      <dataType>xs:double</dataType>
      <meaning>Measure</meaning>
      < uom>kN/m2</ uom>
      <source xlink:href="#DIGGS-CPT-CONE-1-LSFR"/>
    </diggs:Column>
    <diggs:Column gml:id="Dd1e317" index="4">
      <dataType>xs:double</dataType>
      <meaning>Measure</meaning>
      < uom>kN/m2</ uom>
      <source xlink:href="#DIGGS-CPT-CONE-1-PWP"/>
    </diggs:Column>
  </gml:valueComponents>
</gml:CompositeValue>
</gml:rangeParameters>
<gml:tupleList ts="" cs="," decimal="."> 0.1300,0.40,0.0000,0.0013
  0.2400,0.40,0.1.0a,0.0078 0.5500,0.40,0.0040,0.0126 0.6800,0.40,0.0070,-0.0017
  0.7800,0.30,0.0120,-0.0121 0.9000,0.30,0.0150,-0.0161 0.9600,0.40,0.0200,0.0191
  <!-- snip-->
  40.5400,0.40,9999.0000,9999.0000 </gml:tupleList>
</gml:DataBlock>
</gml:rangeSet>
</gml:MultiPointCoverage>
</diggs_geo:coverageData>

```

D.7 Version 1.2.4.a

D.7.1 Change Log 2010-07-07T11:20

Author: Daniel Ponti

Date/Time: Wednesday, July 07, 2010 11:20 PM

Schema Version: V1.2.4.a**Topic:** Major Schema Update Migrating V1.2.3a to V1.2.4a**Description:** The changes reflect modifications based on vendor comments.**Changes to Kernel.xsd:**

1. Added projectRef element to schema - of FeatureReference type, designed to hold a single reference to the project that a feature is associated with for the purpose of this instance document.
2. Added the projectRef element as a mandatory property of the following types
 - a) AbstractTestType
 - b) AbstractLocationType
 - c) UndefinedLocationType
 - d) SamplingActivityType
 - e) AbstractGroupType - not sure this is correct, so made projectRef optional. Does a group need a project reference? What about a group of projects?
 - f) LayerSystemType
 - g) SampleType
3. Made the associatedLocationRef property of SamplingActivity optional. Now that the projectRef is mandatory, there is no need to reference a dummy location to derive the project for a sampling activity that does not occur at a location.
4. Deprecated UndefinedLocation and UndefinedLocationType (did not delete but commented out of schema). No longer needed now that SamplingActivity carries a mandatory projectRef property
5. Added back ProjectGroup element and type(s) from 1.2.1 to allow for grouping projects
6. Replaced ProjectType with ProjectType from v. 1.2.1.a
7. Moved locations, laboratoryTesting, insituTesting, samplingActivities, samples, layerSystems, monitoringInstallations, and groups properties from ProjectType to DiggsType.
8. Added testPosition property of type LocationPositionPropertyType to AbstractInsituTestType.
9. Added inSituTestRef and laboratoryTestRef properties to samplingActivity to allow an activity to reference a test that produces a sample.
10. Added samplingActivityRef properties to AbstractInsituTestType and AbstractLaboratoryTestType to allow a test to reference a sampling activity if in fact a sample is produced from the test.

11. Added optional sensorRef property to AbstractInsituTestType to enable a test to be conducted at a MonitoringLocation.
12. Added nullDataType for values assigned to null and a reason attribute
13. Created AllUnitsType - union of all witsml units.
14. Created CurveInfo element and associated types to handle column definitions of depth-indexed tabular data (eg. CPT and geophysics).
15. Added substituted choice elements for metadataReferencetype properties for equipment, specifications, and business associates, so that users could just substitute a string value instead of having to explicitly reference one of these metadata features
16. Replaced gml3.2Profile_diggs.xsd and gml3.3Profile_diggs.xsd files with the same schema files from the 1.2.3b revision of Burgraff dated 7/5/10.
17. Removed type attribute from ConstituentType (which was an enumerated list) and added a mandatory constituentType property to ConstituentType that is of type gml:CodeType designed to read from a code list instead
18. Changed the type of the following properties from diggs:ReferenceType to gml:ReferenceType
 - a. source property of ColumnType
19. Deleted global diggs:ReferenceType. No longer needed.
20. Added InsituTest, InsituTestType, InsituTest.PropertyType, AbstractInsituTestParameters, AbstractInsituTestParametersType, AbstractInsituTestResults, AbstractInsituTestResultsType elements and complex types to support standardization of test features.
21. Added LaboratoryTest, LaboratoryTestType, LaboratoryTest.PropertyType, AbstractLaboratoryTestParameters, AbstractLaboratoryTestParametersType, AbstractLaboratoryTestResults, AbstractLaboratoryTestResultsType elements and complex types to support standardization of test features.
22. Added parameters and results properties to AbstractInSituTestType and AbstractLaboratoryTestType
23. Added mandatory testType property to AbstractTestType.
24. Added the following elements and types to restrict GML coverage and morph this into types for use with CPT and geophysical logs.
 - a. Log, LogType, Log.PropertyType (derives from gml:AbstractCoverageType)

- b. logPositions, LogPositionsType (substitutes for gml:domainSet)
 - c. logData, LogDataType (substitutes for gml:rangeSet)
 - d. DataBlock, DataBlockType (substitutes for gml:DataBlock)
 - e. curveValues of type gml:CoordinatesType (substitutes for gml:tupleList)
 - f. columns, ColumnsType (substitutes for gml:RangeParameters)
 - g.ColumnInfo, ColumnInfoType (substitutes for gml:ValueArray)
 - h. curves, CurvesType derives from gml:ValueArrayPropertyType and has (substitutes for gml:valueComponents)
 - i. Curve, CurveType, Curve.PropertyType (substitutes for gml:AbstractValue); contains curve/column parameters
25. Changed substitution group for Column element back to diggs:AbstractObject from gml:AbstractValue so this won't substitute for the log type.

Changes to Environmental.xsd

1. Modified AnalysisType to derive from AbstractLaboratoryTestType instead of AbstractFeatureType
2. Added readingMethodRef property to FieldReadingType and renamed methodName to readingMethod and moved into a choice element with this new type (Ref references a Specification)
3. Added readingMethodRef property to WaterLevelReadingType and renamed method to readingMethod and moved into a choice element with this new type (Ref references a Specification)

Changes to Monitoring.xsd

1. Added the projectRef element as a mandatory property of the following type:
 - a) SensorType

Changes to Geotechnical.xsd

1. Modifications to StaticConeTest and associated types:
 - a) grouped all of the test parameter properties into a new type called StaticConeTestParameterType, extending diggs:AbstractSimpleMetadataType
 - b) created new element called StaticConeTestParameters of type StaticConeTestParameterType that extends AbstractInsituTestParametersType
 - c) created new type called StaticConeTestParameter.PropertyType to hold an optional reference to StaticConeTestParameters
 - d) deleted tabularData property (now replaced with gml:coverage construct) from StaticConeTestType

e) created a new element called StaticConeTestResults of type StaticConeTestResultsType that adds cptLogTable property (a coverage type) in extending AbstractInsituTestResultsType.

f) deleted StaticConeTest, StaticConeTestType and StaticConeTestPropertyType - no longer needed.

2. Changed the type of the following properties from diggs:ReferenceType to gml:ReferenceType

- a. sources property of CompactionDetailType
- b. sources property of CompressiveStrengthDetailType
- c. sources property of ConsolidationDetailType
- d. sources property of MCVDetailType
- e. sources property of ShearBoxDetailType

3. Added specimenClampingMethodRef property to SchmidtReboundHardnessType and moved specimenClampingMethod into a choice element with this new type (Ref references a Specification)

4. Added saturationMethodRef property to StaticConeTestParameterType and moved saturationMethod into a choice element with this new type (Ref references a Specification)

5. Modifications to DensityTest:

- a. created new element DensityTestParameters of type DensityTestParametersType
- b. created new element DensityTestResults of type DensityTestResultsType, incorporating DensityMeasurementProperties
- c. deleted Density, Density.PropertyType, DensityType, DensityMeasurement, DensityMeasurementType, and DensityMeasurement.PropertyType - no longer needed.
- d. added DensityResults.PropertyType to support reference to density results in other tests. Changed types from DensityMeasurement.PropertyType of these properties to DensityResults.PropertyType.

Changes to Piling.xsd

1. Changed the type of the following properties from diggs:ReferenceType to gml:ReferenceType

- a. sources property of PileConstructionType

2. Added dearingMethodRef property to MaterialPlacementLogType and moved dearingMethod into a choice element with this new type (Ref references a Specification)

3. Added inspectionMethodRef property to InspectionType and moved inspectionMethod into a choice element with this new type (Ref references a Specification)

4. Added drillingMethodRef property to ShaftMakeupType and moved drillingMethod into a choice element with this new type (Ref references a Specification)

5. Added predictionMethodRef property to PredictedCapacityType and moved predictionMethod into a choice element with this new type (Ref references a Specification)

Changes to `gml3.2Profile_diggs.xsd`

1. Modified RangeSetType to uncomment ValueArray element.
2. Added ValueArray, ValueArrayType, and referenceSystem attribute group to profile (previously not included) so that the change above would validate.

D.8 Version 1.2.4.b

D.8.1 Change Log 2010-07-08T13:03

Author: David Burggraf

Date/Time: July-08-2010 1:03 PM

Schema Version: V1.2.4.b

Topic: Coverage Components

Description:

Changed Kernel.xsd so that ColumnInfo substitutes for gml:CompositeValue instead of gml:ValueArray - following a review of the new coverage components in 1.2.4.a, I noticed the use of gml:ValueArray could have been avoided (i.e. not added to the GML 3.2 profile profile) because gml:CompositeValue is already there and has a very similar content model.

```
<element name="ColumnInfo" type="diggs:ColumnInfoType" substitutionGroup="gml:CompositeValue"/>
<complexType name="ColumnInfoType">
  <complexContent>
    <restriction base="gml:CompositeValueType">
      <sequence>
        <element ref="diggs:curves"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
```

D.8.2 Change Log 2010-07-12T19:14

Author: Daniel Ponti

Date/Time: Monday, July 12, 2010 7:14 PM

Schema Version: V1.2.4.b

Topic: General migration from 1.2.4.a to 1.2.4.b

Description:

Kernel.xsd changes:

1. Switched back to strong typed test features, but using parameters/results model:
 - a. Deletion of parameters, results properties from AbstractLaboratoryTestType
 - b. Deletion of LaboratoryTest element.
 - c. Deletion of LaboratoryTestParameterPropertyType and LaboratoryTestResultsPropertyType.
 - d. Deletion of parameters, results properties from AbstractInsituTestType
 - e. Deletion of InsituTest element.
 - f. Deletion of InsituTestParameterPropertyType and InsituTestResultsPropertyType.
 - g. Deletion of testType from AbstractTestType
2. Added new test elements/types for Geophysical Logs:
 - a) GeophysicalLogTest/Type
 - b) GeophysicalLogTestParameters/Type
 - c) GeophysicalLogTestResults/Type
 - d) ... and associated PropertyTypes
3. Added new complex type to support log parameters (diggs:IndexedParameterType) - modeled from witsml:IndexedObject

Geotechnical.xsd changes:

1. Conversion of StaticConeTest and DensityTest back to strong-typed features but using the parameters/results model:
 - a. Creation of StaticConeTestParametersPropertyType and StaticConeTestResultsPropertyType complex types. StaticConeTestParametersPropertyType extends gml:AbstratMetadataPropertyType.
 - b. Creation of StaticConeTest and StaticConeTestType - derives from AbstractInsituTestType and adds results and parameters properties.
 - c. Creation of DensityTestParametersPropertyType and DensityTestResultsPropertyType complex types. DensityTestParametersPropertyType extends gml:AbstratMetadataPropertyType.
 - d. Creation of DensityTest and DensityTestType - derives from AbstractLaboratoryTestType and adds results and parameters properties.
2. Added Test suffix to all tests and test types

3. Made parameter/results pattern changes to DilatometerTest (as described in e-mail to David Burggraf).
4. Commented out DensityTest/DensityTestType, StaticConeTest/StaticConeTest type, DilatometerTest/DilatometerTestType and GeophysicalLogTest/GeophysicalLogTestType temporarily so that future scripting won't affect these.

D.8.3 Change Log 2010-07-20T17:50

Author: David Burggraf

Date/Time: Monday, July 20, 2010 17:50 PM

Schema Version: V1.2.4.b

Topic: Automated Conversion of Tests

Description:

An automated script was executed following Dan Ponti's algorithm (documented below) to convert most of the existing concrete (non-abstract) test structures to a new scheme structure using a parameters/results property pattern. Before the conversion, each test consisted of at minimum, one element and two types:

- 1) XXXTest (element of type XXXTestType, and belonging to either AbstractInsituTest or AbsractLaboratoryTest substitution groups, depending on the type of test). XXX is the name of the test.
- 2) XXXTestType (derives from either AbstractInsituTestType if an insitu test or AbstractLaboratoryTestType if a lab test)
- 3) XXXTestPropertyType (a complex type that incorporates the XXXTest element by reference).

Some tests have XXXDetail objects/types and detail property types. Detail types derive from diggs:AbstractObjectType. To convert to the new test structure, the XXXTest element remains unchanged, as does XXXTestPropertyType and any XXXDetail types. Only the existing XXXTestType needs to be modified, and then additional elements and types need to be created for each test.

The conversion algorithm followed for the new test test pattern was:

- 1) Convert non-abstract XXXTestType as follows
 - a) change complexType name from XXXTestType to XXXTestResultsType
 - b) change extension base from:
 - i) if extension base is diggs:AbstractIntsitutestType, change extension base to diggs:AbstractInsituTestResultsType

ii) if extension base is diggs:AbstractLaboratoryTestType, change extension base to diggs:AbstractLaboratoryTestResultsType

-- leave the rest of the complex type alone.

2) Create a new complex type named XXXTestParametersType as follows:

```
<complexType name="XXXTestParametersType">
<complexContent>
    <extension base="diggs:AbstractLaboratoryTestParametersType">
        <!-- If test is an insitu test, this line should read: <extension base="diggs:AbstractInsituTestParametersType"> -->
        <sequence>
            <!-- Parameter properties to be inserted into the sequence for each test during domain review -->
        </sequence>
    </extension>
</complexContent>
</complexType>
```

3) Create a new element named XXXTestParameters as follows:

```
<element name="XXXTestParameters" type="namespace:XXXTestParametersType"
    substitutionGroup="diggs:AbstractLaboratoryTestParameters" abstract="false">
    <!-- namespace should be replaced with appropriate namespace identifier for the test -->
    <!-- if test is an insitu test substitutionGroup should be "diggs:AbstractInsituTestParameters" -->
</element>
```

4) Create a new element named XXXTestResults as follows:

```
<element name="XXXTestResults" type="namespace:XXXTestResultsType"
    substitutionGroup="diggs:AbstractLaboratoryTestResults" abstract="false">
    <!-- namespace should be replaced with appropriate namespace identifier for the test -->
    <!-- if test is an insitu test substitutionGroup should be "diggs:AbstractInsituTestResults" -->
</element>
```

5) Create a new complex type named XXXTestParametersPropertyType as follows:

```
<complexType name="XXXTestParametersPropertyType">
<complexContent>
    <extension base="gml:AbstractMetadataPropertyType">
        <sequence>
            <element minOccurs="0" ref="namespace:XXXTestParameters"/> <!-- namespace should be replaced with
            appropriate namespace identifier for the test -->
        </sequence>
    </extension>
</complexContent>
</complexType>
```

6) Create a new complex type named XXXTestResultsPropertyType as follows:

```
<complexType name="XXXTestResultsPropertyType">
  <sequence>
    <element minOccurs="0" ref="namespace:XXXTestResults"/> <!-- namespace should be replaced with
appropriate namespace identifier for the test -->
  </sequence>
</complexType>
```

7) Finally, create a new XXXTestType complex type to replace the original test type that was renamed. It will look like the following

```
<complexType name="XXXTestType">
  <complexContent>
    <extension base="diggs:AbstractLaboratoryTestType"> <!-- or if this is an insitu test, this line changes to
<extension base="diggs:AbstractInSituTestType"> -->
      <sequence>
        <element name="parameters" type="namespace:XXXTestParametersPropertyType"/> <!-- namespace
should be replaced with appropriate namespace identifier for the test -->
        <element name="results" type="namespace:XXXTestResultsPropertyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Did not apply the automated conversion to the complex types in Piling.xsd with the name pattern XXXTestType, which do not currently derive from either AbstractLaboratoryTestType or AbstractInSituTestType (but those that do derive from either of these types were changed). The Piling schema was built independently from the Geotech schema and seems to use different logic. The Piling schema will need to be substantially reworked with the appropriate subject matter experts at a later stage. The conclusion from the last big workshop was that Piling was out of scope for 1.2 - the emphasis was supposed to be on kernel and geotech, and the others if there's time and budget.

Most of the tests were in the diggs:geo namespace but was run on all schema files for completeness. The script searched from complex types that derive from either AbstractInSituTestType or AbstractLaboratoryTestType.

D.8.4 Change Log 2010-07-23T18:49

Author: Daniel Ponti [<mailto:dponti@usgs.gov>]

Date/Time: Friday, July 23, 2010 6:49 PM

Schema Version: V1.2.4.b

Topic: GradingTest element changed and inappropriate base types of some tests corrected

Description:

GradingTest - this element was inappropriately named - it is a GML object as the base type suggests, not a test.

```

<complexType name="GradingTestType">
  <complexContent>
    <extension base="diggs:AbstractObjectType">
      <sequence>
        <element name="percentPassing" type="witsml:generalMeasureType" minOccurs="1"
          maxOccurs="1"/>
        <element name="sieveSize" type="witsml:lengthMeasure" minOccurs="0" maxOccurs="1"/>
        <element name="sieveNumber" type="gml:CodeType" minOccurs="0" maxOccurs="1"/>
        <element name="type" type="gml:CodeType" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

So the following changes were made:

- 1) element GradingTest name changed to Grading
- 2) complex type GradingTestType name changed to GradingType. Its extension base remains AbstractObjectType.
- 3) complex type GradingTestPropertyType name changed to Grading.PropertyType
- 4) All elements of type diggs_geo:GradingTestPropertyType had their types changed to diggs_geo:Grading.PropertyType to associate with the renamed property type.

Many occurrences of XXXTestType that have unexpected base types, i.e. not AbstractLaboratoryTestType or AbstractIntsituTestType (e.g. some base values were found to be: AbstractObjectType, AbstarctTestType, MaterialTestType). For example SlumpTestType:

```

<complexType name="SlumpTestType" mixed="false">
  <complexContent>
    <extension base="diggs_pil:MaterialTestType">
      <sequence/>
    </extension>
  </complexContent>
</complexType>
<complexType name="MaterialTestType" mixed="false">
  <complexContent>
    <extension base="diggs:AbstractTestType">
      <sequence/>
    </extension>
  </complexContent>
</complexType>

```

D.9 Version 1.2.4.c

D.9.1 Change Log 2010-07-28T17:38

Author: David Burggraf

Date/Time: Wednesday, July 28, 2010 5:38 PM

Schema Version: V1.2.4.c

Topic: Automated conversion scripts tweaked and re-executed

Description:

There were 3 Abstract test types in Kernel.xsd (not changed):

```
diggs:AbstractTestType  
diggs:AbstractLaboratoryTestType  
diggs:AbstractInSituTestType
```

There was one test type in Environmental.xsd (changed to new pattern):

```
diggs_env:EnvironmentalTestType
```

One type called ‘diggs_geo:AggregateAbrasionTestValueType’ should have been changed to the new pattern, so was renamed ‘diggs_geo:AggregateAbrasionValueTestType’ (and similarly for its associated elements and types). Then the scripts were re-executed to convert to the new pattern.

All the rest of the test types were in Geotechnical.xsd (changed to new pattern):

```
diggs_geo:AggregateAbrasionValueTestType  
diggs_geo:AggregateCrushingValueTestType  
diggs_geo:AggregateImpactValueTestType  
diggs_geo:AtterbergLimitsTestType  
diggs_geo:CBRTTestType  
diggs_geo:ChalkTestType  
diggs_geo:CompactionTestType  
diggs_geo:CompressiveStrengthTestType  
diggs_geo:ConsolidationTestType  
diggs_geo:DensityTestType  
diggs_geo:DilatometerTestType  
diggs_geo:DrivenPenetrationTestType  
diggs_geo:ElongationIndexTestType  
diggs_geo:FlakinessIndexTestType  
diggs_geo:FrostSusceptibilityTestType  
diggs_geo:HandVaneTestType  
diggs_geo:InsituCBRTTestType  
diggs_geo:InsituDensityTestType  
diggs_geo:InsituPermeabilityTestType  
diggs_geo:InsituResistivityTestType
```

```
diggs_geo:InsituVaneTestType
diggs_geo:LaboratoryPocketPenetrometerTestType
diggs_geo:LaboratoryVelocityTestType
diggs_geo:LosAngelesAbrasionTestType
diggs_geo:MCVTestType
diggs_geo:MoistureContentTestType
diggs_geo:ParticleSizeTestType
diggs_geo:PermeabilityTestType
diggs_geo:PocketPenetrometerTestType
diggs_geo:PointLoadTestType
diggs_geo:PolishedStoneValueTestType
diggs_geo:PorosityTestType
diggs_geo:PressuremeterTestType
diggs_geo:PumpingTestType
diggs_geo:RedoxPotentialTestType
diggs_geo:RelativeDensityTestType
diggs_geo:SchmidtReboundHardnessTestType
diggs_geo:ShearBoxTestType
diggs_geo:ShoreHardnessTestType
diggs_geo:ShrinkageTestType
diggs_geo:SlakeDurabilityTestType
diggs_geo:SoundnessTestType
diggs_geo:StandardPenetrationTestType
diggs_geo:StaticConeTestType
diggs_geo:SuctionTestType
diggs_geo:TenPercentFinesTestType
diggs_geo:WaterAbsorptionTestType
```

The test types that were already changed manually and commented out were restored by un-commenting them:

```
diggs_geo:DensityTestType
diggs_geo:DilatometerTestType
diggs_geo:StaticConeTestType
```

Issues Encountered:

The following elements extend from 'diggs:AbstractLaboratoryTestType', but do not have the TestType suffix in the element name, so were not converted to the new test pattern.

```
diggs_env:AnalysisType
diggs_geo:FlamelonisationDetectorType
diggs_geo:PhotolonisationDetectorType
```

D.10 Version 1.2.4.d

D.10.1 Change Log 2010-08-06T18:09

Author: David Burggraf

Date/Time: August-06-10 6:09 PM

Schema Version: V1.2.4.d

Topic: Resolved missing TestType name suffix issue and generated codelist spreadsheet

Description:

As discussed in the status meeting this week, the name suffix of the following 3 types were changed to:

```
diggs_env:AnalysisTestType
diggs_geo:FlamelonisationDetectorTestType
diggs_geo:PhotolonisationDetectorTestType
```

The scripts were re-executed to get the new outputs.

The codetypes script was also executed and a new spreadsheet was generated and saved as CodeLists/SchemaV1.24CodeTypes.xlsx

Issues Encountered:

As a result of making the TestType name suffix changes described above, I noticed diggs_env:SpectralAnalysisType derives by extension from AnalysisTestType (by adding the single property 'wavelength'). We need to address how to deal with extensions of a *TestType. It was left it as is for now. Below is the SpectralAnalysis example:

```
<element name="SpectralAnalysis" type="diggs_env:SpectralAnalysisType"
  substitutionGroup="diggs_env:AbstractAnalysis" abstract="false"/>
<complexType name="SpectralAnalysisType">
  <complexContent>
    <extension base="diggs_env:AnalysisTestType">
      <sequence>
        <element name="wavelength" type="witsml:lengthMeasure" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

D.11 Version 1.2.4.e

D.11.1 Change Log 2010-08-09T13:08

Author: David Burggraf

Date/Time: August-09-10 1:08 PM

Schema Version: V1.2.4.e

Topic: Draft DIGGS URN RFC submission for IANA

Description:

I drafted the URN RFC submission (The_DIGGS_URN_Standard.docx) following the URN definition template (Appendix A of URN_DefinitionMechanisms_rfc3406.pdf) and based it as a template on the OGC submission to IANA (OGC_URN_rfc5165_submittedToIANA.pdf). Action to Loren: review and complete the highlighted sections of the document.

D.11.2 Change Log 2010-08-11T12:21

Author: Daniel Ponti

Date/Time: Wednesday, August 11, 2010 12:21 PM

Schema Version: V1.2.4.e

Topic: North American CRS's

Description:

Drafted a spreadsheet (\CRS\input\North American CRS.xls) of potential horizontal and vertical CRS's

used in North America (US and Canada) dumped out of the EPSG database (v 6.5.2). The first worksheet lists the horizontal (2D) CRS's (there are 695 of them!). They are ordered by area name, which will make them a bit easier for people to find (as opposed by sorting on EPSG code).

The second worksheet lists the 4 vertical coordinate systems to combine with each of the horizontal ones (including CDN 1928 for Canada).

Finally, the last sheet lists the two compound CRS's already in the EPSG database. We could (and probably should) have DIGGS codes for these as well, but should note that there is an EPSG equivalent.

Each resultant compound CRS will be a combination of a horizontal system with each of the vertical systems (695 x 4 = 2780 CRS's...)

D.11.3 Change Log 2010-08-12T12:25

Author: David Burggraf

Date/Time: Aug 12, 2010, at 12:25 PM

Schema Version: V1.2.4.e**Topic:** North American CRS's**Description:**

There are some EPSG codes specified in the North American spreadsheet that do not exist in the latest version of the EPSG registry (V7.5) and hence are not supported by GeoTools. These were probably retired sometime after V6.5. The following unsupported codes were removed from the spreadsheet \CRS\input\North American CRS.xls:

63266406	WGS 84 (degH)
63266407	WGS 84 (Hdeg)
63266408	WGS 84 (DM)
63266409	WGS 84 (DMH)
63266410	WGS 84 (HDM)
63266412	WGS 84 (HDMS)

D.11.4 Change Log 2010-08-12T16:10

Author: David Burggraf**Date/Time:** August-12-10 4:10 PM**Schema Version:** 1.2.4.e**Topic:** RE: North American CRS's**Description:**

Used an XSLT script to generated the formal CRS dictionary definitions (both as WKT and as GML) see:

CRS\output\DIGGS_GML_NA.xml
 CRS\output\DIGGS_GML_UK.xml
 CRS\output\DIGGS_GML_World.xml

CRS\output\DIGGS_WKT_NA.txt
 CRS\output\DIGGS_WKT_UK.txt
 CRS\output\DIGGS_WKT_World.txt

We used three input spreadsheets to generate these outputs see:

CRS\input\North American CRS.xls
 CRS\input\UK CRS.xls
 CRS\input\World CRS.xls

D.11.5 Change Log 2010-08-12T16:16

Author: David Burggraf**Date/Time:** August-12-10 4:16 PM

Schema Version: V1.2.4.e

Topic: SpectralAnalysis fixed

Description:

Changed diggs_env:SpectralAnalysis element to a test type, renamed to SpectralAnalysisTest and converted all the associated elements and types to the new test pattern as discussed in the Wednesday status meeting. The corresponding testinstance.xml file was also modified so that it was valid.

D.11.6 Change Log 2010-08-16T11:12

Author: David Burggraf

Date/Time: Aug 16, 2010, at 11:12 AM

Schema Version: V1.2.4.e

Topic: Automated Object/property rule check

Description:

Executed the automated GML conformance check using the Galdos GMLSDK and there were no Object/property rule violations detected. There were some naming convention anomalies however. The following types should start with an upper case letter:

percentMeasureType
geotechnicalDensityMeasure

The following types would have the 'Type' suffix:

geotechnicalDensityMeasure
PercentDouble

The following elements should start with the 'Abstract' prefix (because they are abstract)

Reinforcement
PileConstruction
MaterialTest
LoadTestIncrement

D.12 Version 1.2.4.f

D.12.1 Change Log 2010-08-17T15:21

Author: David Burggraf

Date/Time: August-17-10 3:21 PM

Schema Version: V1.2.4.f

Topic: Automated Object/property rule conformance changes

Description:

Made changes as suggested in Change Log 2010-08-16T11:12. However the name change from ‘percentMeasureType’ to ‘PercentMeasureType’ caused an issue because ‘PercentMeasureType’ already exists. Changed the existing ‘percentMeasureType’ to ‘PercentMeasureType’ and the existing ‘PercentMeasureType’ to ‘Valo-100MeasureType’

D.13 Version 1.2.4.g

D.13.1 Change Log 2010-08-26T14:30

Author: David Burggraf

Date/Time: August-26-10 2:30 PM

Schema Version: V1.2.4.g

Topic: referenceEdge, WitsML profile, test instance

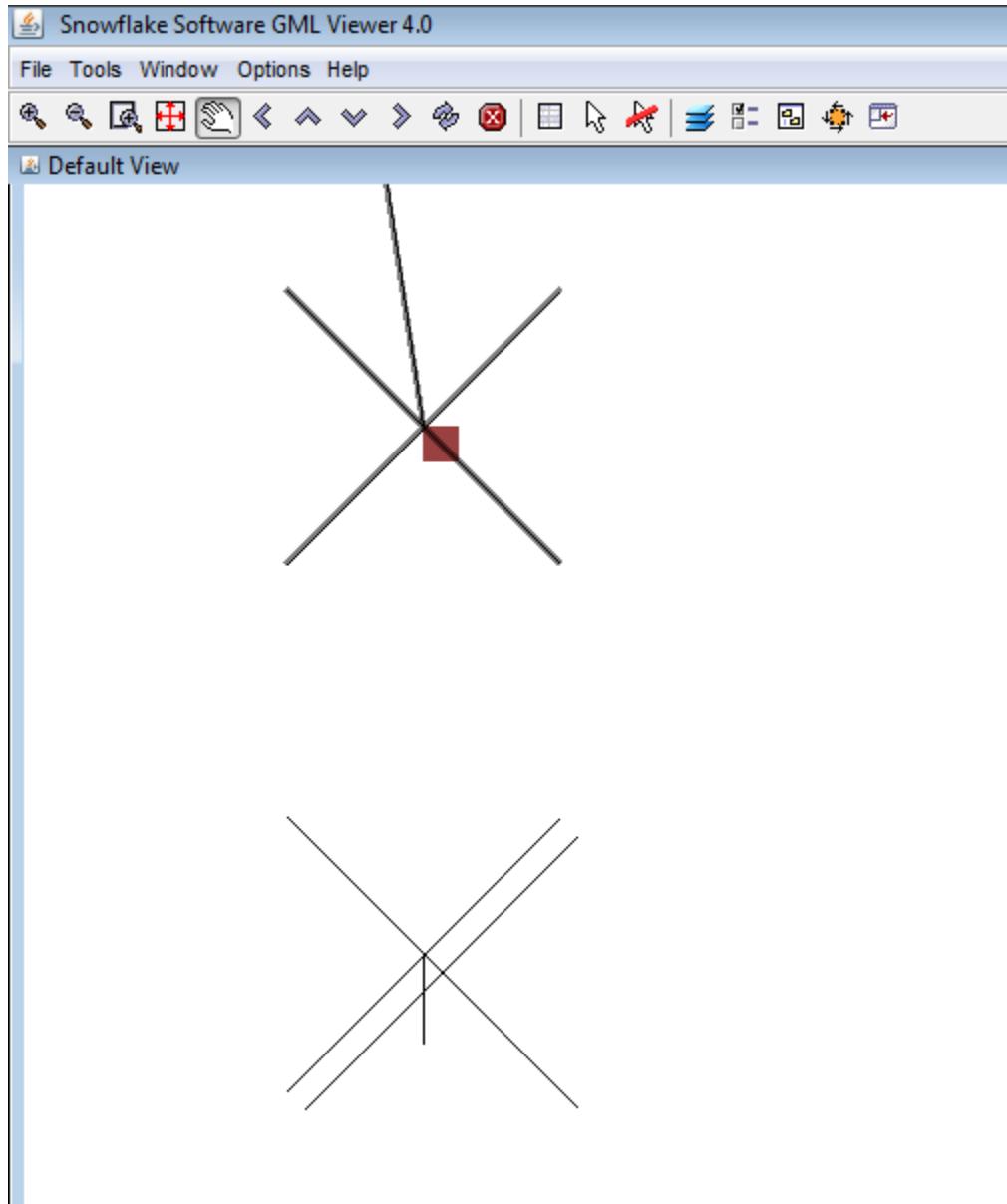
Description:

The following changes were made:

1. Added referenceEdge property to diggs:AbstractPlanarLocationType
2. Removed the 161 unused WitsML schema files
3. Updated testinstance.xml with
 - a. new trench wall instance
 - i. new referenceEdge instance,
 - ii. simple polygon as feature extent
 - b. new urn values (replaced ‘:DIGGSINC’ with ‘:DIGGS’)
 - c. added and corrected srsName values (someref -> epsg 7405)

Note that referenceEdge is mandatory (minOccurs=1), similar to referencePoint. The planar referencing now makes use of the referenceEdge, not the featureExtent:

I checked by viewing the testinstance.xml with the Snowflake GML Viewer 4.0 and the polygon now shows up as shown in the following screenshot (see the red box below).



D.13.2 Change Log 2010-08-26T14:36

Author: David Burggraf
Date/Time: August-26-10 2:36 PM
Schema Version: V1.2.4.g

Topic: DIGGS CRS Dictionary**Description:**

1. Single file instead of multiple files created for GML dictionary and WKT dictionaries:
 - a. CRS\output\DIGGS_GML_CRS_DICTIONARY.xml
 - b. CRS\output\DIGGS_WKT_CRS_DICTIONARY.txt
2. Dictionary Version set to 0.1 on all identifier values in GML dictionary
3. 'DIGGSINC' replaced with 'DIGGS'
4. WKT has no version tag (see documentation
<http://geoapi.sourceforge.net/2.0/javadoc/org/opengis/referencing/doc-files/WKT.html>), so no version has been assigned to WKT defns.

D.14 Version 1.2.4.h**D.14.1 Change Log 2011-01-11T10:52****Author:** Daniel Ponti**Date/Time:** Tuesday, January 11, 2011 10:52 AM**Schema Version:** V1.2.4.h**Topic:** Latest testInstance revision, etc.**Description:**

Revised testInstance.xml as follows:

- 1) Changed the trench wall layer, following the pattern we discussed -with a position property that includes base and top linestrings, and a representative polygon that uses a ring structure. I didn't use the OrientedCurve as it's not available in the current profile, but I ordered the coordinates of the various curve elements to provide a contiguous order for the trench wall layers.
- 2) I added a sample activity and sample using the measuredDepth borehole position so that it can be rendered as a symbol

```
<SamplingActivity gml:id="pointSample">
  <projectRef xlink:href="#p1"/>
  <associatedLocationRef xlink:href="#LB_Webster"/>
  <activityPosition>
    <BoreholePosition gml:id="pt1">
      <measuredDepth>
        <gml:Point gml:id="pt1-1">
          <gml:pos srsName="#sr123" srsDimension="1">30</gml:pos>
        </gml:Point>
      </measuredDepth>
    </BoreholePosition>
  </activityPosition>
</SamplingActivity>
```

```

</measuredDepth>
</BoreholePosition>
</activityPosition>
<samplesProduced>
  <SampleProduced gml:id="ptsp">
    <sampleRef xlink:href="#sampt"/>
    <samplePosition>
      <BoreholePosition gml:id="pdbp">
        <measuredDepth>
          <gml:Point gml:id="pt1-2">
            <gml:pos srsDimension="1" srsName="#sr123">30</gml:pos>
          </gml:Point>
        </measuredDepth>
      </BoreholePosition>
    </samplePosition>
  </SampleProduced>
</samplesProduced>

```

Revised kernel.xsd includes the new elements for encoding positions of layers on trench walls.
The new modifications are:

- 1) added trueTopObserved and trueBaseObserved boolean properties (optional) to the AbstractLocationPosition base class.
- 2) added layerIntersectionPosition property to TrenchWallPositionType as an additional position choice.
- 3) layerIntersectionPosition is of type LayerIntersectionPropertyType that refers to a LayerIntersections object of type LayerIntersectionType. This type inherits from diggs:AbstractObjectType and adds three mandatory properties: a) layerTopIntersection (MultiCurvePropertyType), b) layerBaseIntersection (MultiCurvePropertyType), and c) representativeSurface (1..n; SurfacePropertyType).

D.14.2 Change Log 2011-01-13T09:20

Author: David Burggraf

Date/Time: January-13-11 9:20 AM

Schema Version: V1.2.4.h

Topic: DIGGS KML Output

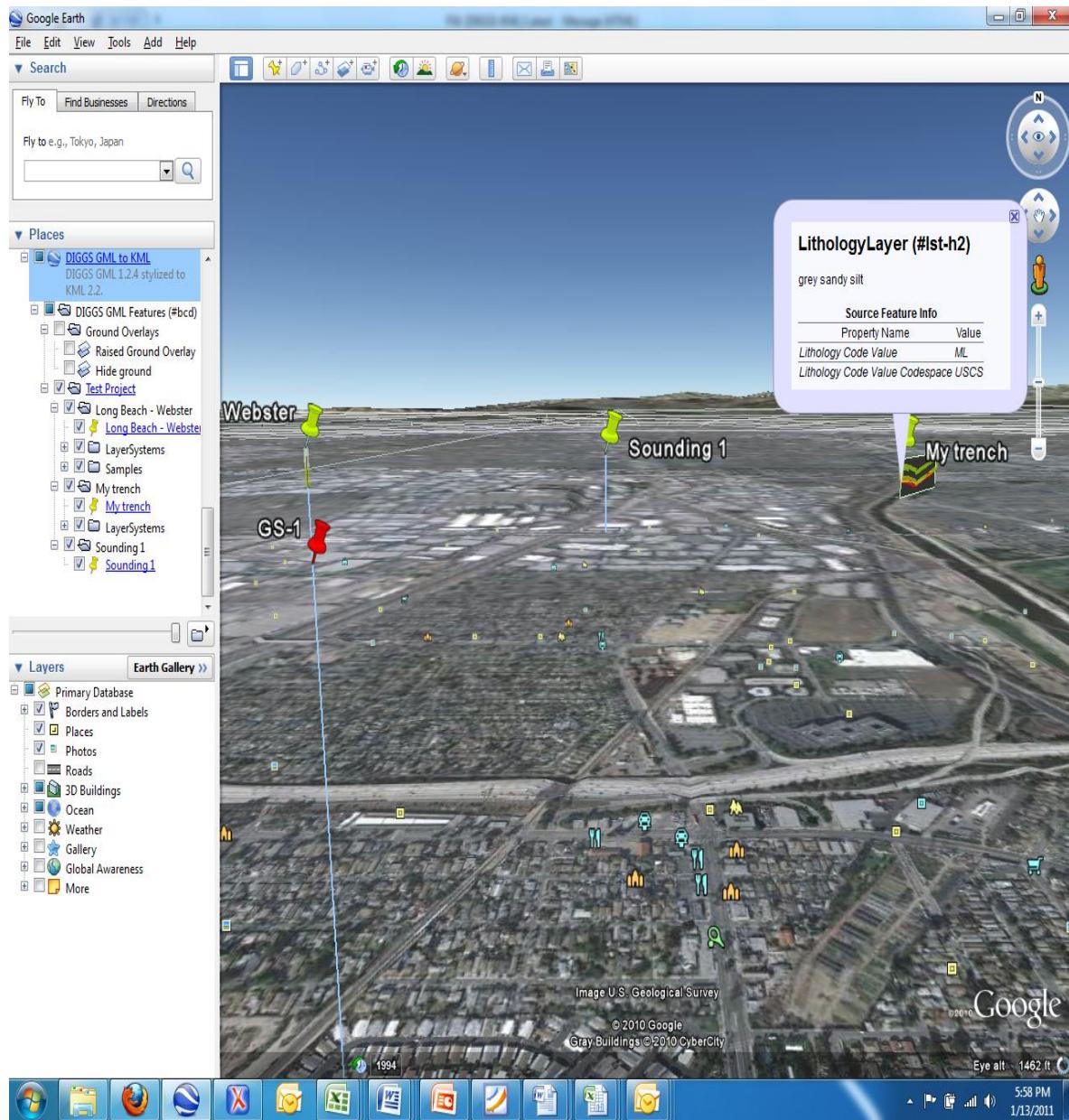
Description:

The KML output was generated for the latest test instance using the KML tool. Linear referencing and planar referencing are both supported in the KML styling. Multiple CRS handling is supported (by GeoTools). The KML CRS WKT has been transcribed from the GML

defn to a WKT now supported by GeoTools as an extension. The KML WKT is as follows (note that the vertical datum begins at the WGS 84 Geoid:

```
COMPD_CS["urn:ogc:def:crs:OGC:LonLat84_5773",
GEOGCS["WGS 84",
  DATUM["World Geodetic System 1984",
    SPHEROID["WGS 84", 6378137.0, 298.257223563, AUTHORITY["EPSG","7030"]],
    AUTHORITY["EPSG","6326"]],
  PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]],
  UNIT["degree", 0.017453292519943295],
  AXIS["Geodetic longitude", EAST],
  AXIS["Geodetic latitude", NORTH],
  AUTHORITY["EPSG","4326"]],
  VERT_CS["EGM96 geoid height",
    VERT_DATUM["EGM96 geoid", 2005, AUTHORITY["EPSG","5171"]],
    UNIT["m", 1.0],
    AXIS["Gravity-related height", UP],
    AUTHORITY["EPSG","5773"]]]
```

In particular, the DIGGS SampleActivity and Sample are supported. Here's a screen shot:



D.14.3 Change Log 2011-02-11T14:32

Author: David Burggraf

Date/Time: February-08-11 2:32 PM

Schema Version: 1.2.4.i

Topic: Revised GML Profile, testInstance.xml with support for OrientableCurve.

Description:

The schemas were updated with a revised GML3.2 profile and testInstance.xml instance that makes use of OrientableCurve (see excerpt in bold copied below).

```

<representativeSurface>
  <gml:Polygon gml:id="lt1-p">
    <gml:exterior>
      <gml:Ring>
        <gml:curveMember xlink:href="#twp-int-11"/>
        <!--Top edge -->
        <gml:curveMember>
          <gml:LineString gml:id="lt1-1">
            <!--Right edge -->
            <gml:posList srsDimension="2" srsName="#lsrs002">50 1 50
              4</gml:posList>
            </gml:LineString>
          </gml:curveMember>
          <gml:curveMember>
            <gml:OrientableCurve gml:id="OC_twp-int-12" orientation="-">
              <gml:baseCurve xlink:href="#twp-int-12"/>
            </gml:OrientableCurve>
          </gml:curveMember>
        <!--Basal edge in reverse orientation -->
        <gml:curveMember>
          <gml:LineString gml:id="lt1-2">
            <!--Left edge -->
            <gml:posList srsDimension="2" srsName="#lsrs002">0 5 0 2</gml:posList>
          </gml:LineString>
        </gml:curveMember>
      </gml:Ring>
    </gml:exterior>
  </gml:Polygon>
</representativeSurface>

```

D.15 Version 1.2.4.j

D.15.1 Change Log 2011-05-12T22:36

Author: Daniel Ponti

Date/Time: May-12-11 10:36 PM

Schema Version: 1.2.4.j

Topic: Tests and Layer changes.

