Author 1

First Name (or initial)	Middle Name (or initial)	Surname	Role (ASABE member, etc.)	Email	Contact author? yes or no
Dennis	G.	Daggett		ddaggett@proag.com	no

Affiliation

Organization	Address	Country	Phone for contact author
ProAg Management, LLC	Amarillo, TX	USA	

Author 2

First Name (or initial)	Middle Name (or initial)	Surname	Role (ASABE member, etc.)	Email	Contact author? yes or no
R.	Andres	Ferreyra	Member	andres.ferreyra@agconnections.com	yes

Affiliation

Organization	Address	Country	Phone for contact author
Ag Connections	1576 Killdeer Trl, Murray, KY	USA	

Author 3

First Name (or initial)	Middle Name (or initial)	Surname	Role (ASABE member, etc.)	Email	Contact author? yes or no
Linga	T.	Reddy		ReddyLingaT@JohnDeere.com	no

Affiliation

Organization	Address	Country	Phone for contact author
John Deere ISG	Urbandale, IA	USA	

Author 4

First Name (or initial)	Middle Name (or initial)	Surname	Role (ASABE member, etc.)	Email	Contact author? yes or no
Stuart	T.	Rhea		stuart.rhea@agconnections.com	no

Affiliation

Organization	Address	Country	Phone for contact author
Ag Connections	1576 Killdeer Trl, Murray, KY	USA	

Author 5

First Name (or initial)	Middle Name (or initial)	Surname	Role (ASABE member, etc.)	Email	Contact author? yes or no
Joe	W.	Tevis		jwtevis@vis4ag.com	no

Affiliation

Organization	Address	Country	Phone for contact author
TOPCON	Minneapolis, MN	USA	

Repeat the Author and Affiliation tables above for each additional author.

Put paper number in box below; pages 1-13
2462418

2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

An ASABE Meeting Presentation
DOI: 10.13031/aim.202462418
Paper Number: 2462418

Filling in the blanks with ContextItems: a lightweight method for extending field operations object models

Daggett, Dennis G.[2]; Ferreyra, R. Andres[1]; Reddy, Linga T.[4]; Rhea, Stuart T.[1]; Tevis, Joe W.[3]
[1]Ag Connections, Murray, KY, United States; [2] ProAg Management, LLC, Amarillo, TX, United States;
[3] TOPCON, Minneapolis, MN, United States; [4] John Deere ISG, Urbandale, IA, United States

Written for presentation at the
2016 ASABE Annual International Meeting
Sponsored by ASABE
Orlando, Florida
July 17-20, 2016

ABSTRACT. *Precision agriculture (PA) is still limited by a lack of hardware/software systems interoperability. AgGateway, a nonprofit consortium of 240+ companies, leveraged its wide cross-section of PA stakeholders to propose a collaborative solution: its ADAPT team created an open-source, field operations common object model. The goal: replace current systems' need to support multiple, incompatible data formats, with a single integration to the common object model and a system of manufacturer-specific format-conversion plug-ins. This enables reading/writing to new systems with marginal development cost. The common object model meets requirements from AgGateway's SPADE and PAIL projects, including compatibility with the ISO11783-10 standard (ISOXML) and participant companies' own systems.*

Internationalization is important for this work, but conflicting requirements must be reconciled: ADAPT developers must seek universality, staying free of regionally-specific clutter. However, different geographies' business processes involve context-specific data (e.g., USA EPA product numbers.) If these "context items" are not accommodated, the common object model's relevance suffers. Additionally, it is desirable to use a controlled vocabulary. However, the dynamic nature of business and regulation requires this vocabulary to be easily extensible. ADAPT reconciled the contradictions by defining an object class, the ContextItem (CI), that can be attached to various other objects in the common object model. A ContextItem is a key/value structure where the "key" code references a ContextItemDefinition (CID) that defines what each CI means. The "value" is composed of a string value along with data needed to interpret it (such as a unit of measure) or a nested list of other CIs (e.g. PLSS cadastral information.) AgGateway's SPADE project implemented a RESTful API to provide a machine-readable vocabulary of CIDs; its Standards & Guidelines Committee created an ad-hoc group to manage the vocabulary. The CI system can be used jointly with ISOXML's feature of associating unique IDs to its own locally-scoped IDs (defined in ISO11783-10 Annex E.) This enables adding geopolitical-context-dependent data to ISOXML's otherwise generic and highly machine-specific scope, with no modifications.

Keywords. *codes, information systems, international, ISO, standards.*

Introduction

Precision agriculture (PA) is a promising set of technologies, but is still limited by a lack of hardware and software systems interoperability. Users of precision agriculture spend a lot of time moving data among various proprietary systems and converting among multiple proprietary formats; this limits adoption and the perceived value of PA.

Concurrently, current trends in sustainability, traceability, and compliance reporting demand an ever-increasing amount of data be gathered as part of everyday operations in modern production agriculture. This requirement usually includes significant amounts of frequently-changing geopolitical-context-dependent information such as identification numbers specific to the government agencies the grower interacts with in their jurisdiction. Fulfilling all of these requirements in the data model of farm management information system (FMIS) software is a moving target, unless it were somehow possible to decouple the infrequently- and frequently-changing aspects of the FMIS data model.

In terms of requirements thus placed on a data model, an FMIS object model should simultaneously be:

- generic, simple and compact enough to be easily understood and used, as well as accepted from an international perspective (which would suggest staying free of regionally-specific clutter), but still be able to support the capture & communication of necessary region-specific (i.e., geopolitical-context-dependent) data needed by growers and their partners as part of their business processes (simple/generic vs comprehensive/specific)
- able to express data with a controlled vocabulary (so everyone can understand what it means), but allowing that controlled vocabulary to be continually updated to match the nature of data requirements (static vs dynamic)

AgGateway (www.aggateway.org), a nonprofit consortium of about 240 companies dedicated to the implementation of standards for eAgriculture, created its Precision Agriculture Council in 2010 to collaboratively tackle these interoperability problems. This led to the creation of the ADAPT team, charged with implementing a toolkit to provide the industry with a **common object model** for field operations as well as a set of **format conversion tools** (AgGateway, 2016).

The ADAPT common object model meets requirements from AgGateway's SPADE (planting, crop care, harvest and post-harvest - themed) and PAIL (irrigation, observations and measurements - themed) projects, as well as compatibility with the ISO11783-10 standard XML format (ISO, 2015) and participant companies' own systems.

The relatively-static core object components of the ADAPT framework were defined in a minimalistic way, but the problem remained of how to represent frequently-changing and/or geopolitical-context-dependent data. The team decided to implement this by providing the core objects with placeholders for attaching key /value pair – derived objects called ContextItems.

Key / value pairs are straightforward, and powerful in their simplicity. The approach has the weakness, however, that the key is limited in how much shared meaning it can effectively communicate without compromising the brevity that is a valuable feature of the approach. The solution chosen by the ADAPT Team was to create a controlled vocabulary of keys that would allow for the detailed description of the ContextItems, albeit *kept separate from the usage of the ContextItems themselves* (but available to all of the parties exchanging data).

AgGateway's SPADE3 project (AgGateway, 2015) implemented a RESTful API to provide a machine-readable vocabulary of ContextItem definitions. Furthermore, AgGateway's Standards & Guidelines Committee created an ad-hoc group to manage the vocabulary.

The same approach can be used to graft additional, semantically-rich data onto objects from other data models such as the elements defined in ISO 11783-10. The ContextItem system can be used jointly with ISOXML's feature of associating unique identifiers to its own locally-scoped identifiers defined in ISO11783-10 Annex E (ISO, 2015). This enables adding geopolitical-context-dependent data to ISOXML's otherwise generic and highly machine-specific scope, with no modifications.

The goal of this paper is to present the ContextItem system as a simultaneous solution for satisfying the aforementioned conflicting sets of requirements: simple/generic vs comprehensive/specific, and static vs dynamic. We will first present some basic ideas regarding how identity is implemented in the ADAPT data model, followed by a general description of the ContextItem system data model, and specific description of the properties (or attributes, we will use the terms interchangeably) of ContextItems and their definitions. We will then present examples of the different types of ContextItem definitions, and discuss implications and future direction of the work.

Basic ADAPT model concept: Identity and the CompoundIdentifier

Many objects specified in the ADAPT common object model in general, and in its subset model the ContextItem system in particular, are used *by reference* in other objects (for example, a grower, farm and field may be referenced in a work order) and thus need identifiers that can be used by the referencing object. Figure 1 shows a Unified Modeling Language (UML) class diagram (ISO/IEC, 2005) of the mechanism used by ADAPT to do this. It centers on an object class called CompoundIdentifier, which serves two purposes:

- First, the CompoundIdentifier allows its parent object to be used by reference from other objects, by providing a simple integer identifier (ReferenceId) for use in the local scope of any particular instance of a data model. In

dealing with especially large datasets, the ability to use objects by reference results in a smaller footprint when persisting to storage or attempting transmission.

- Second, it enables the association of multiple unique identifiers (UniqueId) to an object. This allows for an enhanced exchange of data between different systems because each system can include its internal, unique identifier for an object without compromising any of the others.

A historic pain point in data exchange has been that systems vary in the way chosen to construct unique identifiers. For example, Company A might use integers as their unique (albeit only inside their system) identifier; Company B may use UUIDs (globally unique). The two systems are fundamentally incompatible, but each may be appropriate for the corresponding company's processes. It is for this reason that the UniqueId class uses a combination of string (Id) and enumeration (UniqueIdTypeEnum) to fully describe the unique identifier: it enables the support of a broad variety of identification methods.

Each UniqueId can thus be of four different types (The UniqueId itself is stored as a string, but the type of UniqueId is specified with the CIType attribute.):

- A Universally Unique Identifier, or UUID (Leach et al., 2005).
- An arbitrary string (meant to accommodate proprietary alphanumeric identifiers)
- A long integer (meant to accommodate proprietary integer identifiers)
- A uniform resource identifier, or URI (W3C/IETF, 2001).

CompoundIdentifiers are meant to be used in a distributed context, where a document may circulate among two or more FMIS. Since each FMIS may have its own set of unique identifiers (for chemical or seed products, for example), UniqueIds can specify their originating organization / issuing authority by populating the Source attribute with either a Global Location Number / GLN (GS1, n.d.) or a URI. The SourceType attribute specifies the type of source identifier being used.

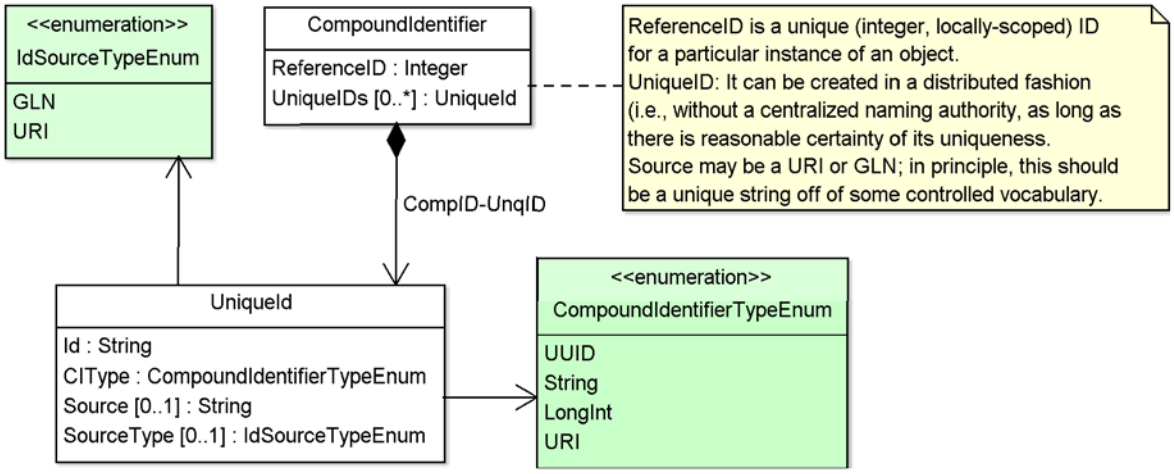

Figure 1: UML Class diagram for CompoundIdentifier and associated classes.