Description:

Changes are primarily for the tests, but there are also some layer changes as well. The main change for the tests is that we're more closely following the O&M paradigm, but essentially generalizing the coverage model we used for CPT tests and geophysical logs to all kinds of measurements. One issue that's come up is how to handle temporally varying measurement info

at both one location as well as a domain of locations (I think we need to add back into our profile more of the gml:Covverage model), as well as handling other tabular types of results that are indexed on other types of informaiton (pressure cycles, etc.)

D.15.2 Change Log 2011-05-12T09:15

Author: David Burggraf

Date/Time: May-12-11 9:15 AM

Schema Version: 1.2.4.j

Topic: Linear Referencing

Description:

Some issues were identified in GML 3.3 linear referencing structure. Specifically the offset plane, being perpendicular to the line string, is discontinuous across angle breaks in the line string. As a result we have now adopted the vector offset approach. The latest changes due to the GML 3.3 Vector Offset developments are contained in glrovProfile_diggs.xsd and are now supported in testInstance.xml and both the Excel and KML tools.

D.15.3 Change Log 2011-05-19T07:15

Author: David Burggraf

Date/Time: May-19-11 7:15 AM

Schema Version: 1.2.4.j

Topic: Enumerated lists

Description:

The original criterion for selecting the codelists was that type="gml:CodeType", and not a based on a string enumeration. All of the string enumerations have now been extracted from the DIGGS 1.2.4.j schemas (in addition to the gml:CodeTypes) and the generated output is in the third 'Enumerations' worksheet of /CodeLists/SchemaV1.24CodeTypes.xlsx.

D.16 Version 1.2.4.k

D.16.1 Change Log 2011-05-19T07:58

Author: Daniel Ponti

Date/Time: May-19-11 7:54 AM

Schema Version: 1.2.4.k**Topic:** Test handling**Description:**

This version implements a proposed change in how DIGGS handles tests:

The top level test features in prior versions have been replaced by a single Measurement feature. This feature is patterned after the OGC Observation feature (but doesn't derive from it). It is defined as an event, whose results are estimates of the value of a property or properties of interest at a position or series of positions on or within the earth. The feature has two main properties: 1) outcome, and 2) procedure. The outcome property contains a MeasurementResult object that derives from a GML coverage feature. This feature contains both the position information for the measurement (eg. where the measurement applies), that derives from the coverage domain, and a results property (derives from the coverage range) that contains information about the property(ies) derived from the measurement, and the value(s) of those properties. The procedure property contains zero to one test object, that describes the metadata for the test procedure and any interim results of value. The existing test features in prior versions of DIGGS are being converted to these test objects - and contain the same parameter properties as before, minus the properties that are the reported test results.

The Geotechnical.xsd schema in this current version only contains a few test oprocedures. Loren is working on converting the others and making sure they are mappable to the AGS 4.0 data dictionary. The file Geotecnical-old.xsd is the prior (1.2.4.j) version.

A new test instance document: testinstance-newMeasurements.xml, shows how the new Measurement structure works for both the prior tests in that document, and a few others.

Advantages/Rationale for the change:

- 1) All "results" of test (eg. soil, chemical, hydrologic properties) occur at a single place in the schema. In this encoding scheme, the "result" is what matters and can be easily extracted from an instance document.
- 2) Measurement results can be reported without the need to instantiate a fake sample or test feature for legacy data where only the result and position are known. While the procedure property is a mandatory part of the Measurement feature, it need not be populated by a measurement object.
- 3) All results are encoded in the same consistent fashion - results of in-situ tests, CPT, geophysical logs, lab tests on samples, etc.
- 4) Provides a more flexible and practice-friendly means of associating results with test procedures and sampling features.

- 4) Provides a basis for developing a parallel structure for time-varying observations (eg. monitoring).

Disadvantages:

- 1) Currently uses a generic property object to describe the property being measured - this means that the property type must come from a controlled list (dictionary) to maintain interoperability.
- 2) Some schema control is lost (more dependence on schematron to ensure that an instance document makes sense).
- 3) GML coverage encoding for all measurement results is not as human readable and somewhat more complex to parse.

However, current DIGGS is already saddled with these problems w/respect to how CPT and geophysical logs need to be encoded and will have similar issues with any kind of test and monitoring results that are tabular in form, so we don't see these issues being a significant detriment to this plan.

In addition, 1.2.4.k makes some relatively minor changes to the layer system structure and in particular the lithology layer and lithology objects to better accommodate US DOT practice.

D.17 Version 2.0a

D.17.1 Change Log 2012-02-10T12:55

Author: Daniel Ponti

Date/Time: February-10-12 12:55 PM

Schema Version: 2.0a

Topic: Migration from V1.2.4.k to 2.0a

Description:

Changes:

- 1) Changed name of Location and all of its siblings to SamplingFeature. This normalizes it more with O&M and lets us use the term location to mean what it normally means in the English language. This mostly only affects abstract elements and base types - there are some but not a lot of impacts to the instances.
- 2) Removed monitoringInstallations as a top level property; replaced with the Monitoring structure, which derives from AbstractMeasurement

- 3) Added investigationTargets as a top-level property and built AbstractInvestigationTarget base type for these targets (equivalent to OM feature-of-interest). The base type right now is the same as diggs:AbstractFeature. This will be the category of features that are the "targets" for DIGGS data - eg. ground investigations, embankments, piles, roadways, etc.
- 4) Created an element and associated base types for Ground, which extends AbstractInvestigationTarget (but adds no additional properties). It is currently the only concrete investigation target in this version of DIGGS.
- 5) Added mandatory investigationTargetRef be identified for every sampling feature, measurement (eg. test and monitoring), and sampling activity - DIGGS features where the locationRef is not mandatory.
- 6) Added Well as a new concrete SamplingFeature.
- 7) Created two additional sampling features: Transect and Face, to represent a generic linear and planar sampling feature (such as a linear transect or outcrop). These use the abstract type without extensions.
- 8) Removed AbstractLocation and type - not needed - diggs:AbstractGeometry contains all needed elements;
- 9) Deleted LayerIntervalType - not needed any longer.
- 10) Removed ValueAtTime, ValueAtDepthbyTime and ValueAtType elements and properties - not needed.
- 11) Eliminated most comments except for CDATA and DSB profile restrictions.
- 12) Changed a number of gml: geometries to their diggs-derived equivalents. Did not do this for referenceEdge, featureExtent, and relativeFeatureBoundary of AbstractPlanarSamplingFeature. Probably should, but want to consult with David first.
- 13) Reduced the number of diggs geometry features to eliminate redundancy (eg. DepthInterval and CurveLocation merged into a single LinearExtent feature)
- 14) Abstracted Measurement and defined a base type (AbstractMeasurement) that holds all of the required and optional references that a measurement might have. Renamed Measurement to Test to be used for measurements that occur at a single time instant or within an interval of time but where time is not an integral part of the result (eg. the position indexed results). Created a Monitoring feature that would be used for measurements taken at a non-varying position or location, but where the results are time-indexed. This covers the end members for measurements.

- 15) For Monitoring, although it parallels the Test structure, there is no real procedure. I replaced the procedure property with a process property that contains the diggs metadata property group to allow recording of specifications and equipment. However, there is some redundancy here; as the Property object of a measurement also contains a property called detectorRef that can reference equipment used to record the specific property. So, should we limit procedure to only reference or describe a specification?
- 16) Created ChemicalAnalysis in the environmental namespace as a test procedure that can store info for chemical tests (such as sample volume, type, etc.) and also modified SpectralAnalysis to extend ChemicalAnalysis. We need to discuss this with the environmental SIG and see if there shouldn't be some other concrete procedures for these chemical tests.
- 17) Removed WaterLevelReading and associated types in Environmental and converted the properties here to result properties, or added to the Property object to be used within the Monitoring structure. There are a couple of properties that either seemed redundant or I didn't understand that I did not transfer. We need to talk with folks in Environmental to decide what to do with these:
 - a) captureQualifier - no clue what this means,
 - b) type property for the reading - unclear what this is supposed to represent - if it is method specific this is handled in the procedure.
 - c)

There are some tradeoffs here where some types of recorded info for fluid levels may result in repetition of values in the data block (eg. like fluid_type) and by measurement-specific reporting constraints. This can be avoided somewhat by expanding the result property definitions we support, but this is likely to get unwieldy. We can discuss this a bit so I can fully explain.

There's still some things in Environmental I don't yet know what to do with - eg Filtration, Purge, TICResult. Filtration sits on its own, and probably should be melded into the samplePreparation structure somehow. Purge is a procedure that might work within a monitoring measurement. How is Purge different from the Pumping test in the Geotechnical schema? TICResult (I think this is total organic carbon), should go into the set of property classes for measurement results, but maybe there should be another test procedure for this? Again, I think we need to consult with the Environmental SIG.
- 18) Commented out a lot of code in Monitoring and Environmental that is no longer needed. I left water level measurements in Environmental for now, but this would

probably again morph into a monitoring measurement with the procedure or detector being a water level test or something like that.

- 19) Fixed a few odds and ends to make sure everything validated.
- 20) Changed the testInstance to validate against the new changes.
- 21) Eliminated the Monitoring schema - with the monitoring feature in kernel, it is no longer needed.

Still need

- 1) Proper typing for the Monitoring results and Time objects - how best to construct this we can leave for David.
- 2) Cleaning up witsml units and base types - incorporating all within the diggs namespace and putting all simple types and unit/enumeration types into a separate .xsd document (David)
- 3) Additional annotations (Dan and Loren)
- 4) Checking dependencies and eliminating orphaned elements and types (David).

D.17.2 Change Log 2012-03-01T14:04

Author: David Burggraf

Date/Time: Thursday, March 01, 2012 2:04 PM

Schema Version: 2.0a

Topic: Property Info

Description:

Created an xslt script according to the mapping rules specified by Dan and ran it on the input .xml file. The input .xml file was derived from a witsml dictionary of log properties, along with their descriptions. The XML file was transformed into an Excel table using the following mapping:

XML input	Spreadsheet output
name	Property Name
description	Description

name	Code (with underscore characters '_' in name value replaced by single white space in Code value)
------	--

The two output formats are: Excel XLS and the more generic CSV.

D.17.3 Change Log 2012-03-14T00:44

Author: David Burggraf

Date/Time: Wednesday, March 14, 2012 12:44 AM

Schema Version: 2.0a

Topic: Review of 2.0a draft

Description:

The following changes were made in this version:

1. Linear referencing finalization to be released as GML 3.3.1 corrigendum
2. Updated schema namespace to V2.0a
3. Updated testInstance.xml to validate against schema changes

Appendix E. DIGGS Meeting Notes

E.1 Teleconference Meeting Notes 2010-01-06

Date: January 6, 2010

Time: 7:30 AM – 8:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray
Fiona Bruce

Agenda: Continue our discussion from last month on the domain-independent issues, but bring Chris Bray into the discussion. The one particular issue that we wanted to discuss with Chris had to do with the organization of the schemas and the use of namespaces. I've included the notes from that last meeting below.

Notes:

- Scripts for object-property rule conversion:
 - David reported that they are currently writing the scripts to deal with the object-property issue and are almost ready to run the scripts on all the DIGGS schemas.
 - Dan requested that David email one of the converted schemas files to him when completed (if that doesn't require extra work) to see the difference in structure when viewed in Altova or Stylus.
- Namespace issue:
 - We reviewed the discussion from the prior meeting regarding the namespace and schema organization issue. We reviewed the two options that were considered:
 - Option (1): Single namespace with 5 schema files (Kernel, Geotechnical, Environmental, Piling, Monitoring). Some reorganization of elements might be needed.
 - Option (2): 5 namespaces and 5 schema files.
 - Dan cited issues with the use of multiple namespaces, namely that substitution groups and use of explicit namespace prefixes are not handled uniformly between XML parsers.
 - David explained the use of explicit namespace prefixes and common conventions:
 - Default namespace doesn't need the prefix and is usually the application namespace that occurs most frequently.
 - In a DIGGS instance file the DIGGS namespace will likely be the default namespace.
- Meetings in the future:
 - David will email a weekly status update on Monday/Tuesday of each week.
 - David, Dan, Chris, Loren will tentatively schedule a weekly teleconference meeting for Thursday mornings 7:30 AM (PST) to follow up on issues identified in the weekly report.
 - Loren will set up the meetings and WebEx and will send meeting invitations.

E.2 Teleconference Meeting Notes 2010-01-12

Date: January 12, 2010

Time: 7:30 AM – 8:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Fiona Bruce
Scott Deaton
Scot Weaver
Salvatore Caronna
Chris Bray

Not Available: Roger Chandler
Tim Spink
Cliff Roblee
Jean Benoit
John Bobbitt
Marc Hoit

Agenda: This purpose of this meeting is to brief the Core SIG on the recently executed contract with Galdos to carry out work under Tasks 2 & 4, and discuss the initial strategies for meetings.

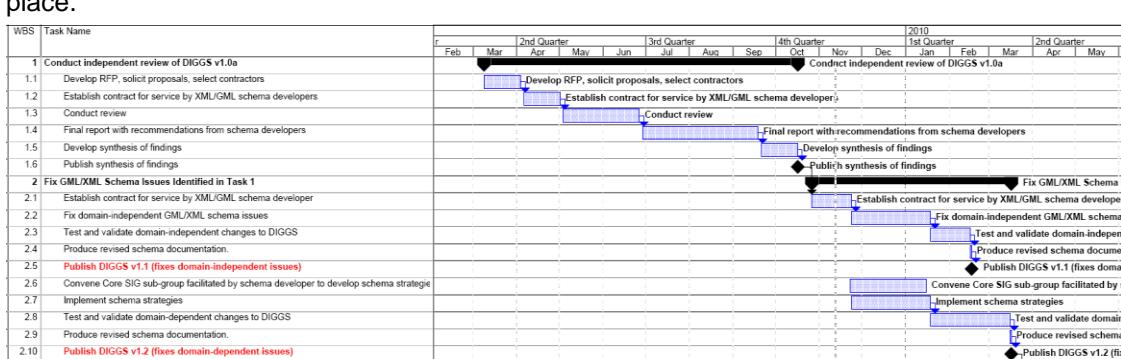
Notes:

- Contract with Galdos was executed on December 22, 2009 to carry out work under Tasks 2 & 4.
 - Domain Independent Schema Work, leading to release of **DIGGS v1.1**.
 - GML “Object-Property” Patterning
 - Update to GML 3.2
 - Inheritance of Objects
 - Use of GML Profiles
 - Organization of DIGGS Schemas
 - Relative Referencing of Schemas
 - Domain Dependent Schema Work, leading to release of **DIGGS v1.2**.
 - Code Tables
 - Key Fields
 - Table Data
- Galdos is proceeding with the domain-independent work right now.
- Ponti and Turner held telecon with Galdos last week to discuss domain-independent work and provided some initial guidance to get them started.
- Regarding the Galdos contract and involvement of the Core SIG, Turner proposed, and Core SIG concurred:
 - Core SIG participates in the kickoff meeting, and the meetings prior to the major DIGGS releases (v1.1, v1.2, and v2.0)

- For the technical schema meetings with Galdos on Task 2 issues, a smaller technical group from the Core SIG works with Galdos more closely, and bring issues to the Core SIG as needed.
- Technical group includes Bray, Ponti, and Turner.
- Turner acts as a coordinator/organizer for the work, bringing design issues to the Core SIG as needed, and communicating to the Core SIG on progress over the course of the contract.
- Established 3 Technical Sub-Groups to work with Galdos on specific domain dependent issues.
 - Work is conducted in parallel to the domain independent work.
 - Begin work this week.
 - Team members:

Technical Sub-Group	Members	Initial Meeting
Key Fields	Salvatore Caronna Scott Deaton Dan Ponti Chris Bray Loren Turner David Burggraf	1/14/10, 8:30 am (PST)
Table Data	Scott Deaton Dan Ponti Chris Bray Loren Turner David Burggraf	1/21/10, 8:30 am (PST)
Code Tables	Tim Spink (to be confirmed) Dan Ponti Chris Bray Loren Turner David Burggraf	1/28/10, 8:30 am (PST) (to be confirmed)

- Technical Sub-Group logistics:
 - 3 scheduled telecom/WebEx meetings per group to discuss specific issue.
 - Turner will make meeting arrangements.
 - Meetings will be scheduled generally on a Thursday 8:30 am (PST).
 - Meetings may last 1-3 hours.
 - Other Core SIG members may join these sub-group meetings if interested.
- Project timeline will be modified to account for 1 month delay in getting Tasks 2 & 4 contracts in place.



E.3 Teleconference Meeting Notes 2010-01-14T07:30

January 14, 2010

Time: 7:30 AM – 8:30 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Chris Bray

Not Available:

Agenda: Weekly status meeting.

Notes:

- David summarized the work for this week resulting in a revised self-contained draft of the DIGGS schemas (emailed to us yesterday). Revisions:
 - Using uppercase for the first letter of each schema file name
 - Adding the Geophysical.xsd import to the Complete.xsd
 - Addressing the Object/Property rule issues (still subject to testing).
 - Used relative paths for schemaLocations
 - The schemas are currently divided into 6 namespaces as follows:
 - xmlns:diggs="http://schemas.diggsml.com/1.0a"
 - xmlns:diggs_geo="http://schemas.diggsml.com/1.0a/geotechnical"
 - xmlns:diggs_env="http://schemas.diggsml.com/1.0a/environmental"
 - xmlns:diggs_mon="http://schemas.diggsml.com/1.0a/monitoring"
 - xmlns:diggs_pil="http://schemas.diggsml.com/1.0a/piling"
 - xmlns:diggs_gph="http://schemas.diggsml.com/1.0a/geophysical"
- Chris confirmed that Geometry.xsd schema was no longer needed and could be completely removed.
- David will create an instance file using the revised schemas for further testing.
- Dan noted that the Geophysical.xsd will need more work following the “table data” discussion next week.
- The namespace issue was discussed at length.
 - Goals:
 - DIGGS not be made more complex than necessary
 - Retain modularity
 - Software vendors may only support specific DIGGS components, not the entire standard. Need a way to accommodate this – ideas include:
 - Retain multiple namespaces as is the case currently
 - Create DIGGS profiles that limit application to specific domains.
 - Considerations for using a single namespace:
 - Fewer include statements in instance documents
 - Reduces issues with certain XML parsers resolving namespaces
 - GML and WITSML have been deployed under a single namespace

- New versions will require updates for users, even though the schemas they use may not have been affected.
- Considerations for using multiple namespaces:
 - DIGGS is a complex schema and warrants it
 - GML is moving to multiple namespaces
 - Distinguishes some elements with similar names, for example “sample” currently takes on different meanings in the different namespaces. (Ponti noted that we should reconsider the concept of “sample” to take on a more global meaning.)
- Path forward at this time:
 - Adopt the six namespaces suggested by Burggraf:
 - xmlns:diggs="http://schemas.diggsml.com/1.0a"
 - xmlns:diggs_geo="http://schemas.diggsml.com/1.0a/geotechnical"
 - xmlns:diggs_env="http://schemas.diggsml.com/1.0a/environmental"
 - xmlns:diggs_mon="http://schemas.diggsml.com/1.0a/monitoring"
 - xmlns:diggs_pil="http://schemas.diggsml.com/1.0a/piling"
 - xmlns:diggs_gph="http://schemas.diggsml.com/1.0a/geophysical"
 - Have separate meeting with Turner, Ponti, and Bray to identify elements such as “hole,” that may be more appropriate to move under Kernel.xsd.

E.4 Teleconference Meeting Notes 2010-01-14T08:30

Date: January 14, 2010

Time: 8:30 AM – 9:30 AM (PST)

Participants: Loren Turner

Dan Ponti

David Burggraf

Chris Bray

Salvatore Caronna

Not Available: Scott Deaton

Agenda: Discussion of “Key Fields” issue.

Notes:

- The “key fields” issue is not an issue that can be addressed in the schema alone.
- The schema is already set up to accommodate the use the key fields, however, rules on how those keys are assigned, applied, and used need to be documented.

- Validation of the keys will likely need to be handled using a combination of schema validation and/or external software tools (e.g. schematron, java, web service, etc.).
- There's a need to establish business rules.
 - Ponti suggested setting up a special team to work on this.
 - Caronna volunteered to lead this team. **Action items:**
 - Enlist help from Scott Deaton, Roger Chandler, Scot Weaver, and Derrick Dasenbrock
 - Draft up a document describing what the team will do, and send draft to Turner this week or next before sending to the team.
 - Carry out the team's work over the next few weeks.
 - Communicate the team's recommendation back to our group (Burggraf, Turner, Ponti, Bray)
 - Will document the proposed business rules in a non-technical manner.
 - Business rules could govern:
 - Positional data (e.g. top depth higher than bottom depth, hole position is within project limits)
 - Test data (e.g. Plasticity Index value is a positive integer between 0 and 100)
 - Formats and standards for key fields and identifiers to assure compliance with database-driven applications such as gINT and Holebase
 - Rules should insure that IDs are unique
 - Other types of data?
 - Not clear how detailed we want to get with application of business rules.
 - At minimum, the rules should be restrictive enough to insure that the software vendors can import/export DIGGS as easily as possible.
- Discussion about IDs:
 - With GML 3.2, IDs are mandatory for all objects.
 - GML IDs must be unique within an instance document.
 - DIGGS identifiers are unique within the DIGGS community:
 - Uses unique 3-alpha minimum organization identifier
 - Each organization insures uniqueness of ID.
 - The organization's ID assures uniqueness in the community
 - The GML ID can be used to replace the DIGGS ID, while applying the DIGGS restrictions in the GML profile.
 - **Action Item** – David will remove DIGGS ID element and apply the DIGGS restrictions to the GML ID in the profile.

E.5 Teleconference Meeting Notes 2010-01-28T07:30

Date: January 28, 2010

Time: 7:30 AM – 9:00 AM (PST)

Participants: Loren Turner
Dan Ponti

David Burggraf
Chris Bray

Not Available:

Agenda: Weekly status meeting.

Notes:

- Burggraf summarized the work for the past two weeks and explained the revised schemas emailed to the group on 1/26/10 (DIGGSdraft0.3.zip).

GeneratedInstances	File Folder	1/28/2010 7:35 AM	
witsml	File Folder	1/28/2010 7:36 AM	
DIGGS_spyProject.spp	1 KB	Altova XMLSpy Project	1/26/2010 1:41 PM
Diggs1ns.xsd	1,788 KB	W3C XML Schema	1/24/2010 4:24 PM
Environmental.xsd	138 KB	W3C XML Schema	1/22/2010 10:16 AM
Geotechnical.xsd	844 KB	W3C XML Schema	1/22/2010 10:16 AM
qml4diggs.xsd	160 KB	W3C XML Schema	1/22/2010 9:45 AM
qml4diggsNoCRS.xsd	72 KB	W3C XML Schema	1/24/2010 4:34 PM
Kernel.xsd	369 KB	W3C XML Schema	1/22/2010 10:17 AM
Monitoring.xsd	41 KB	W3C XML Schema	1/22/2010 10:16 AM
Piling.xsd	416 KB	W3C XML Schema	1/22/2010 10:16 AM
xlinks.xsd	6 KB	W3C XML Schema	6/27/2009 11:11 AM
xml.xsd	9 KB	W3C XML Schema	6/27/2009 7:16 PM

- Burggraf developed a single namespace version of the DIGGS schema (Diggs1ns.xsd) and created and instance document from this. He reported that the schema validated in seconds.
- The group decided at this time to continue with the multiple namespace approach to facilitate development of schemas.
- Discussed the issue of moving elements within the various namespaces. (This may result in a significant structural change to the organization of DIGGS elements.)
 - Intent is to reduce the number of namespaces needed to be included in an instance document.
 - Typical instance document should have 2 namespaces – likely the “Core” plus one other.
 - Ponti suggested that all elements related to location be moved to the kernel schema.
 - The elements “where you observe things” or “ where you collect things” should be in the kernel.
 - Separate the observational information from the location (e.g. hole, trial pit, exposure) where it was observed.
 - Example – “fracture spacing” is a geotechnical feature that can be observed in boreholes, but also in trial pits or on exposures on a rock face. Fracture spacing therefore not unique to the hole element in the geotechnical schema.
 - Bray explained that current schema uses abstract types (?) to facilitate inheriting elements like fracture type in other location types.
 - Only modify the observation (e.g. fracture spacing) in one place in the schema.
 - User ignores the element if they don’t use it.
 - DIGGS v1.0a elements are related by a mix of hierarchical constructs (parent-child) and referencing (xlink:href and id tags).
 - Not clear if the mix of these approaches is good.
 - Burggraf believes that either approach can be used successfully.
 - ACTION ITEM:**
 - Need to have meeting to resolve fundamental issue of schema organization.
Consider questions:
 - Continue development with current schema structure?

- Revise structure to make more use of element ids and referencing?
- Pros/cons of each?
- What's better for the software vendors with respect to implementation?
- What approach is more conducive to mapping to/from databases?
- Extensibility into future?
- Complexity?
- Validation?
- Other considerations?
- Turner organize meeting of the 4 software vendors in the next 2 or 3 weeks:
 - Keynetix
 - gINT
 - Dataforensics
 - Earthsoft
- Ponti and Bray each prepare a presentation to provide context for discussion.
- Burggraf brought up issue of coordinate systems.
 - Bray said that what's in DIGGS right now was there prior to him working on it. No issues with changing it.
 - Ponti suggests that Burggraf recommend how to handle this.
 - Group will discuss this further next week.
- Burggraf summarized progress:
 - Anticipates conversion of the schemas to GML 3.2 prior to the next meeting.
 - Need to discuss validation testing plans after 3.2 conversion.
 - Most domain-independent issues are addressed now.
 - CRS issue could be resolved by next Thursday.

E.6 Teleconference Meeting Notes 2010-01-28T09:00

Date: January 28, 2010

Time: 9:00 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Chris Bray
Scott Deaton

Not Available: n/a

Agenda: Discussion of “table data” handling in DIGGS.

Notes:

- Team reviewed the various options of dealing with table data throughout the DIGGS schema. Bray explained the approach used for DIGGS to date. We reviewed the approaches and options

presented in the Compusult report. Ponti presented an approach used in the cosmosDIGGS schema for CPT data. Three primary options were generally discussed:

- All data in a “block” within a single XML element (DIGGS V1.0a and WITSML).
- Data “rows” captured in separate elements (cosmosDIGGS).
- Data “columns” captured in separate elements (Compusult recommendation)
- Bray reminded group that he had written a series of articles posted to the DIGGSml.com website on the subject of implementing table data. It would be valuable for the team to take a look at these again to gain some context for why the current DIGGS structure is implemented as it is. You can find the articles posted here:
 - <http://www.diggsml.com/monitoring-and-samplingpoint-object-part-3>
 - <http://www.diggsml.com/monitoring-and-samplingpoint-object-part-4>
- Bray’s “part 4” article summarizes the DIGGS implementation:

“It was decided that DIGGSMLs implementation should follow the WITSML implementation as much as possible, but it should also be compatible with SensorML, so a number of objects were renamed to coincide with SensorML, with a view to only having to learn one lexicon when representing monitoring data in XML, some properties were also added to ease this compatibility.”
- Deaton explained that vendors will need to know the specific “names” for the columns in the data block in order to parse the data.
- Conclusions from discussion on table data:
 - Retain the existing table structure currently implemented in DIGGS.
 - Consider restructuring the handling of geophysical data to be consistent with the rest of DIGGS so that all table data is uniformly captured. This might require a departure from WITSML as is currently implemented.
 - DIGGS will need to rely upon schematron to validate the block data.
 - Need to resolve the issue of defining canonical values in codelists so that software vendors can effectively parse table data and clearly identify the meaning of the “columns” of data.
- Discussed the issue of codelists further, since the codelists are needed in order to identify the data within the table data block.
 - DIGGS currently uses the *GML Dictionary* to manage codelists. For example, in DIGGS V1.0a a dictionary file (agsCodeList_V1.xml) is provided and lists all the types of measurements one might do for CPT. The terms are specific to UK-AGS practice. US, French, or other region’s practices may use different terms. How does a software vendor discern, for example, what “tip resistance” is in the French codelist?
 - Codelists can be maintained on any server by any data generator. DIGGS instances can refer to those “external” codelists.
 - Possible solution:
 - Use method other than GML dictionary.
 - Define canonical (standard set) values.

- Implement XML Classification Scheme/Node that supports external identifiers. Burggraf said he can generate this automatically if provided with the various codelists (“havest” the codelists).
- **ACTIONS:**
 - Identify “next steps” in the next meeting.

E.7 Teleconference Meeting Notes 2010-02-04

Date: February 4, 2010

Time: 7:30 AM – 9:15 AM (PST)

Participants: Loren Turner
David Burggraf
Chris Bray

Not Available: Dan Ponti
Scott Deaton

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- David updated us on progress this past week:
 - GML namespace changed from “<http://www.opengis.net/gml>” to “<http://www.opengis.net/gml/3.2>”
 - The derivation from gml:AbstractFeatureCollectionType is removed from the diggs:DiggsType, which is now derived from gml:AbstractFeatureType with “diggs:featureMember” property elements. This follows the GML 3.2 feature collection pattern.
 - All abstract elements with names starting with “_*” are changed to the names “Abstract*”. All the references to these abstract elements are accordingly changed to the names “Abstract*”
 - All components from GML 3.1 that were deprecated are removed from the GML 3.2 profile (e.g. gml:coordinates element).
 - gml:pos and gml posList has replaced gml:coordinates
 - New DIGGS profile forces use of gml:pos and gml posList.
 - gml:coordinates had supported custom delimiters, but that's been eliminated.
 - gml:pos and gml posList forces use of standard delimiters.
- Discussed new **gml:id** attribute and **gml:identifier** element:
 - Every feature and geometry has to have a gml:id attribute.


```
<gml:LineString gml:id="LS001" rsName="urn:ogc:def:crs:EPSG::4326">
  <gml:pos>0 0</gml:pos>
  <gml:pos>1 1</gml:pos>
</gml:LineString>
```
 - Wasn't required in GML 3.1.

- Will notice in data instances.
- Does not have to be globally unique.
- DIGGS id pattern is currently applied.
- **gml:id attribute** is mandatory on all objects that derive from gml abstract types.
 - May change between different generations of the same file
 - Does not need to be unique globally
- **gml:identifier element** used to identify holes
 - Doesn't change
 - Unique
- Implementation:
 - Where we had used diggs:id before, replace with gml:identifier.
 - gml:identifier is used for the main digs objects (e.g. project, hole, sample, test)
 - gml:identifier has pattern restrictions
 - Diggs objects with gml:identifier should also have a corresponding gml:id.
Recommend that the same name for the id be used.
 - gml:id is used for all referencing within document, linking various objects
 - gml:id is used for database handles
 - gml:id has no pattern restrictions
- Codelists
 - Burggraf recommends adopting Registry Information Model (RIM) coding from OASIS.
Example of this was provided in email status update.
 - Similar to current codelist implementation.
 - DIGGS would host a codelist using XML schema/nodes file that establishes a single name for something as well as lists the alternate descriptions for it (e.g. in different languages) and the other names for that same thing.
 - For DIGGS, we would have this single file hosted on DIGGS website:
 - Use version system and release corresponding to the version of DIGGS.
 - The file would establish the primary DIGGS names of everything in the current codelists.
 - We would define the alternate names for the same thing to satisfy UK-AGS and US practices in this first release.
 - This allows software vendors to claim DIGGS compliance with a specific version corresponding to a specific codelist.
- Next steps:
 - Recast DIGGS examples into new schemas (20 examples)
 - Next week, evaluate the examples and make sure they validate.
 - More work on inheritance – after reviewing examples.
 - Revisit discussion on codelists with Ponti and Deaton.

E.8 Teleconference Meeting Notes 2010-02-11

Date: February 11, 2010

Time: 7:30 AM – 9:55 AM (PST)

Participants: Loren Turner

David Burggraf
 Chris Bray
 Dan Ponti
 Scott Deaton

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Continuation of codelist discussion:
 - Team reached consensus on implementing Registry Information Model (RIM) approach for codelists.
 - A single RIM xml file will contain all codes for DIGGS. This file will have a version, consistent with DIGGS release, and will be hosted on the DIGGS site with the schema files.
 - The file will be needed in order to validate DIGGS instances, similar to the schema.
 - The RIM codelist will contain single canonical values defined as the ClassificationNode (e.g. "tip"). Multiple descriptions are supported for different languages or domains. ExternalIdentifiers are used to identify alternative names for the ClassificationNode (e.g. "tipResistance" or "coneResistance").

```
<rim:ClassificationNode id="urn:x-diggs:def:code-list:status:a" code="tip"
  parent="urn:x-diggs:def:code-list:status">
  <rim:Name>
    <rim:LocalizedString xml:lang="en" value="A" />
  </rim:Name>
  <rim:Description>
    <rim:LocalizedString xml:lang="en" value="ACCEPT" />
    <rim:LocalizedString xml:lang="fr" value="ACCEPT" />
    <rim:LocalizedString xml:lang="es" value="ACEPTAR" />
  </rim:Description>
  <rim:ExternalIdentifier id="d1e11" value="tipResistance"
    registryObject="urn:x-diggs:def:code-list:status:a"
    identificationScheme="urn:x-diggs:def:code-list:status" />
  <rim:ExternalIdentifier id="d1e11" value="coneResistance"
    registryObject="urn:x-diggs:def:code-list:status:a"
    identificationScheme="urn:x-diggs:def:code-list:status" />
</rim:ClassificationNode>
```

- DIGGS instances will only use the ClassificationNode (e.g. "tip"). Software that reads DIGGS files can reference the RIM file to discover descriptions or alternative terms for the ClassificationNode. However, use of the RIM file for that purpose isn't required.
- RIM file is used to identify list of valid names in a list, similar to a codelist.
- Software developers need only know the valid ClassificationNode values to successfully map to/from DIGGS files.
- Schematron or other external validation tool would need to be used to assess if the values used in a DIGGS instance conforms to the RIM file values. Schema validation alone cannot accomplish this.
- RIM approach forces interoperability. Prior approach with codelists in V1.0a presented issue where values could be published outside of DIGGS domain.

- We need to identify the canonical values and lists that are in common from the existing codelists in DIGGS.
- Need to harmonize these two existing lists:
 - DIGGS codelist (codelists_V1.xml)
 - AGS codelist (agsCodeList_V1.xml)
- Burggraf has done this using the existing codelists and has created a RIM file.
- Pick through schema and ensure all places in schema where codelists are needed.
- **ACTION ITEM:** Turner will organize meeting of domain experts to:
 - Evaluate RIM codelists element by element.
 - Combine elements from other codelists where applicable.
 - Identify rules for when codelists are used, and where we should use enumerations or other techniques.
- Status update from Burggraf:
 - Continuing to work on converting 20 examples.
 - **ACTION ITEM:** Will send RIM file (and style sheet) to team to look at.
 - **ACTION ITEM:** Will prepare example of RIM implementation with DIGGS instance – use a couple of instances from the 20 examples (e.g. a CPT example and a borehole example).
- Ponti suggests working on structural issues first (see notes from 1/28/10 – Turner action item to organize meeting of four software vendors.)
- How do we know when we're done with v1.1?
 - Finish the 20 examples.
 - Identify remaining issues.
 - Run GML SDK to detect violations.
 - Summarize the efficiencies (reduction in complexity) realized in this new version – reduction in schema size, number of schemas, includes/imports, schema load time, auto instance generation, etc.
 - Instances open in Altova, Oxygen, and Stylus with no need for special configuration.
- Next meeting:
 - Discuss the CRS issue.
 - Discuss inheritance and where this can be minimized.
 - Discuss the 20 examples.

E.9 Teleconference Meeting Notes 2010-02-18

Date: February 18, 2010

Time: 7:30 AM – 11:00 AM (PST)

Participants: Loren Turner
 David Burggraf
 Chris Bray
 Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Burggraf presented the 20 examples.
 - Lots of work to convert due to schema changes due to GML 3.2 and object-property rules.
 - Ponti asked why does <address> need to be a GML object here? Doesn't have geometry, so why does GML app need to recognize this?

```
<address>
  <diggs:Address gml:id="d1e17">
    <gml:identifier codeSpace="DMDC">DMDC-DELTA-address6</gml:identifier>
    <name>Delta Consultants</name>
    <street>35 West Broad Street</street>
    <city>St Paul</city>
    <state>MN</state>
    <country>USA</country>
    <postalCode>54673</postalCode>
  </diggs:Address>
</address>
```

- Burggraf explained:
 - <address> is the property
 - <diggs:Address> is the object
 - Need this construct in order for GML apps to recognize this properly.
 - Can do it either as a GML object, or not, like this

```
<address>
  <gml:identifier codeSpace="DMDC">DMDC-DELTA-address6</gml:identifier>
  <name>Delta Consultants</name>
  <street>35 West Broad Street</street>
  <city>St Paul</city>
  <state>MN</state>
  <country>USA</country>
  <postalCode>54673</postalCode>
</address>
```

- Example: for <address>, if we were to break down <street> into a couple of elements, we'd need a new GML object and property, which requires a gml:id and gml:identifier for the street.
- Ponti suggests we reconsider which items should be GML objects. Where is it redundant to have GML objects? Are there places where nested GML objects can be converted to regular XML complex types? Have this part of the domain discussion.
- GML Feature vs. Object
 - A GML "feature" is a physical entity, represents something in the real world.
 - A GML "object" can represent something in a real world, but not important enough to be passed around alone.
 - GML features require gml:identifier, but GML objects do not.
 - GML features and objects require gml:id.
 - Complex XML elements don't require either.
 - **ACTION:** Review features and objects and determine is we have that right.

- Ponti, Bray, Turner to do this by email, and have telecon if needed.
- Burggraf will generate a list of the features and objects; will note parent property for the objects.
- Team will review list and identify where changes are needed.
- Some objects could probably be changed to complex types.
- Prepare recommendation for Burggraf.
- This will be for DIGGS v1.2
- In the examples, all features and objects have gml:identifiers.
 - **ACTION:** Burggraf will change the examples to remove gml:identifier for objects to be consistent with prior discussion on the requirements for use of gml:identifier and gml:id. gml:identifier should not be there for the objects. This change needed for v1.1.
- Referencing:
 - Schema had this construct:


```
<groupings>
  <Group>
    <diggs:id>DMDC-GROUP_1</diggs:id>
    <type codeSpace="groupingType">QCSamplesFD</type>
    <items>
      <Item></Item>
      <Item></Item>
      <diggs:Ref xlink:href="#DMDC-BH1W_20070613" />
      <diggs:Ref xlink:href="#DMDC-BH1W_20070613b" />
    </items>
  </Group>
```
 - Burggraf notes:
 - Not allowed in GML to comply with object-property rule.
 - Revised implementation:


```
<Group gml:id="dle214">
    <gml:identifier codeSpace="DMDC">DMDC-GROUP_1</gml:identifier>

    <type codeSpace="groupingType">QCSamplesFD</type>
    <itemRef xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#DMDC-BH1W_20070613"/>
    <itemRef xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#DMDC-BH1W_20070613b"/>
    <items>
      <Item></Item>
      <Item></Item>
    </items>
  </Group>
```

 - <Item> is now separated from <itemRef>
 - Ponti notes that we rarely have instances where we have inline content and referencing.
 - Ponti suggests that for <grouping> we require only references, no inline content.
 - Burggraf needs a list of elements where we want to restrict the use of inline content and references.
 - Can deliver this in a spreadsheet.
 - Every GML complex property has this construct.
 - This will be for DIGGS v1.2
 - Burggraf prepared the RIM file:
 - Thought he captured all the codelists.
 - Will do checksum to verify.

- CRS issue:

- Currently implemented as:

```

<defaultCRS>
  <gml:EngineeringCRS gml:id="crs_BH1">
    <gml:identifier codeSpace="DIGGS">DIGGS-EngineeringCRS2</gml:identifier>
    <gml:name>BH1 CRS</gml:name>
    <gml:scope>gml:scope value is required in GML3.2, not provided in source data (not required
      in GML 3.1)</gml:scope>
    <gml:linearCS>
      <gml:LinearCS gml:id="lcs_BH1">
        <gml:identifier codeSpace="DIGGS">DIGGS-LinearCS2</gml:identifier>
        <gml:name>BH1 CS</gml:name>
        <gml:axis>
          <diggs:CoordinateSystemAxis gml:id="axis_BH1" uom="ft">
            <gml:identifier codeSpace="DIGGS">DIGGS-CoordinateSystemAxis2</gml:identifier>
            <gml:name>Depth Down BH1</gml:name>
            <gml:axisAbbrev>D</gml:axisAbbrev>
            <gml:axisDirection codeSpace="">Down</gml:axisDirection>
            <diggs:axisDirection xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#cl_BH1"/>
          </diggs:CoordinateSystemAxis>
        </gml:axis>
      </gml:LinearCS>
    </gml:linearCS>
  </gml:EngineeringCRS>

```

- Burggraf noted problems:

- The xlink:href is currently under the CoordinateSystemAxis, and shouldn't be.
- defaultCRS is inline when should be referenced.

- In order to implement depth in a hole in GML 3.2, we'd need to create individual objects for each depth. Burggraf showed an example where showing a single depth could result in a dozen lines of elements. This is how GML 3.3 would do it. The example included lots of xlink:href links.

```

<abc:featureMember>
  <abc:MyFeature gml:id="MF002">
    <abc:position>
      <glr:PositionExpression gml:id="PE002">
        <glr:linearElement xlink:href="#LE001"/>
        <glr:iml xlink:href="#LRM001"/>
        <glr:distanceExpression>
          <glr:DistanceExpression gml:id="DE001">
            <glr:distanceAlong uom="m">200</glr:distanceAlong>
          </glr:DistanceExpression>
        </glr:distanceExpression>
        <glr:PositionExpression>
      </abc:position>
    </abc:MyFeature>
  <abc:featureMember>
    <abc:MyFeatureCollection>

```

- Can collapse to:

```

<distanceAlong curveRef="#cl_cpt17" uom="m">200</distanceAlong>

<diggs:geometry>
  <gml:MultiCurve gml:id="dle325">
    <gml:identifier codeSpace="DIGGS">DIGGS-MultiCurve2</gml:identifier>
    <gml:curveMember>
      <gml:LineString srsName="urn:ogc:def:crs:epsg:6.9:27700" srsDimension="3" gml:id="cl_cpt17">
        <gml:identifier codeSpace="DIGGS">DIGGS-LineString2</gml:identifier>
        <gml:posList>0 0 0 1 0 1 1 0 1 0</gml:posList>
      </gml:LineString>
    </gml:curveMember>
  </gml:MultiCurve>
</diggs:geometry>

```

- Can also make the curveRef and uom as optional, and the defaults would be used.