ContextItem System Data Model

Figure 2 shows a simplified (in that the classes' attributes are not shown) data model of the ContextItem system, emphasizing the relationships among the classes. (The CompoundIdentifier and associated classes shown in Figure 1 are not shown, for clarity.)

Table 1 provides a brief description of the purpose of each class, including whether it is a simple enumeration, whether it is used by value or reference, and how it relates to others.

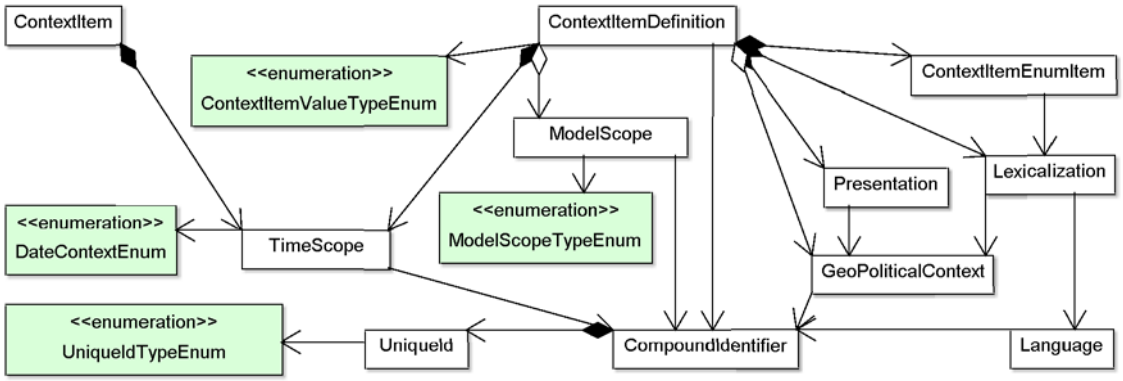

Figure 2: Simple class diagram showing the ContextItem system classes and their relationships. Solid diamonds represent lists assembled with composition relationships (i.e., the child objects are directly inserted by value into the parent) whereas hollow diamonds represent lists assembled through aggregation (the child objects "stand alone" and are referenced using the ReferenceId of their CompoundIdentifier. Arrows with no diamonds represent relationships with single instances of objects, typically by composition.

Table 1: ContextItem system class descriptions

Class	Type	Description
ContextItem	Object	A key / value pair used to attach geopolitical-context-dependent information to other objects in ADAPT, ISO11783 files, etc.
ContextItemDefinition	Object	A definition of a specific kind of ContextItem, including the type of data it can contain, what classes in an object model it can be attached to, and how to display/enter instances of it.
ContextItemValueTypeEnum	Enumeration	An enumerated type that describes the type of value carried by the ContextItem (Integer, Boolean, Double-precision floating point, string, enumeration, datetime, nested..)
ContextItemEnumItem	Object	Description of specific enumeration items for ContextItemDefinitions with an enumerated ContextItemValueType
GeoPoliticalContext	Object	Describes a particular jurisdiction or geopolitical context that a ContextItemDefinition, Lexicalization or Presentation is to be used in the context of.
Language	Object	Describes a language (e.g., "en-us" for US English, or "pt-br" for Brazilian Portuguese) used to express terms in Lexicalizations.
Lexicalization	Object	Represents a way to express something in a given combination of GeoPoliticalContext and Language.
ModelScope	Object	Used by ContextItemDefinition to denote a data model class that a corresponding ContextItem instance can be attached to.
ModelScopeTypeEnum	Enumeration	Describes what data model a particular class belongs to. The system currently supports ADAPT and ISO11783 data models.
Presentation	Object	Describes how to enter and display a ContextItem instance corresponding to a given ContextItemDefinition.
CompoundIdentifier	Object	Provides a mechanism to reference an object from other objects, as well as a mechanism of associating one or more external unique identifiers to an object.
UniqueId	Object	Captures a unique identifier as part of a CompoundIdentifier.
UniqueIdTypeEnum	Enumeration	An enumerated type that describes what kind of unique identifier is contained in a UniqueId (Integer, String, URI, UUID).
TimeScope	Object	Associates a date/time or range thereof to an object. Also encapsulates the meaning of the timestamp / time interval.
DateContextEnum	Enumeration	Specifies the meaning of a TimeScope.

Properties of ContextItem and ContextItemDefinition

Figure 3 shows the full data ContextItem system model (except for CompoundIdentifier, shown in fig. 1). Specifics about the attributes / properties of the ContextItem and ContextItemDefinition classes follow.

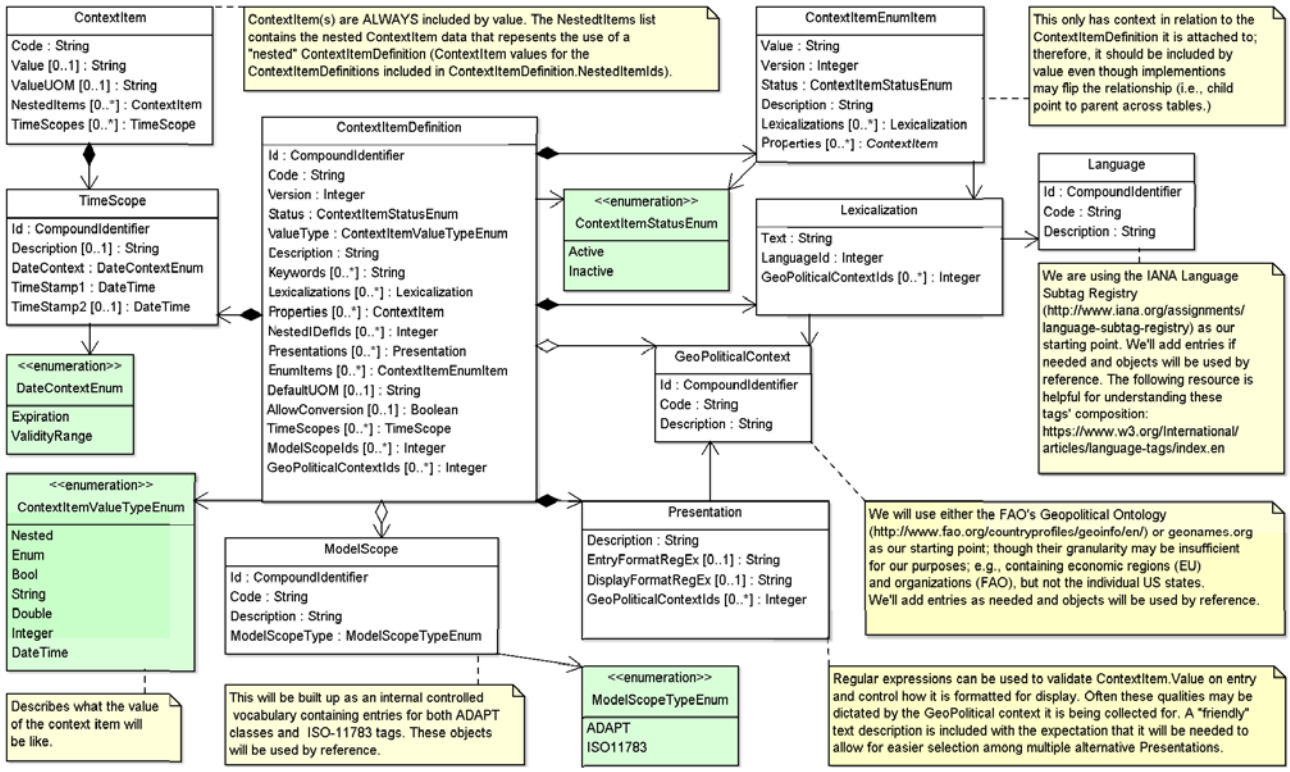

Figure 3: The ContextItem system data model. CompoundIdentifier (see Figure 1) is not shown, for clarity.

ContextItem

A ContextItem is a single, specific, use of a ContextItemDefinition. Put a simpler way, a ContextItem is a key / value pair. Its Code property is the "key" (corresponding to the unique Code property of a single ContextItemDefinition), while the rest of its properties describe the "value".

- The Value property is ALWAYS expressed as a string even though that may not be how it was collected or how it is expected to be used elsewhere. The ValueType property of the associated ContextItemDefinition supplies the user with this data type information. In some cases, like the presence of NestedItems, there is no real Value to record so this property is optional.
- The ValueUOM property contains the optional unit of measure, further defining the Value. A DefaultUOM property is included with the associated ContextItemDefinition and may be used instead of including it locally with the ContextItem. In an effort to foster the broadest appeal, UN Rec 20 codes will be the default vocabulary for expressing units of measure.
- If the ContextItemDefinition has a ValueType of "Nested", the NestedItems collection allows a resulting ContextItem to support a hierarchical structure of other instances of ContextItem. See the below discussion about the NestedItemIds property of ContextItemDefinition for further information.
- Notice the inclusion of an optional collection of TimeScope objects. This is used to record the various relationships a given Value may have with time. For example, there could be a TimeScope that captures when the Value was recorded and another TimeScope that expresses the duration for which the Value is considered valid.

ContextItemDefinition: Basic Properties

The ContextItemDefinition defines a kind of ContextItem: what it means, how to enter it, display it, where it can be used, and so forth. It should communicate everything needed to enable that ContextItem to be captured and displayed appropriately. What follows is a description of the properties of the ContextItemDefinition object, and some of the rationale behind its design.

- The **Code** property
    - is expected to be universally unique within the ContextItemDefinition domain,
    - is used as part of the URI that forms the identity of a ContextItemDefinition,
    - is the same "key" used by **ContextItem** in its Code property, and
    - is issued by a central authority to ensure its uniqueness.
- **Version:** Enables change detection. Whenever a member property of ContextItemDefinition is changed, the Version number is incremented. This allows users who have a cached version of a ContextItemDefinition to realize that a change has taken place. It does NOT communicate what change was made, by whom, or why it occurred; only that it has happened. This approach was deemed more reliable than trying to use a "modified date" time stamp. All ContextItemDefinition(s) start with a Version value of 0.
- **Status:** Selected from an enumeration of two values: "Active" or "Inactive". There will be situations where a ContextItemDefinition or one of its ContextItemEnumItems will need to be "retired" from use. However, those ContextItemDefinition(s) may still be needed in order to interpret historical data. As a result we choose not to delete things, but to instead mark them as "Inactive". It will be the responsibility of the user to avoid the use of "Inactive" objects in the generation of new data.
- **ValueType**: Describes the expected data type for the "value" of a ContextItem. It is expressed as an enumeration with the values of Bool, String, Double, Integer, DateTime, Enum, and Nested.
    - The first five data types (Bool, String, Double, Integer, DateTime) can be easily represented as a string and then parsed back to their original form. This is the reason why the Value property of ContextItem is a string.
    - The "Enum" data type indicates that this ContextItemDefinition is an encoded enumerated list. The items in the list are ContextItemEnumItem objects and are included by value in the EnumItems collection property. When creating a ContextItem using an "Enum" type ContextItemDefinition, the ContextItem Value property corresponds to the Value property of the selected ContextItemEnumItem.
    - The "Nested" data type indicates that this ContextItemDefinition is a container that encapsulates a group of other ContextItemDefinition(s). When creating a ContextItem using a "Nested" type ContextItemDefinition, the ContextItem Value property is left empty but its NestedItems property contains a collection of ContextItem(s).
- **Description**: A human-readable name that makes it easier to correctly choose the ContextItemDefinition of interest from a pick list. It is not meant to be a lengthy explanation.