- Here's how it would be implemented as a <top> element:


```
<top curveRef="#cl_cpt17" uom="m">0</top>
<base curveRef="#cl_cpt17" uom="m">18</base>

<stop>
  <gml:Point gml:id="dle1775">
    <gml:identifier codeSpace="DMDC">DMDC-BH1-top16</gml:identifier>
    <gml:pos>0</gml:pos>
  </gml:Point>
</top>
<!-- define top and base incase the CRS position is not top of hole --&gt;
&lt;base&gt;
  &lt;gml:Point gml:id="dle1783"&gt;
    &lt;gml:identifier codeSpace="DMDC"&gt;DMDC-BH1-base20&lt;/gml:identifier&gt;
    &lt;gml:pos&gt;18&lt;/gml:pos&gt;
  &lt;/gml:Point&gt;
&lt;/base&gt;</pre>

```
- Note that the first two lines would replace the lower nested structures.
- **ACTIONS:** Burggraf will:
 - Eliminate MultiCurve, since DIGGS only uses LineStrings, Points, Polygons, Planes, and Volumes.
 - Take a look at how GeoSCIml handles all this.
 - Develop an example implementation of the GML 3.3 implementation in a compact encoding with default settings. Discuss the example at next week's meeting.
 - Consider that DIGGS will need to capture more than just linestring positions.
- Next meeting agenda:
 - Review revised examples that address:
 - Revised CRS structure.
 - Fixed gml:identifier issue.
 - Review validation of RIM files --- results of the checksum exercise.
 - Follow up on DIGGS v1.1 testing – How do we know when we're done with v1.1?
 - Finalize the 20 examples.
 - Identify remaining issues.
 - Run GML SDK to detect violations.
 - Summarize the efficiencies (reduction in complexity) realized in this new version – reduction in schema size, number of schemas, includes/imports, schema load time, auto instance generation, etc.
 - Instances open in Altova, Oxygen, and Stylus with no need for special configuration.
 - Discuss inheritance and where this can be minimized – maybe

E.10 Teleconference Meeting Notes 2010-02-25

Date: February 25, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
 David Burggraf
 Chris Bray

Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Report from Burggraf:
 - The code list check sum passed – there are 53 code-list classification schemes in the XML registry package encoding and 53 code-list files in the directory:
DIGGS\1.0a\source\Codelists\default
 - The ‘gml:Multi’ geometries have been removed and replaced with gml:Point, gml:LineString, gml:Polygon geometries
 - The ‘geometry’ property has not yet been changed to either
 - GML profile only has these three.
 - Will add “composite surface” to the profile to handle surfaces that are made up of multiple polygons or multiple lines.
 - We can restrict features to specific geometries.
 - For example, force boreholes to be line strings.
 - For project features, can be point or bounding polygon, or bounding surface. Project features are not required.
 - **ACTION: Implement restrictions to geometry types:**
 - Turner, Bray, Ponti review feature table provided by Burggraf ([DIGGS_FeatureTable.htm](#)) and define which type of geometry applies, and if required or not. Add two columns to the table to list the applicable geometries and the requirements.
 - Burggraf will add composite surface as an option for the geometry .
 - Burggraf will implement the result of feature table work to enforce in gml profile and schema.
 - Piles have different geometries that change with depth. Can capture the cross-sections as polygon geometries that change over depth.
 - A “composite curve” requires an end-to-end connection with no branching. Not needed at this point. Use groupings.
 - A “multi curve” would handle the situation, for example, where multiple bores in the same hole that go different directions at the bottom. Not needed at this point. Use groupings.
- I’ve taken another stab at the ID issue in the DIGGS schemas and data. So far I have generated the data examples that have:
 - A gml:id value (random id value representing a DB handle, unique within the XML documents only) for all objects and features, for the mandatory gml:id attribute in GML 3.2
 - A gml:identifier property element for all identifiable features, i.e. elements that derive ultimately from diggs:IdentifiedFeatureType or diggs:IdentifiedObjectType (note that these are both GML Features, i.e. they derive from gml:AbstractFeatureType)

- No `gml:identifier` property element for all GML Objects that are not identifiable Features (as described above).
- There are some important schema design choices to make that have varying degrees of enforcement of the above practice for generating the data. First, I need to point out what objects the DIGGS schemas have and then I'll need some clarification of the intent of these objects:
 - `diggs:AbstractDiggsObject`, `diggs:DiggsObject`, `diggs:AbstractDiggsBase`, `diggs:DiggsBase`: These are currently typed as a GML Feature (i.e. derive from `gml:AbstractFeatureType`). Should these be typed as a mere GML Object? I.e. derive from `gml:AbstractGMLType` and not `gml:AbstractFeatureType`? Is it alright if these elements do not have a `gml:identifier`?
 - `diggs:AbstractIdentifiedObject`: This is currently typed as a GML Feature. It seems clear that the intention is to have a mandatory `gml:identifier` on these elements (should these be typed as a mere GML Object? I.e. derive from `gml:AbstractGMLType` and not `gml:AbstractFeatureType`?)
 - `diggs:AbstractFeature`: This is currently typed as a GML Feature. Is it alright if these elements do not have a `gml:identifier`?
 - `diggs:AbstractIdentifiedFeature`: This is currently typed as a GML Feature. It seems clear that the intention is to have a mandatory `gml:identifier` on these elements.
 - Several elements (e.g. `diggs:Address`) in the substitutionGroup “`gml:AbstractGML`” (originally in substitutionGroup “`gml:_Object`” in 1.0a) that derive from `gml:AbstractGMLType`. These are mere GML Objects (not GML Features). Some of these may remain GML Objects and some may be demoted to generic complex types as discussed in the last telecon.
- Currently I have taken the liberty of declaring `gml:identifier`:
 - mandatory on `diggs:AbstractIdentifiedObject` and `diggs:AbstractIdentifiedFeature`
 - absent on `diggs:AbstractDiggsObject`, `diggs:DiggsObject`, `diggs:AbstractDiggsBase`, `diggs:DiggsBase`, `diggs:AbstractFeature`
 - absent on all elements that are mere GML objects (e.g. `diggs:Address`, `diggs:Role`, etc)
- I'll need feedback on the questions above and the assumptions about the `gml:identifier` declarations I made above. Note that changes to the assumptions I made above will likely have implications on the design of the Feature and Object hierarchy in the Diggs schemas.
- GeoSciML uses a “coverage encoding” approach. It can be difficult for the reader to understand.
- Bray responded that entities to derive from one of three common bases:
 - ‘Object derived’ (`diggs:Object?`) - Typed as GML Object, geometry optional, identifier prohibited, transmission without context prohibited, identification outside instance document impossible, identification inside instance document possible via `gml:id` (was `diggs:AbstractDiggsObject`, `diggs:DiggsObject`, `diggs:AbstractDiggsBase`, `diggs:DiggsBase`)
 - ‘Feature derived’ (`diggs:Feature?`) - Typed as GML Feature, geometry option, identifier mandatory, transmission without context allowed, identification outside instance document possible via `gml:identifier`, identification inside instance document possible

- via `gml:id` or `gml:identified` (was `diggs:AbstractIdentifiedObject`, `diggs:AbstractFeature`, `diggs:AbstractIdentifiedFeature`)
- o 'generic complex type' - No GML, geometry prohibited, identified prohibited, transmission without context prohibited, identification outside or inside instance document impossible (many others, as identified)
- o Although it's still a good idea if all `diggs:*` entities (that are not generic complex types) to derive from either `diggs:Object` (which derives from `gml:Object`) and `diggs:Feature` (which derives from `gml:Feature`) rather than `gml:Object` and `gml:Feature` directly. Having a common root makes entity identification easier for consumers as well as making it easier for consuming applications to attempt to deal with custom extensions without having to implement the whole of GML.
- **ACTION:** Need to clean up where we use abstract types (prune the hierarchy).
 - o When reviewing the (`DIGGS_FeatureTable.htm`), examine the column "substitutes for" column to determine if an abstract type is really needed.
 - o For example, `Hole` derives from `AbstractHole`, which derives from `AbstractLocation`. In this case we could eliminate `AbstractHole`.
 - o Burggraf will eliminate a lot of the superfluous annotations throughout. Add another column to the `DIGGS_FeatureTable.htm` to contain what the annotation field should be.
 - o Burggraf will convert `DIGGS_FeatureTable.htm` to an Excel spreadsheet and will add the columns that we need to populate.
 - o Turner will post table on Google Docs for the team to edit.
- Burggraf suggested:
 - o We can merge the `diggs:AbstractIdentifiedFeature` and `diggs:AbstractFeature` hierarchies as both should derive from GML Feature and have mandatory `gml:identifier`. If everyone agrees, I'll replace the `diggs:AbstractFeature` substitution group with the `diggs:AbstractIdentifiedFeature` substitution group.
 - o We can demote `diggs:AbstractDiggsObject`, `diggs:DiggsObject`, `diggs:AbstractDiggsBase`, and `diggs:DiggsBase` to GML Object. If everyone agrees, I'll have these derive from `gml:AbstractGMLType` and not `gml:AbstractFeatureType`
 - o **ACTION:**
 - These are currently features and will be changed to objects.
 - Burggraf will change the Excel spreadsheet before sending to us.
 - o We still need to decide whether several elements (e.g. `diggs:Address`) in the `substitutionGroup "gml:AbstractGML"` (originally in `substitutionGroup "gml:_Object"`) remain GML Objects or be demoted to generic complex types as discussed in the last telecon. The Feature and Object Tables, should help with this decision making.

E.11 Teleconference Meeting Notes 2010-03-04

Date: March 4, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
David Burggraf

Dan Ponti

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Edit the spreadsheet:
 - **ACTION:** Ponti, Turner, Bray complete
 - Determine if Abstract:
 - *Abstract* – Will never appear in instance document. Meant only to be used as a base type.
 - Identify Type:
 - *Feature* – Top level object at root of structure, shared. Identified with a `gml:identifier` that would be globally unique in addition to a `gml:id`. (e.g. project, hole, samples, business associates)
 - *Object* – `gml` object, complex type, requires a `gml:id`, would never be passed around on its own; always be nested in line with something; something else would reference it; feature would reference it. (e.g. geometry, remarks)
 - *Complex* – Complex element that has no `gml:id`; internal elements not recognized as `gml` properties.
 - *Remove* – Object is no longer needed.
 - Determine if Metadata:
 - Indicate: (Y) yes, entire object is metadata; (N) no; (C) contains simple elements that are metadata
 - `Gml` objects would by typed `gml` metadata.
 - Would extend from `AbstractMetadataType` instead of `AbstractGMLtype`.
 - GML applications could discover metadata.
 - No significant structural change in an instance document
 - **ACTION:** Ponti, Turner, Bray consider structural change to “group” metadata elements into a metadata object, separate from the data elements similar to `cosmosDIGGS`.
- Use of `gml:identifier` and `gml:id`.
 - Recommended practice to use the same `gml:identifier` as the `gml:id`.


```
<Project xmlns:gml="http://www.opengis.net/gml/3.2" gml:id="DIGGSINC-DIGGSV1">
  <gml:name codeSpace="diggsml.com" xmlns="">DIGGS Example files</gml:name>
  <gml:identifier codeSpace="DIGGSINC" xmlns="">DIGGSINC-DIGGSV1</gml:identifier>
```
- Consider a formal way to construct the `gml:identifier` similar to the way the geometry is handled (e.g. `srsName="urn:ogc:def:crs:epsg:6.9:27700"`).
 - For DIGGS `gml:identifier`, we'd have something like: “`urn:diggs:fi:caltrans:diggsv1`”
 - Could also be used to point to dictionaries, for example, for color: “`urn:diggs:dict:munsel:colorcode`”
 - Follows a pattern:
 - URN
 - Namespace identifier (NID), e.g. `diggs`
 - Object type, e.g. feature instance, dictionary, feature type
 - Authority, e.g. `caltrans`

- Code (defined by the authority, unique to the authority)
 - Typically used as the gml:id value
 - Can be the code for a codelist item
 - Can be used in xlink:href to reference various types of objects.
 - Would need to submit a RFC (Request For Comment) to IANA (Internet Assigned Numbers Authority).
 - No cost – need to verify
 - RFC process – Burggraf can draft
 - Implementation:


```
<Project gml:id="d1e44">
  <gml:name codeSpace="diggsml.com">DIGGS Example files</gml:name>
  <gml:identifier codeSpace="http://dot.ca.gov/about-caltrans-diggs-ids.htm">urn:diggs:fi:caltrans:diggsv1</gml:identifier>
```
 - Can make the gml:id and gml:identifier the same (but doesn't have to be):


```
<Project gml:id="diggsv1">
  <gml:name codeSpace="diggsml.com">DIGGS Example files</gml:name>
  <gml:identifier codeSpace="http://dot.ca.gov/about-caltrans-diggs-ids.htm">urn:diggs:fi:caltrans:diggsv1</gml:identifier>
```
 - **ACTIONS:**
 - Burggraf draft the RFC for the URN structure to be submitted to IANA.
 - DIGGS organization (Turner) will submit to IANA.
 - Make changes to implement pattern restrictions for gml:identifier for DIGGS v1.2.
 - Burggraf will revisit code list encoding to use URN and email with any issues identified.
- Burggraf presented revised linear referencing approach. This was implemented in the Example 18 included in an email attachment on 3/3/10.
 - Accommodates multiple linear referencing of a single hole.
 - **ACTION:** Ponti will send Burggraf example WITSML data instance (blocks of data) to see how the new linear referencing would encoded it.
- **ACTION:** Punchlist to deliver DIGGS v1.1
 - Implement linear referencing approach.
 - Revise the 20 examples.
 - Run GML SDK to detect violations.
 - Summarize the efficiencies (reduction in complexity) realized in this new version -- reduction in schema size, number of schemas, includes/imports, schema load time, auto instance generation, etc.
 - Verify that instance documents open in Altova, Oxygen, and Stylus with no need for special configuration.
 - Generate online documentation.
 - CoreSIG meeting to present changes in DIGGS v1.1. (Proposed 3/18/10)
- **ACTION:** Project status
 - Burggraf will send update on billing and hours to date.
 - Skip telecon next week. Reschedule for early following week.

E.12 Teleconference Meeting Notes 2010-03-16

Date: March 16, 2010

Time: 7:30 AM – 9:30 AM (PST)

Participants: Loren Turner
David Burggraf

Dan Ponti

Not Available: Chris Bray

Agenda: DIGGS-Galdos technical meeting.

Notes:

- Ponti presented two proposed changes to the schema (both documented in his email on 3/15/10):
 - Create metadata groupings in base types – from Dan's email:

"2) Created two element groups - FeatureMetaDataProperties and TestMetaDataProperties. The FeatureMetaDataProperties group contains three elements - a) associatedFile b) roles and c) remark. associatedFile is of type gml:referenceType; roles is an object-property type that contains role objects, and remark is a gml:stringOrRef type - to reference either a remarks object or hold a simple string. Added associatedFile and remark object-properties to Project (eg. these features are maintained at the Project level). FeatureMetaDataProperties added to AbstractFeature and AbstractNamedFeature only (not in objects). TestMetaDataProperties contains references to equipment and specificatilon and is added to diggs:TestType. TestType is made abstract. Probably should rename to AbstractTestType to conform with the abstract element as I did with objects and features, but I didn't want to fix all of the broken references..."
 - Removing redundant abstract elements.

"10) Removed many, but not all Abstract objects and features in the kernal and geotechnical namespaces where not needed. Abstract objects were deleted when they only had one single concrete object/feature defined as a substitution.. There is still much cleaning up to do with these."
- General concurrence from Turner and Burggraf on these two proposed changes.
- **ACTION:** Bray review proposed changes.
- For metadata grouping change, Burggraf will provide Ponti with "input schemas". Ponti will implement the metadata grouping in schemas, as proposed, and return schemas to Burggraf, who will run scripts to implement.
- For removing abstract elements, David will do this via scripting using logic whereby Abstract objects will be deleted when they only had one single concrete object/feature defined as a substitution.
- Other suggestions proposed by Ponti will be discussed at Thursday's meeting.
- Burggraf still needs geometry column completed in the spreadsheet in order to complete v1.1.
- DIGGS v1.1 likely delayed by a week due to these changes.

E.13 Teleconference Meeting Notes 2010-03-18

Date: March 18, 2010

Time: 7:30 AM – 10:45 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Modification of the “input” schemas to remove inline or reference elements where we want to limit the use to one or the other.
 - Use input schemas.
 - Comment out elements to be deleted.
 - Scripts will not carry over comments.
 - Script will look for complex type for which no element uses it.
 - **ACTIONS:**
 - Burggraf will modify (with abstract elements removed) and send new input kernel schema (3/19)
 - Ponti will edit and comment (3/25)
 - Burggraf will re-run scripts (3/26)
- Implement metadata grouping discussed earlier for roles, associated files, specifications, and equipment.
 - **ACTION:**
 - Ponti to comment these changes in the input schema.
 - Burggraf to modify scripts to implement.
- Scripting had produced reference types with annotations and 1:1 corresponding elements. These types were based on `gml:ReferenceType`. Suggestion to eliminate the reference types, change to `gml:ReferenceType`, then move the annotations to the element level.

```

<element name="businessAssociateRef" type="gml:ReferenceType" minOccurs="0"
         maxOccurs="unbounded">
    <annotation>
        <documentation>A property element that supports a value by reference only. To
indicated in the appinfo element.</documentation>
        <appinfo source="urn:x-gml:targetElement">diggs:BusinessAssociate</appinfo>
    </annotation>
</element>

```

- Use `gml:ReferenceType` for most. Use `diggs:ReferenceType` when the index and percentage is applicable.


```

<complexType name="ReferenceType">
    <complexContent>
        <extension base="gml:ReferenceType">
            <attribute name="index" type="integer"/>
            <attribute name="percentage" type="double"/>
        </extension>
    </complexContent>
</complexType>

```

- Move the <annotation><appinfo> to the element level which includes target info.
- Annotate either the type or the element, but typically not both. Burggraf will check to make sure the annotation
- Handling remarks as string or inline or reference:
 - GML doesn't support the "string or an object" construct. It would be mixed content and difficult to map from a database.
 - Could create a <remark> property, with three elements who/when/what, the what is the only required, object within a remarks. Cons – need a gml:id;
 - Bray stated it needed a gml:identifier in an earlier email. Is a gml:id sufficient?
 - **ACTION:** Dan will document the various options discussed in the meeting (to keep remarks inline) and email to the group for further consideration.
 - Choice of remarks property or object
 - "Flatten" remarks – create complex content, make who/what/when elements
- Positional accuracy property – suggest to add this as optional metadata to geometry.
 - Treat as metadata.
 - Consider adding a positional accuracy metadata element to a base type.
 - Leave it as-is for DIGGS v1.1.
- Completion of spreadsheet?
 - Not as relevant for v1.1 any longer.
 - Scripting work and changes made to input schemas will address most issues.
 - Feature vs. Object – can resolve this for v1.2.
- Status of DIGGS v1.1 release.
 - Complete input schema changes and run scripts as above.
 - Run validation.
 - Create documentation.
 - Update example instances.
 - Finish input schema mods (3/25)
 - Finish v1.1 by end of March (3/26)
 - Another week to wrap up everything else (4/9)

E.14 Teleconference Meeting Notes 2010-03-25T07:30

Date: March 25, 2010

Time: 7:30 AM – 10:30 AM (PST)

Participants: Loren Turner

David Burggraf

Dan Ponti

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Discussed approach for divorcing position information in the schema:
 - Positions of observations at location features are defined in the context of that location feature
 - Eg. for a hole, positions are defined by measured depth – either at a point in the hole or an interval that defines a segment of the hole centerline.
 - These positions are defined by the location, not by the observation itself, and therefore can logically be coded as a property of the location feature, and not of the observation,
 - Divorcing positions from observations will make extensibility easier and allow for easy re-use of observation features.
- Dan will work on an example implementation.
- Example of schema with this implementation:

```
<Hole gml:id = "h1">

    <centerline>

        <gml:LineString xmlns:gml="http://www.opengis.net/gml/3.2"
            srsName="urn:ogc:def:crs:epsg:6.9:27700" srsDimension="3" gml:id="cl_BH18">
            <gml:posList>407829 268621 23.93 407415 268600 8.43</gml:posList>
        </gml:LineString>

    </centerline>

    < linearReferencing>
        < PositionExpression gml:id="PE001">
            .....
        </PositionExpression>

    </linearReferencing>

    <measuredDepths>

        <MeasuredDepths gml:id = "md1">

            <md>
```

```

<gml:Point gml:id="d1e483" srsName="#PE001" srsDimension="1">
    <gml:pos>0</gml:pos>
</gml:Point>

</md>
<md>

    <gml:Point gml:id="d1e484" srsName="#PE001" srsDimension="1">
        <gml:pos>15</gml:pos>
    </gml:Point>

</md>

</MeasuredDepths>

</measuredDepths>

<depthIntervals>

    <DepthIntervals gml:id="di1">

        <di>

            <gml:LineString gml:id="dil-1" srsName="#PE001" srsDimension="1">
                <gml:posList>5 7.5</gml:posList>
            </gml:LineString>

        </di>

    </DepthIntervals>

<depthIntervals>

</Hole>

<SamplingActivity gml:id="samp1">

    <gml:identifier>urn:diggs:fi:usgs:samp1</gml:identifier>

    <position xlink:href="#dle484"/>

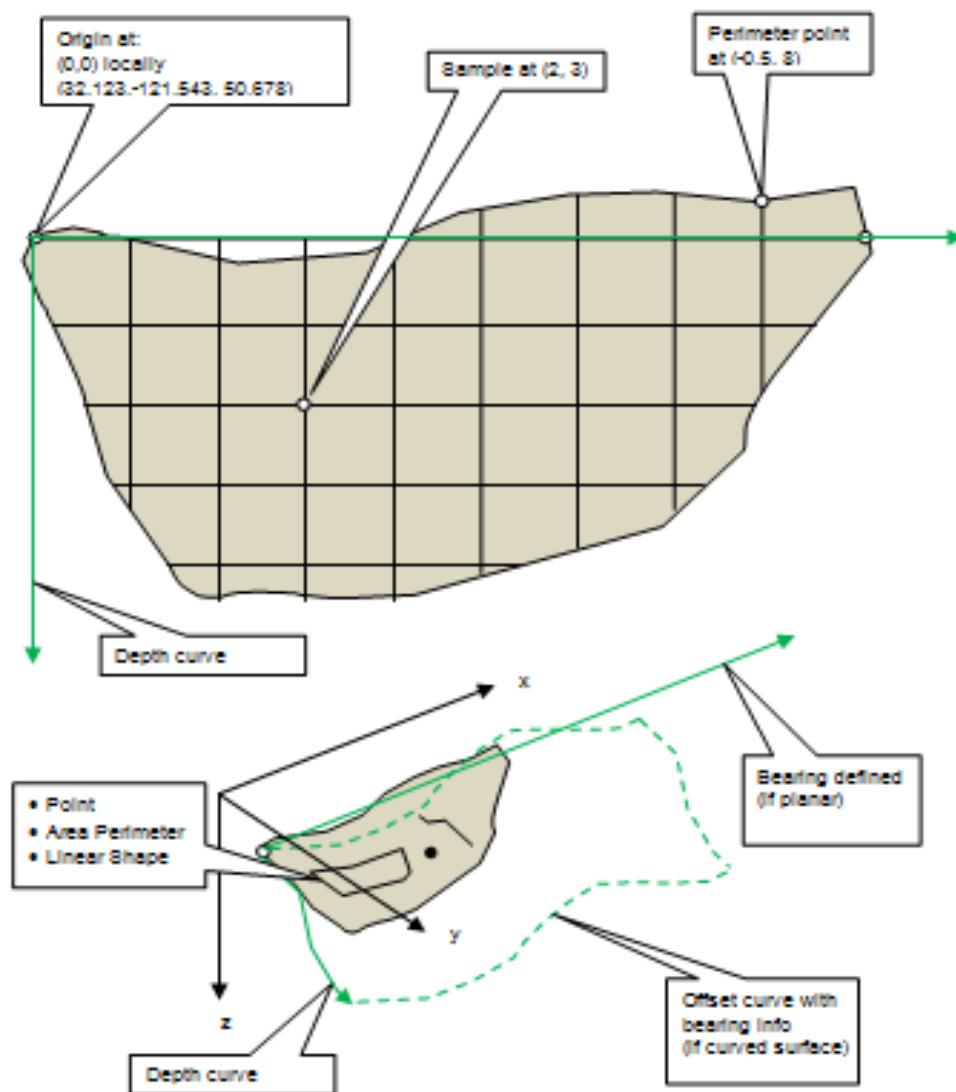
    ...

</SamplingActivity>

```

- Discussed how this new schema approach would facilitate implementation of trench logs and similar features.

- Can reuse features like samples without having to create new sample features that support specific geometries.
- Discussed method of defining geometry for trench logs and exposures.
 - Field practices:
 - Establish origin in upper left of trench wall
 - Create grid
 - Map relative to grid
 - Grid face is roughly vertical
 - To model this in schema:
 - Establish “depth curve”
 - Create “offset curve”
 - “point”, “area perimeter”, and “linear shape” features can then be identified.



E.15 Teleconference Meeting Notes 2010-03-25T10:30

Date: March 25, 2010

Time: 7:30 AM – 10:30 AM (PST)

Participants:

- Loren Turner
- Dan Pontl
- Chris Bray
- Salvatore Caronna
- Roger Chandler
- Scot Weaver
- Scott Deaton

Not Available: n/a

Agenda: DIGGS software vendors meeting.

Notes:

- Summary of work to date on DIGGS v1.1
 - Fixed gml Object-Property rule
 - Example of object property rule.


```
<Diggs xmlns="http://schemas.diggsml.com/1.0a" xmlns:gml="http://www.opengis.net/gml/3.2"
        xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:witsml="http://www.witsml.org/schemas/131" xmlns:diggs="http://schemas.diggsml.com/1.0a"
        xmlns:diggs_geo="http://schemas.diggsml.com/1.0a/geotechnical"
        xmlns:diggs_env="http://schemas.diggsml.com/1.0a/environmental"
        xmlns:diggs_mon="http://schemas.diggsml.com/1.0a/monitoring"
        xmlns:diggs_pil="http://schemas.diggsml.com/1.0a/piling" gml:id="d1e1"
        xsi:schemaLocation="http://schemas.diggsml.com/1.0a ../../Complete.xsd">
<businessAssociates>
```
 - BusinessAssociate


```
<BusinessAssociate xmlns:gml="http://www.opengis.net/gml/3.2" gml:id="d1e5">
<gml:name xmlns="">Bill Mallard</gml:name>
<gml:identifier codeSpace="DIGGS" xmlns="">DIGGS-BM</gml:identifier>
<address>
<Address gml:id="ad123">
<street>345 Middlefield Road</street>
<city>Menlo Park</city>
<state>CA</state>
<postalCode>94025</postalCode>
</Address>
</address>
</BusinessAssociate>
</businessAssociates>
```
 - Fixed import/includes – no longer need OASIS catalog.
 - Seconds to load schemas
 - No special configuration needed with tools.
 - Migrated DIGGS to GML 3.2
 - ISO standard
 - GML profile created for DIGGS
 - Reorganized to 5 namespace schemas with one file per namespace
 - gml:identifier and gml:id
 - Dropped diggs:id

- gml:identifier for globally unique id (use URN); only applies to gml features (locations, projects, samples, layer systems, tests)
- gml:id required for all features and objects for referencing (database handle); objects are complex types that are “recognized” as properties in gml.
- Use of GML identifier (URN encoding)
 - For DIGGS gml:identifier, we'd have something like:
“urn:diggs:fi:caltrans:diggsV1”
 - Could also be used to point to dictionaries, for example, for color:
“urn:diggs:dict:munsel:colorcode”
 - Follows a pattern with components separated by “:”
 - URN
 - Namespace identifier (NID), e.g. diggs
 - Object type, e.g. feature instance, dictionary, feature type
 - Authority, e.g. caltrans
 - Code (defined by the authority, unique to the authority)
 - Typically used as the gml:id value
 - Can be the code for a codelist item
- Can be used in xlink:href to reference various types of objects.
- Would need to submit a RFC (Request For Comment) to IANA (Internet Assigned Numbers Authority).
- No cost – need to verify.
- RFC process – Galdos will facilitate this.
- Implementation – can make the gml:id and gml:identifier the same (but doesn't have to be).
- Removed unnecessary abstract types
- Implemented RIM for codelists in lieu of gml dictionaries.
 - Allows for single vocabulary with translations.
 - Consolidates DIGGS codelists.
 - “Nodes” become the single identifier in the DIGGS file. Naming systems and localization for language included.
 - Example of RIM encoding for codelists:


```
<rim:ClassificationScheme id="urn:x-diggs:def:code-list:driven_penetration_test_type" lid="urn:x-diggs:def:code-list:driven_penetration_test_type" objectType="urn: oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ClassificationScheme" status="urn: oasis:names:tc:ebxml-regrep>StatusType:Submitted" isInternal="true" nodeType="urn: oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
  <rim:Name>
    <rim:LocalizedString xml:lang="en" value="Driven Penetration Test Type" />
  </rim:Name>
  <rim:Description>
    <rim:LocalizedString xml:lang="en" value="AGS" />
  </rim:Description>
  <rim:ClassificationNode id="urn:x-diggs:def:code-list:driven_penetration_test_type:s" lid="urn:x-diggs:def:code-list:driven_penetration_test_type:s" objectType="urn: oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ClassificationNode" status="urn: oasis:names:tc:ebxml-regrep>StatusType:Submitted" parent="urn:x-diggs:def:code-list:driven_penetration_test_type" code="S" path="/urn:x-diggs:def:code-list:driven_penetration_test_type/S">
    <rim:Name>
      <rim:LocalizedString xml:lang="en" value="S" />
    </rim:Name>
    <rim:Description>
      <rim:LocalizedString xml:lang="en" value="SPT Split spoon" />
    </rim:Description>
    <rim:ClassificationNode id="urn:x-diggs:def:code-list:driven_penetration_test_type:c" lid="urn:x-diggs:def:code-list:driven_penetration_test_type:c" objectType="urn: oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ClassificationNode" status="urn: oasis:names:tc:ebxml-regrep>StatusType:Submitted" parent="urn:x-diggs:def:code-list:driven_penetration_test_type" code="C" path="/urn:x-diggs:def:code-list:driven_penetration_test_type/C">
      <rim:Name>
        <rim:LocalizedString xml:lang="en" value="C" />
      </rim:Name>
```

```

<rim:Description>
  <rim:LocalizedString xml:lang="en" value="SPT Cone" />
</rim:Description>
<rim:ClassificationNode>
</rim:ClassificationScheme>

```

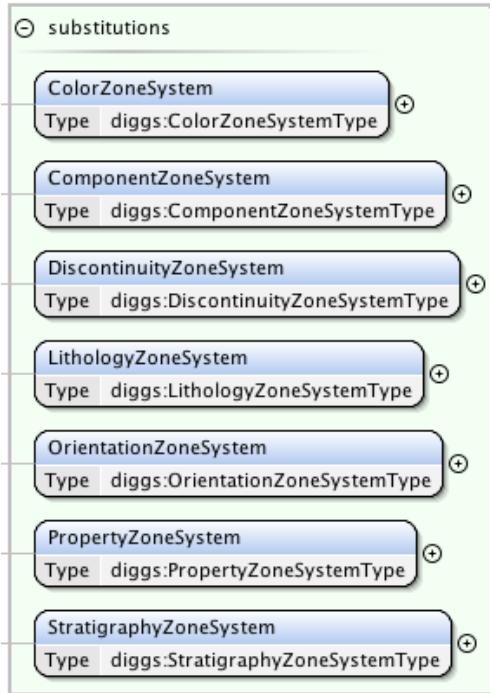
- Tabular data
 - Retain existing structure, but implement code lists (in RIM) in v1.2 to restrict column types.
 - Removed generic table property.
 - Tables to be included under specific test features only.
- Geometry
 - DIGGS V1.0a had generic geometry. DIGGS v1.1 restricts geometry bases on the feature.
 - Projects can have three types of geometry – reference point, linear extent, areal extent.
 - Linear referencing – gml method to reference positions in a borehole will be adopted in gml 3.3.
 - Example:


```

<Hole xmlns="http://schemas.diggsml.com/1.0a/geotechnical" gml:id="d1e96">
  <gml:identifier codeSpace="DIGGS">DIGGS-BH18</gml:identifier>
  <diggs:linearReferencing>
    <!-- the PositionExpression element contains the Linear Referencing definition and only needs to appear once in the scope of a feature or feature collection -->
    <diggs:PositionExpression gml:id="PE001">
      <diggs:linearElement xlink:href="#cl_BH18">
      </diggs:linearElement>
      <diggs:lrm>
        <diggs:LinearReferencingMethod gml:id="LRM001">
          <diggs:name>chainage</diggs:name>
          <diggs:type>absolute</diggs:type>
          <diggs:units uom="m"/>
        </diggs:LinearReferencingMethod>
      </diggs:lrm>
      </diggs:PositionExpression>
    </diggs:linearReferencing>
    <construction>
      <CylindricalConstruction gml:id="d1e481">
        <gml:identifier codeSpace="DIGGS">DIGGS-CC2</gml:identifier>
        <top>
          <gml:Point gml:id="d1e483" srsName="#PE001" srsDimension="1" uomLabels="m">
            <gml:pos>0</gml:pos>
          </gml:Point>
        </top>
        <base>
          <gml:Point gml:id="d1e489" srsName="#PE001" srsDimension="1" uomLabels="m">
            <gml:pos>15.5</gml:pos>
          </gml:Point>
        </base>
        <type>Cable Percussion</type>
        <diameter uom="mm">150</diameter>
      </CylindricalConstruction>
    </construction>
  </Hole>
```
 - Galdos proposed a GML referencing model (that's being proposed for use in DIGGS) to OGC which was adopted and should be in GML 3.3.
- Metadata
 - AssociatedFiles, Roles, Remarks, Specifications, Equipment, BusinessAssociates, Contracts are cast as gml metadata so that gml aware applications will recognize those objects as metadata.
 - No longer assigning metadata properties at the base level to prevent recursion.
 - All features carry associatedFile, roles, and remarks metadata properties.
 - All objects carry remarks metadata properties.

- All tests are features and carry specifications and equipment metadata properties as default.
 - More work needed on this for v1.2.
- Necessary tasks for DIGGS v1.2
 - Codelists need to be fleshed out; more explicit definitions are required.
 - Tighten up object and feature designations. Possibly demote some objects to complex types.
 - All properties and features in the schemas need to be annotated.
 - Identify business rules to be implemented in Schematron.
- Business rules (Caronna).
 - Roger Chandler, Will Holmes, Salvatore Caronna, Scott Deaton met.
 - Need to come up with business rules and mechanism to come up with them.
 - Four levels of business rules identified:
 - Level 1 – Enforcing definitions as they appears in DIGGS, e.g.:
 - The concatenation of the key values must be unique for each record.
 - Within a layer system, layers cannot overlap nor can there be gaps.
 - Level 2 – Impossible data, e.g.:
 - Total core recovery < RQD
 - Bottom depth > Top depth
 - Level 3 – Warnings on relationships.
 - Level 4 – Warnings, out of reasonable range, e.g.:
 - Permeability = 10 cm/sec
 - Dry Density = 140 pcf
 - Recommend pursuing Level 1 issues first using schematron.
 - Rules encoded in files separate from the schema and instance files.
 - Suggestion to use online forums to discuss rules.
 - Need the business rules in “plain English” in order to specify the contract work needed for the development of schematron.
 - Turner needs enough info on business rules in order to write a scope of work for a schematron contractor.
 - List of rules
 - Turner will set up discussion forum to facilitate discussion of rules.
 - Will need more info on “the concatenation of the key values must be unique for each record.”
 - What are the key values?
 - What is getting concatenated?
 - Look at the AGS model.
 - What’s the uniqueness that needs to be enforced for every feature that requires a unique key?
 - Need to look at examples of DIGGS instances to identify.
 - What combination of object properties need to be unique?
 - Use annotation within the schema to flag “key fields”.
- Future vision of DIGGS (Bray) – gml:identifier enables “atomic features” for DIGGS.
 - Samples, holes, and other features can be transmitted independently.
 - Enables DIGGS features to be self-contained.
 - Web Feature Services (WFS) could be developed, for example, that returns collection of DIGGS features.
- Proposed structural changes for DIGGS v1.2 (Ponti):

- Add solid and composite area geometries to project.
- Sample and Sampling Activity – separate the physical sample from the activity that produces it.
 - Issues:
 - Current sample feature conflates the physical sample with the activity that produces it.
 - Samples need to be transmitted without any info about the sampling activity in order to preserve anonymity for blanks and standards
 - Currently to do this, there is a generic source property that can reference either another sample or a hole. Position (eg. depth) is still part of Sample, so this info is still transmitted.
 - Samples that are subsamples or aggregates require a kludge using generic source property to indicate where sample came from, but this is not a direct association – only information about a sample's source.
 - Generic source property is difficult to map to (non-specified target – violates *gml* referencing rules that suggest target specification for references in `<appInfo>` element.
 - Recommended solution:
 - Create a separate SamplingActivity feature as a child of the Diggs root element
 - SamplingActivity contains properties that references the position and location to which it belongs; if the activity does not occur at a specific position, (such as an aggregate or subsample) then it references the Project only and the source sample ID's in properties specific for that purpose.
 - SamplingActivity also contains info about the sampling activity itself.
 - Sample contains only info about the physical sample and chain of custody information
 - Sample references its parent sample activity only. No ambiguous association reference.
 - Advantages:
 - Can account for an activity without a sample (eg. core run with no recovery)
 - Reduces duplication where a single activity creates multiple physical samples.
 - Aggregations and subsamples easily handled
 - Sample objects themselves provide no information on how they were obtained/created – perfectly blind.
 - Disadvantages:
 - Most practice conflates activities and physical samples; requires creating 2 features in xml for each sample record in a database
- Layer Systems – more specificity for classes of layer systems (interval data)



- Namespace changes:
 - All samples, base types, and common types of locations placed in kernel.
 - Geotechnical, environmental, and piling namespaces contain discipline-specific tests and locations only.
 - Intent is to reduce the likelihood for instance documents to have to use multiple namespaces.
 -
- Multi-track insitu tests (geophysical logs, CPT):
 - Geophysical logs as type of insitu test.
 - Currently considering using WITSML log structure for geophysical logs and DIGGS table structure for CPT.
 - Harmonize to a DIGGS namespace structure for both?
 - Use WITSML for CPT?
- Divorce position information at locations from results of tests and observations
 - Position information is inherent in the type of location feature where observations are made and tests are run.
 - Intent to provide better extensibility for different location types without having to redesign observation features
 - More compact position encoding
 - Positions of observations at location features are defined in the context of that location feature.
 - Eg. for a hole, positions are defined by measured depth – either at a point in the hole or an interval that defines a segment of the hole centerline.
 - These positions are defined by the location, not by the observation itself, and therefore can logically be coded as a property of the location feature, and not of the observation.
 - Divorcing positions from observations will make extensibility easier and allow for easy re-use of observation features.
- For the proposed schema changes (i.e. sampling and sampling activity, layer systems, and divorcing position information), Ponti will create example instance documents to help illustrate the impact of the change using a real example. Example instance documents will use the existing 20 examples that were created earlier in AGS and DIGGS formats.

E.16 Teleconference Meeting Notes 2010-04-01

Date: April 1, 2010

Time: 7:30 AM – 8:10 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti
Chris Bray

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Ponti status:
 - Taking a little longer than anticipated to modify input schemas
 - Lots of changes
 - Main kernel file is finished and will send to Burggraf by end of this week.
- Burggraf:
 - Will start working on schemas on Monday
 - Should complete everything by end of next week
- Schema versioning
 - Implementation:
 - Will use version attribute on the root schema element <diggs/>.
 - Will use namespace/filepath to determine version.
 - Backwards compatibility – need to consider what this means.
 - Manage changes in such a way that 'structural' changes require a minor version number bump, whereby simpler changes only required a revision number bump. So 1.1.1 and 1.1.2 would be compatible where 1.1.1 and 1.2.1 would not, this is identified by the 1.1 and 1.2 at the start without the need for a longer version number.
 - This is similar to the versioning policy for all xml encoding standards at the OGC. The DIGGS community should publish their own as well.
 - Version defined by X.Y.Z:
 - X is the major version number, meant for radical changes - no backwards compatibility is guaranteed when this number changes.
 - Y is the (first) minor version number, backwards compatibility is strongly encouraged but not required. Changes in this number must be accompanied by a change of namespace.
 - Z is the 2nd minor version number, meant for corrigenda or bugfixes. Backwards compatibility is implied and no change of namespace is accompanied by a change in Z. Typically backwards compatible bugfixes are treated at this revision level. Also clarification/annotation to the usage of elements/types is included at this revision level
- Use of identifiers for “remarks” – will proceed with Ponti’s suggestion from prior meeting.

E.17 Teleconference Meeting Notes 2010-04-08

Date: April 8, 2010

Time: 7:30 AM – 8:10 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti
Chris Bray
Roger Chandler

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Ponti prepared the input schemas and delivered last Saturday.
- Ponti went through the “Closing the book on v1.1” email from 3/18/10 (attached). The group concurred with recommendations.
- Burggraf needed to modify in order validate.
 - No major issues
 - Looks correct
 - Lots of special cases needing changes in scripts
 - Ran 2 or 3 scripts so far
 - Should be done within a few hours
 - Ponti will provide schema snippets for linear referencing for holes.
 - Will pass back to Ponti to verify that changes were implemented.
- For 1.2 development, scripts will be run against the 1.1 version.
- Punchlist for 1.1 release
 - Finish running the scripts (as above).
 - Verification with Ponti.
 - GML SDK check (prior check didn’t identify any GML issues).
 - Documentation – automatic generation.
 - What’s new in 1.1 document.
 - Roll-out meeting for week of April 12.



Daniel Ponti <pontifamily@gmail.com>
03/18/2010 09:04 PM

To	Loren Turner <loren_turner@dot.ca.gov>, David Burggraf <dburggraf@galdosinc.com>, Chris Bray	<input type="button" value="..."/>
cc	<input type="text"/>	
Subject	Closing the book on v 1.1	

All -

I was thinking about the loose ends to wrap up to close the book on 1.1. Here are a few and some suggested pathways for now:

1) Feature/object determinations and their geometries:

We've dealt with some of these, but not systematically. David's scripting out of abstract elements will help with the clutter of going through all of these. I would suggest that we deal with this in a blanket fashion for now, and then we can look closer at this for 1.2:

1) Locations (eg. holes, sampling stations): FEATURES. For 1.1, derive these from abstract location types that carry geometries consistent with location type (eg. holes derive from a linear location type).

2) Projects: FEATURES. For geometries in 1.1, I would suggest that project carry these three optional geometry properties: a) referencePoint (gml:PointPropertyType), b) arealExtent (gml:SurfacePropertyType), and c) linearExtent (gml:CurvePropertyType). We talked about defining solids for projects and composite areas as well, but these geometries are not currently in our GML profile. We can revisit and add others if necessary for 1.2.

3) Samples: FEATURES - can be transmitted on their own, although for full context, one must know the location where sample was taken or process by which sample is created. A current problem with samples is that it has a generic source property(ies) that can reference any uncontrolled feature/object. For 1.2 I think we need to tighten this up. One way to do this is to divorce the process of sampling from the sample itself so that samples can travel on their own without revealing anything to a lab about how they were created, and the sampling activity feature can then contain explicit references to either a position at a location, or to source samples.