ContextItemDefinitions: Advanced Properties

At this point we have discussed all the properties that are required to define a simple ContextItemDefinition. This minimum set of information, while sufficient to allow data capture, does little to convey any real meaning. The goal of the remaining properties is to enable a fuller, semantically-rich description of the target concept and where it can be used.

- **Keywords**: A collection of strings intended to be an aid for querying. They are expected to be single words.
- **Lexicalizations**: A collection of Lexicalization objects. The concept that a ContextItemDefinition represents may be expressed verbally in a number of different ways; not just in different languages, but also using different terms in the same language. For example, regional differences in the common name of a pest. A Lexicalization object contains the text, a reference to the language the text is in, and a list of GeoPoliticalContext objects that describe "where" that terminology is used. We have chosen to use the Internet Assigned Numbers Authority (IANA) Language Subtag Registry (IANA, n.d.) as the controlled vocabulary for languages.
- **Properties**: A collection of ContextItem objects. This is how we can supply additional information needed to use the definition correctly. For example, if a ContextItemDefinition involves capturing values for latitude & longitude, it might include in the Properties a ContextItem that indicates the geodetic datum is expected to be WGS84.
- **NestedDefIds**: A collection of references to other ContextItemDefinition(s). This is only populated if ValueType is set to "Nested". Sometimes there are groups of data points that need to be collected together rather than individually. For example, the Public Land Survey System used in the United States uses several attributes (such as the Principal Meridian, Township, Range, and Section) to specify the location of a piece of land for cadastral purposes. Each attribute is defined through its own ContextItemDefinition, and are represented as ContextItem(s) that are included by value in the NestedItems property of the PLSS ContextItem. The examples shown below build up to a PLSS ContextItemDefinition.
- **Presentations**: a collection of Presentation objects. One of the challenges in collecting quality data is being able to make sure that it is entered properly. The Presentation object contains a "friendly" name description, regular expressions (Kleene, 1956) that define how the value is supposed to look when entered or displayed, and a list of GeoPoliticalContext objects that describe "where" this presentation is used. A regular expression is a sequence of one or more characters, alone or in groups, which can be used to describe an expected pattern. It is used to test a

given string to see if it follows the pattern.

- **EnumItems**: a collection of ContextItemEnumItem objects. This is how we encode the enumerated values for a ContextItemDefinition of ValueType "Enum". The ContextItemEnumItem contains some of the same properties that ContextItemDefinition does. Instead of having a Code property, it has a Value. This Value is expected to be unique within the domain of the ContextItemDefinition it is attached to. When creating a ContextItem using an "Enum" type ContextItemDefinition, the ContextItem Value property corresponds to the Value property of the selected ContextItemEnumItem.
- **DefaultUOM**: An optional string describing the expected unit of measure. In an effort to foster the broadest appeal, UN Rec 20 codes (UN CEFACT, 2005) will be the default vocabulary for expressing units of measure.
- **AllowConversion**: an optional Boolean value that is used in conjunction with the DefaultUOM. This serves as a flag to determine if the user can allow the value to be entered in a unit compatible with DefaultUOM, or are they required to use the default unit.
- **TimeScopes**: an optional collection of TimeScope objects. This collection enables attaching a variety of time-related attributes to the ContextItemDefinition. For example, a TimeScope could describe when its ContextItemDefinition was created, another when it was updated, and a third the date range it is valid for.
- **ModelScopeIds**: A collection of references to ModelScope objects. The ModelScope object represents a business object in either the ADAPT model or ISO11783 (potentially other models as well). For example, there is a ModelScope object for the ADAPT Farm class as well as a separate one for the ISO11783 Farm element (FRM). This coded list of business objects forms the controlled vocabulary we use to specify which objects a given ContextItemDefinition can be used to describe. Throughout the ContextItem system we have tried to reuse existing controlled vocabularies where ever we could; the language and geopolitical vocabularies used both point to external sources. In this case, however, we were forced to create our own.
- **GeoPoliticalContextIds**: A collection of references to GeoPoliticalContext objects. The GeoPoliticalContext object represents an entry in an external controlled vocabulary that describes a particular geographic/political domain or organization. This allows us to tag ContextItemDefinition(s) with a marker that conveys "where" the data it represents is relevant.

Examples

Simple integer and string data entry ContextItemDefinitions

Figure 4 shows the relevant properties / attributes of an example ContextItemDefinition representing a USDA Farm Service Agency (FSA) Farm Number, an identifier very frequently used by growers in the United States. Comments:

- It has an (arbitrary, simply meant to be unique within the set of ContextItemDefinitions) Id.ReferenceId.
- It also has a Code, shown as 101, which represents the FSA Farm Number.
- The contents of an embedded (i.e., used by value) Presentation object are shown. Note the regular expression (Kleene, 1956) used to convey information about how to enter/display the FSA Farm Number value.
- There is a list of keywords, shown in curly braces.
- ModelScopeIds is really a list of integer identifiers; the figure shows them dereferenced into ADAPT object model class names, in brackets.
- Similarly, the GPCIds list shows a dereferenced identifier for the United States. An FMIS in Belgium, for example, would know that the FSA Farm Number is not relevant to their geopolitical context.

Attribute	Value / Comment
Id	ReferenceId = 5 (Could be any value)
Code	101
Version	0
Status	Active
ValueType	Integer
Description	FSA Farm Number
Keywords	{USDA, FSA, farm}
Presentations	Description = "Minimum 1, maximum 7 digits." EntryFormatRegEx = "[0-9]{1,7}"
AllowConversion	FALSE
ModelScopeIds	{[Farm], [Field], [Cropzone]}
GPCIds	{[USA]}

Figure 4: FSA Farm Number, an example of a simple integer-valued ContextItemDefinition.

Numeric ContextItemDefinitions for auxiliary purposes (e.g., ContextItemEnumItems' Properties)

Figure 5 shows the relevant attributes of an example ContextItemDefinition representing Latitude and Longitude values. The purpose of this kind of ContextItemDefinition is to provide the infrastructure for ContextItemEnumItems to express properties. A subsequent example will show how the ContextItemDefinition for Public Land Survey System Principal Meridians can express the latitude and longitude of each enumerated item's reference point. This is done through the use of the Latitude and Longitude ContextItems shown. Comments:

- They have (arbitrary, simply meant to be unique within the set of ContextItemDefinitions) Id.ReferenceIds.
- They have unique Codes, shown as 107 and 108, which represents the Latitude and Longitude concepts.
- Presentation objects are not included, because these ContextItem definitions are not meant to be user-entered.

	Latitude	Longitude
Attribute	**Value / Comment**	**Value / Comment**
Id	ReferenceId = 13 (Could be any value)	ReferenceId = 14 (Could be any value)
Code	107 (Represents "Latitude")	108 (Represents "Longitude")
Version	0	0
Status	Active	Active
ValueType	Double	Double
Description	Latitude	Longitude
Keywords	{Geographical coordinates, geodesy}	{Geographical coordinates, geodesy}
DefaultUoM	[degrees]	[degrees]
AllowConversion	TRUE	TRUE
ModelScopeIds	{[Farm], [Field], [Cropzone]}	{[Farm], [Field], [Cropzone]}

Figure 5: Latitude and Longitude, examples of floating-point-valued ContextItemDefinitions.

Enumerated ContextItemDefinition

Figure 6 shows the relevant properties / attributes of an example ContextItemDefinition representing a World Meteorological Organization (WMO) code for cloud type. Comments:

- It has an (arbitrary, simply meant to be unique within the set of ContextItemDefinitions) Id.ReferenceId.
- It also has a Code, shown as 130, which represents the Cloud type / genus concept.
- The ValueType is shown as "enum". That implies that there must be a list of ContextItemenumItems. There is no need for a Presentation object, since the user would be shown the ContextItemEnumItems' Lexicalization strings corresponding to the user's Language and GeoPoliticalContext (or, in their absence, the Description). What would "travel" with the ContextItem, however, would be the ContextItemEnumItems' Value.

ContextItemDefinition

Attribute	Value / Comment
Id	ReferenceId = 20 (Could be any value)
Code	130 (Represents "Cloud type/genus")
Version	0
Status	Active
ValueType	Enum
Description	WMO Cloud type / genus
Keywords	Weather, clouds
EnumItems	(See table at right)
AllowConversion	FALSE

EnumItems

Value	Version	Status	Description
/	0	Active	cloud not visible
0	0	Active	cirrus (CI)
1	0	Active	cirricumulus (CC)
2	0	Active	cirrostratus (CS)
3	0	Active	altocumulus (AC)
4	0	Active	altostratus (AS)
5	0	Active	nimbostratus (NS)
6	0	Active	stratocumulus (SC)
7	0	Active	stratus (ST)
8	0	Active	cumulus (CU)
9	0	Active	cumulonimbus (CB)

Figure 6: WMO Cloud type / genus, an example of an enumerated ContextItemDefinition.

Enumerated ContextItemDefinition with Properties

Figure 7 takes the concept of an enumerated ContextItemDefinition further, with an example that includes Properties on the ContextItemEnumItems. Comments:

- In this case we are only showing three of the ContextItemEnumItems corresponding to the (much longer) list of PLSS Principal Meridians, each with its own attribute table.
- Each of the ContextItemEnumItems has its own, private, list of ContextItems: the Properties!
- Each of the Properties shows a forrmat consistent with the ContextItem shown in the model of Figure 2:
    - The Code values match the Code specified for Latitude and Longitude in Figure 5.
    - A ValueUoM (unit of measure) is specified to remove any ambiguity regarding what the latitude / longitude Values represent.

ContextItemDefinition

Attribute	Value / Comment
Id	ReferenceId = 7 (Could be any value)
Code	103 (Represents "PLSS Principal Meridian")
Version	0
Status	Active
ValueType	Enum
Description	PLSS Principal Meridian
Keywords	{PLSS, cadastre, surveying}
EnumItems	(See column at right)
AllowConversion	FALSE
ModelScopeIds	{[Farm], [Field], [Cropzone]}
GPCIds	[USA]

EnumItems

Attribute	Value / Comment
Value	FirstPrincipal
Version	0
Status	Active
Description	First Principal Meridian
Properties	(See column at right)

Attribute	Value / Comment
Value	SecondPrincipal
Version	0
Status	Active
Description	Second Principal Meridian
Properties	(See column at right)

Attribute	Value / Comment
Value	ThirdPrincipal
Version	0
Status	Active
Description	Third Principal Meridian
Properties	(See column at right)

Properties

Code	Value	ValueUoM
107	40.989444	[degrees]
108	84.803056	[degrees]

Code	Value	ValueUoM
107	38.470556	[degrees]
108	86.455833	[degrees]

Code	Value	ValueUoM
107	38.474167	[degrees]
108	89.148333	[degrees]

Figure 7: PLSS Principal Meridian, an example of an enumerated ContextItemDefinition with Properties

Nested ContextItem

The final example, shown in Figure 8, features a nested ContextItemDefinition: a simplified form of the Public Land Survey System (PLSS) data mentioned earlier. The ContextItemDefinitions referenced by the NestedDefIds list are summarized in a separate table. Note how the Id.ReferenceId is used to link the two tables, and not the **Code**.