4) LayerSystems: FEATURES. These could be objects, too, but layer systems apply to almost any location feature and therefore could be referenced. We can debate this more. Currently, LayerSystem is an object and the child Layer is a feature. This is backward. For 1.1, I would suggest making LayerSystem a feature, and Layer an object. Then we can carry this debate into 1.2. For 1.2, I would like to see specific classes of layer systems. Right now they are too open ended and confusing for average folk to get a handle on how to implement and encode.

5) In-Situ Tests: FEATURES. In-situ tests only occur in context of a particular location type (maybe there are a few exceptions), and perhaps could be demoted to objects - now they are features because they extend from the same base type as lab tests. Let's leave them that way for now. How we categorize use in-situ tests should be another 1.2 discussion. Geophysics are another in-situ test and we need to build the correct hooks in 1.2 for Witsml, if we choose to stay with that schema design.

6) Laboratory Tests: FEATURES.

7) Groups. These are currently FEATURES held within a root level property type. Keep as is for 1.1, but -- are groups metadata?

8) Tables. Tables are currently objects and should remain as such as this is a construct to be used as properties of test results (eg. static cone, etc.) or other features. There is a tabularData property type to hold generic table data at the root level. Table data should be encapsulated in the context of a test or other relevant feature - it doesn't belong here, I don't think. For 1.1, I suggest we delete (eg. comment out) this property type at the Diggs level. Note - I'm not suggesting that we drop the Table element or property type, just that we no longer carry a root level property that only carries generic table objects.

For 1.1, for all of the features above:

a) leave the point property types that define top and base alone. They will work with the linear referencing property type that David introduced. We can discuss more compact encoding options for 1.2, including divorcing positions from tests and other observations. GML encoding of depth information for tabular data (eg. table data structure as implemented for static cone tests, or via witsml for geophysical logs) needs to be handled differently and this will take some thought and David's tutelage. I believe David suggested this could be done by typing the depth column as a geometry and referencing the srsName to the lrm???. Anyway, we need to fix this to get geophysics and static cone tests to work right - but this is 1.2.

b) All property type elements that now are contained within the above features should probably be converted to objects. I believe that is the case for most of these already, if not all. Getting this issue buttoned up is a 1.2 exercise. I don't think it's necessary and productive to hold up a 1.1 release for this at the present.

IF THERE IS AGREEMENT, I'LL MAKE ANY REQUIRED CHANGES TO THE INPUT SCHEMAS THAT DAVID WILL SEND ME TO GET THE FEATURE CLASSES ABOVE PROPERLY TYPED AND REFERENCED TO THE APPROPRIATE BASE TYPES, AND CARRY OUT THE OTHER SUGGESTIONS I MENTIONED ABOVE FOR 1.1.

2) Metadata

The following Diggs objects can be classed as metadata:

- 1) AssociatedFiles
- 2) Roles
- 3) Remarks
- 4) Specifications
- 5) Equipment
- 6) BusinessAssociates - this is a weird one. BusinessAssociates are physical features, and can exist on their own, but within Diggs' context, Business Associates are properties of roles, contracts and other metadata objects, they are not direct properties of the Diggs features themselves.
- 7) Contracts

For 1.1: Type the above objects as metadata (change to derive from AbstractMetadataPropertyType).

As I discussed in my prior long e-mail from Monday, I suggest that metadata properties as above be placed in the schema more judiciously than before to avoid unnecessary and irrelevant use of these properties, and to avoid recursion. I suggested that we create two element groups - one that contains properties for associated files, roles and remarks (or if we go with option 4 from my earlier e-mail, remarks would be a separate element group. This group (FeatureMetadataProperties) is assigned to base Feature types only. Remarks would be assigned to base object types, only. Specification and equipment properties would be ganged into another element group (TestMetadataProperties) and attached to base test types (which also would carry the feature metadata as well). Other features that might use equipment, say, (like a hole) would have those properties added to the concrete feature type only.

IF THERE'S AGREEMENT, I'LL MAKE THE TYPE CHANGES AND MODIFY THE FEATURE AND OBJECT BASE TYPES IN THE INPUT SCHEMA FILES TO IMPLEMENT THIS FOR 1.1.

3) Referencing

In keeping with Chris' desire to keep as much stuff inline as possible, for 1.1 we should eliminate as many referencing properties as possible and require inline coding of property elements. Chris discussed the value of this with remarks and WFS implementation and I provided some options that Loren, David and I discussed to do this. There are some metadata properties, though, that I can see should be referenced, because to do otherwise will likely result in lots of duplication of data within instances. These "shared" objects are AssociatedFiles, Roles,

Specifications, Equipment, and BusinessAssociates. So, properties that include these objects should exclusively use gml:ReferenceType to point to an external object that would sit at the root level.

Since the above objects would be referenced only, those data would have to be transmitted in a Diggs object along with the features that refer to them - something Chris wanted to avoid. So perhaps it would be worth revisiting stringOrRefType again for these properties (not for remarks). I envision many use cases where references to equipment, people, and specifications would be a simple string (eg. a person's name, or a standard test spec, or the name of a piece of equipment), and therefore use of stringOrRef would allow compact encoding and that information to be included inline and not referenced when detailed specs or business associates are not called for. I mentioned that stringOrRefType is being deprecated in 3.2, but we could create an identical type in the diggs namespace to do this.

Recommendation: For 1.1 any properties of features/objects that identify associated files, roles, specifications or equipment be only of type gml:ReferenceType. We can continue the discussion about whether we'd want to resurrect stringOrRef type for these properties.

Also for 1.1, to avoid any mandatory elements having to be "made up" to wrap a transmission of Diggs features in the Diggs object, add a mandatory version attribute to the Diggs element. Modify the transmissionInformation property of Diggs to be optional, and remove the version property from the TransmissionInformation object.

IF THERE'S AGREEMENT, I'LL MAKE THIS CHANGE TO THE INPUT SCHEMAS. THEN SEND TO DAVID TO RUN HIS SCRIPTS ON AND TEST TO MAKE SURE WE HAVE A VALID, GML COMPLIANT APPLICATION SCHEMA. VERSION 1.1 WILL NOT BE AN IMPLEMENTABLE SCHEMA QUITE YET, BUT IT WILL BE FAR EASIER FOR THE CORE SIG AND OTHERS TO BEGIN TO EXAMINE AND PROVIDE INPUT TO A 1.2 VERSION.

4) GML profiling of geometry types

Our gml4diggs profile has restricted base GML object and feature types so that all GML objects do not inherit unnecessary gml elements. However, such restrictions have not been placed on gml geometries. GML geometries inherit from unrestricted gml base types and all of the elements come over into our types. David - for 1.1 can you restrict gml: AbstractGeometry to remove the gml:StandardObjectProperties? Is doing so unwise?

E.18 Teleconference Meeting Notes 2010-04-15

Date: April 15, 2010

Time: 7:30 AM – 11:00 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti
Chris Bray

Not Available: n/a

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- First 1.5 hours – meeting for release of v1.1.
- **ACTION:** Burggraf needs to reconcile remaining v1.1 changes that Ponti sent this week.
 - Needs to generate the instances to make sure they are valid.
 - **Inline and referenced elements caused issues with the examples.**
- Can features be transmitted on their own?
 - Doesn't have to be at root level to be GML compliant.
 - Ponti suggests having all features at the root level and use referencing.
 - For example the hole is at the root level and references the project.
 - If layers aren't going to be transmitted separately, maybe it should be an object rather than a feature.
 - Burggraf – WFS can serve any feature. Doesn't have to be wrapped in the DIGGS element. WFS can be set up to transmit the parent object too, or some filtered version of the WFS.
 - If all features are moved to root level.
 - If a feature is referencing a feature in a child-parent relationship, probably need to reference each other (with a "back pointer"), or decide on a case by case basis.
 - Advantages of nesting:
 - Don't have to rely on referencing.
 - Don't need schematron to validate.
 - Quicker to parse.
 - Ponti suggestions:
 - Probably want to maintain as much hierarchy as we can allow.
 - Where should we use referencing?
 - Holes to projects
 - Samples to holes
 - Where should we keep hierarchy?
 - Layers
 - Insitu tests – some of them(?)
 - Should probably engage broader group in this discussion. Meeting might include Roger Chandler, Chris Power, Marc Hoit
 - Burggraf suggests using the analog of a website to explain the concept of feature and objects.
 - **ACTION:** Turner set up meeting with Core SIG to suggest that an XML technical group convene to discuss this further.
- Encode tabular data in the most compact way possible.
 - **ACTION:** Burggraf will come up with recommendation.
 - Will probably stick to something close to what we have.
 - Can produce script for GML to KML to show how it works.
- Need to review Galdos contract and work with Burggraf to estimate hours needed for 1.2 and 2.0.
- Versioning:
 - Starting with v2.0, implement versioning convention.
 - Use 1.x series during development (1.1 and 1.2) even though it doesn't conform to the new convention.
 - Consider renaming the 1.2 as 1.5 to imply the significant changes.
 - Changes from 1.2 to 2.0 likely to be small.

E.19 Teleconference Meeting Notes 2010-04-22

Date: April 22, 2010

Time: 7:30 AM – 9:00 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti
Chris Bray

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Ponti and Burggraf have been corresponding this past week and resolving issues with the schema.
- Burggraf currently working through the 20 example instance documents and identifying validation errors.
 - Order of elements.
 - Deprecated elements
- Ponti suggests needing to parse out work:
 - Every element needs a definition
 - Need annotations in the schema
- Defining geometry of hole – call it “linearExtent” or “centerline”?
 - linearExtent is used for project geometries
 - centerline is commonly used in geotech practice.
 - Consensus to go with the term centerline
- We will use GML 3.3 for some of the geometry elements in the future.
 - Linear referencing would reference GML 3.3.
 - For now in v1.1:
 - Move linear referencing to separate schema file.
 - Use different prefix for it.
 - Use the namespace prefix “G3.3” as a placeholder (different from kernel).
 - Instance documents would reference both the GML 3.2 and G3.3 namespaces.
 - This schema will become the profile for GML 3.3.
- Timeline for Burggraf:
 - Schemas are mostly done today.
 - Implement URN pattern restrictions in schemas and in examples.
 - Change scripts.
 - 1-3 days.
- **ACTION:** Turner to set up a new forum on DIGGS site for punchlist for v1.2 issues.

E.20 Teleconference Meeting Notes 2010-04-29

Date: April 29, 2010

Time: 7:30 AM – 9:00 AM (PST)

Participants: Loren Turner
David Burggraf

Not Available: Dan Ponti
Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Update from Burggraf
 - Emailed latest v1.1 schemas yesterday.
 - Also sent all 20 examples converted and validated.
 - Updated the Example 16 schema to extend DIGGS.
 - Made all changes discussed in last couple of weeks.
 - URNs
 - Centerline
 - GML 3.3 profile
 - New position element
- **ACTIONS:** Burggraf to:
 - Update DIGGS namespace in example instance docs to 1.1.
 - Move declaration of gml 3.3 to top.
 - Will post process and clean up multiple namespace declarations that Oxygen put everywhere.
 - Repackage the v1.1 and send out.
- **ACTIONS:**
 - Ponti, Turner, Bray review the 0.98 zip package and give feedback to Burggraf.

E.21 Teleconference Meeting Notes 2010-05-13

Date: May 13, 2010

Time: 7:30 AM – 8:00 AM (PST)

Participants: Loren Turner
David Burggraf
Chris Bray

Not Available: Dan Ponti

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Update from Burggraf

- Cleaning up final changes to examples from Turner's comments.
- Should complete examples by Friday and send final package to group by Tuesday.
- Start developing URN RFC for IANA.
 - Need to identify the group of people and contact info (3 to 5 people).
 - Need naming authority.
- Bray to post v1.1 schemas on Tuesday.
- Turner to post announcement and v1.1 schemas, examples, codelists, and documentation on diggsm website on Tuesday.
- Bray has presentation to AGS on v1.1 on May 25.

E.22 Teleconference Meeting Notes 2010-05-20

Date: May 20, 2010

Time: 7:30 AM – 8:00 AM (PST)

Participants: Loren Turner
David Burggraf
Dan Ponti

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Approach to deal with codelists for v1.2.
 - Identify elements that have codespace attributes.
 - Check these elements to make sure they have a codelist created already.
 - Make sure that the codelists are annotated properly.
 - Identify elements that use enumerated lists. Should those be codelists instead?
 - **ACTION:** Burggraf will create list as an excel file.
 - **ACTION:** Turner set up meeting(s) with Core SIG members.
- Need to get Hoit to start looking at the piling schema.
- Should we try to standardize a format for tests? Have more continuity in form.
 - For example, organize elements in groups:
 - Metadata object
 - Results object
 - Burggraf will share examples from GeoSCIml.
- Proposed structural changes for DIGGS v1.2:
 - **ACTION:** Burggraf will create example demonstrate how it looks using coverage model encoding. Ponti suggests using the CPT example file (Example 02 - CPT Final.xml). Ponti, Turner, Bray, and Burggraf discuss this further after looking at example to decide if this should be presented to the Core SIG.
 - **ACTION:** Create examples showing adding solid and composite area geometries to project. Burggraf will create example.
 - **ACTION:** Sample and Sampling Activity – separate the physical sample from the activity that produces it. Ponti will work with Burggraf to create scripts to implement changes.

- **ACTION:** Layer systems. Ponti will work with Burggraf to create scripts to implement changes
- Namespace issue – hold off on this for now.
- Divorce position information at locations from results of tests and observations. Ponti to consider this further. Need to decide where position information goes. Discuss further next week.
- Approach for dealing with structural issues:
 - **ACTION:** Turner set up meeting(s) with Core SIG and software vendors for first week of June.
 - Decide on three issues:
 - Solid and composite area geometries to project
 - Sample and Sampling Activity
 - Layer systems
 - Present examples to demonstrate impacts of changes
 - Goal of meeting is to make decision on whether to adopt change or not.
- **ACTION:** Turner go back through notes and emails from v1.1 development phase and compile punchlist.

E.23 Teleconference Meeting Notes 2010-05-27

Date: May 27, 2010

Time: 7:30 AM – 10:30 AM (PST)

Participants: Loren Turner
David Burggraf
Chris Bray
Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Discussion on codelist work:
 - For codelists, focus first on:
 - Kernel
 - Geotechnical
 - Monitoring
 - **ACTION:** Burggraf update codelist:
 - URN codes to include authority.
 - **ACTION:** Burggraf to revise spreadsheet to populate:
 - Parent Element
 - Documentation
 - “Codes” worksheet
- Ponti presented work underway with schema changes.
 - Features are at DIGGS root level.

- Features cross-reference each other. For example, projects point to locations, and locations point back to projects.
- **ACTION:** Ponti will send out later today.
- **ACITON:** Burggraf will:
 - Review the schema
 - Check for GML compliance
 - Generate 4 converted examples – use Example files 2, 3, 6, 11, 18

E.24 Teleconference Meeting Notes 2010-06-03

Date: June 3, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner

David Burggraf

Chris Bray

Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Update from Burggraf – Finishing up work on codelist Excel spreadsheet and will send to us in the next couple of days.
- Ponti briefed us on the changes made to the proposed v1.2a. He sent the schemas and some documentation to the group this week.
- Ponti and Burggraf are working on the encoding for trenches.

E.25 Teleconference Meeting Notes 2010-06-10

Date: June 10, 2010

Time: 7:30 AM – 9:30 AM (PST)

Participants: Loren Turner

David Burggraf

Dan Ponti

Scott Deaton

Salvatore Caronna

Roger Chandler

Cliff Roblee

Agenda: DIGGS software vendors meeting

Notes:

The notes below were taken from an email sent by Ponti to the group on 6/10/10, and further edited by Turner to include figures from presentation materials used during the meeting. Key decisions on elements of the proposed DIGGS v1.2a were made following discussion during the meeting. A summary of that discussion and decisions are documented throughout.

Following a discussion we had with software vendors a couple of months ago, we have implemented some new structural changes in an test version of DIGGS v1.2 for evaluation (v1.2a). While the recently released DIGGS v1.1 corrects GML and other schema complexity issues that made the original version of DIGGS very difficult to work with, this 1.2a makes some

significant structural changes in schema organization that have relevance with respect to how the schema is used in practice and mapped to. The intent of these changes were to:

- Allow from mapping to/from databases to be simpler and more direct and understandable
- Allow for easier transfer of partial information associated with a project or a location
- Allow for easier extension of the schema to include additional location types (places where data are collected) and associated observations and tests.

Some of the groundwork for the above was laid in v1.1 with changes made to abstract data types. The main purpose of the conference call is to get input from you as to whether the proposed changes we're making appears to be helping to achieve the above objectives.

Quick Primer on GML Terminology, DIGGS Nomenclature, and Data Organization

DIGGS is a *GML application schema* which means that all schema elements must derive from abstract GML data constructs and follow GML's object-property rules, which governs how schema elements and XML instance documents are constructed. GML describes the world in terms of geographic entities called *features*, although not all features must contain geometry properties. Features are simply complex data types that hold properties and geometries about the entities they represent. In DIGGS v1.2, the GML features are real world entities (eg. holes, samples, etc.) or processes - that "stand alone" and do not depend on other features for their definition and context. GML *objects* are structurally the same as features - they are collections of properties and geometries as well, but typically do not occur out of context with the features they relate to. In DIGGS, objects are really just complex properties of features (a complex property is one that contains a number of simple properties). For example a layer system defining soil descriptions is a DIGGS feature, whereas the individual layers contained within a layer system are objects. Properties are simple attributes of a feature. For example, a numeric result of a test is a property of the test feature. *Metadata objects* are specially typed in GML to define objects that provide ancillary information about features and objects. Generic GML applications can extract metadata from GML instances or handle that information differently because of the special typing. In DIGGS, things like Remarks and Associated Files are typed as metadata.

In DIGGS v1.1, we defined 9 classes of features (as shown in Figure 1 below), each of which derives from its own common base type. In v1.2, the goal is to formalize this organization so

that all features fall within these classes and derive from their "class" base types. In v1.2a, most current features have been so categorized, but there are some in the Piling, Environmental, and Monitoring namespaces that still need to be assigned to a class, reclassified into objects, or possibly deleted.

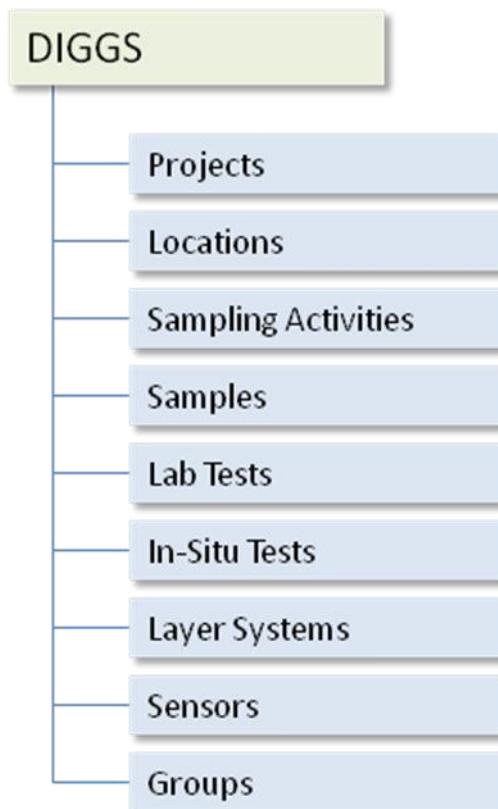


Figure 1 – There are 9 top-level feature classes in DIGGS v1.2a.

The nine feature classes are:

- **Projects** - business activities that collect, compile, and process information from locations [Process]
- **Locations** - real world places and constructions from which observations are made, samples are collected, or tests are run. [Entity]
- **Sampling Activities** - the process of sample creation or collection [Process]
- **Samples** - earth material, fluids, or gases collected or created for observation and testing [Entity]

- **Layer Systems** - ordered interval observations or interpretations of earth materials, properties or features at a location [Entity]
- **Laboratory Tests** - analyses performed on samples collected from locations, or created via a sampling activity [Process]
- **In-Situ Tests** - analyses or observations at a location [Process]
- **Sensors** (I'd suggest a nomenclature change to Installations or Monitoring Installations) - equipment or devices installed at locations that collect repeated measurements or observations [Entity]
- **Groups** - collections of projects, locations, samples or groups of these, for the purpose of providing meaningful context to observations and measurements.

All features in DIGGS carry a mandatory id (*gml:id*), required by GML and used for referencing and linking with other features. All features also carry a mandatory identifier (*gml:identifier*), required by DIGGS, which is a globally unique key for the feature, and uses a URN pattern. Optional properties of all feature classes include status (needs clearer definition), description, and associated file, role, and remarks metadata objects.

Discussion and Recommendation for Change to v1.2 Proposal

Concerns were raised about the requirement for a *gml:identifier* for all features. For some software developers, this represents a significant change to their backend software structure with the need to generate and retain the *gml:identifier* in their data structures and implemented in user interfaces. There were additional concerns about generating *gml:identifiers* for features that typically aren't identified (e.g. layers) versus those that typically are identified (e.g. borehole, samples). It appears that the issue of use of globally unique identifiers is also unresolved within the AGS standard.

As such, the consensus from the group was to make the *gml:identifier* an optional item. This approach provides the most flexibility in applications and appears to meet all stakeholder needs.. Users that don't have a need to construct full data sets from various component data sets from others can use DIGGS without identifiers. DIGGS files without identifiers will have internal integrity and uniqueness within the context of that single file, however, cannot be combined with other DIGGS files which may contain identical identifiers causing conflicts.

Projects, Locations, Samples, Layer Systems, Sensors, and Groups are "named" features. In addition to the properties above, they also carry a mandatory name property.

All objects (complex properties of features) must carry a mandatory id; optional properties of all objects are description, status, and the remarks metadata objects. Some metadata objects are named (eg. equipment and specifications), and carry a mandatory name property.

Metadata objects currently defined are:

- **Document Information** - information about the specific XML instance document
- **Associated Files** - references to non-XML documents or records outside of the XML instance
- **Business Associates** - persons and institutions
- **Equipment** - well, you know, equipment :-)
- **Specifications** - test specifications or procedures

Organization of features in an XML instance.

DIGGS v1.0 and v1.1 represents the associations between features in a hybrid form - both in a fixed hierarchical tree structure with some features that sit outside the tree and relate to features in the tree by reference properties.

In DIGGS v1.2, all features are global, and associations between features are defined by specific reference properties that carry both the id and the globally unique identifier of every object it associates with. All features are organized in collections at the root level of the document according to their classes. For example, consider a single project that has one borehole:

```
<DIGGS>

  <projects>

    <Project gml:id ="A"/>

      <gml:identifier codespace =
```

```
usgs">urn: DIGGS: def: fi: USGS: usgs\_A</gml: identifier>

<originalLocationRef xlink:href="#1"
identifierRef="urn: DIGGS: def: fi: USGS: usgs\_1" />

<Project gml:id = 'B' />

</project>

<locations>

<Borehole gml:id = "1">

<gml: identifier codespace =
"usgs">urn: DIGGS: def: fi: USGS: usgs\_1</gml: identifier>

<originalProjectRef xlink:href="#A"
identifierRef="urn: DIGGS: def: fi: USGS: usgs\_A" />

</Borehole>

<TrialPit ... />

</locations>

...

</DIGGS>
```

Note, in this example:

- The Project and the Borehole are at the same level in the DIGGS hierarchy.
- The Project “A” carries a reference to the Location/Borehole “1”.
- Location/Borehole “1” carries a reference to Project “A”.

Another way to look at this without the need to interpret the XML syntax is shown in Figure 2. In this example Project “A” contains Borehole “1” where two samples were taken.

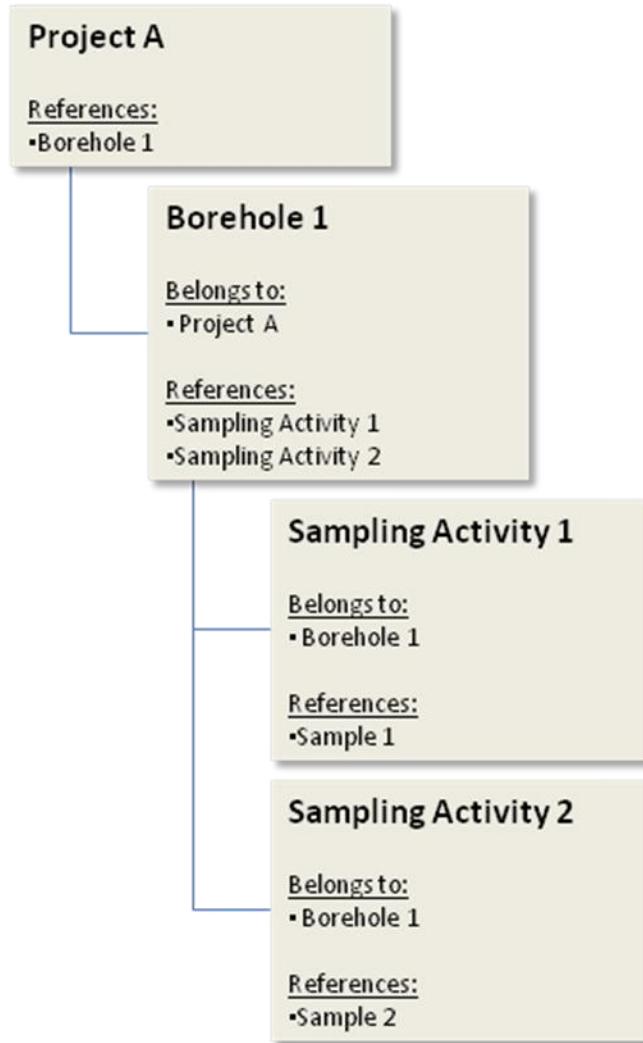


Figure 2 – Example implementation of feature associations.

This reason for organizing features in this manner is that this structure both maintains strict two-way associations between features (i.e. parent-child associations are preserved where appropriate), while allowing individual feature classes to be transmitted in XML instance documents without having to also transmit information about the objects they are associated with, except for their id's and identifiers. The reason that the referencing properties carry both the id and identifier is that an xlink:href may not actually resolve to a real object in an XML file somewhere (it could reside in a database instead), but the global identifier (the key) is still maintained in the referencing feature, so that associations can link up later, in a database or processing software. This allows flexibility in business practice and reduces redundancy and requirements to update records in databases when there is no need to do so. When a hole is drilled, the driller can transmit the hole information in one instance file, next the geologist can independently transmit the layer descriptions with a reference to the hole, and finally, the logging company can deliver the geophysical log with a hole reference - all in separate instance documents but the information can be compiled together (and transmitted later together in a single instance document as part of a final report) because they will all carry references to the

hole feature that they associate with. This can be done with or without a requirement to send along project information.

Figure 3 shows an illustration of how the feature classes are associated in DIGGS.

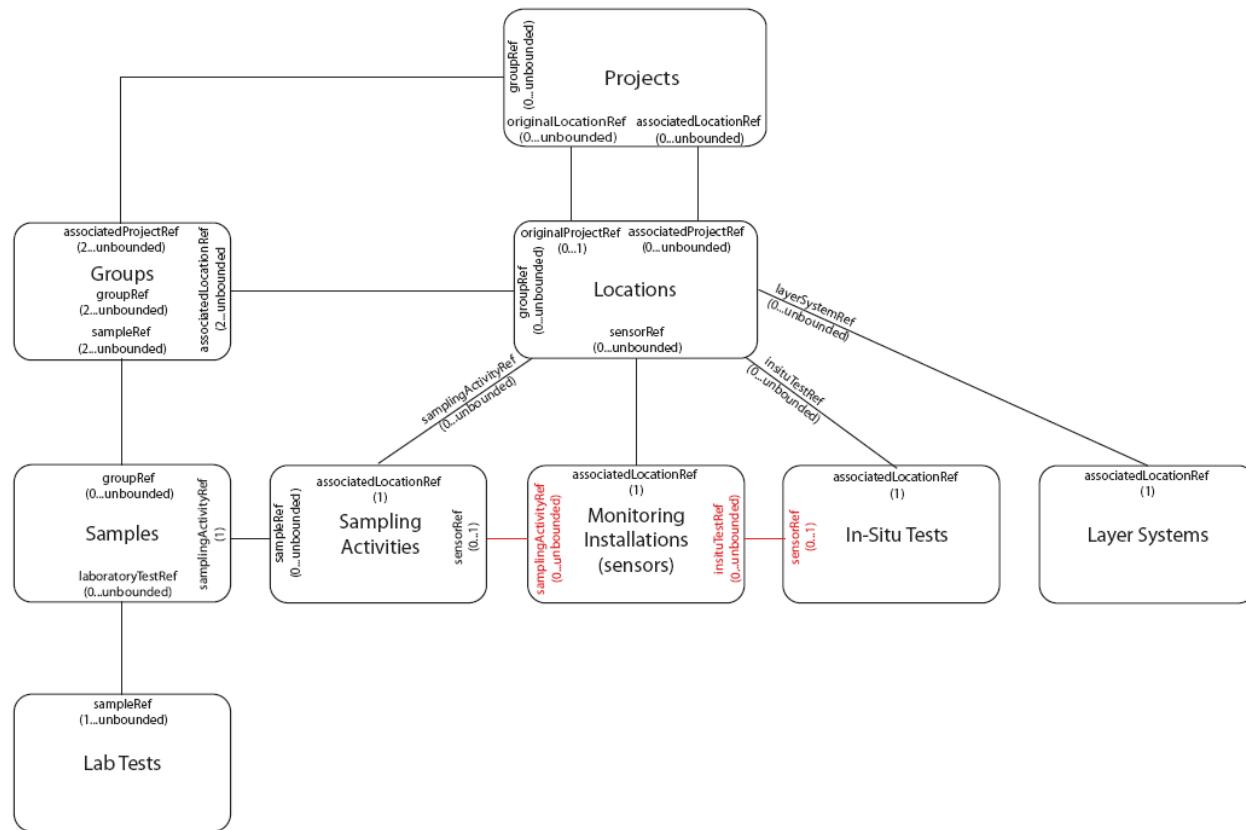


Figure 3 – Proposed feature class associations in DIGGS v1.2a.

Discussion and Recommendation for Change to v1.2 Proposal

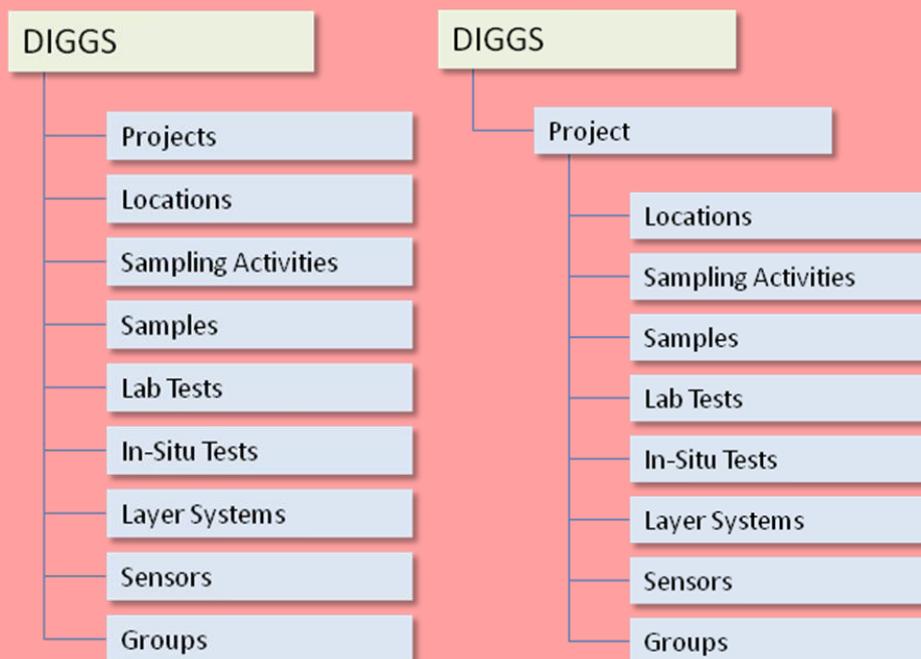
Concerns were raised regarding the “atomic” nature of the feature classes. On one hand, the ability to transmit feature data independently had merit for specific use cases (e.g. blind testing in environmental applications). On the other hand, the lack of a project reference in an independently transmitted DIGGS feature would cause more significant issues when compiling information from various DIGGS files.

A suggestion was made to include a mandatory reference in all features to the parent project.

Although this solves the problem described above, it creates the need for a project reference in all of the features.

A second suggestion was made to make “Project” a higher level feature, thereby implementing a hierarchy in the DIGGS structure. This would assure that Project data would be present in all DIGGS instances due to the hierarchical structure.

The proposed v1.2a structure would be reorganized from that shown on the left, to that shown on the right:



The AGS had adopted a single project hierarchical structure in its new format. In general, the geotechnical software vendors use a structure more like this and is more common in practice. Adopting the single-project hierarchical structure would likely be better received by the DIGGS stakeholders.

By implementing this approach, however, the ability to reference multiple projects is lost. This feature could be retained to some degree with an optional element within each feature to point to other source projects. This would be an informational element and not a structural

one.

E.26 Teleconference Meeting Notes 2010-06-24

Date: June 24, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
David Burggraf
Chris Bray
Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Discussed the implications of making the change to the DIGGS structure where the major features were moved under the Project feature in a hierarchy. As we discussed this further, it became apparent that this particular change would have implications to XML data mapping methods, and would limit backwards compatibility with subsequent DIGGS revisions if the standard, at some point, is extended to accomodate more varied use cases.
- Two major requirements of the DIGGS structure (with respect to Projects) from the software vendor's meeting:
 - Project information must be included in every DIGGS file.
 - All features (samples, holes, etc.) must be "under" a single project in a DIGGS file.
- Given this, we propose to:
 - Retain the structure originally presented at the meeting where all features are at the root level.
 - Make the Project feature mandatory in all DIGGS instances.
 - Restrict Project feature so that only one Project is permitted in a DIGGS file.
 - All features in the DIGGS file must fall "under" the single project (via href).
- Functionally, this achieves the same thing that the hierarchy does, and technically, it keeps the overarching DIGGS structure consistent.

E.27 Teleconference Meeting Notes 2010-07-01

July 1, 2010

Time: 7:30 AM – 9:30 AM (PST)

Participants: Loren Turner
Dan Ponti
Scott Deaton

Salvatore Caronna
Roger Chandler
Cliff Roblee

Agenda: 2nd DIGGS software vendors meeting

Notes:

A summary of meeting agenda items discussed is as follows:

- Recap from last meeting on June 10, 2010:
 - Gml:identifier is optional.
 - Project was moved to top hierarchy.
- Revised recommendation on Project hierarchy structure.
 - Retain the structure originally presented at the meeting where all features are at the root level.
 - Make the Project feature mandatory in all DIGGS instances.
 - Restrict Project feature so that only one Project is permitted in a DIGGS file.
 - All features in the DIGGS file must fall "under" the single project (via href).
- Review remaining proposed v1.2 changes:
 - Location Features
 - Sampling and Sampling Activity

The notes below were taken from an email sent by Ponti to the group on 6/10/10, and further edited by Turner to include figures from presentation materials used during the meeting. A summary of that discussion and decisions are documented throughout.

Organization of features in an XML instance.

Discussion and Recommendation (updated 7/1/10)

Concerns were raised regarding the “atomic” nature of the feature classes. On one hand, the ability to transmit feature data independently had merit for specific use cases (e.g. blind testing in environmental applications). On the other hand, the lack of a project reference in an independently transmitted DIGGS feature would cause more significant issues when compiling information from various DIGGS files.

A suggestion was made to include a mandatory reference in all features to the parent project. Although this solves the problem described above, it creates the need for a project reference in all of the features.

A second suggestion was made to make “Project” a higher level feature, thereby implementing a hierarchy in the DIGGS structure. This would assure that Project data would

be present in all DIGGS instances due to the hierarchical structure.

During the weekly DIGGS/Galdos status meeting following the 1st software vendors meeting, the team discussed at length the implications of making the change to the DIGGS structure where the major features were moved under the Project feature in a hierarchy. It became apparent to the team that this particular change would have implications to XML data mapping methods, and would limit backwards compatibility with subsequent DIGGS revisions if the standard, at some point, is extended to accommodate more varied use cases.

We recalled two major requirements of the DIGGS structure (with respect to Projects) from the meeting:

- Project information must be included in every DIGGS file.
- All features (samples, holes, etc.) must be "under" a single project in a DIGGS file.

Given this, we propose to:

- Retain the structure originally presented at the meeting where all features are at the root level.
- Make the Project feature mandatory in all DIGGS instances.
- Restrict Project feature so that only one Project is permitted in a DIGGS file.
- All features in the DIGGS file must fall "under" the single project (via href).

Functionally, this achieves the same end that the hierarchy does, and technically, it keeps the overarching DIGGS structure consistent.

The AGS had adopted a single project hierarchical structure in its new format. In general, the geotechnical software vendors use a structure more like this and is more common in practice. Adopting the single-project structure with an implied hierarchy would likely be better received by the DIGGS stakeholders.

Location Features

In v 1.1, the Hole feature was somewhat generic, it was set up to handle properties of a geotechnical borehole, but could be other types that were of similar geometry but not really boreholes (eg. a transect or trial pit). In 1.2a, there are now specific Location features that derive from either AbstractPointLocation, AbstractLinearLocation, or AbstractPlanarLocation location feature types, each of which have properties specific to that type of feature. This way, we can model more types of features in a straightforward fashion in the future, such as embankments, tunnels, roads, etc.

Diggs 1.2a now has defined in it the following location features:

- Borehole - very similar to Hole in v 1.1, but modified to handle 1.2 constructs. (Linear Location)
- Trial Pit - a shallow excavation - legacy to handle current AGS trial pit constructs. (Linear Location)
- Trench Wall - designed to supplant Trial Pit in the future. A wall of a trench or pit represented by a vertical planar surface. (Planar Location); in 1.2a this is only partially built
- Station - a point on the earth's surface (Point Location)

Discussion and Recommendation (7/1/10)

The group concurred that these four types are about the right breakdown for location feature types. CPT would be considered a borehole.

Since all of the above location features could/would be utilized by the geotechnical, environmental, and even piling disciplines, these concrete location feature types are placed in Kernel (main Diggs namespace) in 1.2a.

Note, 1.2a has left over from 1.1 three concrete location features that are not yet classed as either point, linear, or planar and need to be modified to bring these in line with the other locations. They are:

- diggs_mon:MonitoringLocation
- diggs_pil:FoundationGroup
- diggs_pil:FoundationGroupInstance

Samples and Sampling Activities

Diggs 1.2a takes the v1.1 Sample feature and bifurcates the various properties into two features - Sample and SamplingActivity. Sample only contains properties describing the physical aspects of the sample and its chain of custody. The Sample feature contains a mandatory

samplingActivityRef property that identifies the activity that produced it, as well as references to lab tests subsequently conducted on it. SamplingActivity features carry properties on the location and position of the sample, the methods of sample creation, the sampling environment, and equipment used.

The reasons for doing this are as follows:

- Creates logical separation between the process of creating a sample and the physical sample itself.
- Completely isolates creation information from the Sample, ensuring that samples can be transmitted with no information that might reveal its sources or source process.
- Creates a single referencing property from a sample to its creating process and henceforth up the hierarchy - instead of the generic source property in 1.1 that could point to either location or sample targets.
- Allows a sampling activity to be reported where an actual sample is not produced (no associated Sample feature need exist).
- Multiple samples can be created from one activity without having to duplicate sampling process information for each associated sample.
- Easier handling of subsamples and aggregates.

Samples can be created (the activity) at either locations (collected) or as a result of a business activity (eg. a sample blank or an aggregate sample from several locations). In order that there be no ambiguity in what a sampling activity references, an unidentified location feature has been defined in Diggs 1.2 (has no geometry), that allows the sampling activity to reference a project via a location (without a real location feature actually having to exist). This carries a bit of overhead in the xml, but ensures unambiguous associations. A similar construct is also employed that lets a test result associate to a position at a location via inferred or virtual sample and sampling activity features.

Discussion and Recommendation (7/1/10)

The proposed structure of splitting a single sample feature into two, a sample and the activity of sampling, was discussed. Observations and comments:

- Typical use case is to combine information on the sample with the information on the collection of that sample. That is, the sample and the activity are typically considered a single collection of information.
- If implemented as two separate features, most current software would need to deconstruct sample data when mapping to DIGGS, then combine the data again when importing DIGGS data. This is an extra mapping step that the software vendors would need to accommodate. With some software, this mapping would need to be identified by the user.
- In many cases the mapping would be one-to-one. That is, each sample would have a corresponding sampling activity.
- The general approach of creating separate features was supported by the group, although the implementation and acceptance by the user community was uncertain.

The group concurred on the following path forward at this time:

- Proceed with the proposal of separating *samples* and *sampling activity* as separate features.
- Solicit feedback from Earthsoft and Dataforensics on the environmental use cases and applicability of the proposal.
- Reassess the proposed structure following more testing after v1.2 deployment and use.
- Add references from samples to insitu test and lab test.

There was additional discussion about the idea of developing a database representation of DIGGS. This would likely take the form of the Excel spreadsheet proposed as part of Task 3. The spreadsheet/database wouldn't represent all possible DIGGS data, but would provide a simplified tool to demonstrate mapping to and from DIGGS for typical data. This would also make assessing the implications of sample/sampling activity features more tangible. However, a database/spreadsheet representation of a DIGGS instance shouldn't be presented to the community as a representative of the entire standard, or as a model for setting up a database.

E.28 Teleconference Meeting Notes 2010-07-06

Date: July 6, 2010

Time: 7:30 AM – 9:30 AM (PST)

Participants: Loren Turner
Dan Ponti
Scott Deaton
Salvatore Caronna
Chris Bray
Scott Weaver

Agenda: 3rd DIGGS software vendors meeting

Notes:

A summary of the meeting is as follows:

- Recap of key actions/decisions from last two meetings (June 10 & July 1, 2010):
 - *gml:identifier* is optional.
 - The *Project* feature will be mandatory in all DIGGS instances and will be restricted so that only one Project is permitted in a DIGGS file. All features in the DIGGS file must reference the single project.

- Implement the four types of location features as proposed – *borehole*, *trial pit*, *trench wall*, and *station*.
- Implement *Sampling* and *Sampling Activity* as separate features as proposed
- Key actions/decisions from this meeting (July 6, 2010):
 - Implement layer systems as proposed and use codelists to define constituents.
 - Implement the GML coverage model for encoding CPT, geophysical, and similar data types. Use the profile to restrict use to MultiPointCoverage element only.

The notes below were taken from an email sent by Ponti to the group on 6/10/10, and further edited by Turner to include discussion notes from the meeting.

Layer Systems

Layer systems become more specific to make data mapping to DIGGS easier and to provide more specificity in coding. The Layer system object now carries mandatory properties that indicates which type of layer system it is, and that layer system's subtype. Layer system types come from an enumerated list hard-coded into the schema; sub-types are code types that would be defined in referenced code lists. The types of layer systems (enumerations) are:

- *Color* – describes the color of materials encountered
- *Component* – describe details of earth materials encountered
- *Discontinuity* – describes fractures and joints and their spacing
- *Lithology* – describe the earth materials encountered
- *Orientation* – describes the geometry of vectors or planar surfaces encountered at a location, such as bedding, joints, cross-beds, etc.
- *Other* – describes a layer system of unknown type, using name-value pairs.
- *Property* – describes a layer system where the results are simple text or numeric values - usually interpreted as a result of some lab or in-situ test (eg. porosity).
- *Stratigraphy* – describes ordered bodies of rock or soil, such as formations, biostratigraphic units or aquifers.

The layer objects that are properties of layer systems are defined separately for each type of layer system. For example, a layer system of type lithology has a layer of type LithologyLayerType, which has properties unique to lithology layer systems.

Note - Discontinuity layer systems replace the discontinuity and fracture spacing properties of the Hole feature in v 1.1.

Discussion and Recommendation (7/6/10)

The group concurred that the constituent types (e.g. density, consistency, moisture, plasticity, etc. in the Lithology layer structure) should be based on codelists rather than enumerated lists, since differences in practice in the community may require different types of constituents with different meanings.

There was general consensus to support the proposed layer system construct.

Positions of Sampling Activities, Layers, Tests and Sensors at Locations

Diggs v 1.1 uses point properties of top and base to describe the positions of observations (eg. sampling activities layers, tests, sensors) at locations. This position type is suitable only for holes and hole type features where the positions are described along a vertical (or nearly so) linear reference. This is constraining and does not allow for easy reuse of these observation features.

In 1.2a, all of these observations carry a position property type that contains a LocationPosition object that is defined by the specific type of Location feature where the observation occurs. So, position properties for a borehole feature are contained within a BoreholePosition position object. A BoreholePosition object consists of a choice of two properties:

- measuredDepth (a gml:PointProperty in 1D)
- depthInterval (gml:CurveProperty in 1D).

A TrenchWallPosition object consists of a choice between a pointPosition (2D Point property), linearElementPosition (2D Curve property), or a surfacePosition (2D surface property).

For example, a layer described in a hole would contain a BoreHolePosition object in its position property - typically with a depthInterval property that defines the top and base of the layer. If the layer were associated with a trench wall, its position property would contain a TrenchWallPosition object to define its position on a trench wall in a 2D reference system; this would most likely be a surface property (polygon) that defines the exposure of the layer in the trench wall. The layer feature's other properties remain exactly the same; there is no need to redefine a layer for different location features.

Coverage Model

The coverage model in a GML construct that has two main sections:

- <domainSet> – region which the values are assigned. (e.g. for CPT, this is the block of depth data.) Can be one of two types:
 - <RectifiedGridCoverage> – define top and bottom depths and the interval, spacing is regular
 - <MultiPointCoverage> – not regularly spaced depths
- <rangeSet> – contains the data values.
 - <CurveInfo> - defines the columns in the data block
 - <curveClass> - column headers from a codelist
 - <tupleList> - data block; has attributes for value separators and line separators.

Discussion and Recommendation (7/6/10)

General consensus that the coverage model approach should be implemented. For the initial implementation, use the profile to restrict use of MultiPointCoverage only, since most software will encode using the MultiPointCoverage element. The RectifiedGridCoverage element provides some file size savings, but makes encoding more complex for most cases.

Groups

In version 1.1, Groups are generic and can associate any kind of feature with generic reference properties that do not have a target defined. In 1.2a, there is now an AbstractGroupType and then several base group features developed from this, each of which can group only one type of feature. The concrete groups are:

- ProjectGroup - a group of projects
- LocationGroup - a group of locations
- SampleGroup - a group of samples
- GroupGroup - a group of groups

This construct could be extended further for even more specific group types. For example, rather than a FoundationGroup being a location feature, it probably should be group of FoundationInstances, (extension of a GroupGroup) which in turn is a group of Piles (extension of LocationGroup).

E.29 Teleconference Meeting Notes 2010-07-08

July 8, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
David Burggraf
Chris Bray
Dan Ponti

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

Review actions from software vendor's meetings over last few weeks:

- *gml:identifier* is optional.

- This is now done with v1.2.4a.
- The *Project* feature will be mandatory in all DIGGS instances and will be restricted so that only one Project is permitted in a DIGGS file. All features in the DIGGS file must reference the single project.
 - This is now done with v1.2.4a.
 - No action needed.
- Implement the four types of location features as proposed – *borehole*, *trial pit*, *trench wall*, and *station*.
 - This is now done with v1.2.4a.
 - No action needed.
- Implement *Sampling* and *Sampling Activity* as separate features as proposed.
 - This is now done with v1.2.4a.
 - No action needed.
- Implement layer systems as proposed and use codelists to define constituents.
 - This is now done with v1.2.4a.
 - No action needed.
- Implement the GML coverage model for encoding CPT, geophysical, and similar data types. Use the profile to restrict use to MultiPointCoverage element only.
 - This is now done with v1.2.4a.
 - Need to check the implementation of this.
- **ACTION:** Burggraf will run validation checks for GML compliance for these changes with v1.2.4a. Specifically, check the implementation of the coverage model (i.e. nesting structure with DataBlock, columns).

Revised structure for laboratory tests and insitu tests.

- General structure has:
 - parameters
 - results
- For example:


```
<DensityTest gml:id="d123">
    <gml:description>Average soil density for all samples at the project site</gml:description>
    <projectRef xlink:href="#p1"/>
    <sampleRef xlink:href="#s321"/>
    <parameters>
      <diggs_geo:DensityTestParameters>
        <samplePreparation>cut 1 inch block from sample for test</samplePreparation>
      </diggs_geo:DensityTestParameters>
    </parameters>
    <results>
      <diggs_geo:DensityTestResults gml:id="dm123">
        <diggs_geo:bulkDensity uom="g/cm3">2.07</diggs_geo:bulkDensity>
        <diggs_geo:isNatural>true</diggs_geo:isNatural>
      </diggs_geo:DensityTestResults>
    </results>
  </DensityTest>
```
- All the tests need to be converted into this new structure.
 - Domain experts will eventually need to be involved and review this. This cannot be completely automated.
 - Convert all existing test elements to <results> elements.
 - **ACTION:** Ponti will provide instructions to Burggraf on doing the initial conversion.

Description of task on codelists:

- Review the “CodeTypes” worksheet.
 - Check that all the elements are still needed.
 - Use the schema documentation to see that it's still used.

- For each element in the “CodeTypes” worksheet, check that the corresponding list is present in the “Codes” worksheet.
 - Are all possible codes present?
 - Check against earlier data dictionaries.
 - Build the codelist if it doesn’t exist.
 - Authority should reflect the standard being referenced. Use “DIGGS” if no other authority can be referenced.
- See example spreadsheet (below)
- Codes shouldn’t use spaces or special characters. Use upper camel case. (e.g. *StaticCone*)
- **ACTION:** Burggraf will regenerate spreadsheet with v1.2.4a.

	A	B	C	D	E	
1	File	Parent Element(s)	Element Name	Documentation	Type	Corresponding List
2	Geotechnical.xsd	Classification	system	The system used to describe the Object. This is derived from gml:CodeType		
3	Geotechnical.xsd	Classification	code	The value of the legend code for the pictorial representation. This is derived from gml:CodeType		
4	Geotechnical.xsd	Compaction	compactionMouldType	Compaction mould type	gml:CodeType	
5	Geotechnical.xsd	Compaction	compactionTestType	Compaction test type	gml:CodeType	
6	Geotechnical.xsd	Component	system	The system used to describe the Object. This is derived from gml:CodeType		
7	Geotechnical.xsd	Component	code	The value of the legend code for the pictorial representation. This is derived from gml:CodeType		
8	Geotechnical.xsd	CompressiveStrength	triaxialTestType	Test type	gml:CodeType	
9	Geotechnical.xsd	CompressiveStrengthDetail	failureMode	Mode of failure	gml:CodeType	
10	Geotechnical.xsd	CompressiveStrengthDetail	modulus	Modulus	gml:CodeType	
11	Geotechnical.xsd	Consolidation	consolidationTestType	Oedometer or Rowe, primary or secondary consolidation test type	gml:CodeType	
12	Geotechnical.xsd	AbstractConstructionEvent, Construct	type	The type of Group represented by this Object. TODC	gml:CodeType	
13	Geotechnical.xsd	Description	system	The system used to describe the Object. This is derived from gml:CodeType		
14	Geotechnical.xsd	Discontinuity	type	The type of Group represented by this Object. TODC	gml:CodeType	urn:x-diggs:def:code-type
15	Geotechnical.xsd	DrivenPenetrationTest	drivenPenetrationTestEquip	Type of driven test equipment	gml:CodeType	
16	Geotechnical.xsd	DrivenPenetrationTest	hammerRelease	The mechanism used to lift and drop the hammer or probe	gml:CodeType	
17	Geotechnical.xsd	DrivenPenetrationTest	hammerType	The type of hammer or drive-weight assembly used	gml:CodeType	
18		Curve	curveClass			urn:x-diggs:def:code-type
19	Geotechnical.xsd	Grading	sieveNumber	Sieve Number of This Sieve	gml:CodeType	
20	Geotechnical.xsd	Grading	type	The type of Group represented by this Object. TODC	gml:CodeType	
21	Geotechnical.xsd	Hole	type	The type of Group represented by this Object. TODC	gml:CodeType	
22	Geotechnical.xsd	InSituPermeability	type	The type of Group represented by this Object. TODC	gml:CodeType	
23	Geotechnical.xsd	LayerSystem	type	The type of Group represented by this Object. TODC	gml:CodeType	
24	Geotechnical.xsd	PointLoad	pointLoadTestType	Point load test type (A, D, L or P)	gml:CodeType	
25	Geotechnical.xsd	Pressuremeter	type	The type of Group represented by this Object. TODC	gml:CodeType	
26	Geotechnical.xsd	ShearBox	shearBoxTestType	Test type e.g. small shear box, large shear box, ring	gml:CodeType	
27	Geotechnical.xsd	StandardPenetrationTest	type	The type of Group represented by this Object. TODC	gml:CodeType	
28	Geotechnical.xsd	StaticConeTest	piezoconeType	The type of Piezocone is defined in part by the position	gml:CodeType	
29	Geotechnical.xsd	StaticConeTest	porousElementType	The type of material used as porous filter element.	gml:CodeType	
30	Geotechnical.xsd	StaticConeTest	saturationFluid	The fluid used to saturate the porous filter element	gml:CodeType	
31	Geotechnical.xsd	StaticConeTest	type	The type of Group represented by this Object. TODC	gml:CodeType	
32	Environmental.xsd	AbstractAnalysis, Analysis	basis	The Basis of the Analysis	The Basis of the Analysis	gml:CodeType
33	Environmental.xsd	AbstractAnalysis, Analysis	labType	Fixed, Mobile		gml:CodeType
34	Environmental.xsd	AbstractAnalysis, Analysis	type	The type of Group represented by this Object. TODC	gml:CodeType	
35	Environmental.xsd	DetectionLimit	constraint	Is this the Upper boundary of detection or Lower bound?	gml:CodeType	
36	Environmental.xsd	DetectionLimit	type	The type of Group represented by this Object. TODC	gml:CodeType	
37	Environmental.xsd	EnvironmentalTest	labType	Fixed, Mobile		gml:CodeType

SchemaV1.2CodeTypes - example.xlsx - Microsoft Excel

Table Tools

Home Insert Page Layout Formulas Data Review View Developer Add-Ins Acrobat Design

Cut Copy Format Painter Clipboard Font Alignment Number Conditional Formatting as Table Normal Bad Good Styles Insert Delete Format Cells

F472 DIGGS

A	B	C	D	E	F	G	External Identifier
1	Name of List	List URN	Code	Description	See Also	Authority	Code URN
442	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	135TMB	1,3,5 -	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:id_135tmb	
443	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	CFP	Chloroenvinping	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:cfp	
444	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	2346TCP	2,3,4,6 -	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:id_2346tcp	
445	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	CFM	Chloroform	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:cfn	
446	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	HARDW	Calcium	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:hardw	
447	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	B	Boron	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:b	
448	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	GMETH	Methane	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:gmeth	
449	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	PHEIDX	Phenol Index	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:pheidx	
450	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	HEXPB	Hexachlorobi	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:hexpb	
451	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	HEPPB	Heptachlorob	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:heppb	
452	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	TETC	Tetrachloroet	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:tetc	
453	Chemical Determinand	urnx-diggs:def:code-list:AGS:chemical_determinand	T12DE	Trans - 1,2 -	AGS	urnx-diggs:def:code-list:AGS:chemical_determinand:t12de	
454	Discontinuity Type	urnx-diggs:def:code-list:DIGGS:discontinuity_type	shear		DIGGS	urnx-diggs:def:code-list:DIGGS:discontinuity_type:shear	urnx-diggs:def:code-list:DIGGS:discontinuity_type:shear
455	Discontinuity Type	urnx-diggs:def:code-list:DIGGS:discontinuity_type	fault		DIGGS		urnx-diggs:def:code-list:DIGGS:discontinuity_type:fault
456	Discontinuity Type	urnx-diggs:def:code-list:DIGGS:discontinuity_type	joint		DIGGS		urnx-diggs:def:code-list:DIGGS:discontinuity_type:joint
457	Discontinuity Type	urnx-diggs:def:code-list:DIGGS:discontinuity_type	fracture		DIGGS		urnx-diggs:def:code-list:DIGGS:discontinuity_type:fracture
458	Discontinuity Type	urnx-diggs:def:code-list:AGS:discontinuity_type	shearing		AGS	urnx-diggs:def:code-list:AGS:discontinuity_type:shearing	
459	Discontinuity Type	urnx-diggs:def:code-list:AGS:discontinuity_type	faulting		AGS		urnx-diggs:def:code-list:AGS:discontinuity_type:faulting
460	Discontinuity Type	urnx-diggs:def:code-list:AGS:discontinuity_type	jointing		AGS		urnx-diggs:def:code-list:AGS:discontinuity_type:jointing
461	Discontinuity Type	urnx-diggs:def:code-list:AGS:discontinuity_type	fracturing		AGS		urnx-diggs:def:code-list:AGS:discontinuity_type:fracturing
462	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	In situ test equipment		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:insitu_test_equipment	
463	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Oil or Gas	Hydrocarbon	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:oil_or_gas	
464	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Surveying equipment		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:surveying_equipment	
465	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Forensic	Exploration	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:forensic	
466	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Drilling rig		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:drilling_rig	
467	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Stratigraphic	General	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:stratigraphic	
468	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Static probe		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:static_probe	
469	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Material Survey	Exploration	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:material_survey	
470	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Environmental	Environment	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:environmental	
471	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Geophysical	Geophysical	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:geophysical	
472	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Construction Quality	Field	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:construction_quality	
473	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Geothermal	Geothermal	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:geothermal	
474	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Geohazard	Exploration	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:geohazard	
475	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Dynamic probe		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:dynamic_probe	
476	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Chemical testing		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:chemical_testing	
477	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Excavator		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:excavator	
478	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Mixed/Other	Other type or	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:mixed_other	
479	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Geotechnical	Engineering	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:geotechnical	
480	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Laboratory index testing		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:laboratory_index_testing	
481	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Pump		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:pump	
482	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Laboratory strength		DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:laboratory_strength	
483	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Ground Water	Ground water	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:ground_water	
484	Equipment Class	urnx-diggs:def:code-list:DIGGS:equipment_class	Mineral exploration	Exploration	DIGGS	urnx-diggs:def:code-list:DIGGS:equipment_class:mineral_exploration	

E.30 Teleconference Meeting Notes 2010-07-29

Date: July 29, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Chris Bray
Dan Ponti

Not Available: David Burggraf

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Revisions from software vendor's meeting.
 - The changes have been implemented in 1.2.4b.
- Implementation of revised test data structure.
 - Skipped the piling schema.
 - Test procedure vs. test result. Consider a structure where soil/rock index properties can be transmitted with or without reference to the test procedure that produced it.
 - **ACTION:** Turner to set up meeting with Turner, Ponti, Roblee, and Ponti in Sacramento to discuss testing parameters and procedures.
- New codelist worksheet.
 - Ask Burggraf to regenerate this based on the latest changes.
 - **ACTION:** Burggraf to send revised spreadsheet based upon 1.2.4c.
- 3D compound CRS issue.
 - EPSG doesn't have many combined horizontal and vertical datums for US.
 - For UK practice, generally there is one.
 - Need to combine horizontal and vertical and create CRS.
 - DIGGS can define a common set of CRS used in US and UK practice and host these on the DIGGS site? Creates a maintenance issue.
 - Or, develop a construct within the DIGGS file that combines two SRS? Need to ask Burggraf about this. Maybe create a SRS feature that declares both. Then everything else references the gml:id for that feature.
 - **ACTION:** Burggraf assess options on this.

E.31 Teleconference Meeting Notes 2010-08-05

Date: August 5, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Implementation of revised test data structure.
 - Test procedure vs. test result. Consider a structure where soil/rock index properties can be transmitted with or without reference to the test procedure that produced it.
 - **ACTION:** Turner to set up meeting with Turner, Ponti, Roblee, and Ponti in Sacramento to discuss testing parameters and procedures.
 - Consider using the gml Observation structure:

Option 1 -- Observations and procedures are separate objects.

```
Observation1
    Shear strength (ref: procedure1)
Observation2
    Shear strength (ref: procedure2)
Observation3
    Shear modulus (ref: procedure1)
Procedure1
    Pressuremeter
Procedure2
    Triaxial
```

Option 2 -- Observations and procedures are inline.

```
Observation1
    Shear strength
    Shear modulus
    Procedure1
        Pressuremeter
```

Observation2
Shear strength
Procedure1
Unknown

Current structure.

Pressuremeter Test
Results
Shear strength
Shear modulus
Parameters
Pressuremeter test parameters
Reference to sample (mandatory)

Generic Shear Strength Observation
Results
Shear strength
Parameters
Unknown

Proposed structure.

Test Observation 1
Results
Shear strength
Shear modulus
Parameters
Pressuremeter test parameters
Position (relative to location object) (optional)
Reference to sample (optional)

Test Observation 2
Results
Shear strength
Parameters
Unknown
Position (relative to location object) (optional)
Reference to sample (optional)

Test Observation 3
Results
Blow Count
Relative Density
Shear Strength

Porosity
 Parameters
 unknown
 Position (relative to location object) (optional)
 Reference to sample (optional)

- New codelist worksheet.
 - **ACTION:** Burggraf to send revised spreadsheet based upon 1.2.4c. Will do by early next week.
- 3D compound CRS issue.
 - EPSG doesn't have many combined horizontal and vertical datums for US.
 - For UK practice, generally there is one.
 - Need to combine horizontal and vertical and create CRS.
 - DIGGS can define a common set of CRS used in US and UK practice and host these on the DIGGS site? Creates a maintenance issue.
 - Or, develop a construct within the DIGGS file that combines two SRS? Need to ask Burggraf about this. Maybe create a SRS feature that declares both. Then everything else references the gml:id for that feature.
 - **ACTION:** Burggraf assess options on this.
 - Create GML compound CRS
 - Need SR IDs for all the various combinations
 - Use WKT (Well known text) and build library

E.32 Teleconference Meeting Notes 2010-08-11

Date: August 11, 2010

Time: 7:30 AM – 9:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- **ACTION:** Create GML compound CRS.
 - Need SR IDs for all the various combinations.
 - Use WKT (Well known text) and build library.

- Ponti will identify combinations for US.
 - Determine all horizontal coordinate systems in the US
 - Combine with the three vertical datums: NAVD88, NGVD29, MSL
- Burggraf will generate compound CRS files.
- **ACTION:** Burggraf to:
 - Test the 1.2.4d schema for object-property rule.
 - Update the testinstance DIGGS file to 1.2.4d
 - Coordinate with Ponti if 1.2.4d needs to be fixed.

E.33 Teleconference Meeting Notes 2010-08-18

Date: August 18, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Discussed ideas about revising the Lab test structure.
 - Ponti, Turner, Roblee, and Hannenian (Caltrans Geotech) met yesterday to review State DOT lab testing procedures. The group identified a test data construct that appears to address the various DIGGS use cases and common practices.
 - Adopt data construct similar to OGC's "Observations and Measurement" standard.
 - The structure of the test data would look something like this:

```

Test (Observation)
  Date
  Time
  ....(other metadata)
  Reference to the Property/Result
  Reference to sample (optional)
Procedure/Parameters
  Triaxial test parameters
  Specification
  Load rate
  Pressure
  ....(other metadata)

```

Property (Result)

Shear strength
 Shear modulus
(other metadata)
 Position
 Reference to location feature
 Reference to the Test

Borehole

Reference to the Property/Result

- There would be three main test objects – **Test, Procedure, and Property**,
 - **Test** – this is the observation that contains the info about the execution of a procedure. For example, the date, time, and person that did the test.
 - **Procedure** – this has the info about the actual test itself, such as test type, specification, and any interim test results used to arrive at the final result, or property.
 - **Property** – This is the value resulting from the test that is a property of the material. This can be a measurement (3500 tsf), classification (Sandy Clay), a Boolean (is organic), etc.
- The structure would allow the DIGGS creator to transmit a property without reference to a test, common with legacy data on boring logs. (For example, shear strength of a soil at a depth with no indication of how that value was determined.)
- Multiple tests can produce the same properties.
- Multiple properties can be generated from one test.
- Schematron could be used to insure that valid properties are associated with the tests.
- For CPT data, the construct would have the coverage model within the property.

Test (Observation)

Date
 Time
 Reference to the Property/Result
 Reference to sample (optional)
 Procedure/Parameters
 CPT test parameters

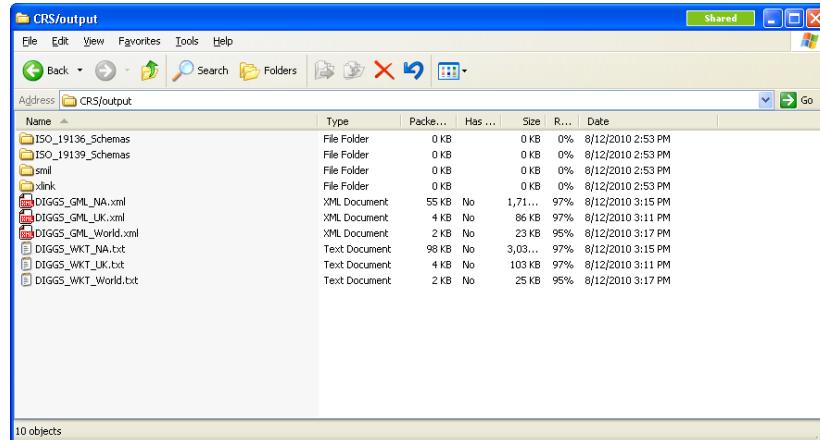
Property (Result)

Coverage model data
 Domain
 Range
 Tip
 Sleeve
(other metadata)
 Reference to location feature
 Reference to the Test/Observation

Borehole

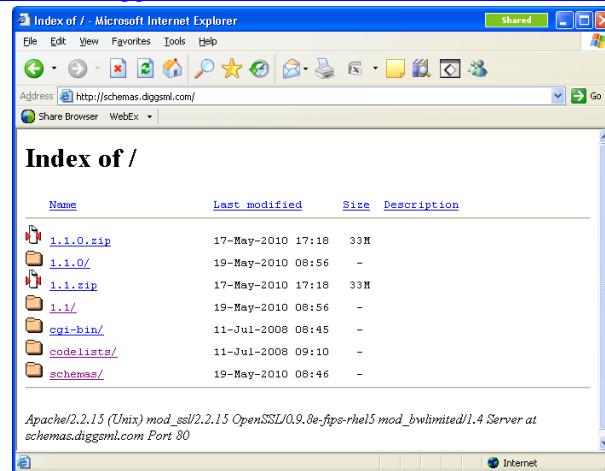
Reference to the Property/Result

- Completed creation of GML compound CRS.
 - Used WKT (Well known text) and built the library.
 - Ponti developed combinations for US.
 - Determined all horizontal coordinate systems in the US
 - Combined with the three vertical datums: NAVD88, NGVD29, MSL
 - Burggraf generated compound CRS dictionary. (Emailed to group this week.)
 - 3 files:
 - North America
 - UK
 - World
 - 2 versions of each of the three files – XML and TXT (WKT).



Name	Type	Packe...	Has ...	Size	R...	Date
ISO_19136_Schemas	File Folder	0 KB	0 KB	0%	8/12/2010 2:53 PM	
ISO_19139_Schemas	File Folder	0 KB	0 KB	0%	8/12/2010 2:53 PM	
smil	File Folder	0 KB	0 KB	0%	8/12/2010 2:53 PM	
xlink	File Folder	0 KB	0 KB	0%	8/12/2010 2:53 PM	
DIGGS_GML_NA.xml	XML Document	55 KB	No	1,71...	97%	8/12/2010 3:15 PM
DIGGS_GML_UK.xml	XML Document	4 KB	No	86 KB	97%	8/12/2010 3:11 PM
DIGGS_GML_World.xml	XML Document	2 KB	No	23 KB	95%	8/12/2010 3:17 PM
DIGGS_WKT_NA.txt	Text Document	98 KB	No	3,03...	97%	8/12/2010 3:15 PM
DIGGS_WKT_UK.txt	Text Document	4 KB	No	103 KB	97%	8/12/2010 3:11 PM
DIGGS_WKT_World.txt	Text Document	2 KB	No	25 KB	95%	8/12/2010 3:17 PM

- Need to publish these on the DIGGS website in a new “dictionaries” directory.
- **ACTION:** Bray to post the XML and TXT (WKT) files at <http://schemas.diggsml.com/> in a new directory to be created called “dictionaries”. (Rename the XML and TXT files to include “CRS” in the filename.) For example:
http://schemas.diggsml.com/dictionaries/DIGGS_CRS_GML_NA.xml



Name	Last modified	Size	Description
1.1.0.zip	17-May-2010 17:18	33M	
1.1.0/	19-May-2010 08:56	-	
1.1.zip	17-May-2010 17:18	33M	
1.1/	19-May-2010 08:56	-	
cgi-bin/	11-Jul-2008 08:45	-	
codelists/	11-Jul-2008 09:10	-	
schemas/	19-May-2010 08:46	-	

Apache/2.2.15 (Unix) mod_ssl/2.2.15 OpenSSL/0.9.8e-fips-rhel5 mod_bwlimited/1.4 Server at schemas.diggsml.com Port 80

- Created new EPSG type codes that combine the EPSG codes from the horizontal and vertical. For example: "<EPSG code>_<EPSG code>"
- URN in DIGGS files will point to the DIGGS namespace.
- How to use these?
 - KML prototype application will demonstrate how these are used by applications.
- Discussed the use of WITSML units schemas in DIGGS.
 - Since DIGGS no longer uses the WITSML construct for geophysical or CPT data (replaced by the GML coverage model), WITSML is now only used for the units schemas. So, there's no longer a need to reference the full WITSML schemas.
 - Discussed the idea of taking the units schemas from WITSML and integrating them into DIGGS. This would require many changes throughout DIGGS, but could probably be done via scripts.
 - Also, considered repackaging the relevant WITSML schemas under a "wrapper" to include with DIGGS.
 - **ACTION:** Burggraf to:
 - Identify the units schemas from WITSML used by DIGGS.
 - Consider how these schemas are imported currently.
 - Look into schema "wrapper" or other approach to make use of units schemas more efficient.
 - Provide recommendation on how to proceed.
 - Include the changes in 1.2.4f.
- Status of 1.2.4e
 - Tested the 1.2.4d schema for object-property rule. Everything looked fine.
 - Naming convention warning came up.
 - Update the testinstance DIGGS file to 1.2.4f
 - **ACTION:** Burggraf will fix and send out 1.2.4f.

E.34 Teleconference Meeting Notes 2010-08-26

Date: August 26, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
 Dan Ponti
 David Burggraf
 Chris Bray

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- CRS Dictionaries
 - Bray posted files on the DIGGS website.
 - Combine the 3 dictionaries into a single dictionary. One XML file and one WKT file.
 - Should we version the CRS files or the dictionary?
 - Carry the version in the URL
(<http://schemas.diggsml.com/dictionaries/0.1/>)
 - Add an attribute to the XML files to carry the version. May add this to the WKT files as well.
 - Begin with version “0.1”
 - Every CRS carries a version identifier. For example:
urn:diggs:def:crs:DIGGS:0.1:27700_5701
 - Example in a DIGGS instance:

```
<g3.3:offsetVector
  srsName="urn:diggs:def:crs:DIGGS:0.1:27700_5701">0 0 -
  10</g3.3:offsetVector>
```
 - CRS URN references in DIGGS instances will carry the CRS version as a parameter.
 - **ACTION:** Burggraf to modify files (add version attributes, change “DIGGSINC” to “DIGGS”). Bray to create directory on website and upload new files.
- Schema wrapper for WITSML units.
 - Didn’t need to use a wrapper.
 - Import just “typ_dataTypes” schema
 - Eliminate 161 schema files from the WITSML directory.
 - typ_dataTypes includes 5 schemas that reference units:
 - typ_baseType
 - typ_catalog
 - typ_dataTypes
 - typ_measureType
 - typ_quantityClass
- Updated test instance for 1.2.4f
- Polygons for trenchwalls – didn’t display in Snowflake GML Viewer software.
 - Add a reference edge
 - Use a simple polygon.
 - **ACTION:** Burggraf to make changes, create revised test instance, test in Snowflake.
- Need to demonstrate use of DIGGS files in an off the shelf application:
 - Works partially in Snowflake.
 - Does not work in Gaia.
 - **ACTION:** Ponti to try using the testinstance using Geotools.
 - **ACTION:** Turner to work with Caronna to test in Bentley software.

- O&M schema evaluation – need to consider this approach further. Continue this discussion next week.

Test (Observation)

Date
Time
....(other metadata)
Reference to the Property/Result
Reference to sample (optional)
Procedure/Parameters
Triaxial test parameters
Specification
Load rate
Pressure
....(other metadata)

Property (Result)

Shear strength
Shear modulus
....(other metadata)
Reference to location feature
Reference to the Test (optional)

Borehole

Reference to the Property/Result

- Burggraf out until Sept 14. Next meeting on Sept 16.

E.35 Teleconference Meeting Notes 2010-09-16

Date: September 16, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Recap tests from prior weeks on viewing DIGGS test instance file in Snowflake's *GML Viewer* free software. Observations:
 - Appears to recognize and display 3D SRS coordinates.

- However, file needs to include the optional attribute for SRS dimension in order for Snowflake to recognize 3D coordinates.
- Viewer supports GML 3.2 constructs.
- Observations on viewing DIGGS test instance file in Gaia:
 - Does not recognize 3D SRS attribute tag.
 - Parses coordinates as 2D.
- Tim Spink emailed last week expressing interest in reviewing encoding of trial pits, trench walls, and similar location features.
 - **ACTION:** Turner to set up meeting with Spink, Ponti, Bray on 9/30 to brief Spink on DIGGS handling of these location features.
- No work has been done on codelist since last meeting.
 - **ACTION:** Turner to begin this work.
- Lab test data structure
 - **ACTION:** Dan and Loren will have meeting to continue discussion.
- Next team status meeting on 9/30.
- Punchlist for DIGGS v1.2 at this point:
 - Need to identify changes (if any) to lab test data structure and implement.
 - Need to finalize codelists and implement.

E.36 Teleconference Meeting Notes 2010-09-30

Date: September 30, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Discussed the approach for handling codelists in DIGGS. Identified issues with the past approach and identified a revised approach for implementation in DIGGS v1.2.
- The past approach attempted to maintain a comprehensive DIGGS codelist, containing a compilation of commonly used standards by DIGGS stakeholders. This included various categories of codes, such as:

Type A: Codes to describe in more detail a specific data element, where the data element cannot be controlled or validated by the schema alone (e.g. table data and CPT parameter names).

- Type B:** Codes created, maintained, and published by recognized standards organizations, used in practice, and commonly referenced with or without software (e.g. USCS Group Symbols for soil classification, Munsell color codes, EPSG spatial reference codes).
- Type C:** Codes created by an organization, government agency, trade group, or company to standardize nomenclature and terms across a specific user base (e.g. Roles, titles, equipment names, test names).
- The proposed design approach for DIGGS with regards to managing the three types of codes:
 - If the code is absolutely necessary for DIGGS to function and be unambiguous for source and target data interchange, then these codes should be implemented into enumerated lists. Enumerated lists are part of the schema and are validated by schema alone. All of the “Type A” codes fall in this category.
 - For codes that are commonly referenced, nomenclature and abbreviations well documented, and maintained by a standards body, these should be implemented in DIGGS using codetype and codespace attributes. “Type B” codes fall in this category. DIGGS might require that some codetype and codespace attributes be mandatory. Although the codespace would reference the standards organization (e.g. USCS, AASHTO), the full list of codes (e.g. SP, SW) would not be in the codelist, since the standards organization maintains this list, and it would be left to the users to comply with the standards published by that standards organization.
 - Codes that are used in localized practice, as described by the “Type C” codes, should be made available for integration into DIGGS as needed. Codespace and codetype attributes would be optional. This would be applicable, for example, for codes such as “roles” where the value itself likely carries meaning without other external references. However, specific user groups may want to standardize the possible values being used. Three possibilities:
 - DIGGS file authors could simply use codes (uncontrolled) without any reference to a codetype or codespace. However, the recipient of the DIGGS file would not know what standards are being referenced.
 - The DIGGS author could populate the codetype and codespace attributes. Since these are optional and the format uncontrolled, the recipient may still be unable to resolve the references in a systematic manner.
 - The DIGGS author could reference a published codespace that can be validated with schematron.
 - Codespace validation in schematron.
 - This should evaluate codespace compliance in a generic way.

E.37 Teleconference Meeting Notes 2010-10-07

Date: October 7, 2010

Time: 7:30 AM – 9:30 AM (PST)

Participants:

- Loren Turner
- Dan Ponti
- David Burggraf
- Chris Bray

Not Available:

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- Continued discussion on the approach for handling codelists in DIGGS. General consensus reached on proposed approach from prior meeting. Three generalized types of codes and proposal for handling in DIGGS are summarized:

Type	Description	Proposed DIGGS Implementation
A	Codes to describe in more detail a specific data element, where the data element cannot be controlled or validated by the schema alone (e.g. table data and CPT parameter names).	If the code is absolutely necessary for DIGGS to function and be unambiguous for source and target data interchange, then these codes should be implemented into enumerated lists. Enumerated lists are part of the schema and are validated by schema alone.
B	Codes created, maintained, and published by recognized standards organizations, used in practice, and commonly referenced with or without software (e.g. USCS Group Symbols for soil classification, Munsell color codes, EPSG spatial reference codes).	For codes that are commonly referenced, nomenclature and abbreviations well documented, and maintained by a standards body, these should be implemented in DIGGS using codetype and codespace attributes. DIGGS might require that some codetype and codespace attributes be mandatory. Although the codespace would reference the standards organization (e.g. USCS, AASHTO), the full list of codes (e.g. SP, SW) would not be in the codelist, since the standards organization maintains this list, and it would be left to the users to comply with the standards published by that standards organization.
C	Codes created by an organization, government agency, trade group, or company to standardize nomenclature and terms across a specific user base (e.g. roles, titles, equipment names, test names).	Codes that are used in localized practice should be made available for integration into DIGGS as needed. Codespace and codetype attributes would be optional. This would be applicable, for example, for codes such as "roles" where the value itself likely carries meaning without other external references. However, specific user groups may want to standardize the possible values being used. Three possibilities: <ul style="list-style-type: none"> • DIGGS file authors could simply use codes (uncontrolled) without any reference to a codetype or codespace. However, the recipient of the DIGGS file would not know what standards are being referenced. • The DIGGS author could populate the codetype and codespace attributes. Since these are optional and the format uncontrolled, the recipient may still be unable to resolve the references in a systematic manner.

- The DIGGS author could reference a published codespace that can be validated with schematron.
- Implementation of codelists in DIGGS using codetype elements:
 - *Codetype element* – has one attribute, *codespace*
 - *Codespace* – points to the authority or dictionary. This can be a text string, URN referencing a specific codelist, or URL to a website where the published standard can be found. For example:

```
<soilClassification codeSpace="Caltrans Logging Practice">SAND</soilClassification>
<soilClassification codeSpace="urn:diggs:def:soilclass:Caltrans ">SAND</soilClassification>
<soilClassification codeSpace="http://dot.ca.gov/logging_standards.pdf">SAND</soilClassification>
```
 - *Codetype with authority* requires the codespace attribute populated.
- For codetype elements, it is not possible to validate the codespace attribute with XML schema validation tools alone. However, validation is possible through the use of schematron, a schema validation language that some software could use to further validate XML against specific business rules. This is typically an extra step in the validation process. For the codetype elements, the schematron could be set up to validate all codetype elements in a uniform and generic manner. For example, whenever a codetype element is encountered, the schematron would:
 - Evaluate if the URN points to a specific codelist. If so, it checks the element value against the codelist.
 - Evaluate if the URN points to a valid URL.
 - If the URN or URL is not valid, the schematron would report a validation error to the user.
- Users can choose whether or not to validate DIGGS files with schematron. For those that require strict validation with business rules, the schematron validation step may be worthwhile. However, for cases where strict adherence to business rules are not necessary (e.g. source and target software use well-defined and mutually compatible codes), schematron may not provide additional benefit.
- Development and maintenance of “Type C” codelists.
 - The DIGGS organization will not publish official codelists as part of a standard release.
 - Codelists will be developed and maintained by user groups.
- The DIGGS organization will provide an online community forum that facilitates the formation of practice groups.
 - These groups could be regional-based, practice-based, or organized based on specific needs of a particular stakeholder group.

- The forum will allow posting of codelists under stakeholder groups.
- Stakeholder groups can establish review committees to approve changes to codelists, if desired.
- Forum will have upload tools to make it easy to create codelists and upload them for community usage. (e.g. Excel-based, or web-form driven)

E.38 Teleconference Meeting Notes 2010-11-03

Date: November 3, 2010

Time: 7:30 AM – 10:15 AM (PST)

Participants: Loren Turner
Dan Ponti
Chris Bray
Roger Chandler

Not Available:

Agenda: Meeting to discuss schema change proposals.

Notes:

- Agenda:
 - Proposal to combine monitoring into the kernel
 - Consider combining environmental with monitoring.
- Issues with capturing data from wells:
 - Multiple wells in a single hole.
 - Backfill, however, is in one hole.
 - Wells are not locations.
 - AGS made distinction on process of how the water level was made (piezometer vs. tape) and stored in two different ways. However, resulting measurement was the same.
 - In AGS, monitoring is a location type and can be a child of a borehole.
- Actions:
 - Need to fix monitoring in DIGGS.
 - Ponti come up with strawman construct. Review with Chandler and Bray.
 - Followup meeting in a couple weeks.
 - Include the monitoring feature in the kernel.
 - Split monitoring into:
 - Wells
 - Sensors

E.39 Teleconference Meeting Notes 2010-12-15

Date: December 15, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
Tim Spink

Not Available:

Agenda: Discussion on the proposed approach for encoding trenchwalls and trial pits in DIGGS v1.2

Background:

Provided by D. Ponti in a 12/14/10 email. (Additional figures inserted by L. Turner):

To start, it might be best to back up a bit and cover some of the changes and structure in DIGGS v. 1.2 as this may help you better understand the origin of the trench wall treatment.

First off, The earlier hierarchical structure is flattened in v.1.2, insofar as the major feature classes in DIGGS are all carried at the top level of the instance document. This is done so that partial transmissions of information can be sent atomically, without having to send redundant data. It also allows for more flexibility in associating features with one another, which was getting us into trouble in v. 1. The current main feature classes in DIGGS are:

- 1) Projects (the business activity that produces or contains other DIGGS features)
- 2) Locations (features where samples are collected or observations made)
- 3) Sampling Activities (eg. the process of taking a sample - includes sample position information)
- 4) Samples (the physical sample itself)
- 5) Layer Systems (eg. geologic layers, etc)
- 6) In-situ tests
- 7) Laboratory tests
- 8) Groups

And some metadata feature classes:

- 1) Business Associates
- 2) Associated Files
- 3) Contracts
- 4) Equipment
- 5) Specifications

Location features are the heart of DIGGS. They are places where samples are taken, tests are run, and/or observations are made. I mentioned in our last phone call that I was becoming uncomfortable with the use of the term Location for these things, as the term location is more commonly thought of as only a position (e.g. coordinate, etc.), as opposed to a real-world feature that can be complex and contain many properties. I suggested the use of the term Site when we last talked, which you didn't like because of other connotations there. Nonetheless, while I can live with Location, I'd like to suggest we think of a different term. OGC's O&M evolving standard refers to things like boreholes as "sampling features" which is a good description, but since we don't directly inherit from their feature types, we probably shouldn't use the same term.

One other thing - in the following, I'll be using the term observation. In this context, I'm describing any feature or process that provides information about an earth property "observed" at a location feature. So sampling activities, samples, layers, and tests are all "observations" at a location feature in the context of our discussion here.

Anyway, in version 1 most of the effort went into the "Hole" feature, although we also had a few others (eg. Station, FoundationGroup), and hole largely paralleled its usage in AGS where it is used as a catch-all for a geotechnical borehole, trial pit, etc.). In v. 1.1 and 1.2, we've created a structure where a basic location is abstracted and then very specific "concrete" location features are built from the abstract types. BTW, to be clear, an abstract element in XML is one that exists in schema but isn't used in an instance document directly. Instead, concrete elements (that are used in an XML document) are defined in the schema that substitute for the abstract types and contain properties (other elements or tags) from the abstract element. In DIGGS 1.2, AbstractLocation derives from the basic GML feature type (required for a GML application schema) and adds to it properties that reference other features that the location may be associated with (such as its project, layers, etc.) Using AbstractLocation as a base type, there are 3 abstract location elements defined that can substitute for abstract location and add geometric elements that define the geometries of the Location feature. AbstractLinearLocation serves as a base type for Location features whose geometries are modeled as a line or curve (linestring) - such as a borehole. AbstractPointLocation serves as a base type for location features whose geometries are modeled as a point in space (eg. a sampling station), and AbstractPlanarLocation is used as a base type for Location features whose geometries can be modeled as planes (although this isn't strictly true, as I'll explain below).

All location features have one or more properties that define the geometry of the feature. All features have a mandatory property called a referencePoint, which is of type `gml:PointPropertyType` and defines a point in (typically) a defined coordinate reference system (eg. a combined CRS with British Grid and a vertical datum). For point location types, this is the only geometry property. Linear location features adds one other mandatory geometry property - centerLine, of type `gml:CurvePropertyType`, which defines the centerline of the linear feature (eg. a borehole or roadway centerline). Like the referencePoint, the centerLine's coordinates are typically in some defined geographic or projected CRS. Planar location features have a referencePoint and a similarly structured (eg. it's a curve) property to centerLine, but instead is called a referenceEdge. I'll get more into its use in a bit. In addition, planar location features have two optional geometry properties that define the physical boundaries of the feature - these are of type `gml:SurfacePropertyType` (eg. polygons). The first, featureExtent, defines the feature's boundary in a concrete geographic or projected CRS and is used to display the feature in typical simple mapping software that supports GML 3.2 and below. After our conversation last time when you questioned why the perimeter of a trench is coded in a 3D CRS whereas other positions are relative, the answer is so that trench locations (or their outlines) can be displayed in existing

software. After discussing this with Galdos, we added a second optional element called a relativeFeatureBoundary, which is functionally the same as featureExtent, but is defined in the linear reference system of the trench (more on this is in a bit). Future GML 3.3 savvy software (such as the KML viewer that Galdos is now building for us) will be able to understand and display a relativeFeatureBoundary. Instance authors can populate either, both, or none of these properties, as the feature boundary isn't an essential property of a planar location feature (although a good one to include).