ContextItemDefinition

Attribute	Value / Comment
Id	ReferenceId = 6 (Could be any value)
Code	102 (Represents "Public Land Survey System")
Version	0
Status	Active
ValueType	Nested
Description	FSA Farm Number
Keywords	{PLSS, cadastre, land management, surveying}
NestedDefIds	{7, 8, 9, 10} (Should match Id of nested items)
AllowConversion	FALSE
ModelScopeIds	{[Farm], [Field], [Cropzone]}
GPCIds	[USA] (Could also include relevant US states)

Nested ContextItemDefs

Id	Code	Version	Status	ValueType	Description
7	103	0	Active	Enum	PLSS Principal Meridian
8	104	0	Active	String	PLSS Township
9	105	0	Active	String	PLSS Range
10	106	0	Active	String	PLSS Section

Figure 8: PLSS, an example of a nested ContextItemDefinition.

Discussion

Enabling incremental progress

The ContextItem system provides a way to preserve the simplicity of data models that utilize it by allowing these models to first focus on expressing "universal" ideas with their core objects and then enhancing those objects with geopolitical-context-dependent specifics through ContextItem(s). New models (like ADAPT) can thus start simple and grow organically over time. Likewise, existing models (like ISO11783) can be extended in a dynamic, data-driven way. This is all possible because the system provides a powerful test bench for data model improvements: a ContextItemDefinition (conceivably not restricted to geopolitical-context-dependent attributes) can be proposed, tested in real world usage, and subsequently either incorporated into the data model as an object class attribute, kept for use as a ContextItem, or abandoned entirely.

Extensibility is decoupled from data model versions

The ContextItem system allows for the common object model (and other standards) to be extended without requiring the release of a new version of the standard. This is critically important in industrial IT environments, where updating to a new version of a data standard generally represents a huge expenditure of resources in migration, training, security audits, and so forth. Following a data-driven approach enables users to retain the same standard version implementation for a longer period while also allowing progress outside of the traditionally slow standards making process.

The use of external code lists to support data exchange in an industry is not a novel concept: a prime example of their use is the ISO20022 standard used internationally by the financial industry. There is usually a tradeoff in such usage, in that using external code lists makes it harder to validate a particular instance of the data model (because the code list is not built into the model itself.) We believe that the impact of this problem is minimized in the ContextItem system, because all of the ContextItemDefinition(s) (and their associated enumeration items) are readily available through the ContextItem API in a single-format. Thus, implementing the mechanism to validate a ContextItem against all of the (limited number of) possible ContextItem value types enables validating all possible ContextItems. This makes for very efficient and scalable use of the system.

Minimal a priori knowledge needed for use

FMIS developers have heretofore often been forced to hard-code geopolitical-context-dependent attributes (often in rapidly-changing regulatory contexts), and have had to manage multiple geopolitical-context-specific versions of their software. This increases costs, and limits the implementation scalability (and market expansion) of FMIS products.

The ContextItem system should bring welcome relief, because it allows for the collection and communication of data yet does not require the facilitating software to understand what that data means. This has rather revolutionary implications for farm management information systems (FMIS):

When allowing the user to enter data to describe a given object (say, a person, a field, or a document) the FMIS can search the ContextItem API by ModelScope and/or GeoPoliticalContext to find what ContextItemDefinitions are available for the object being entered.

The ContextItem API can then deliver, for each of the available ContextItemDefinitions for that ModelScope, all the data the FMIS needs to present the user with a user interface to populate the ContextItem.

Thus, by virtue of integrating once with the ContextItem system, a FMIS can allow users in multiple geographies to enter data specific to those geographies, without making any changes to the code of the FMIS. Moreover, as the list of ContextItemDefinitions for a given GeoPoliticalContext grows, the FMIS becomes progressively able to enter more and more data pertaining to the local business processes.

A starting point for richer semantics in field operations data exchange

Business-process-specific data exchange among different FMIS is currently very limited by proprietary implementations of geopolitical-context-dependent data. In practice this translates to inter-FMIS data exchange being very infrequent. The ContextItem system is a major step toward building a semantically-rich vocabulary for industry-supported, local-business-process-aware data exchange in production agriculture field operations. The authors hope this will translate into greater electronic communication between growers and their trusted partners, a corresponding greater accuracy and efficiency, and less opportunity for error.

Enabling the use of existing controlled vocabularies

Various communities (research, industry, government) have made a great collective effort over time to develop controlled vocabularies for use in agriculture. Examples include the AGROVOC thesaurus developed by FAO (Caracciolo et al., 2013), the US National Resource Conservation Service's list of management templates (NRCS, n.d.) , and the European and Mediterranean Plant Protection Organization (EPPO) lists of plants, pests and pathogens (EPPO, 2015). An informal survey of industry participants suggested that more widespread adoption of these vocabularies has been limited by the need for ad-hoc implementations in FMIS to enable their use. The "one-size-fits-all' ContextItem system enables the widespread use of controlled vocabularies: if it can be encoded as a ContextitemDefinition, any ContextItem-enabled FMIS can instantly use it.

Encoding proprietary payloads

During the development of the AgGateway Reference Data API system, several manufacturers expressed interest in leveraging their investment in Reference Data API infrastructure to deliver premium content to selected subsets of users of the API (e.g., paying or otherwise special customers.) The ContextItem system is consistent with this idea of enabling premium (and/or proprietary) content delivery.