In addition, linear and planar location features carry a mandatory complex property called linearReferencing, that defines the relative spatial reference system that observations use to identify their positions on the location feature. This referencing system is part of GML 3.3, which isn't released yet, but provides a simpler way of referring to positions than was required in the past (where engineering systems needed to be defined and where existing software didn't know how to handle this stuff). So, going in this direction looks to the future, results in simpler encoding, and, while no existing software supports GML 3.3, the idea is that this is a more standard way to handle relative positional referencing, so that this will be supported down the road. In the meantime, Galdos is writing for us a translation package that will support GML 3.3 and allow DIGGS files to be viewed in GoogleEarth.

From these abstract location types, DIGGS 1.2 has several "concrete" location features defined - these are the ones that can show up in an XML instance document. Borehole and TrialPit features derive from AbstractLinearLocation; TrenchWall derives from AbstractPlanarLocation, and Station derives from AbstractPointLocation. Now that we have the basic structure defined, it's a relatively simple process to add more location features to DIGGS (eg. piles, embankments, etc.). Note, a trial pit in DIGGS 1.2 functions identically to a trial pit in AGS, in that it is modeled as a linear feature, only it is explicitly called a trial pit in DIGGS, as opposed to using the hole group in AGS. Trial pits in DIGGS, therefore, are meant to serve as a legacy feature for older AGS structured data. In the future, the idea would be to use the TrenchWall feature instead of a TrialPit.

Ok, so now that we have defined the basic structure of our location features we need to be able to refer to the "positions" of observations from those location features. We don't need to do this for point locations, as these are essentially 0-dimension features - all observations at such locations are defined at the point of the location feature. But for other location features, observations can be made at a number of positions on the feature. In DIGGS 1 and 1.1, that positional information (eg. top and base) was hard coded into the properties of our observations, which "locked" in an observation so that it was only relevant to a location feature. So, we have a sample defined for a borehole, but what about a trench wall? With the old structure, you'd have to define another sample feature for trenches. And this extends to all other relevant observations that could be made at different location features as well, and will eventually lead to extreme schema bloat. If you think about it, the actual properties of a sample or a geologic layer are independent of the feature from which it is being described. The only aspect of a sample collected from a trench wall, for example, that is different from one collected in a borehole is how the positional information is encoded (sampling process may be different, too, but structurally, this information is recorded in the same way). But if we can generalize the position property for observations, we can then define one observation type, say sample or layer, and then reuse this type regardless of what location feature the sample or layer comes from. This makes extension much simpler and cleaner.

How this works in practice is that for every concrete location feature, we define a position object that contains properties that defines the relative positions that are possible for observations within a given location feature. This position object is then substituted into the position property for an observation. So for the position of a sampling activity in a borehole, the XML looks something like this:

```
<SamplingActivity gml:id="xyz">
  <projectRef xlink:href="#p1"/>
  <associatedLocationRef xlink:href="#h123"/> <!-- refers to a borehole feature -->
  <activityPosition>
    <BoreholePosition gml:id="bpp1">
      <depthInterval>
        <gml:LineString gml:id="pl1">
          <gml:posList srsName="#sr123" srsDimension="1">5 7</gml:posList>
        </gml:LineString>
      </depthInterval>
    </BoreholePosition>
  </activityPosition>
</Sampling Activity>
```

whereas a sampling activity in a trench would look like this (for a channel sample):

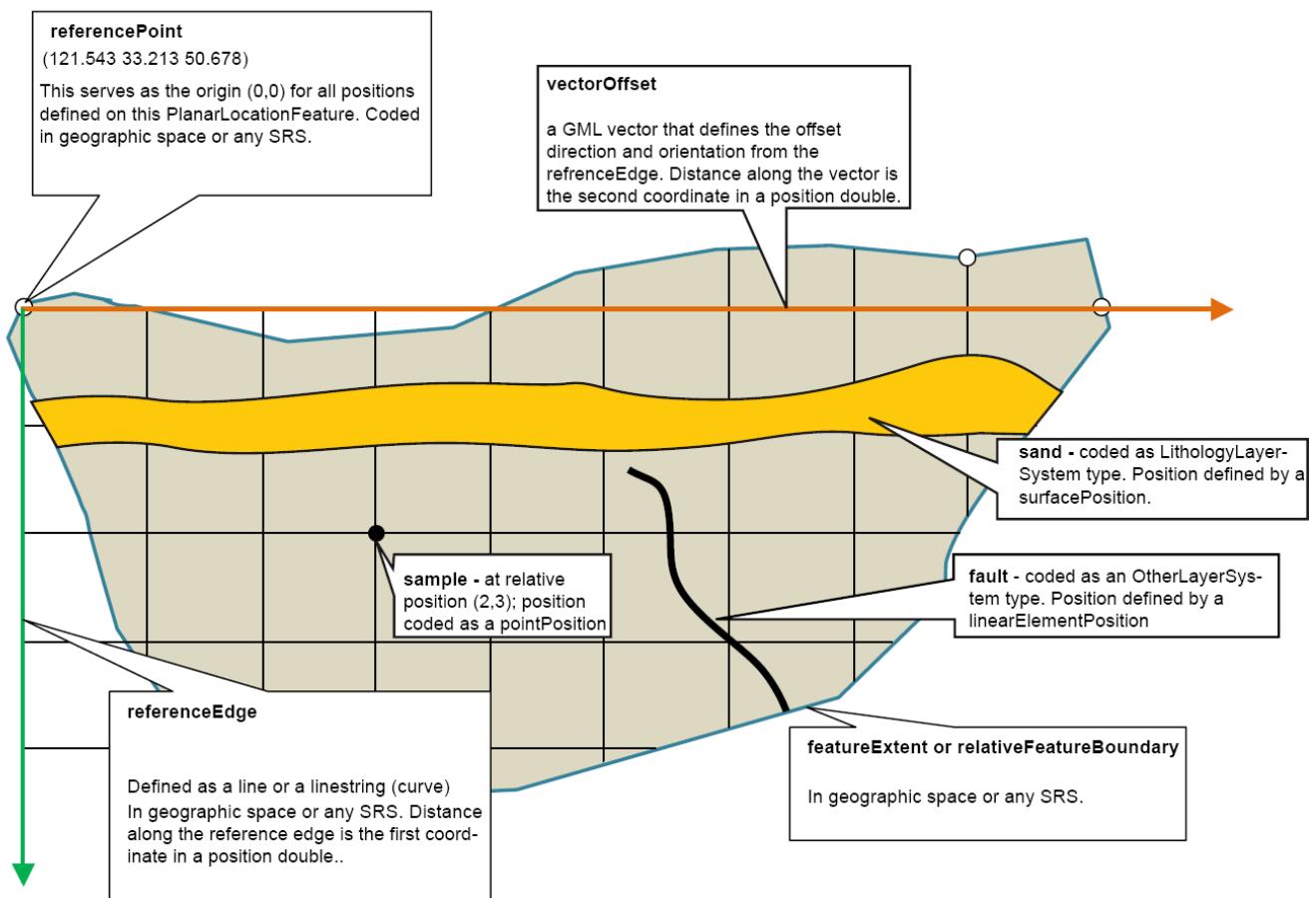
```
<SamplingActivity gml:id="xyz">
  <projectRef xlink:href="#p1"/>
  <associatedLocationRef xlink:href="#t123"/> <!-- refers to a trenchwall feature -->
  <activityPosition>
    <TrenchWallPosition gml:id="bpp2">
      <linearElementPosition>
        <gml:LineString gml:id="pl2">
          <gml:posList srsName="#srt123" srsDimension="2">5 7 5 9</gml:posList>
        </gml:LineString>
      </linearElementPosition>
    </TrenchWallPosition>
  </activityPosition>
</Sampling Activity>
```

For boreholes, the BoreholePosition object substitutes in for the activityPosition, whereas TrenchWallPosition is used if the sampling activity occurs at a TrenchWall feature.

In DIGGS 1.2, position objects are defined for boreholes, trial pits, and trench walls. For borehole positions, there is a choice of two properties (only one can be used for a given position) - either measuredDepth (a point position) or depthInterval (a line string that defines an interval within the hole). Note that top and base are no longer used, as top and base are inferred from the linestring definition. In fact, while convention will probably define depth intervals from top to base, they could be defined from base to top - it doesn't matter. What's relevant in the position is the location of the interval within the borehole. Trial pit position objects use the same position properties (depthInterval and measuredDepth), and add a stratumReference property as it is used in AGS. Trench walls work a bit differently. Positions on a trench wall can either be points, linear elements (eg. a channel sample can be defined as a line on the trench wall surface), or areas (polygons) defined on the trench wall surface (layers would be encoded this way). Trench wall position

objects therefore offer a choice of a pointPosition (point), linearElementPosition (linestring), or surfacePosition (polygon) -- all defined in the linear reference system for the trench wall.

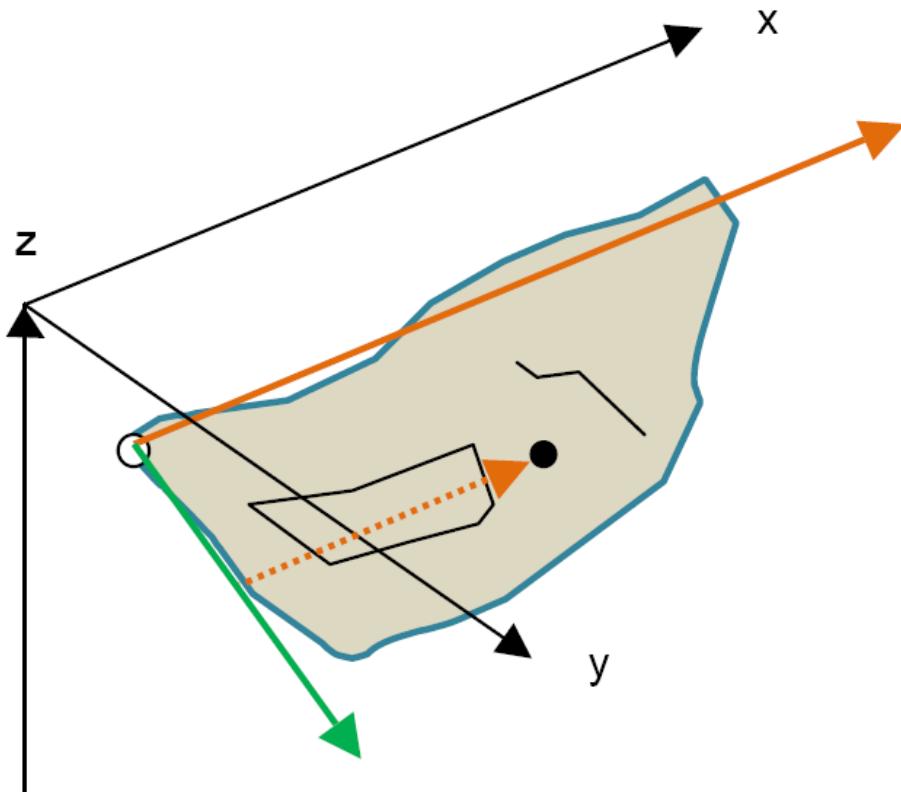
Ok, so now to the referencing part for trench walls. Attached is a diagram that can (hopefully) help define the terms.



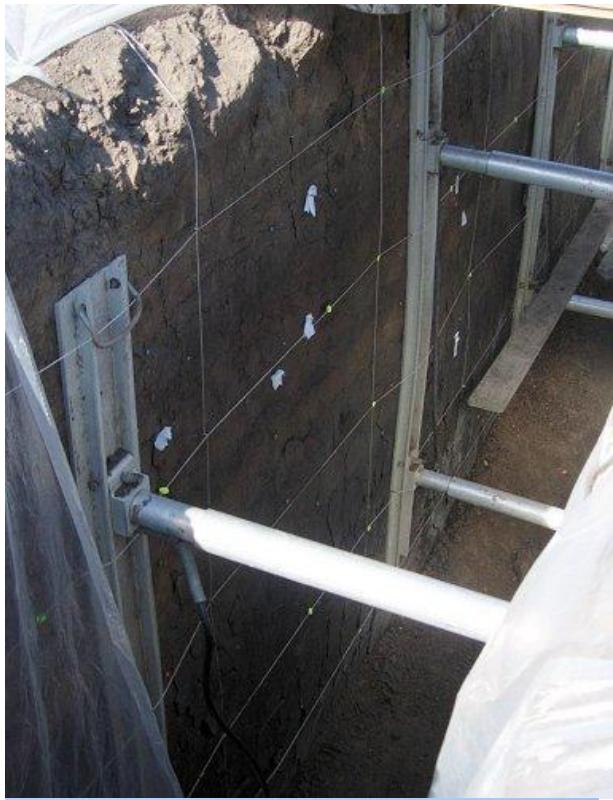
Trench walls are typically logged as planar features (or at least modeled that way) and the construct in DIGGS is designed to support this representation - positions on the trench wall are represented by coordinate pairs in the relative reference system that are distances along the referenceEdge, and distance along the offsetVector (see diagram), so these two properties define the orientation of the relative coordinate system of the trench. In contrast, positions in a borehole are represented by a single numeric value that represents distance along the centerLine from the reference point only. In the top diagram, if we assume it represents a vertical trench wall on an E-W bearing, the referenceEdge is defined as a vertical line (straight in this example), and the vectorOffset is a horizontal line with an E-W bearing. It's perfectly ok to reverse the definitions with the referenceEdge along the horizontal and the vector along the "vertical" dimension. The only difference would be that the coordinate pairs would reverse in the position properties - distances along the reference edge are given first in the coordinate double, then the offset distance. It's also not required that the coordinate system that is set up by defining the

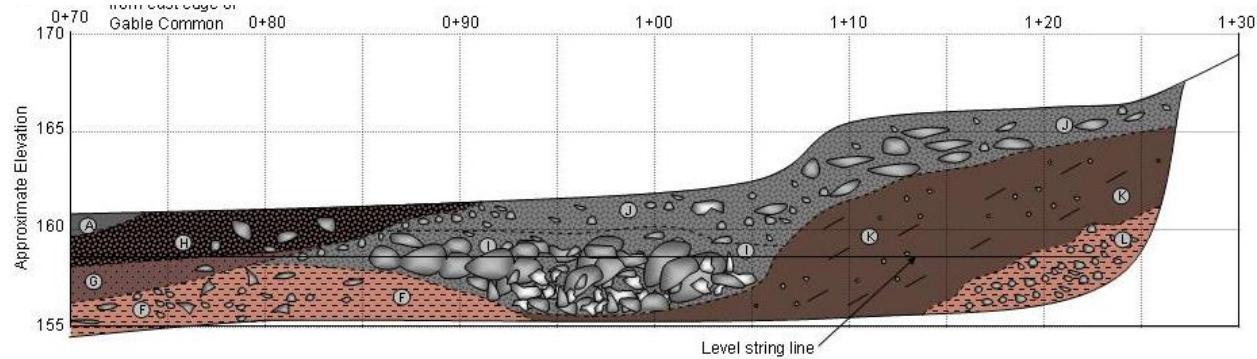
referenceEdge and offsetVector be orthogonal, although I'm not sure why someone would want to do that...

Even though the trench wall is modeled as a planar feature, that feature can have any strike and dip orientation in space (doesn't have to be vertical), nor does it strictly have to be a flat planar surface. This is because the reference edge can be any curve - not just a straight linear segment - and it is defined in geographic space. The vectorOffset, however, is a vector, and can only be a straight line with bearing and plunge defined in the context of the 3D CRS (this is a GML limitation of the linearReferenceSystem constructs). So, you could model corrugated roofing or even a cylindrical trench wall with this construct, but not a sphere, or any type of complex surface. A GML 3.3 aware application will be able to manage the transformations to plot the positions of observations in true geographic space.



In typical practice, though, we represent a trench as a truly planar surface, with the vertical dimension being "down" the face of the trench wall and the horizontal dimension along the bearing of the wall, and this construct handles this well. In the field, the old way of logging trenches here is the US was to lay out a string grid that is used to scale observations that are drawn on a log sheet.





More recently, features are often shot in using a total station or other surveying means, usually using a local coordinate system that is oriented (in the x-y) parallel and perpendicular to the average bearing of the trench face (or can be transformed into trench-parallel and trench-perpendicular coordinates. I'd be interested in hearing of other practice standards that may go on in the UK to make sure this structure will work for those, too. Either way, positions can be recorded (or digitized from a log sheet) in an orthogonal coordinate system and the trench wall construct handles this well. When surveyed measurements are then translated for drafting, they are usually projected onto a planar surface (eg. the piece of paper). If the logging coordinates are set up or transformed so that x is parallel to the trench wall, then the position coordinate pairs are plotted using the x-z coordinates and the y-values (the dimensions into or out of the paper), are ignored, and if the trench is straight, are insignificant. For analog (paper) trench logs, observation positions can be digitized and translated to true distance units directly for transfer via DIGGS as coordinate pairs in the position properties. Currently, DIGGS doesn't support true storage/transmission of the full 3D position of observations on a trench wall that could be derived from a total station. To do so would require defining a different location feature type that can handle an irregular surface - GML doesn't currently handle such a feature in a relative sense, although there are constructs that can do so in absolute coordinate space.

Below is an XML snippet of a trench wall feature, as an example of how the definition of a trench wall might be encoded. We can go over this when we talk on Wed.

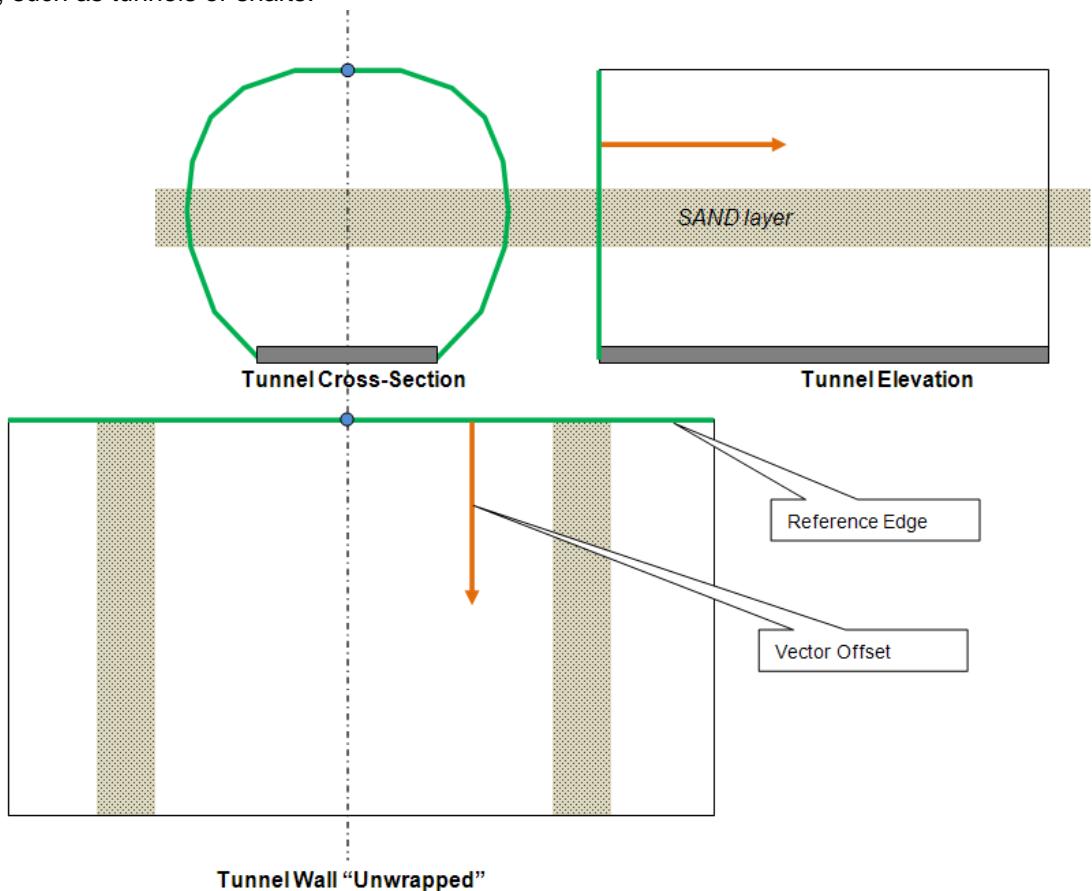
```

<TrenchWall gml:id="a22">
  <gml:name>My trench</gml:name>
  <gml:identifier codeSpace="usgs">urn:DIGGS:def:fi:USGS:usgs_a22</gml:identifier>
  <projectRef xlink:href="#p1"/>
  <groupRef xlink:href="#g1" identifierRef="urn:DIGGS:def:fi:USGS:usgs_g1"/>
  <referencePoint>
    <gml:Point srsName="urn:ogc:def:crs:EPSG::7405" srsDimension="3" gml:id="a34">
      <gml:pos>33 -117 10</gml:pos>
    </gml:Point>
  </referencePoint>
  <referenceEdge>
    <gml:LineString srsName="urn:ogc:def:crs:EPSG::7405" gml:id="ply1_top">
      <gml:posList>33 -117 10 33.1 -117.1 10</gml:posList>
    </gml:LineString>
  
```

```
</referenceEdge>
<!-- In this case the reference edge is horizontal and lies along the top of the trench -->
<featureExtent>
  <gml:Polygon srsName="urn:ogc:def:crs:EPSG::7405" gml:id="ply1" srsDimension="3">
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>33 -117 10 33.2 -117 10 33.2 -117.2 0 33 -117.2 0 33 -117 10</gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</featureExtent>
<planarReferencing>
  <LinearSpatialReferenceSystem gml:id="lsrs002">
    <g3.3:linearElement xlink:href="#ply1_top"/>
    <g3.3:lrn>
      <g3.3:LinearReferencingMethod gml:id="lr123">
        <g3.3:name>chainage</g3.3:name>
        <g3.3:type>absolute</g3.3:type>
        <g3.3:units uom="m"/>
      </g3.3:LinearReferencingMethod>
    </g3.3:lrn>
    <g3.3:vectorOffsetExpression>
      <g3.3:offsetVector srsName="urn:ogc:def:crs:EPSG::7405">0 0 -1</g3.3:offsetVector> <!-- Offset is
vertical down>
    </g3.3:vectorOffsetExpression>
    <linearElementAccuracy>
      <PositionalAccuracy>
        <measurementMethod>GPS</measurementMethod>
        <result uom="m">5</result>
      </PositionalAccuracy>
    </linearElementAccuracy>
    <linearReferencingMethodAccuracy>
      <PositionalAccuracy>
        <measurementMethod>tape</measurementMethod>
        <result uom="ftUS">1</result>
      </PositionalAccuracy>
    </linearReferencingMethodAccuracy>
  </LinearSpatialReferenceSystem>
</planarReferencing>
<constructionMethods>
  <ConstructionMethod gml:id="cm22">
    <gml:description>backhoe</gml:description>
    <remarks>
      <Remark>
        <content>Backhoe was brand new and worked like a charm</content>
      </Remark>
    </remarks>
  </ConstructionMethod>
</constructionMethods>
</TrenchWall>
```

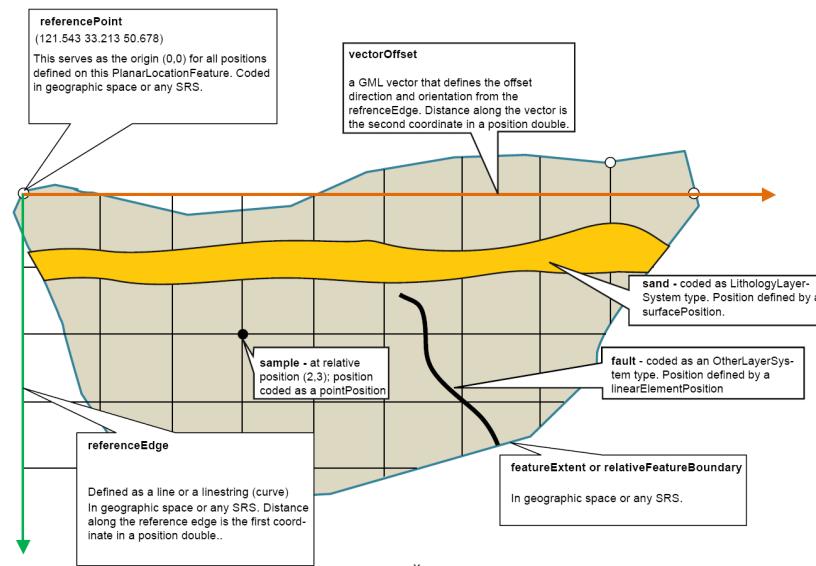
-
- **Meeting Notes:**
 -
 - Summary of key concepts of on encoding spatial data as implemented in DIGGS v1.2:
 - The locations of boreholes (identified by a point at the top of hole), trenchwalls/trialpits (identified by a polygon area), and other DIGGS “location features”:
 - Are encoded in real-world 3D coordinates (e.g. latitude, longitude, elevation).
 - Use GML 3.2 schema structures.
 - Can be interpreted by GML 3.2 aware applications – *Snowflake GML Viewer* (3D), *Gaia* (2D only).
 - Will be visualized in GoogleEarth tool.
 - The *Linear Referencing* method is used to identify the location of observations within the borehole or on the trenchwall. This method:
 - Creates a localized spatial reference system within the borehole or trenchwall.
 - For boreholes – a curve is defined in real-world coordinates; the depth (or interval) along that defined path serves as a localized spatial reference system.
 - For trenches – a curve and a polygon are defined in real-world coordinates; the depth (or interval) along that defined path and a vector offset serves as the localized spatial reference system.
 - Is built from GML 3.3 draft standards.
 - Uses GML 3.3 which has not yet been published.
 - Concerns raised about the *Linear Referencing* method during the meeting:
 - There's currently no commercial software that supports GML 3.3 linear referencing. (Likewise, there's no commercial software that currently supports the “relative” engineering referencing structures used in borehole and trenchwall logging practices.)
 - Support for this kind of referencing will need to be built into geotechnical software.
 - This method will not handle a trenchwall face with XYZ geometry, e.g. mapping a spherical surface. However, there was general concurrence that for most cases, forcing a planar mapping model, as proposed, would handle most practical use cases.
 - There were questions as to whether this method supports a reference line that closes in on itself. E.g. cylindrical wall of a shaft where the reference line is the circular cross section line and the vector is used for depth encoding. **ACTION: Discuss this issue with Burggraf.**
 - T. Spink expressed reservations with use of this GML 3.3 construct in DIGGS, since it has not been released as a standard, and that DIGGS may be adopting this too early.

- **ACTION:** Turner will present this to the software vendors to assess if this will be a barrier to implementation in commercial products.
- If DIGGS doesn't use the GML 3.3 linear referencing , what are the alternatives?
 - Use 3.2 or 3.0? This presents a different, but challenging, set of issues:
 - Requires more verbose XML in order to achieve similar 3.3 functionality.
 - 3.0 is not an ISO standard; 3.2 is ISO standard; applications that are implementing GML are going with 3.2.
 - Create DIGGS specific structure? This would result in an encoding approach equally unsupported by software.
- The approach used for trenchwalls could also be used to map other geotechnical features, such as tunnels or shafts.



- - For a tunnel, the cross sectional outline (“horseshoe”) is the reference edge
 - Vector offset is the position along the centerline of the tunnel
 - Issues:
 - Cross section can change shape.
 - Tunnels aren't straight, but the vector offset must be in the same direction.

- In this example the sand layer get represented twice when the tunnel is “unwrapped”. **ACTION: Ask Burggraf on how to associate the two represented sand layers as being the same.**
- Mapping tunnels isn’t likely to be a common use case at the moment. However, it is useful to explore the limitations on the linear referencing approach to ensure that DIGGS remains extensible in the future.
- Issue with encoding stratum on trench wall face:
 - In example below, the yellow area represents a sand layer as a polygon. However, common practice is to define layer boundaries, which, in a cross-section in a trench, would be a polyline upper boundary and another polyline lower boundary.



- Recommendation is to use a construct similar to the borehole with “top” and “bottom” to define the interval layer.
- Use top and bottom bounding lines to represent stratum limits
- Consideration needs to be given to capture that the bottom boundary may not necessarily be the same as a layer bottom. This happens with boreholes too – bottom of the drilled hole may not be the same thing as the bottom of a layer.

ACTION: Ask Burggraf on how to capture this.

E.40 Teleconference Meeting Notes 2010-12-09

Date: December 9, 2010

Time: 7:30 AM – 10:00 AM (PST)

Participants: Loren Turner
 Dan Ponti
 David Burggraf
 Yang Zhu

Not Available: Chris Bray

Agenda: DIGGS-Galdos weekly status meeting.

Notes:

- An issue was identified with the AllUnits element, possibly related to the use of the union structure when mapping. Yang Zhu (COSMOS contractor) has been working to map Caltrans CPT data sets to DIGGS v1.2.4g. He encountered an issue while attempting to generate an XSLT mapping using Altova's MapForce 2009 software. From Zhu's email:
 - *I created a mapping project with MapForce, ComosDIGGS schema and DIGGS1.2a. Then, I connected only 1 element from cosmosDIGGS to DIGGS1.2a.*
 - *The following error occurred when MapForce tried to create an XSLT file from the mapping.*

ERROR G102: Internal error: Base type ({<http://schemas.diggsml.com/1.2a>}) of {<http://schemas.diggsml.com/1.2a>}AllUnits missing

Place where the error might be:

Element AllUnits is defined in Kernal.xsd with a union of many other simple data types, as follows:

```
<simpleType name="AllUnits">
  <restriction>
    <simpleType>
      <union memberTypes="witsml:anglePerLengthUom witsml:anglePerTimeUom
witsml:areaPerAreaUom witsml:areaUom witsml:densityUom witsml:dimensionlessUom
witsml:dynamicViscosityUom witsml:electricCurrentUom witsml:electricPotentialUom
witsml:energyPerAreaUom witsml:equivalentPerMassUom witsml:forcePerLengthUom
witsml:forcePerVolumeUom witsml:forceUom witsml:frequencyUom witsml:illuminanceUom
witsml:lengthPerLengthUom witsml:lengthUom witsml:magneticFieldStrengthUom
witsml:magneticInductionUom witsml:massConcentrationUom witsml:massPerLengthUom
witsml:massUom witsml:MeasuredDepthUom witsml:momentOfForceUom witsml:PercentUom
witsml:perLengthUom witsml:planeAngleUom witsml:powerUom witsml:pressureUom
witsml:relativePowerUom witsml:specificVolumeUom witsml:timeUom witsml:velocityUom
witsml:volumeFlowRateUom witsml:volumePerVolumeUom witsml:volumeUom"/>
    </simpleType>
  </restriction>
</simpleType>
```

What I did:

I found most of the union members were defined in witsml/1.3.1.1/typ_quantityClass.xsd. However, when I compared the members in the union with all the types defined in

witsml/1.3.1.1/typ_quantityClass.xsd, I found a couple of differences. First, **accelerationLinearUom** and **thermodynamicTemperatureUom** were defined in witsml/1.3.1.1/typ_quantityClass.xsd, but did not appear in AllUnits. Second, **MeasuredDepthUom**, **PercentUom** were found in AllUnits, but were not defined in witsml/1.3.1.1/typ_quantityClass.xsd.

Problem persists when I tried to redefined the AllUnits by randomly pick 1 or 2 members to keep, and delete the rest of the union members:

```
<simpleType name="AllUnits">
  <restriction>
    <simpleType>
      <union memberTypes="witsml:anglePerLengthUom
      </simpleType>
    </restriction>
  </simpleType>
```

Problem solved if I redefined the AllUnits in the following manner:

```
<simpleType name="AllUnits">
  <restriction base="string">
    <maxLength value="64"/>
    <pattern value="[^ ]*"/>
  </restriction>
</simpleType >
```

Action taken:

I proceeded mappings with this temporary fix as I couldn't seem to be able to fix the problem.

- **ACTION ITEM:** Burggraf will look into this issue further and verify if the union construct is the problem. Send to Yang to test that it works in MapForce
- Possible solutions, if the union is the issue:
 - Create a AllUnits type within the diggs namespace that combines all units, systematically extracted from the WITSML schemas.
 - Create a new schema in diggs called "units" which includes all witsml units.
- No meeting next week. Burggraf out of town.

E.41 Teleconference Meeting Notes 2011-01-06

Teleconference Meeting Notes

Date: January 6, 2011 **Time:** 8:30 AM – 11:00 AM (PST)

Participants: Loren Turner

Dan Ponti

David Burggraf

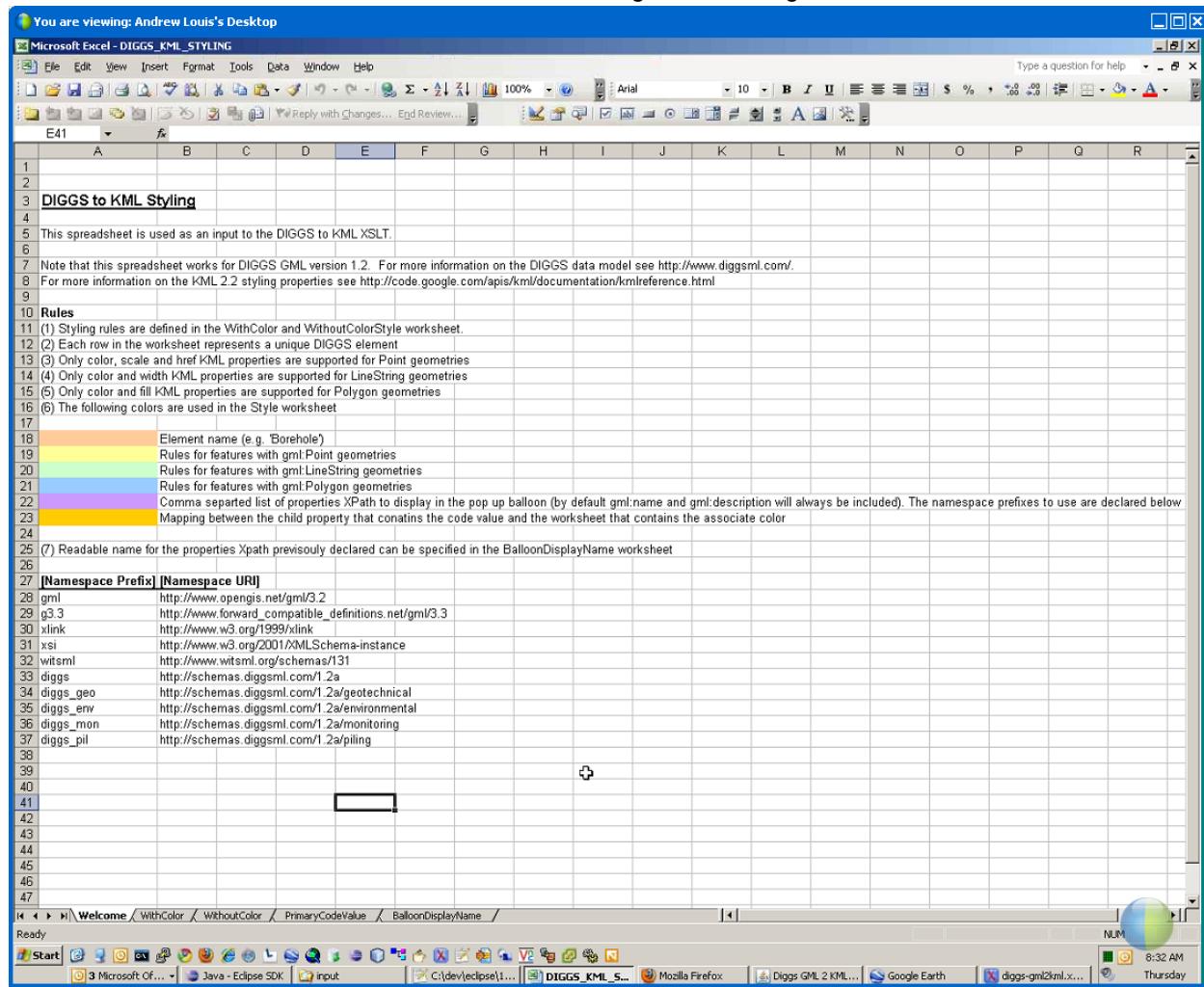
Andrew Louis

Agenda: DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

DIGGS to KML Tool

Excel file is used to maintain GoogleEarth KML mapping display configuration settings. The XSLT file uses the Excel XML file to obtain these settings used during the transformation.



The screenshot shows a Microsoft Excel spreadsheet titled "DIGGS to KML_STYLING". The "WithColor" sheet is active. The content includes:

- Rules:**
 - (1) Styling rules are defined in the WithColor and WithoutColorStyle worksheet.
 - (2) Each row in the worksheet represents a unique DIGGS element.
 - (3) Only color, scale and href KML properties are supported for Point geometries
 - (4) Only color and width KML properties are supported for LineString geometries
 - (5) Only color and fill KML properties are supported for Polygon geometries
 - (6) The following colors are used in the Style worksheet
- Element name (e.g. Borehole)**: A column containing the names of DIGGS elements.
- Rules for features with gml:Point geometries**, **Rules for features with gml:LineString geometries**, **Rules for features with gml:Polygon geometries**: Columns defining styling rules for different geometry types.
- Comma separated list of properties XPath to display in the pop up balloon (by default gml:name and gml:description will always be included). The namespace prefixes to use are declared below**: A column defining the properties to display in the GoogleEarth balloon.
- Mapping between the child property that contains the code value and the worksheet that contains the associate color**: A column defining the mapping between properties and worksheets.
- (7) Readable name for the properties Xpath previously declared can be specified in the BalloonDisplayName worksheet**: A column defining the readable name for properties.
- [Namespace Prefix] [Namespace URI]**: A table mapping namespace prefixes to URIs.

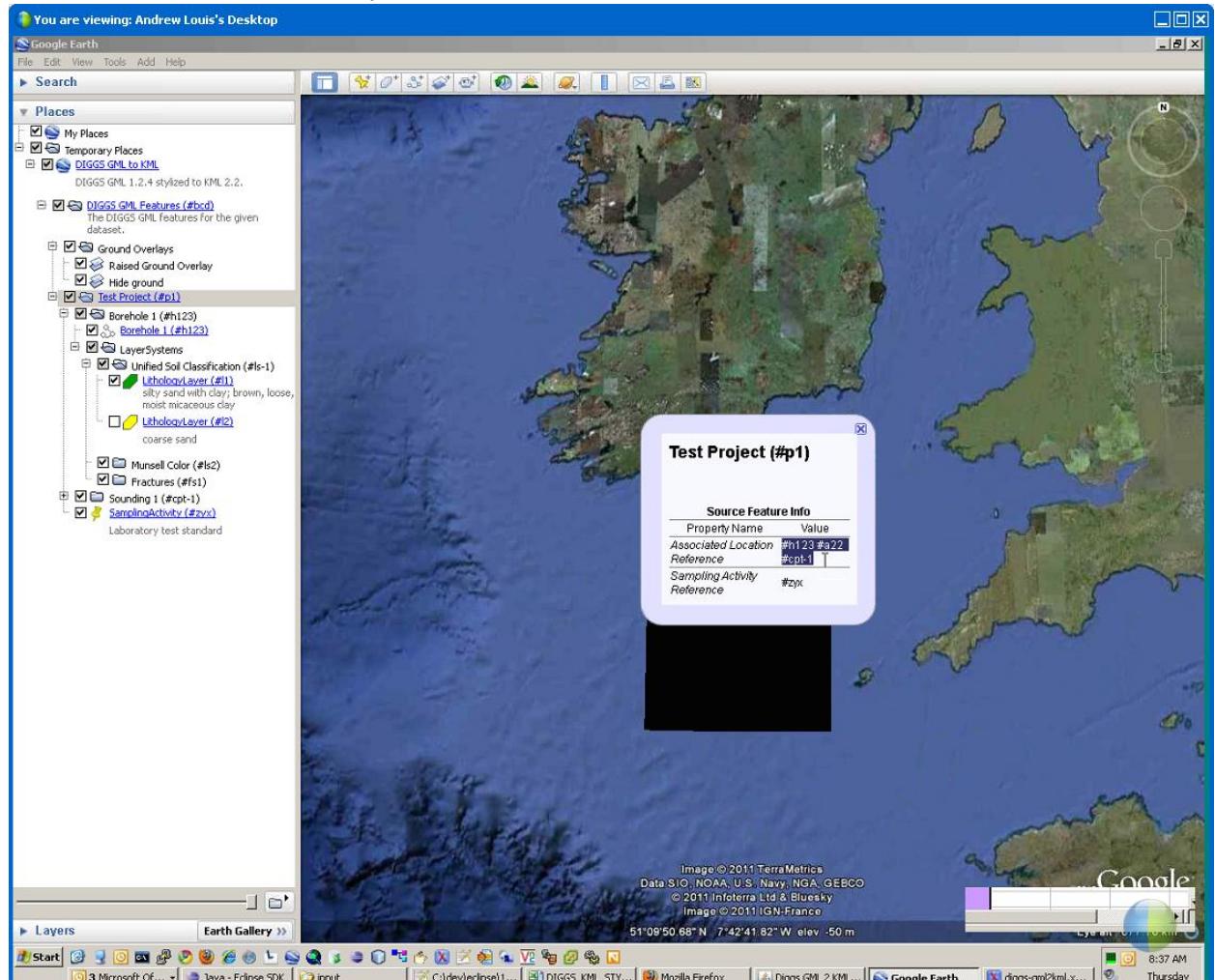
Features can be configured in this worksheet. This is how the points, lines, and polygons are presented in the GoogleEarth interface. If nothing is entered, the default GoogleEarth style will be used.

You are viewing: Andrew Louis's Desktop

Microsoft Excel - DIGGS_KML_STYLING

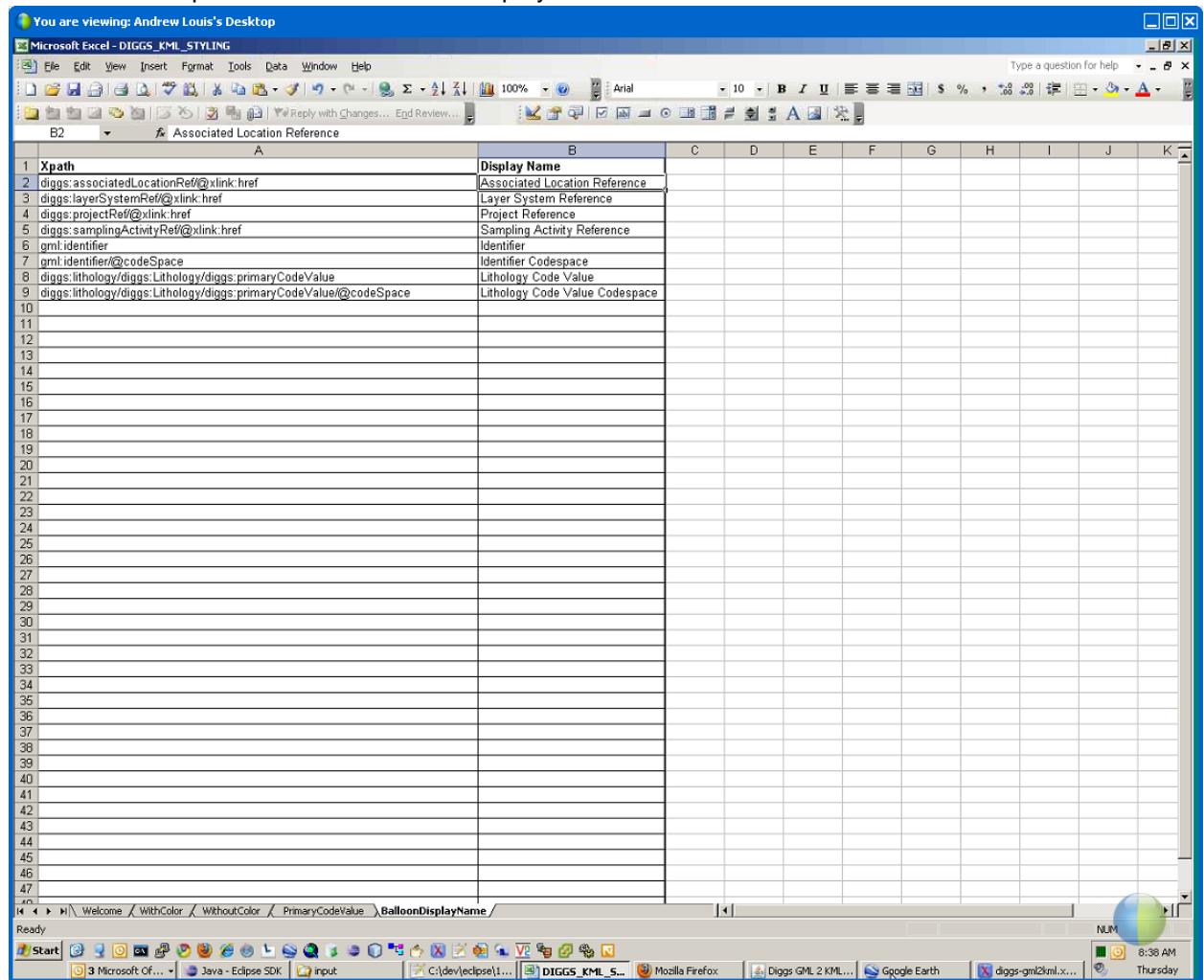
	A	B	C	D	E	F	G	H	I	J	K	L
1		Point		LineString	Polygon				Balloon			
2	DIGGS Element Name	color	scale	href	color	width	color	fill				
3	Project											
4	Borehole											
5	TrialPit											
6	SamplingActivity											
7	Sample											
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30												
31												
32												
33												
34												
35												
36												
37												
38												
39												
40												
41												
42												
43												
44												
45												
46												
47												

Balloon info is identified in the spreadsheet column "Balloons".



It currently shows the `gml:id`, but it would be good to resolve the `xlink:href` to show the feature's name, which is likely more meaningful to the user.

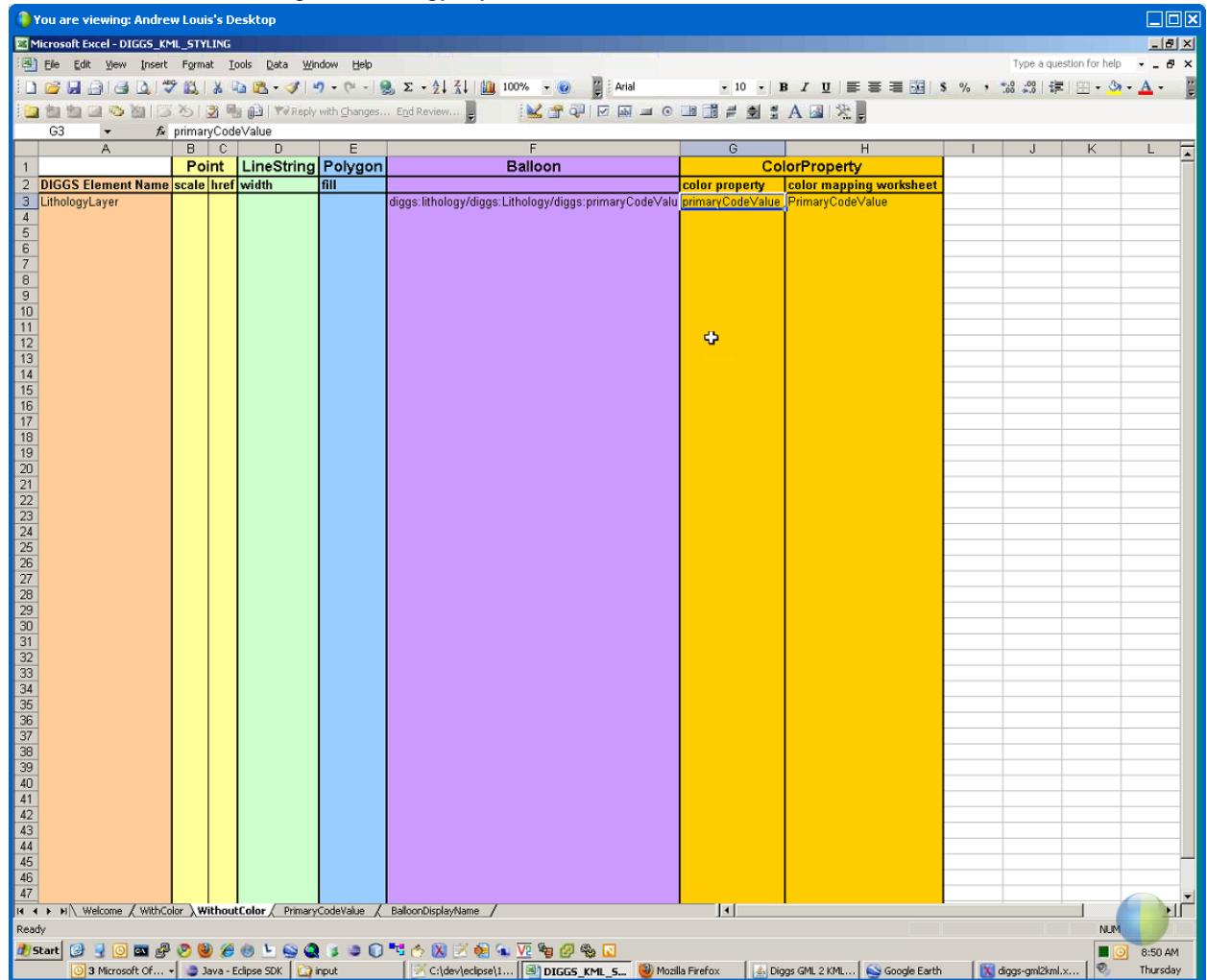
A worksheet maps the xlink:href to the a display name that makes more sense to the reader.



The screenshot shows a Microsoft Excel spreadsheet titled "Microsoft Excel - DIGGS_KML_STYLING". The spreadsheet contains a single sheet named "Associated Location Reference". Column A lists various XPaths, and column B lists their corresponding display names. The data is as follows:

Xpath	Display Name
diggs:associatedLocationRef/@xlink:href	Associated Location Reference
diggs:layerSystemRef/@xlink:href	Layer System Reference
diggs:projectRef/@xlink:href	Project Reference
diggs:samplingActivityRef/@xlink:href	Sampling Activity Reference
gml:identifier	Identifier
gml:identifier/@codeSpace	Identifier Codespace
diggs:lithology/diggs:Lithology/diggs:primaryCodeValue	Lithology Code Value
diggs:lithology/diggs:Lithology/diggs:primaryCodeValue/@codeSpace	Lithology Code Value Codespace

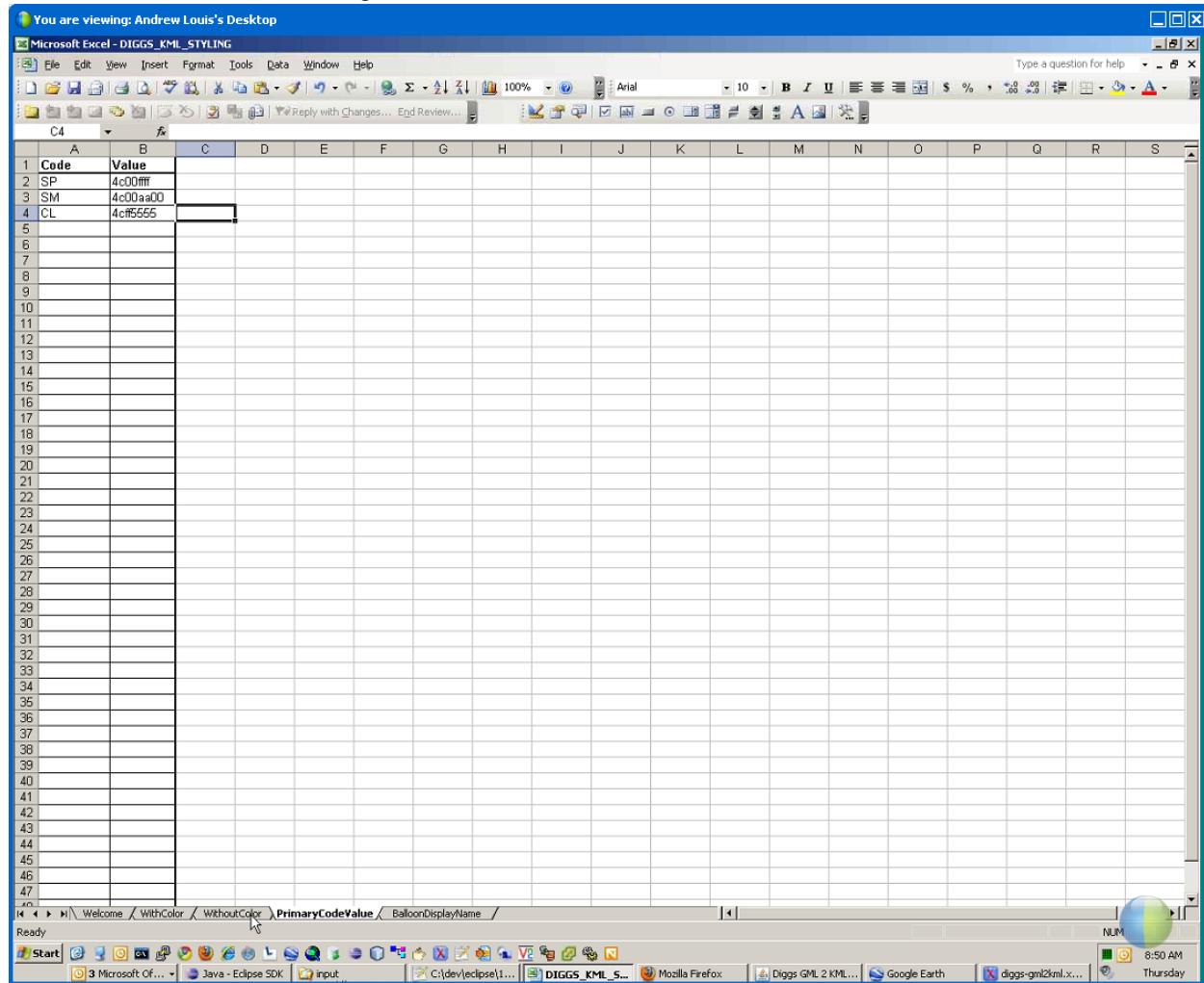
Codes can be defined, e.g. for lithology layers.



The screenshot shows a Microsoft Excel spreadsheet titled "Microsoft Excel - DIGGS_KML_STYLING". The spreadsheet contains a table with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L
1		Point	LineString	Polygon	Balloon		ColorProperty					
2	DIGGS Element Name	scale	href	width	fill	diggs:lithology/diggs:Lithology/diggs:primaryCodeValue	color property	color mapping worksheet				
3	LithologyLayer						primaryCodeValue	PrimaryCodeValue				
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30												
31												
32												
33												
34												
35												
36												
37												
38												
39												
40												
41												
42												
43												
44												
45												
46												
47												

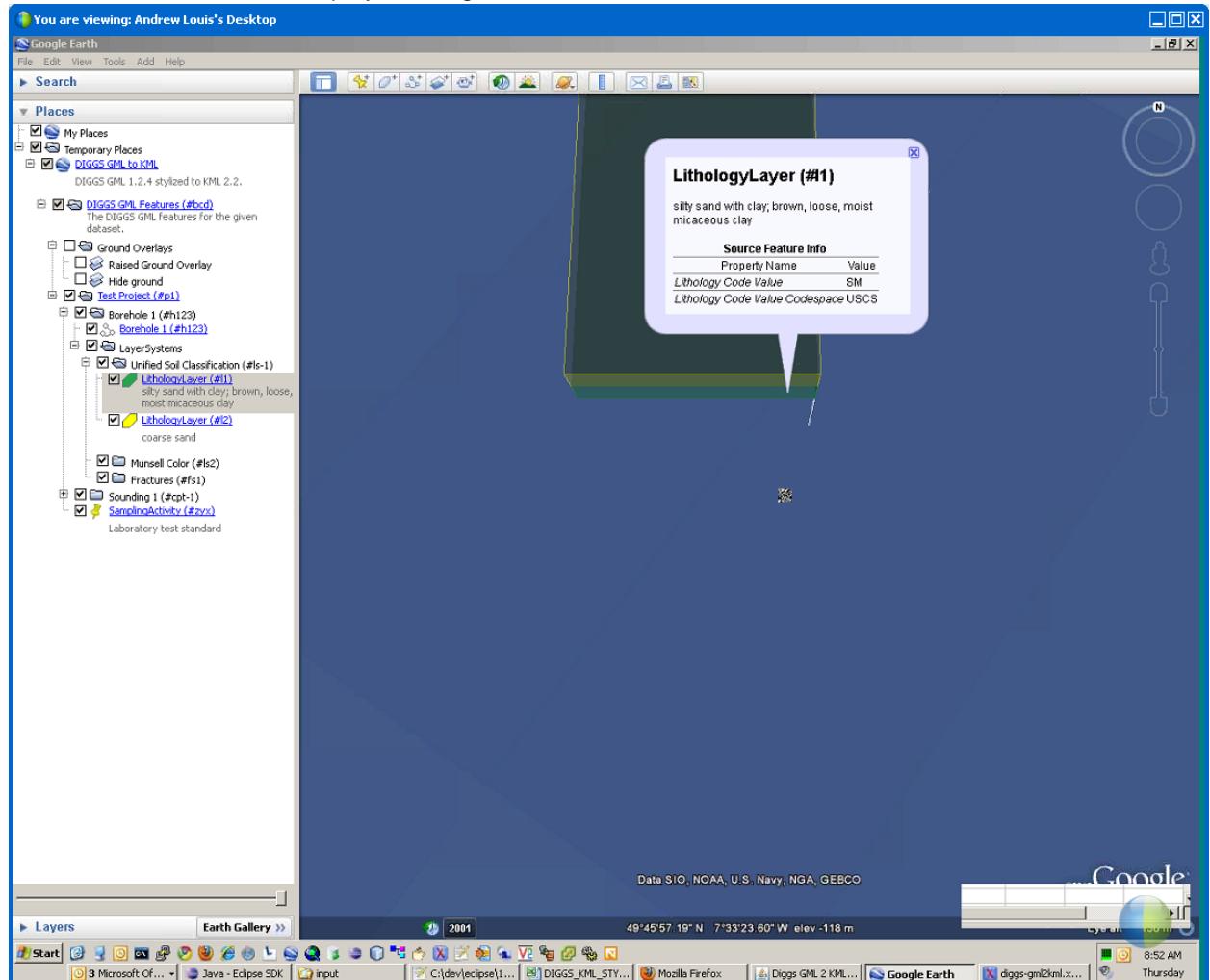
Codes further defined and assigned to colors.



The screenshot shows a Microsoft Excel spreadsheet titled "Microsoft Excel - DIGGS_KML_STYLING". The table has two columns: "Code" and "Value". The "Code" column lists values from 1 to 47, and the "Value" column lists corresponding hex color codes. The first four rows are as follows:

Code	Value
SP	4c00ffff
SM	4c00aa00
CL	4cff5555
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	

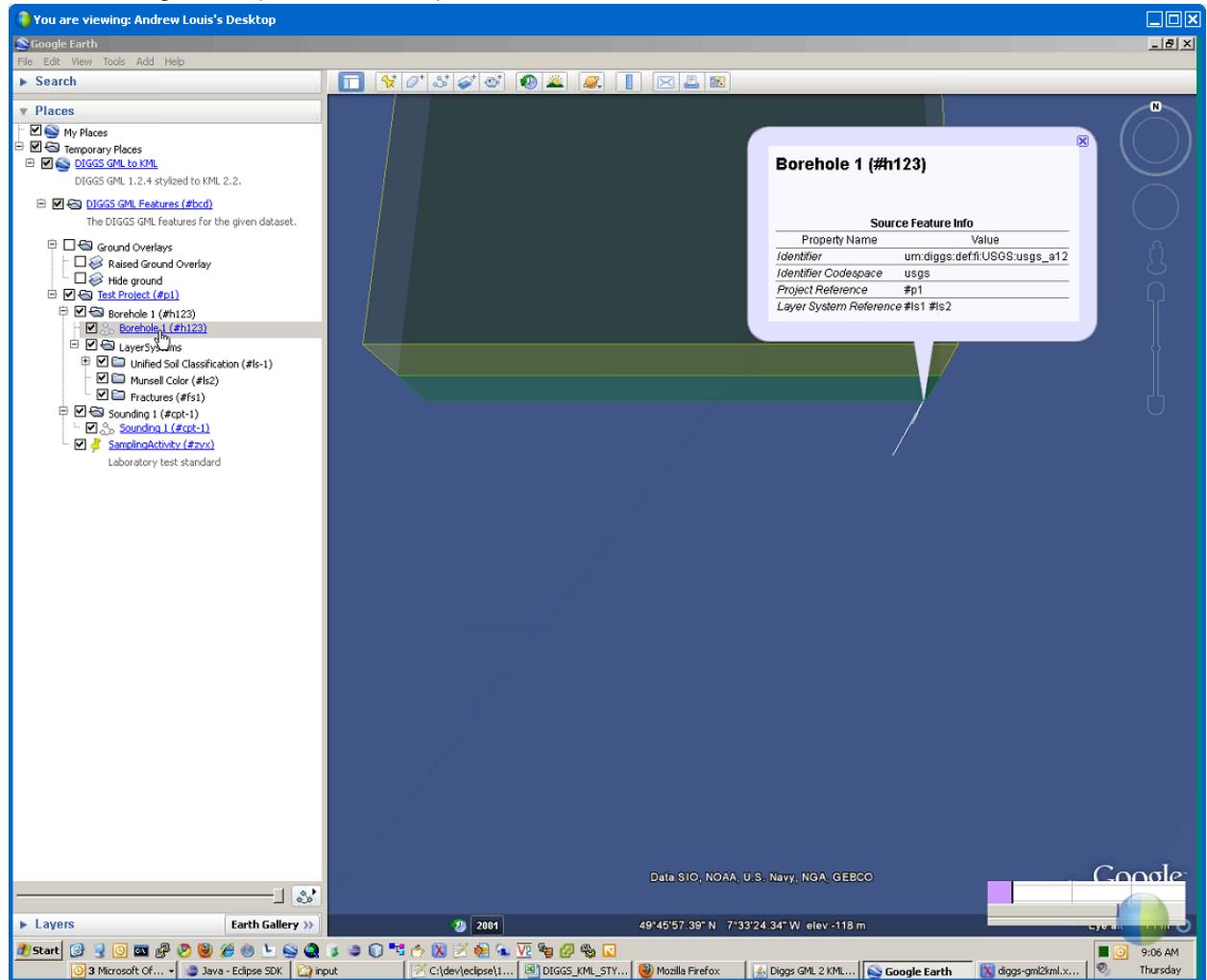
These codes are used to display in GoogleEarth.



This shows the lithology associated with the bounding box of the project, but should really be associated with the borehole location feature. Suggested changes:

- Borehole feature will be a single polyline using default GE display.
- Lithology layers will be line segments.
- Configuration will specify line color and line thickness.

KMZ file is organized (see left sidebar)

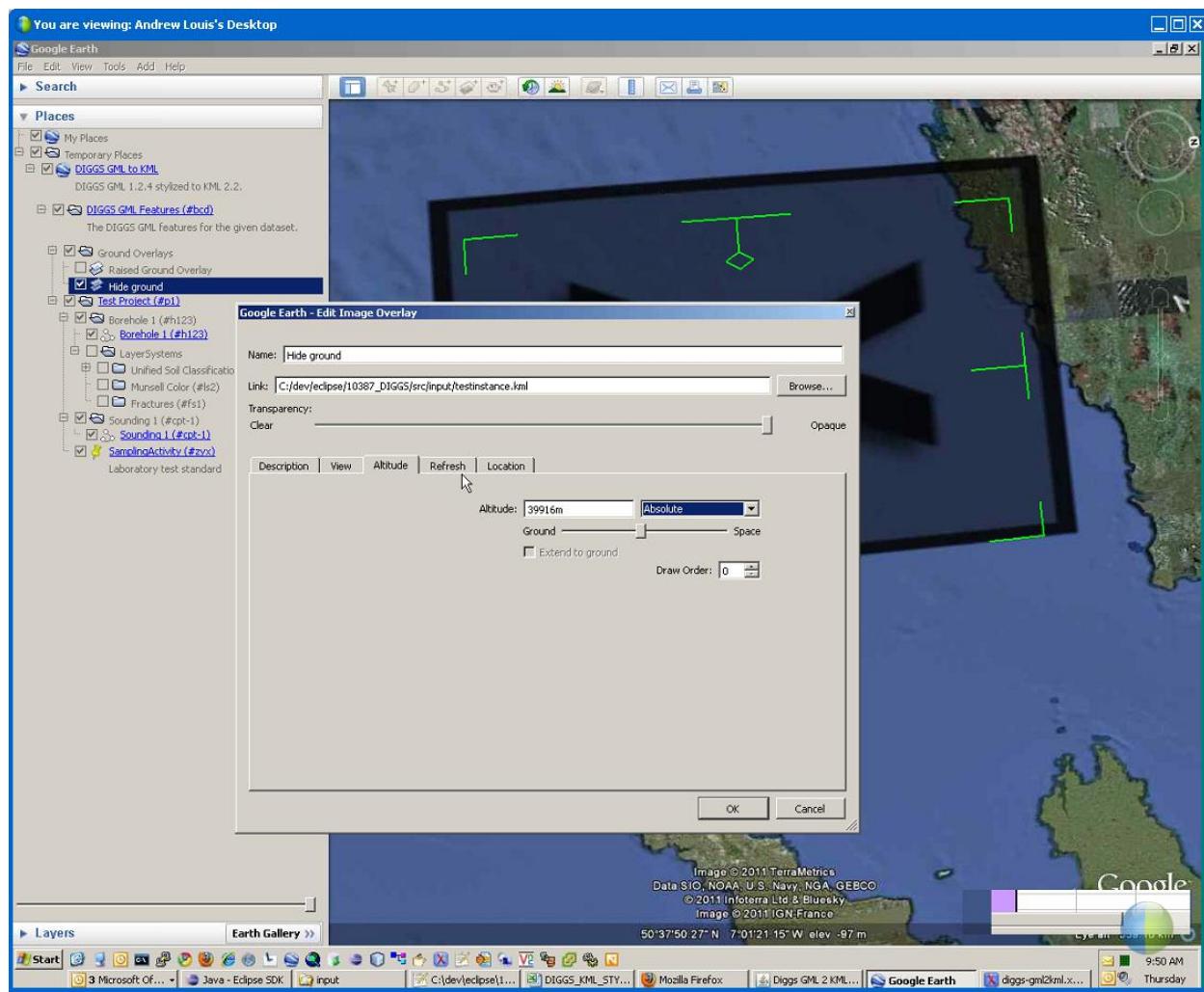


Raised ground overlay.

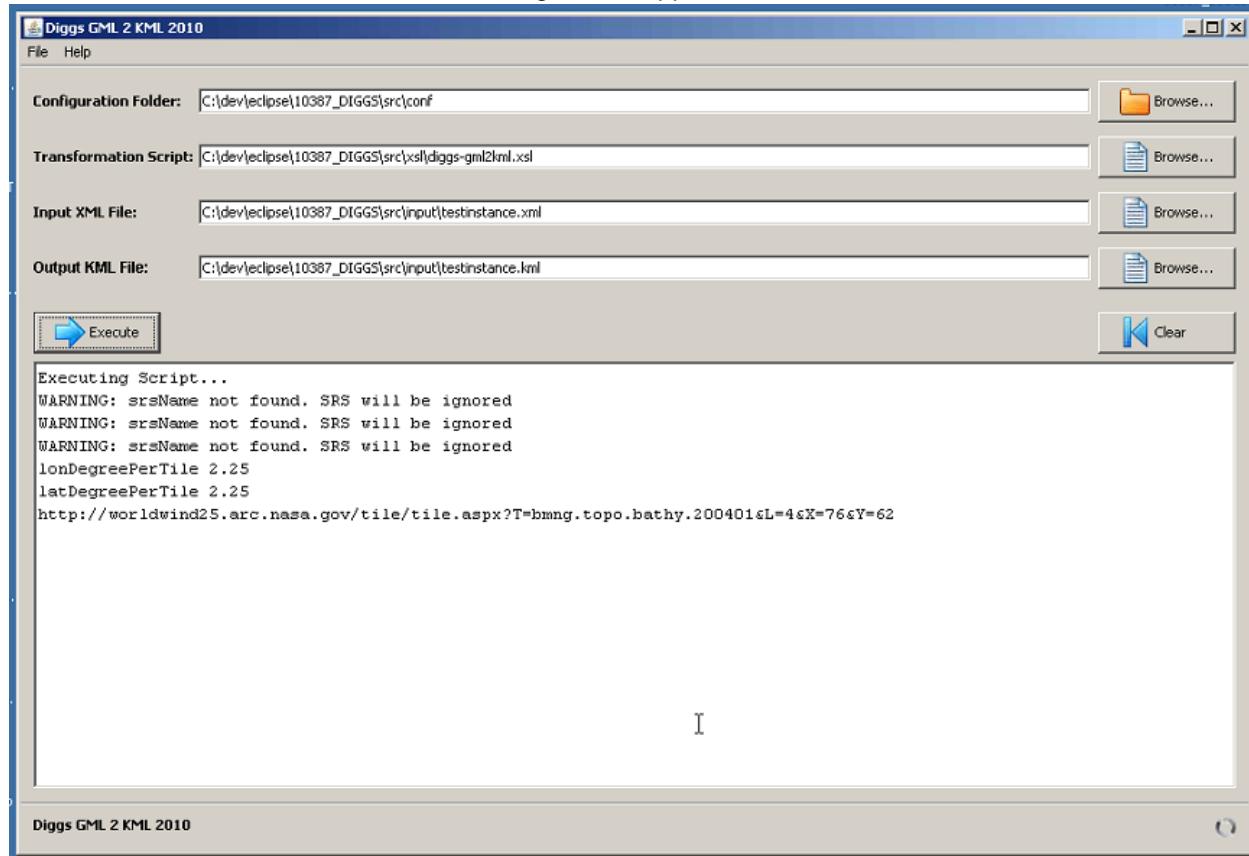
Fetches surface tiles from open-source web map service, *Worldwind*.

Issues:

- Ground overlay tile should be about the size of the project boundary. Need to crop the tiles? This might require a lot more work though.
- Overlay should be at elevation approximately at top of location features.
- Consider holding off on the current ground overlay approach if requires lots of work.
- Alternatively, display a overlay surface with a KML ground overlay using tessellate parameter – Galdos will look into this option further.



Execute the DIGGS to KML conversion, using a Java application.



DIGGS to Excel Tool

Burggraf presented issues with Excel mapping. The translation is challenging – the nested structures in DIGGS requires specific mapping in how to present rows and columns of data in the Excel.

Burggraf suggests:

- Separate worksheets for each object
- Do some specific nesting for table data to reduce number of worksheets

Burggraf will

- Redo the translation in an automatic way to create a first draft spreadsheet.
- Team will look at this first draft and identify worksheets to consolidate.
- Target completion is next week – will email.
- Meeting on 1/18 to recap.

E.42 Teleconference Meeting Notes 2011-01-28

Date: January 28, 2011

Time: 8:00 AM – 11:00 AM (PST)

Participants: Loren Turner

Dan Ponti

David Burggraf

Andrew Louis

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Trenchwall encoding approach:
 - A layer's position is defined by:
 - Top intersection (line)
 - Base intersection (line)
 - Representative surface
 - Defines the left and right edges
 - Combines with the top and bottom to create a polygon
 - All three are mandatory
 - **ACTION:** Burggraf review the trenchwall encoding schema changes. Implement curve orientation. Review changes to make sure valid with GML rules. Issue new version v1.2.4i.
 - **ACTION:** Ponti/Turner close loop with Spink on what was done.
- DIGGS to KML tool
 - Work planning -- approx 5 working days remaining
 - **ACTION:** Ponti/Turner review spreadsheet and provide feedback to Galdos.
- DIGGS to Excel
 - **ACTION:** Ponti and Turner modify spreadsheet to represent how we want to see the data presented.

- DIGGS schema changes for log data:
 - Harmonize structure between geophysical logs and CPT
 - **ACTION:** Ponti will make changes. Burggraf review changes to make sure valid with GML rules and issue v1.2.5.

E.43 Teleconference Meeting Notes 2011-02-03

Date: February 3, 2011

Time: 8:00 AM – 9:30 AM (PST)

Participants: Loren Turner

Dan Ponti

David Burggraf

Chris Bray

Roger Chandler

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Briefing to Bray and Chandler on KML and Excel tools.
 - **ACTION:** Turner will provide files to Bray and Chandler to review.
- No progress this week from last week's meeting. Continue working on action items from last week's meeting.
 - Trenchwall encoding approach:
 - **ACTION:** Burggraf review the trenchwall encoding schema changes. Implement curve orientation. Review changes to make sure valid with GML rules. Issue new version v1.2.4i.
 - **ACTION:** Ponti/Turner close loop with Spink on what was done.
 - DIGGS to KML tool
 - **ACTION:** Ponti/Turner review spreadsheet and provide feedback to Galdos.
 - DIGGS to Excel
 - **ACTION:** Ponti and Turner modify spreadsheet to represent how we want to see the data presented.
 - DIGGS schema changes for log data:
 - **ACTION:** Ponti will make changes. Burggraf review changes to make sure valid with GML rules and issue v1.2.5.

E.44 Teleconference Meeting Notes 2011-02-09

Date: February 9, 2011 **Time:** 8:00 AM – 9:30 AM (PST)

Participants: Loren Turner

Dan Ponti

David Burggraf

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Trenchwall encoding approach:
 - Ponti implemented changes to trenchwall encoding in DIGGS v1.2.4h and provided to Burggraf.
 - Burggraf reviewed the trenchwall encoding schema changes and implemented the OrientableCurve element (part of GML 3.2), provided in DIGGS v.1.2.4i, as in the example snippet.
 - ```
<representativeSurface>
 <gml:Polygon gml:id="lt1-p">
 <gml:exterior>
 <gml:Ring>
 <gml:curveMember xlink:href="#twp-int-11"/>
 <!--Top edge -->
 <gml:curveMember>
 <gml:LineString gml:id="lt1-1">
 <!--Right edge -->
 <gml:posList srsDimension="2" srsName="#lsrs002">50 1 50
 4</gml:posList>
 </gml:LineString>
 </gml:curveMember>
 <gml:curveMember>
 <gml:OrientableCurve gml:id="OC_twp-int-12" orientation="-">
 <gml:baseCurve xlink:href="#twp-int-12"/>
 </gml:OrientableCurve>
 </gml:curveMember>
 <!--Basal edge in reverse orientation -->
 <gml:curveMember>
 <gml:LineString gml:id="lt1-2">
 <!--Left edge -->
 <gml:posList srsDimension="2" srsName="#lsrs002">0 5 0 2</gml:posList>
 </gml:LineString>
 </gml:curveMember>
 <gml:Ring>
 </gml:exterior>
 </gml:Polygon>
 • </representativeSurface>
 •
```
  - Ponti added trueTopObserved and trueBaseObserved boolean properties (optional) to the AbstractLocationPosition base class. This allows the DIGGS instance author to state whether the top or base are known. There was discussion as to the need for a “default” state. We decided that there should be no default, as the absence of this implies that the value is not known or could not be determined.

- ```

<sequence>
  <element minOccurs="0" name="trueTopObserved" type="boolean"
default="true">
    <annotation>
      <documentation>True or false, indicating whether the geometry of the top
position of
      the feature represents the true top of the feature being defined (true) or (false)
      whether the top represents the upper extent of the feature that is defined by the
      extent of the location feature itself.</documentation>
    </annotation>
  </element>
  <element minOccurs="0" name="trueBaseObserved" type="boolean"
default="true">
    <annotation>
      <documentation>True or false, indicating whether the geometry of the top
position of
      the feature represents the true top of the feature being defined (true) or (false)
      whether the top represents the upper extent of the feature that is defined by the
      extent of the location feature itself.</documentation>
    </annotation>
  </element>
  • </sequence>

```
- Ponti explained that layerIntersectionPosition is of type LayerIntersectionPropertyType that refers to a LayerIntersections object of type LayerIntersectionType. This type inherits from diggs:AbstractObjectType and adds three mandatory properties: a) layerTopIntersection (MultiCurvePropertyType), b) layerBaseIntersection (MultiCurvePropertyType), and c) representativeSurface (1..n; SurfacePropertyType). Burggraf identified a design inconsistency and suggested the following modification, which was adopted:
 - ```

<sequence>
 <element name="layerTopIntersection" type="gml:CurvePropertyType"
maxOccurs="unbounded">
 <annotation>
 <documentation>The position of the top surface of a layer (a volume of earth material)
 that intersects a trench wall. Depending on how a trench wall is encoded, there
 could be multiple line strings that define this top surface.</documentation>
 </annotation>
 </element>
 <element name="layerBaseIntersection" type="gml:CurvePropertyType"
maxOccurs="unbounded">
 <annotation>
 <documentation>The position of the basal surface of a layer (a volume of earth
 material) that intersects a trench wall. Depending on how a trench wall is encoded,
 there could be multiple line strings that define this basal surface.</documentation>
 </annotation>
 </element>

```

- ```

<element name="representativeSurface" type="gml:SurfacePropertyType" maxOccurs="unbounded">
  <annotation>
    <documentation>One or more polygons that represent the layer volume in the plane of
      the trench wall. This would typically be a ring polygon with curve members that
      consist of the layer top intersection, layer bottom intersection, and two trench
      wall edges.</documentation>
  </annotation>
</element>

```
- - **ACTION:** Ponti will implement changes to schema and include in the v1.2.5 release. Burggraf will review changes and validate.
 - **ACTION:** Ponti/Turner close loop with Spink on recent changes to trenchwall encoding.
 - DIGGS to KML tool
 - Turner has been reviewing and iterating with Galdos on the styling configuration for the KML tool. A second version was issued yesterday.
 - Java runtime issues seem to be resolved in the latest version.
 - **ACTION:** Ponti/Turner/Bray continue to review the conversion tool and provide feedback to Galdos.
 - DIGGS to Excel
 - No review has conducted yet on this.
 - **ACTION:** Ponti and Turner modify spreadsheet to represent how we want to see the data presented. Provide feedback to Galdos.
 - DIGGS schema changes for lab testing and log data:
 - **ACTION:** Turner and Ponti will meet next week Tuesday to discuss approach for schema for lab testing and insitu log data. Ponti will develop sample schema to see how it looks. We'll discuss this at next Thursday's status meeting. Changes will eventually be part of DIGGS v1.2.5.

E.45 Teleconference Meeting Notes 2011-02-17

Date: February 17, 2011

Time: 8:00 AM – 8:30 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Turner and Ponti met earlier in the week to discuss the test data structure. The proposed structure would focus on the results rather than the process. We'd create an Abstract Test Result that has the following:
 - Derives from DIGGS abstract feature
 - Sits at top level of hierarchy
 - Result property contains result object (e.g. density, gravity, etc.)
 - Process property, can reference more than one test object
 - Reference to a location, **mandatory**
 - Reference to a sample, **optional**
 - Position(s) object, **mandatory**
- An example instance:

```

<MoistureContent gml:id = ""/>
  <location xlink:href = "#b123"/>
  <sampleRef xlink:href = "#s345"/>
  <position>
    <BoreholePosition>
      ...
      ...
    </BoreholePosition>
  </position>
  <result>
    <MoistureContentResult gml:id="">
      <result uom="%>22.4</result>
      <isNatural>true</isNatural>
    </MoistureContentResult>
  </result>
  <process>
    <MoistureContentTest gml:id="">
      <tare>55.5</tare>
      <tareplussoil>66.6</tareplussoil>
      <specifications>
      <description>
        <equipment xlink:href = "#abc123">
      </MoistureContentTest>
    </process>
  </MoistureContent>

```

- Turner began to look at results and processes by compiling a matrix.
- Ponti will send Burggraf some example instances that show how the revised test results schema might look. Burggraf will check to make sure complies with GML and fits within the schema design.
- We still need to follow up with Chandler on the monitoring structure.
- Turner will set up meetings with GMS and Core SIG to provide an update on the status of work on DIGGS.

- Turner emailed initial feedback on Excel tool:
 - The table data (e.g. for CPT, geophysics) doesn't import into the Excel, likely due to the size of the block of data. Modify how this is mapped to Excel such that the data in the block is parsed by the delimiters, where each data point occupies a single cell. Ideally, column data (e.g. depths from the coverage element) would be presented in a single column, and subsequent data presented in adjacent columns).
 - In the "Attribute Name" column:
 - Remove the "@" symbol that precedes the name of the attribute.
 - Right justify attributes within the cells. Use a colon, e.g. "Identifier Ref:"
 - Don't display the type, such as [Text], [Inline Object/Feature]. Just leave cell blank.
 - Don't display "@XLink HREF" -- leave the cell blank where ever that occurs, since the first column indicates it is a reference anyways.
 - Maybe, don't show the namespace prefix, since the color coding indicates it. e.g. instead of "@GML ID", use "ID"
- Turner will continue to review KML and Excel tools.

E.46 Teleconference Meeting Notes 2011-03-24

Date: March 24, 2011

Time: 8:00 AM – 9:30 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Chris Bray

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Impact of GML 3.3 on DIGGS.
 - GML 3.3 draft has gone through major reviews through the OGC and needs to be ratified, likely in September 2011. GML 3.3 is part 2 of the ISO standard established under GML 3.2.
 - Linear referencing changed a bit from the version we implemented in v1.2
 - GML 3.3 has been modularized into 3 namespaces.
 - Offset vector is now described by angle, not a vector.
 - Compact geometry encoding – not really applicable to DIGGS.
 - GML needs to be changed.
 - Other schemas that import need to import the multiple 3.3 namespaces.

- **ACTION:** Burggraf to modify the GML profile and generate DIGGS v1.2.4j. Also, will point out where the changes impact DIGGS. Provide by next week's meeting.
- Status of KML and Excel tools.
 - Jeremy fixed the one issue with the referencing link in the Excel tool.
 - GML 3.3 changes will require minor changes to Excel and KML tools.
 - Offset vector change will impact KML for the trenchwall.
 - **ACTION:** Make the changes to the tools for v1.2.4j. Update the *testinstance.xml* file. Provide by next week's meeting.
 - No likely changes in the Excel tool.
 - Approx 2 days remaining for Excel work, 5 days for KML work. Hold off on further work on this until we make changes in 1.2 and 2.0.
- Punchlist to finish DIGGS v1.2:
 - Modifications to the test data structure. Ponti and Turner to talk early next week.
 - Modifications to the monitoring data structure.
 - Finalize codelists.
 - Update the 20 examples.
 - Implement recently updated GML 3.3 (ISO) structures.
 - Create schema documentation.
- Priorities for remaining project term?
 - Schematron assertions defined.
 - Web authoring tool
- Next week Thursday meeting at 8:00 AM

Date: March 24, 2011

Time: 8:00 AM – 9:30 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Chris Bray

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Impact of GML 3.3 on DIGGS.
 - GML 3.3 draft has gone through major reviews through the OGC and needs to be ratified, likely in September 2011. GML 3.3 is part 2 of the ISO standard established under GML 3.2.
 - Linear referencing changed a bit from the version we implemented in v1.2
 - GML 3.3 has been modularized into 3 namespaces.
 - Offset vector is now described by angle, not a vector.
 - Compact geometry encoding – not really applicable to DIGGS.
 - GML needs to be changed.

- Other schemas that import need to import the multiple 3.3 namespaces.
- **ACTION:** Burgraff to modify the GML profile and generate DIGGS v1.2.4j. Also, will point out where the changes impact DIGGS. Provide by next week's meeting.
- Status of KML and Excel tools.
 - Jeremy fixed the one issue with the referencing link in the Excel tool.
 - GML 3.3 changes will require minor changes to Excel and KML tools.
 - Offset vector change will impact KML for the trenchwall.
 - **ACTION:** Make the changes to the tools for v1.2.4j. Update the *testinstance.xml* file. Provide by next week's meeting.
 - No likely changes in the Excel tool.
 - Approx 2 days remaining for Excel work, 5 days for KML work. Hold off on further work on this until we make changes in 1.2 and 2.0.
- Punchlist to finish DIGGS v1.2:
 - Modifications to the test data structure. Ponti and Turner to talk early next week.
 - Modifications to the monitoring data structure.
 - Finalize codelists.
 - Update the 20 examples.
 - Implement recently updated GML 3.3 (ISO) structures.
 - Create schema documentation.
- Priorities for remaining project term?
 - Schematron assertions defined.
 - Web authoring tool
- Next week Thursday meeting at 8:00 AM

E.47 Teleconference Meeting Notes 2011-04-01

Date: April 1, 2011

Time: 8:00 AM – 9:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burgraff

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Burgraff provided revised DIGGS v1.2.4j which incorporates GML 3.3. Updated the *testinstance.xml* file.
- Status of KML and Excel tools.
 - **ACTION:** Make the changes to the tools for v1.2.4j. Update the *testinstance.xml* file. Provide by next week's meeting.