The proposed syntax for proprietary codes is: `Pr_[GLN]_[Proprietary suffix]`

Example: Pr_1234567890123_T45 would represent a proprietary code created by an organization with a Global Location Number or GLN (GS1, n.d.) of 1234567890123. The meaning of the code, presumed known by the sender and receiver of the data, is represented by the alphanumeric suffix T45. Note that only the initial "Pr_" prefix is required; organizations lacking GLNs, or who choose not to include them into the code, can use any arbitrary alphanumeric syntax following the initial underscore ("_") character.

Future Development

There is a process under development for anyone to submit new ContextItemDefinition(s) through AgGateway's Standards and Guidelines committee. The expected publication date of the process is late 2016.

ContextItemDefinition(s) are made available through a RESTful web service (The expected publication date of the API documentation is late 2016) but could (and should, to prevent unnecessary traffic) be cached locally in users' systems.

Another avenue of future development involves adding mechanisms to assert relationships between ContextItemDefinition(s), ContextItemEnumItem(s), and external sources of information. This will enable linking ContextItemDefinitions or ContextItemEnumItems to definitions such as those found in AGROVOC or AgGateway's AgGlossary (www.agglossary.org), and asserting relationships among ContextItemEnumItems from different vocabularies (e.g., different machinery manufacturers' crop lists).

Conclusions

Current trends in sustainability, traceability, and compliance reporting demand an ever-increasing amount of data be gathered as part of everyday modern production agriculture operations. Specific requirements for what data is collected, how often it is collected, and the format it must be reported in, are constantly evolving and highly geo-political-context-dependent. This makes fulfilling all of those requirements in a common object model an ever-moving target, unless it is possible to decouple these frequently-changing data from an infrequently-changing core.

The solution adopted by the ADAPT team was to provide the relatively static and "universal" core components of the ADAPT framework with placeholders for attaching ContextItem(s). Users of ISO 11783-10 task files can use the system as well, attaching ContextItems via the link list file defined in ISO 11783-10 Annex E.

A controlled vocabulary is essential in properly communicating the meaning of data and, through its consistent use, improves data quality. However, the dynamic nature of business and regulation requires this vocabulary to be easily extensible. If this controlled vocabulary is allowed to become "stale", again, the data model's relevance suffers. The ContextItem system is intended to be a living resource, continuously updated cooperatively by the ag industry, and distributed through a RESTful web service. It is a lightweight method for extending field operations object models, and an elegant way to reconcile conflicting requirements.

Acknowledgements

The authors gratefully acknowledge Arren Mund (Ag Leader; AgGateway/SPADE3's Reference Data API Product Team Chair), Jim Wilson (AgGateway; Standards Director), Ben Craker (AGCO; AgGateway/SPADE3 Project Chair), Shannon Haringx (Syngenta; AgGateway Precision Ag Council Communications Chair), the members of the AgGateway/SPADE2-3 Reference Data API and Regulatory product teams, and the members of AEF's PG9 team (especially Martin Sperlich of CLAAS and Daniel Martini of KTBL) for their comments, suggestions and support.

References

AgGateway (2015). The SPADE Project. <http://s3.amazonaws.com/aggateway_public/AgGatewayWeb/About%20Us/CommunicationsKit/AgGatewaySPADE3_11415.pdf> (June 9, 2016)

AgGateway (2016). The ADAPT Toolkit: Implementing Interoperability in Precision Agriculture. <http://s3.amazonaws.com/aggateway_public/AgGatewayWeb/About%20Us/AgGateway_ADAPT_Toolkit_5616.pdf> (June 9, 2016)

Caracciolo, C., A. Stellato, A. Morshed, G. Johannsen, S. Rajbhandari, Y. Jaques and J. Keizer. (2013) The AGROVOC Linked Dataset. *Semantic Web* 4(3): 341-348.

European and Mediterranean Plant Protection Organization (2015) EPPO codes: a brief description. <http://www.eppo.int/DATABASES/GD&Codes/A4_EPPO_Codes_2015.pdf> (June 9, 2016)

GS1 (n.d.) Global Location Number (GLN). <http://www.gs1.org/gln> (June 9, 2016)

IANA (n.d.) – Language Subtag Registry <http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry> (June 9, 2016)

International Organization for Standardization. (2005). ISO/IEC 19501:2005 Information technology -- Open Distributed Processing -- Unified Modeling Language (UML) Version 1.4.2. International Organization for Standardization, Geneva, Switzerland.

International Organization for Standardization. (2013) ISO 20022 Financial services -- Universal financial industry message scheme -- Part 1: Metamodel. International Organization for Standardization, Geneva, Switzerland. (See also <http://www.iso20022.org>)

International Organization for Standardization. (2015). ISO 11783-10:2015 Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 10: Task controller and management information system data interchange. International Organization for Standardization, Geneva, Switzerland.

Kleene, Stephen C. (1956). Shannon, Claude E.; McCarthy, John, eds. Representation of Events in Nerve Nets and Finite Automata. Automata Studies Princeton University Press. pp. 3–42.

Leach, P. J., Mealling, M., and Salz, R. (2005). “RFC 4122 - A Universally Unique IDentifier (UUID) URN Namespace.” *IETF Tools*, <https://tools.ietf.org/html/rfc4122> (Jun. 9, 2016).

Lewis, T. (1998) Evolution of farm management information systems. *Computers and Electronics in Agriculture* 19(3): 233-248.

NRCS (n.d.) Crop Management Templates <http://www.nrcs.usda.gov/wps/portal/nrcs/main/national/technical/tools/weps/cropmgnt/> (June 9, 2016)

UN CEFACT (2005), Recommendation No. 20 – Units of Measure used in International Trade. <http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_rev3_Annex3e.pdf> (June 9, 2016)

W3C/IETF. (2001). “URIs, URLs, and URNs: Clarifications and Recommendations 1.0.” *World Wide Web Consortium (W3C)*, <https://www.w3.org/TR/uri-clarification/> (Jun. 9, 2016)