- Punchlist to finish DIGGS v1.2:
 - Modifications to the test data structure. Ponti and Turner to talk early next week.
 - Modifications to the monitoring data structure.
 - Finalize codelists.
 - Update the 20 examples.
 - Implement recently updated GML 3.3 (ISO) structures.
 - Create schema documentation.
- Turner presented preliminary work on codelist spreadsheets. **ACTION:** Finalize draft prior to next meeting.
- Turner and Ponti meet next Tuesday to discuss lab test data schema
- Next week Thursday meeting at 8:00 AM

E.48 Teleconference Meeting Notes 2011-04-07

Date: April 7, 2011

Time: 8:00 AM – 9:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burgraf

Agenda: DIGGS schema update; DIGGS to KML tool & DIGGS to Excel Tool Discussion

Discussion:

- Some issues identified in GML 3.3 linear referencing structure. Specifically the offset plane, perpendicular to the line string, creates an issue when referencing points at angle breaks in the line string. The GML team will fix this quickly, likely in the next few days. The new changes will be propagated throughout DIGGS and into the tools and testinstance. Likely change will be to go back to the vector approach, or a choice between vector or planar. DIGGS will restrict via profile to use vector approach.
- **ACTION:** Make the changes to the schemas, tools, and the *testinstance.xml* file. Provide when done in DIGGS v1.2.4k.
- Recap – punchlist to finish DIGGS v1.2:
 - Modifications to the test data structure. Ponti and Turner to talk early next week.
 - Modifications to the monitoring data structure.
 - Finalize codelists.
 - Update the 20 examples.
 - Implement recently updated GML 3.3 (ISO) structures.
 - Create schema documentation.
- Turner is still working on codelist spreadsheets – about half completed. **ACTION:** Finalize draft prior to next meeting.
- Next week Friday meeting at 8:00 AM.

E.49 Teleconference Meeting Notes 2011-05-05

Date: May 5, 2011

Time: 8:00 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Agenda: Discuss DIGGS codelist work

Discussion:

- Turner briefed the team on recent work on the codelists. Codetypes listed in the spreadsheet were located within the schema, matched to an existing codelist if available, categorized by type A/B/C, assigned to be with or without authority, and given a recommendation on any actions. Most of this codelist has been done, with a dozen or so remaining. Observations so far:
 - Most codetype elements will likely be without authority. That is, the codespace will not be required on most of these elements.
 - There will be a handful of elements requiring codespace, such as lithology codes.
 - Most of the codetype elements fell into categories “B” or “C”. No type “A” elements identified so far.

| Type | Description | Proposed DIGGS Implementation |
|------|--|---|
| A | Codes to describe in more detail a specific data element, where the data element cannot be controlled or validated by the schema alone (e.g. table data and CPT parameter names). | If the code is absolutely necessary for DIGGS to function and be unambiguous of source and target data interchange, then these codes should be implemented into enumerated lists. Enumerated lists are part of the schema and are validated by schema alone. |
| B | Codes created, maintained, and published by recognized standards organizations, used in practice, and commonly referenced with or without software (e.g. USCS Group Symbols for soil classification, Munsell color codes, EPSG spatial reference codes). | For codes that are commonly referenced, nomenclature and abbreviations well documented, and maintained by a standards body, these should be implemented in DIGGS using codetype and codespace attributes. DIGGS might require that some codetype and codespace attributes be mandatory. Although the codespace would reference the standards organization (e.g. USCS, AASHTO), the full list of codes (e.g. SP, SW) would not be in the codelist, since the standards organization maintains this list, and it would be left to the users to comply with the standards published by that standards organization. |
| C | Codes created by an organization, government agency, trade group, or company to standardize nomenclature and terms across a specific user base (e.g. roles, titles, equipment names, test names). | Codes that are used in localized practice should be made available for integration into DIGGS as needed. Codespace and codetype attributes would be optional. This would be applicable, for example, for codes such as "roles" where the value itself likely carries meaning without other external references. However, specific user groups may want to standardize the possible values being used. Three possibilities: <ul style="list-style-type: none"> • DIGGS file authors could simply use codes (uncontrolled) without any reference to a codetype or codespace. However, the recipient of the DIGGS file would not know what standards are being referenced. • The DIGGS author could populate the codetype and codespace attributes. Since these are optional and the format uncontrolled, the recipient may still be unable to resolve the references in a systematic manner. • The DIGGS author could reference a published codespace that can be validated with schematron. |

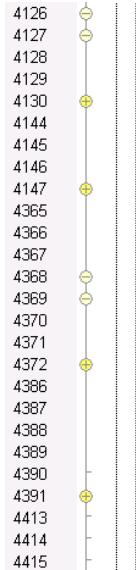
- Most of the codes will be eliminated from the standard DIGGS distribution. In the future each authority will have a separate codelist that can be referenced by the codespace attribute. Those codelists will be created by the DIGGS team and posted to the DIGGS site initially, serving as an example of how the codelists are assembled, referenced, and maintained.
- The enumerations in the spreadsheet seemed to be missing a lot of "Enum" type elements. This needs to be looked at further.

- **ACTIONS:**

- Burggraf regenerate enumeration spreadsheet.
- Turner to complete the remaining codelist evaluation.
- Ponti to review Turner's work and discuss next week to reach consensus.
- Burggraf to implement codelist recommendations, including:
 - Create individual codelist for each authority.
 - Modify codetype elements per recommendations
 - Modify enumerations per recommendations
- Use of "status"? Talk with Roger about this.
- Turner identified a potential issue with the implementation of the constituent element.
 - In Caltrans practice the constituent is typically used to describe a sample or layer lithology. In the following example the underlined text would be a constituent of a layer. Note that the constituent includes descriptive terms for particle aspect ratio ("flat and elongated") and particle shape ("subrounded to rounded").

*Well-graded SAND with GRAVEL (SW); medium dense; brown and light gray; wet; 75% SAND, from coarse to fine, rounded; **20% GRAVEL, coarse, subrounded to rounded, flat and elongated;** 5% fines; weak cementation.*

- In DIGGS, the particle aspect ratio and shape are not part of the constituent element. This is because the constituent can be used to describe more than just lithologic information.

| | |
|--|---|
|  | <pre> 4126 <diggs:lithology> 4127 <diggs:Lithology gml:id="ID_192" xml:lang="en-us" diggs:technique="unknown"> 4128 <gml:description xlink:role="http://www.altova.com" xlink:type="simple" nilReason="inapplicable" xlink:title="</gml:description> 4129 <diggs:status codeSpace="http://www.altova.com">String</diggs:status> 4130 <diggs:remarks owns="false"> 4131 <diggs:primaryCodeValue codeSpace="http://www.altova.com">String</diggs:primaryCodeValue> 4132 <diggs:alternateCode codeSpace="http://www.altova.com">String</diggs:alternateCode> 4133 <diggs:alternateCode codeSpace="http://www.altova.com">String</diggs:alternateCode> 4134 <diggs:particleSizes> 4135 <diggs:particleShape codeSpace="http://www.altova.com">String</diggs:particleShape> 4136 <diggs:particleSorting codeSpace="http://www.altova.com">String</diggs:particleSorting> 4137 <diggs:particleAspectRatio codeSpace="http://www.altova.com">String</diggs:particleAspectRatio> 4138 <diggs:constituents> 4139 <diggs:Constituent gml:id="ID_203" xml:lang="en-us" diggs:technique="unknown"> 4140 <gml:description xlink:role="http://www.altova.com" xlink:type="simple" nilReason="inapplicable" xlink:title="</gml:description> 4141 <diggs:status codeSpace="http://www.altova.com">String</diggs:status> 4142 <diggs:remarks owns="false"> 4143 <diggs:constituentType codeSpace="http://www.altova.com">String</diggs:constituentType> 4144 <diggs:value codeSpace="http://www.altova.com">String</diggs:value> 4145 <diggs:abundanceCode codeSpace="http://www.altova.com">String</diggs:abundanceCode> 4146 <diggs:abundancePercent uom "%">0</diggs:abundancePercent> 4147 </diggs:Constituent> 4148 <diggs:Constituent gml:id="ID_204" xml:lang="en-us" diggs:technique="unknown"> 4149 </diggs:constituents> 4150 </diggs:Lithology> 4151 </diggs:constituents> 4152 </diggs:lithology> </pre> |
|--|---|

- The proposed solution is to create two types of constituent elements, one specific to lithologic constituents (containing the particle elements), and another that's generic and could be used to describe all other constituents.
- **ACTION:** Need to identify how we do this, and who will do this. Turner and Ponti should discuss this further to see if the proposed solution works for other kinds of scenarios before proceeding.
- Issue identified in prior meeting:
 - Some issues identified in GML 3.3 linear referencing structure. Specifically the offset plane, perpendicular to the line string, creates an issue when referencing points at angle breaks in the line string. The GML team will fix this quickly, likely in the next few days. The new changes will be propagated throughout DIGGS and into the tools and testinstance. Likely change will be to go back to the vector approach, or a choice between vector or planar. DIGGS will restrict via profile to use vector approach.
 - **ACTION:** If not already done, Burggraf make the changes to the schemas, tools, and the *testinstance.xml* file. Provide when done in DIGGS v1.2.4k.
- Recap – punchlist to finish DIGGS v1.2:
 - Modifications to the test data structure.
 - Modifications to the monitoring data structure.
 - Finalize codelists.
 - Update the 20 examples.
 - Implement recently updated GML 3.3 (ISO) structures.
 - Create schema documentation.

- Next group meeting will be Thursday 5/19/11 at 8:00 AM.
- Dan & Loren will meet on Thursday 5/12/11 at 9:00 AM to finalize work on codelist and lab test data structure.

E.50 Teleconference Meeting Notes 2011-05-19

Date: May 19, 2011

Time: 8:00 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf
Roger Chandler

Agenda: Discuss proposal for revised DIGGS test data structure

Discussion:

- Ponti and Turner have been working on a revised approach for encoding lab and insitu test data in DIGGS that more closely resembles the structure used for the current CPT data.
- Ponti implemented the major structural changes in the DIGGS kernel, packed in 1.2.4.k
- Turner has been evaluating each test, comparing with AGS4 and State DOT practices, and creating new test procedures as needed.
- Ponti explained new test data structure to group. (From 5/19/11 Ponti email):

Attached is an interim 1.2.4.k version that implements a proposed change in how DIGGS handles tests. Will explain the reasons for the change and the specifics in more detail in the call, but in brief:

The top level test features in prior versions have been replaced by a single Measurement feature. This feature is patterned after the OGC Observation feature (but doesn't derive from it). It is defined as "An act ("event"), whose results are estimates of the value of a property or properties of interest at a position or series of positions on or within the earth." The feature has two main properties: 1) outcome, and 2) procedure. The outcome property contains a MeasurementResult object that derives from a GML coverage feature. This feature contains both the position information for the measurement (eg. where the measurement applies), that derives from the coverage domain, and a results property (derives from the coverage range)that contains information about the property(ies) derived from the measurement, and the value(s) of those properties. The procedure property contains zero to one test object, that describes the metadata for the test procedure and any interim results of value. The existing test features in prior versions

of DIGGS are being converted to these test objects - and contain the same parameter properties as before, minus the properties that are the reported test results.

The Geotechnical.xsd schema in this current version only contains a few test procedures. Loren is working on converting the others and making sure they are mappable to the AGS 4.0 data dictionary. The file Geotechnical-old.xsd is the prior (1.2.4.j) version.

A new test instance document: testinstance-newMeasurements.xml, shows how the new Measurement structure works for both the prior tests in that document, and a few others.

Advantages/Rationale for the change:

- 1) All "results" of test (eg. soil, chemical, hydrologic properties) occur at a single place in the schema. In this encoding scheme, the "result" is what matters and can be easily extracted from an instance document.
- 2) Measurement results can be reported without the need to instantiate a fake sample or test feature for legacy data where only the result and position are known. While the procedure property is a mandatory part of the Measurement feature, it need not be populated by a measurement object.
- 3) All results are encoded in the same consistent fashion - results of in-situ tests, CPT, geophysical logs, lab tests on samples, etc.
- 4) Provides a more flexible and practice-friendly means of associating results with test procedures and sampling features.
- 5) Provides a basis for developing a parallel structure for time-varying observations (eg. monitoring).

Disadvantages:

- 1) Currently uses a generic property object to describe the property being measured - this means that the property type must come from a controlled list (dictionary) to maintain interoperability.
- 2) Some schema control is lost (more dependence on schematron to ensure that an instance document makes sense).
- 3) GML coverage encoding for all measurement results is not as human readable and somewhat more complex to parse.

However, current DIGGS is already saddled with these problems w/respect to how CPT and geophysical logs need to be encoded and will have similar issues with any kind of test and monitoring results that are tabular in form, so we don't see these issues and being a significant detriment to this plan.

In addition, 1.2.4.k makes some relatively minor changes to the layer system structure and in particular the lithology layer and lithology objects to better accommodate US DOT practice. We can discuss these as well, but they are of less consequence than the above.

- There was general concurrence from the group that the proposed approach should be pursued further.

E.51 Teleconference Meeting Notes 2011-11-02

Date: November 2, 2011

Time: 10:00 AM – 12:00 PM (PST)

Participants: Loren Turner
Dan Ponti

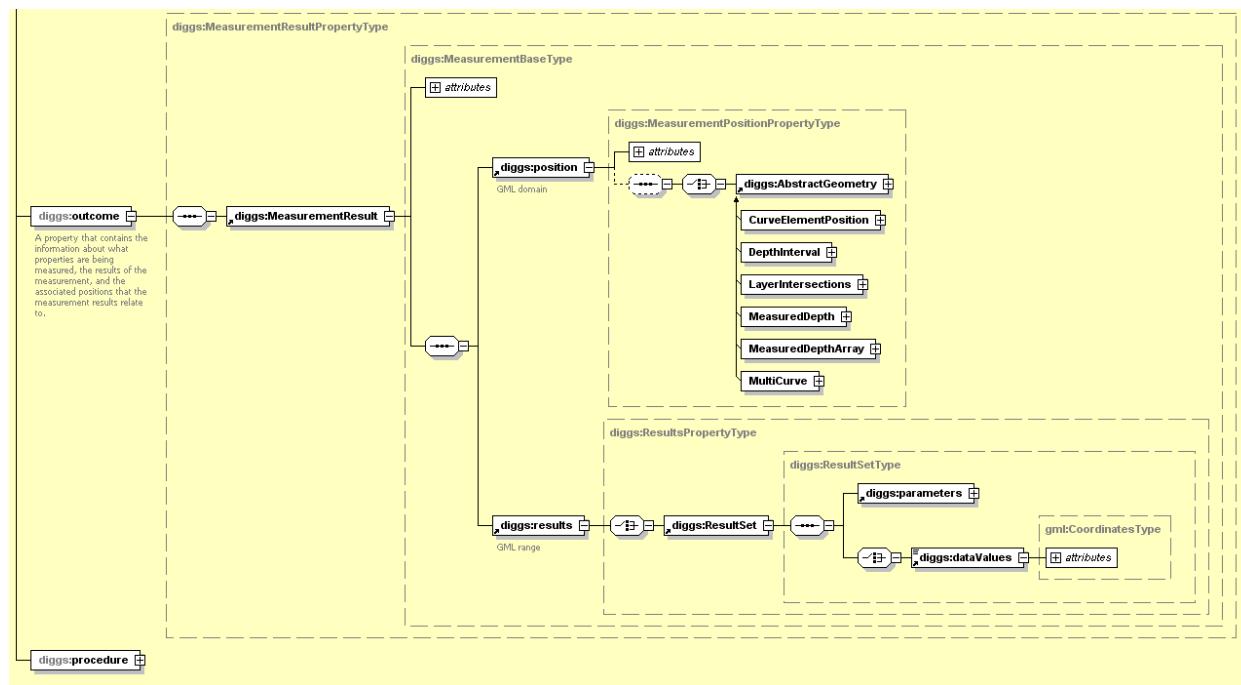
Agenda: Discuss testing and monitoring data structures

Discussion:

Turner and Ponti reviewed the current state of DIGGS v1.2.4.k.j and the structure of the “Measurement” object. This is the object used to capture all test information – the procedure used and the results. At a high level, this object is structured as follows:

- Measurement
 - Outcome
 - Position (*GML Domain Set*)
 - Results (*GML Range Set*)
 - Procedure

The results and the position of the results are contained in the “Outcome” element (e.g. shear strength). The test procedure used is contained in the “Procedure” element (e.g. triaxial test).



This structure is based on the GML 3.3 “coverage” model, where the *Domain Set* defines the position and the *Range Set* contains the values. The structure is flexible such that simple result sets or large sets of values (e.g. CPT) can be encoded. For example, multiple results at a single depth interval:

```

<!-- Laboratory Particle Size analysis results on same sample as above -->
<Measurement gml:id="ps1">
  <projectRef xlink:href="#p1"/>
  <associatedLocationRef xlink:href="#LB_Webster"/>
  <sampleRef xlink:href="#s123"/>
  <samplingTime>
    <TimeInterval gml:id="ti13">
      <start>2004-11-12</start>
    </TimeInterval>
  </samplingTime>
  <resultTime>
    <TimeInterval gml:id="ti14">
      <start>2005-01-16</start>
    </TimeInterval>
  </resultTime>
  <outcome>
    <MeasurementResult gml:id="m102">
      <position>
        <DepthInterval gml:id="gm124">
          <gml:posList srsName="#sr123" srsDimension="1">5 6</gml:posList>
        </DepthInterval>
      </position>
      <results>
        <ResultSet>
          <parameters>
            <PropertyParameters gml:id="pp1">
              <properties>
                <Property index="1" gml:id="psp2">
                  <mnemonic>Percent Fines</mnemonic>
                  <typeData>double</typeData>
                  <propertyName codeSpace="urn:x-diggs:def:code-list:DIGGS:property">percent_fines</propertyName>
                  < uom-%>
                </Property>
                <Property index="2" gml:id="psp3">
                  <mnemonic>Coef. of Uniformity</mnemonic>
                  <typeData>double</typeData>
                  <propertyName codeSpace="urn:x-diggs:def:code-list:DIGGS:property">coef_uniformity</propertyName>
                </Property>
                <Property index="3" gml:id="psp4">

```

```

<mnemonic>Median Grainsize</mnemonic>
<typeData>double</typeData>
<propertyName codeSpace="urn:x-diggs:def:code-list:DIGGS:property">d50</propertyName>
< uom>mm</uom>
</Property>
</properties>
</PropertyParameters>
</parameters>
<dataValues>18.5,1.39,2.5</dataValues>
</ResultSet>
</results>
</MeasurementResult>
</outcome>
</procedure>
<diggs_geo:ParticleSizeTest gml:id="pst">
<specification>ASTM D422-63(2002)</specification>
<diggs_geo:gradingData>
<diggs_geo:Grading gml:id="g1">
<diggs_geo:particleSize uom="mm">4</diggs_geo:particleSize>
<diggs_geo:percentPassing uom "%">84</diggs_geo:percentPassing>
</diggs_geo:Grading>
<diggs_geo:Grading gml:id="g2">
<diggs_geo:particleSize uom="mm">2</diggs_geo:particleSize>
<diggs_geo:percentPassing uom "%">56</diggs_geo:percentPassing>
</diggs_geo:Grading>
<diggs_geo:Grading gml:id="g3">
<diggs_geo:particleSize uom="mm">0.5</diggs_geo:particleSize>
<diggs_geo:percentPassing uom "%">10</diggs_geo:percentPassing>
</diggs_geo:Grading>
</diggs_geo:gradingData>
</diggs_geo:ParticleSizeTest>
</procedure>
</Measurement>

```

Another example for a collection of results:

```

<!-- CPT Sounding -->
<Measurement gml:id="cpttest-1">
<projectRef xlink:href="#p1"/>
<associatedLocationRef xlink:href="#cpt-1"/>
<outcome>
<MeasurementResult gml:id="m103">
<position>
<MeasuredDepthArray srsName="#cptsr1" srsDimension="1" gml:id="MP001">
<gml:posList>0.010 0.020 0.030 0.040 0.050 0.060 0.070 0.080 0.090 0.100 0.110
0.120 0.130 0.140 0.150 0.160 0.170 0.180 0.190 0.200 0.210 0.220 0.230 0.240
0.250 0.260 0.270 0.280 0.290 0.300 0.310 0.320 0.330 0.340 0.350 0.360 0.370
0.380 0.390 0.400 0.410 0.420 0.430 0.440 0.450 0.460 0.470 0.480 0.490 0.500
0.510 0.520 0.530 0.540 0.550 0.560 0.570 0.580 0.590 0.600 0.610 0.620 0.630
0.640 0.650 0.660 0.670 0.680 0.690 0.700 0.710 0.720 0.730 0.740 0.750 0.760
0.770 0.780 0.790 0.800 0.810 0.820 0.830 0.840 0.850 0.860 0.870 0.880 0.890
0.900 0.910 0.920 0.930 0.940 0.950 0.960 0.970 0.980 0.990 1.000 1.010 1.020
1.030 1.040 1.050 1.060 1.070 1.080 1.090 1.100 1.110 1.120 1.130 1.140 1.150
</gml:posList>
<MeasuredDepthArray>
</position>
</results>
<ResultSet>
<parameters>
<PropertyParameters gml:id="cptpr-1">
<properties>
<Property gml:id="Ddle267" index="1">
<mnemonic>Qc</mnemonic>
<typeData>double</typeData>
<propertyName codeSpace="urn:x-diggs:def:code-list:property">tip_resistance</propertyName>
< uom>kN/m2</uom>
< nullValue reason="missing">9999</nullValue>
</Property>
<Property gml:id="Dd1e284" index="2">
<mnemonic>Fs</mnemonic>
<typeData>double</typeData>
<propertyName codeSpace="urn:x-diggs:def:code-list:property">sleeve_friction</propertyName>
< uom>kN/m2</uom>
< nullValue reason="missing">9999</nullValue>
</Property>
<Property gml:id="Dd1e301" index="3">
<mnemonic>Friction Ratio</mnemonic>
<typeData>double</typeData>
<propertyName codeSpace="urn:x-diggs:def:code-list:property">friction_ratio</propertyName>
< nullValue reason="missing">9999</nullValue>
</Property>
<Property gml:id="Dd1e318" index="4">
<mnemonic>u1</mnemonic>
<typeData>double</typeData>
<propertyName codeSpace="urn:x-diggs:def:code-list:property">pore_water_pressure</propertyName>

```

```

< uom>kN/m2</ uom>
< nullValue reason="missing">9999</ nullValue>
</ Property>
</ properties>
</ PropertyParameters>
</ parameters>
< dataValues> cs="" ts="" decimal=""> 0.1300,0.40,0.0000,0.0013
0.2400,0.40,0.1000,0.0078 0.5500,0.40,0.0040,0.0126 0.6800,0.40,0.0070,-0.0017
0.7800,0.30,0.0120,-0.0121 0.9000,0.30,0.0150,-0.0161 0.9600,0.40,0.0200,0.0191
1.0400,0.40,0.0240,-0.0120 1.0700,0.30,0.0270,-0.0129 1.1000,0.30,0.1000,-0.0123
1.1300,0.40,0.0350,-0.0176 1.1800,0.30,0.0400,-0.0234 1.2400,0.40,0.0430,-0.0206
1.2600,0.40,0.0460,-0.0277 1.2600,0.40,0.0480,-0.0303 1.2800,0.40,0.0490,-0.0413
1.2900,0.40,0.0500,-0.0482 1.2600,0.30,0.0500,-0.0455 1.2600,0.40,0.0500,-0.0514
1.2633,0.40,0.0500,-0.0486 1.2667,0.40,0.0490,-0.0483 1.2700,0.40,0.0480,-0.0506
1.2600,0.40,0.0470,-0.0539 1.2500,0.40,0.0460,-0.0560 1.2100,0.40,0.0430,-0.0604
1.2000,0.40,0.0400,-0.0604 1.1900,0.40,0.0360,-0.0061 1.1900,0.40,0.0340,-0.0162
1.1300,0.40,0.0320,-0.0359 1.0800,0.40,0.0300,-0.0353 0.9400,0.40,0.0290,-0.0095
0.8700,0.40,0.0280,-0.0075 0.8300,0.40,0.0270,-0.0077 0.8100,0.40,0.0270,-0.0080
0.8300,0.40,0.0270,-0.0075 0.7600,0.40,0.0250,-0.0059 0.6900,0.40,0.0240,-0.0049
0.7000,0.40,0.0230,-0.0044 0.6500,0.40,0.0220,-0.0040 0.6200,0.40,0.1000,-0.0040
0.6000,0.40,0.1000,-0.0043 0.6000,0.40,0.1000,-0.0038 0.5900,0.40,0.1000,-0.0035
0.5800,0.40,0.1000,-0.0032 0.5500,0.40,0.0200,-0.0084 0.5500,0.40,0.0180,-0.0154
0.5400,0.40,0.0190,-0.0187 0.5300,0.40,0.0180,-0.0220 0.5200,0.40,0.0180,-0.0296
0.5300,0.40,0.0180,-0.0073 0.5400,0.40,0.0170,-0.0039 0.5500,0.40,0.0160,-0.0074
0.5500,0.40,0.0150,-0.0129 0.5500,0.40,0.0150,-0.0169 0.5700,0.40,0.0150,-0.0174
0.5800,0.40,0.0160,-0.0141 0.6000,0.40,0.0170,-0.1000 0.6300,0.40,0.0190,-0.0027
0.6700,0.40,0.0200,-0.0031 0.6800,0.40,0.1000,-0.0028 0.6900,0.40,0.0220,-0.0040
0.7100,0.40,0.0230,-0.0070 0.6900,0.40,0.0230,-0.0095 0.6900,0.40,0.0240,-0.0077
0.6867,0.40,0.0240,-0.0074 0.6833,0.40,0.0250,-0.0137 0.6800,0.40,0.0250,-0.0120
0.6900,0.40,0.0250,-0.0123 0.7000,0.40,0.0240,-0.0116 0.6900,0.40,0.0240,-0.0107
0.7300,0.40,0.0220,-0.0117 0.7300,0.40,0.1000,-0.0167 0.7300,0.40,0.1000,-0.0203
0.7400,0.40,0.0200,-0.0228 0.7200,0.40,0.0200,-0.0222 0.7100,0.40,0.0190,-0.0222
0.7100,0.40,0.0190,-0.0200 0.7000,0.40,0.0180,-0.0213 0.6900,0.40,0.0170,-0.0202</ dataValues>
</ ResultSet>
</ results>
</ MeasurementResult>
</ outcome>
</ procedure>
< diggs_geo:StaticConePenetrationTest gml:id="d1e242" >
< equipmentRef xlink:href="cone-1">
< diggs_geo:distanceTipToSleeve uom="cm">15</ diggs_geo:distanceTipToSleeve>
< diggs_geo:penetrometerType>piezocene</ diggs_geo:penetrometerType>
< diggs_geo:tipArea uom="cm2">15</ diggs_geo:tipArea>
</ diggs_geo:StaticConePenetrationTest>
</ procedure>
</ Measurement>

```

Capturing monitoring data in this structure creates issues. In GML the Domain Set is a spatial reference only. However, the nature of monitoring data is that it's primarily in a time based domain.

Recommendation at this time – Create a new element in the Measurement object that defines a time-based domain. This new element would reside in the Outcome element, as follows:

- Measurement
 - Outcome
 - Position (*GML Domain Set*)
 - **Time (new DIGGS Time Domain Set)**
 - Results (*GML Range Set*)
 - Procedure

This extension of the GML coverage structure would allow capture of time-based monitoring data within the same data structure. This effectively adds a 3rd dimension to the results array. Examples (in very simplified format) are provided to illustrate concepts.

Notes:

- Position domain would be mandatory. We'll always be able to define position value at a location feature.
- Time domain can be optional.
- Use schematron that checks that the time domain has either 1 set of values (used for each position domain), or one per each depth, but nothing in between.
- The iteration through the result set (mapping the results to the appropriate position and time), by convention, iterates through all of the time values before incrementing to the next position value.

Example 1 – Borehole with measurements at 3 depths, each with 3 results, and results reported multiple times at different time intervals.

```

Position (GML Domain Set):
2, 3, 4

Time (new DIGGS Time Domain Set):
0, 1, 2
0.0, 1.1, 2.2, 3.3, 4.4
0.4, 1.4, 2.4

Results (GML Range Set):
100, 200, 30 <---- this corresponds to depth=2, time=0
101, 201, 30 <---- this corresponds to depth=2, time=1
102, 202, 30 <---- this corresponds to depth=2, time=2

100, 200, 30 <---- this corresponds to depth=3, time=0.0
101, 201, 30 <---- this corresponds to depth=3, time=1.1
102, 202, 30 <---- this corresponds to depth=3, time=2.2
102, 202, 30 <---- this corresponds to depth=3, time=3.3
102, 202, 30 <---- this corresponds to depth=3, time=4.4

100, 200, 30 <---- this corresponds to depth=4, time=0.4
101, 201, 30 <---- this corresponds to depth=4, time=1.4
102, 202, 30 <---- this corresponds to depth=4, time=2.4

```

Example 2 – Borehole with measurements at 3 depths, each with 2 measurements, and measured 3 times at consistent time intervals.

```

Position (GML Domain Set):
2, 3, 4

Time (new DIGGS Time Domain Set):
0, 1, 2

Results (GML Range Set):
100, 200 <---- this corresponds to depth=2, time=0
101, 201 <---- this corresponds to depth=2, time=1
102, 202 <---- this corresponds to depth=2, time=2

100, 200 <---- this corresponds to depth=3, time=0
101, 201 <---- this corresponds to depth=3, time=1
102, 202 <---- this corresponds to depth=3, time=2

```

```

100, 200  <---- this corresponds to depth=4, time=0
101, 201  <---- this corresponds to depth=4, time=1
102, 202  <---- this corresponds to depth=4, time=2

```

Example 3 – Borehole with measurements at 3 depths, each with 2 results, and results only reported once (no multiple time-based results).

Position (GML Domain Set):
 $\underline{2, 3, 4}$

Time (new DIGGS Time Domain Set):
 $\underline{0}$

Results (GML Range Set):
100, 200 <---- this corresponds to depth=2, time=0
100, 200 <---- this corresponds to depth=3, time=0
100, 200 <---- this corresponds to depth=4, time=0

Example 4 – Borehole with measurements at 1 depth, 2 results, and results reported 3 times at consistent intervals.

Position (GML Domain Set):
 $\underline{2}$

Time (new DIGGS Time Domain Set):
 $\underline{0, 1, 2}$

Results (GML Range Set):
100, 200 <---- this corresponds to depth=2, time=0
101, 201 <---- this corresponds to depth=2, time=1
102, 202 <---- this corresponds to depth=2, time=2

E.52 Teleconference Meeting Notes 2012-01-24

Date: January 24, 2012

Time: 9:00 AM – 10:00 AM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Agenda: Project Team Status and Planning

Discussion:

- Project planning issues:
 - Need to come up with punchlist to wrap up 1.2.
 - Release 1.2 by late February 2012.

- Galdos begin prep on documentation.
- Technical issues – Burggraf had emailed comments in December in response to meeting notes from Turner/Ponti about time-domain monitoring structure. Four suggestions offered by Burggraf:
 - (1) Change DepthInterval to be a GML geometry, e.g. MultiPoint, that way all GML-aware software will recognize it as such and be able to process it accordingly. If it is a custom geometry, then some GML-aware software will recognize it (e.g. Galdos), but not too many.

```

<MeasurementResult gml:id="m102">
  <position>
    <DepthInterval gml:id="gm124">
      <gml:posList srsName="#sr123" srsDimension="1">5
    </gml:posList>
    </DepthInterval>
  </position>
  <results>
    ...
  
```

Perhaps the finer-grained semantics can be put into the property name instead of the Object name? Here's an example:

```

<MeasurementResult gml:id="m102">
  <depthInterval>
    <g3.3:MultiPoint gml:id="gm124">
      <gml:posList srsName="#sr123" srsDimension="1">5
    </g3.3:MultiPoint>
  </depthInterval>
  <results>
    ...
  
```

- (2) Same comment as above for the other custom geometries (e.g. MeasuredDepthArray, ...)
- (3) <typeData> looks like it should be code-list valued (e.g. the standard XSD data types could be allowed values in the code-list)

```

<PropertyParameters gml:id="psp1">
  <properties>
    <Property index="1" gml:id="psp2">
      <mnemonic>Percent Fines</mnemonic>
      <typeData>double</typeData>
      <propertyName codeSpace="urn:x-diggs:def:code-
        list:DIGGS:property">percent_fines</propertyName>
    
```

```
<uom>%</uom>
</Property>
```

- (4) For the encoding of position + time domains, I would recommend a separate <posList> for the separate time intervals. I.e. for the following example:

```
Position (GML Domain Set) :
2, 3, 4

Time (new DIGGS Time Domain Set) :
0, 1, 2
0.0, 1.1, 2.2, 3.3, 4.4
0.4, 1.4, 2.4
```

Encode it something like this:

```
<position>
  <g3.3:SimpleMultiPoint srsDimension="1">
    <gml:posList>2 3 4</gml:posList>
  </g3.3:SimpleMultiPoint>
</position>
<time>
  <TimeDomain>
    <timePositionList>2 3 4</timePositionList>
    <timePositionList>0.0 1.1 2.2 3.3 4.4</timePositionList>
    <timePositionList>0.4 1.4 2.4</timePositionList>
  </TimeDomain>
</time>
```

- Priorities for technical schema work for next couple of weeks:
 - **First priority** is to come to consensus on the time-domain structure suggested in Burggraf's item (4).
 - **ACTION:** Turner/Ponti/Burggraf will have conference call next Tuesday to decide on structure and move ahead.
 - **Second priority** is to reconsider the position element (DepthInterval) and how it's currently encoded in DIGGS as suggested in Burggraf's items (1) & (2).
 - **ACTION:** Need to decide if the change suggested by Burggraf can/should be implemented at this time. Discuss on call next Tuesday.
 - **Third priority** -- consider Burggraf's item (3).
- **ACTION:** Turner/Ponti meet next week to review status of codelists.

E.53 Teleconference Meeting Notes 2012-02-03

Date: February 3, 2012

Time: 1:00 PM – 4:00 PM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Agenda: Project Team Status and Planning

Discussion:

- Documentation to be prepared by Burggraf.
 - **Turner** will send Burggraf prior documentation from 1.0a phase work for reference.
 - Need primer on how to deal with various features and relationships.
 - **Ponti** will ask Bray if interested to contribute.
- Change this version to “2.0a” (version 2 alpha) due to major changes.
- Remaining work:
 - **GML 3.3 Changes.**
 - Not sure if latest version has all the draft GML 3.3 changes.
 - **Burggraf** will send all draft GML 3.3 files again.
 - **Ponti** will check if it validates with the new DIGGS 2.0a to confirm.
 - We know that some new changes are needed to make DIGGS fully GML 3.3 compliant. However, 3.3 is still under public review and could result in other changes that impact DIGGS. For now, we won’t try to make further changes to anticipate what the final GML 3.3 will contain, and we will proceed with our current understanding of GML 3.3.
 - **Update 20 Examples.**
 - **Update Schema Documentation.**
 - **Monitoring Structure.** We had suggested back in November that a hybrid position/time domain structure be created. However, this requires making changes to GML coverage structures that we hadn’t considered at that time. Path forward – retain the current GML coverage for position based coverages, as we described in the “Example 3” from earlier notes (with the time part removed).

Example 3 – Borehole with measurements at 3 depths, each with 2 results, and results only reported once (no multiple time-based results).

Position (GML Domain Set):
2, 3, 4

Time (new DIGGS Time Domain Set):
θ

Results (GML Range Set):
100, 200 <---- this corresponds to depth=2, time=0
100, 200 <---- this corresponds to depth=3, time=0
100, 200 <---- this corresponds to depth=4, time=0

For monitoring data, we'll create a new time-domain based structure, as we described in "Example 4" (with the position part removed).

Example 4 – Borehole with measurements at 1 depth, 2 results, and results reported 3 times at consistent intervals.

```

Position (GML Domain Set):
2

Time (new DIGGS Time Domain Set):
0, 1, 2

Results (GML Range Set):
100, 200    <---- this corresponds to depth=2, time=0
101, 201    <---- this corresponds to depth=2, time=1
102, 202    <---- this corresponds to depth=2, time=2

```

Burggraf will create a new time-based structure in the DIGGS domain and send out for review.

- Still need to address this issue from prior meeting: Change DepthInterval to be a GML geometry, e.g. MultiPoint, that way all GML-aware software will recognize it as such and be able to process it accordingly. If it is a custom geometry, then some GML-aware software will recognize it (e.g. Galdos), but not too many.

```

<MeasurementResult gml:id="m102">
  <position>
    <DepthInterval gml:id="gm124">
      <gml:posList srsName="#sr123" srsDimension="1">5
    </gml:posList>
    </DepthInterval>
  </position>
  <results>
    ...
  
```

Perhaps the finer-grained semantics can be put into the property name instead of the Object name? Here's an example:

```

<MeasurementResult gml:id="m102">
  <depthInterval>
    <g3.3:MultiPoint gml:id="gm124">
      <gml:posList srsName="#sr123" srsDimension="1">5
    </gml:posList>
    </g3.3:MultiPoint>
  </depthInterval>
  <results>
    ...
  
```

- Bring all WITSML measure types, DIGGS measure types and bring all into DIGGS namespace. This eliminates WITSML from DIGGS.
 - **Burggraf** will carry out this work.
 - Follow GML-like naming conventions.
 - Identify orphans.
- Wireline result parameters not included in Turner's prior spreadsheet. **Ponti** to send instructions to Burggraf on what we need. **Turner** will set up meeting with Ponti for more in-depth discussion.
- May be able to eliminate Monitoring and Environmental schemas. Need to discuss with Environmental SIGs. **Ponti/Turner** to identify who needs to be in discussion.

E.54 Teleconference Meeting Notes 2012-02-10

Date: February 10, 2012

Time: 1:00 PM – 3:00 PM (PST)

Participants: Loren Turner
Dan Ponti
David Burggraf

Agenda: Project Team Status and Planning

Discussion:

- Status of draft Final Report.
- Changes Ponti made between 1.2.4j to 1.2.4k (2.0a). From Dan's 2/10/12 email:
 - 1) *Changed name of Location and all of its siblings to SamplingFeature. This normalizes it more with O&M and lets us use the term location to mean what it normally means in the English language. This mostly only affects abstract elements and base types - there some but not a lot of impacts to the instances.*
 - 2) *Removed monitoringInstallations as a top level property; replaced with the Monitoring structure, which derives from AbstractMeasurement*
 - 3) *Added investigationTargets as a top-level property and built AbstractInvestigationTarget base type for these targets (equivalent to OM feature-of-interest). The base type right now is the same as diggs:AbstractFeature. This will be the category of features that are the "targets" for DIGGS data - eg. ground investigations, embankments, piles, roadways, etc.*
 - 4) *Created an element and associated base types for Ground, which extends AbstractInvestigationTarget (but adds no additional properties). It is currently the only concrete investigation target in this version of DIGGS.*
 - 5) *Added mandatory investigationTargetRef be identified for every sampling feature, measurement (eg. test and monitoring), and sampling activity - DIGGS features where the locationRef is not mandatory.*

- 6) Added Well as a new concrete SamplingFeature.
- 7) Created two additional sampling features: Transect and Face, to represent a generic linear and planar sampling feature (such as a linear transect or outcrop). These use the abstract type without extensions.
- 8) Removed AbstractLocation and type - not needed - diggs:AbstractGeometry contains all needed elements;
- 9) Deleted LayerIntervalType - not needed any longer.
- 10) Removed ValueAtTime, ValueAtDepthbyTime and ValueAtType elements and properties - not needed.
- 11) Eliminated most comments except for CDATA and DSB profile restrictions.
- 12) Changed a number of gml: geometries to their diggs-derived equivalents. Did not do this for referenceEdge, featureExtent, and relativeFeatureBoundary of AbstractPlanarSamplingFeature. Probably should, but want to consult with David first.
- 13) Reduced the number of diggs geometry features to eliminate redundancy (eg. DepthInterval and CurveLocation merged into a single LinearExtent feature)
- 14) Abstracted Measurement and defined a base type (AbstractMeasurement) that holds all of the required and optional references that a measurement might have. Renamed Measurement to Test to be used for measurements that occur at a single time instant or within an interval of time but where time is not an integral part of the result (eg. the position indexed results). Created a Monitoring feature that would be used for measurements taken at a non-varying position or location, but where the results are time-indexed. This covers the end members for measurements.
- 15) For Monitoring, although it parallels the Test structure, there is no real procedure. I replaced the procedure property with a process property that contains the diggs metadata property group to allow recording of specifications and equipment. However, there is some redundancy here; as the Property object of a measurement also contains a property called detectorRef that can reference equipment used to record the specific property. So, should we limit procedure to only reference or describe a specification?
- 16) Created ChemicalAnalysis in the environmental namespace as a test procedure that can store info for chemical tests (such as sample volume, type, etc.). and also modified SpectralAnalysis to extend ChemicalAnalysis. We need to discuss this with the environmental SIG and see if there shouldn't be some other concrete procedures for these chemical tests.
- 17) Removed WaterLevelReading and associated types in Environmental and converted the properties here to result properties, or added to the Property object to be used within the Monitoring structure. There are a couple of properties that either seemed redundant or I didn't understand that I did not transfer. We need to talk with folks in Environmental to decide what to do with these:
- 18) captureQualifier - no clue what this means,
- 19) type property for the reading - unclear what this is supposed to represent - if it is method specific this is handled in the procedure.
- 20) c)
- 21) There are some tradeoffs here where some types of recorded info for fluid levels may result in repetition of values in the data block (eg. like fluid_type) and by measurement-specific reporting constraints. This can be avoided somewhat by expanding the result property definitions we support, but this is likely to get unwieldy. We can discuss this a bit so I can fully explain.
- 22) There's still some things in Environmental I don't yet know what to do with - eg Filtration, Purge, TICResult. Filtration sits on its own, and probably should be melded into the samplePreparation structure somehow. Purge is a procedure that might work within a monitoring measurement. How is Purge different from the Pumping test in the Geotechnical schema? TICResult (I think this is total organic carbon), should go into the set of property classes for measurement results, but maybe there should be another test procedure for this? Again, I think we need to consult with the Environmental SIG.
- 23) Commented out a lot of code in Monitoring and Environmental that is no longer needed. I left water level measurements in Environmental for now, but this would probably again morph into a monitoring measurement with the procedure or detector being a water level test or something like that.
- 24) Fixed a few odds and ends to make sure everything validated.
- 25) Changed the testInstance to validate against the new changes.
- 26) Eliminated the Monitoring schema - with the monitoring feature in kernel, it is no longer needed.

- GML 3.3 changes – **Ponti** will send the new 2.0a to **Burggraf** to check.
- Monitoring structure.
 - Ponti created a draft structure that separates the primarily position-based measurements from primarily time-based measurements.
 - *Test* – measurement using a GML position coverage.
 - Uses the GML coverage model as set up before.
 - *Monitoring* – measurement using a time coverage.
 - Encoding/declaring the times is not clear.
 - Actions:
 - **Burggraf** will review, modify the new structure, and test for GML compliance.
- Path forward with WITSML units schemas.
 - Hold off until later time.
 - Lower priority relative to other remaining issues.
- Regroup elements in the Kernel.
 - Group like elements together.
 - Abstract and base types
 - Support types
 - Alphabetize within sections
 - Use comments in the schema text file.
 - **Burggraf** will do this.
- Check dependencies and eliminate orphaned elements and types.
 - Are there any types not being used?
 - **Burggraf** will run GML SDK and another script tool to identify orphan elements. Will send to **Ponti** for review.
- Consistency in DIGGS types – if defined in DIGGS namespace, is it used consistently everywhere?
 - **Ponti** will check into how DIGGS string type is used throughout as well as other types.
- Updating the 20 examples – need to do this manually. Likely will require Ponti/Turner to do this.
- WITSML wireline parameters into dictionary.
- Elimination of monitoring and environmental schemas
- DepthInterval issue.