ADAPT: A Rosetta Stone for Agricultural Data

B. Craker[1], D.D. Danford[2], R.A. Ferreyra[3,*], K. Nelson[4], S.T. Rhea[3], M.W. Stelford[5], J.A. Wilson[6]

[1] Ag Data Coalition (Springfield, OH, USA); [2] CNH Industrial (Burr Ridge, IL, USA); [3] Ag Connections, LLC (Murray, KY, USA); [4] Farmbelt North, Inc. (Minneapolis, MN, USA); [5] Premier Crop Systems, LLC (Des Moines, IA, USA); [6] AgGateway (Washington DC, USA)

Modern farming requires increasing amounts of data exchange among hardware and software systems. Precision agriculture technologies were meant to enable growers to have information at their fingertips to keep accurate farm records (and calculate production costs), improve decision-making and promote efficiencies in crop management, enable greater traceability, and so forth. The attainment of these goals has been limited by the plethora of proprietary, incompatible data formats among equipment manufactures and farm management information systems (FMIS), along with a lack of common semantics (meaning) in the industry. Proposed partial solutions exist; e.g., the ISO11783.10 standard XML format is well-known and respected, but it is machinery-specific and does not include business-process details needed by growers' FMIS.

AgGateway is an industry consortium of 200+ companies in the agricultural industry. In 2013-14, its SPADE project explored the feasibility of the industry developing an open-source format conversion toolkit. This experience led to what is now its ADAPT Committee.

The ADAPT team created a common object model or "Application Data Model" (ADM), a super-set of field operations data models presented by participating companies. The goal: to replace the current, fragile situation, where FMIS must support multiple hardware data formats, and each machinery manufacturer has to interact with multiple software companies, with a single ADM integration mediated by a framework (currently built on .NET Framework 4.5.1 / .NET Core 2.x; ADAPT can run on Windows, Mac or Linux) from which manufacturer-specific plug-ins convert to and from proprietary formats. This enables the FMIS to read/write to a wide variety of systems with little incremental effort, using ADAPT as a form of a digital agriculture Rosetta Stone. A special emphasis was placed on developing a data-driven approach to managing geopolitical-context-dependent information, and on delivering shared meanings (semantic resources) through application programming interfaces (APIs).

Licensing is an important consideration when seeking to promote the wide adoption of a software platform. The ADAPT Committee selected the well-known, and broadly accepted, open-source Eclipse Public License for the ADM, the conversion framework, and community plug-ins. The licensing model for proprietary plug-ins is different from that of the community-supported tools: each plug-in writer can choose whatever licensing and distribution model best fits their business model.

Several machinery manufacturers have already begun writing plug-ins for their hardware; their projects are at different stages of development. There are currently two community-supported plug-ins: one to convert ISO ISO11783-10 XML files; and another to perform lossless serialization and de-serialization of ADM instances. The former serves as a template for machinery companies that use the ISOXML format to customize, and the latter enables FMIS-to-FMIS communication, a critically-important function that the industry has been lacking. Future plans for community-supported plug-ins include one for the Precision Agriculture Irrigation Language (PAIL) format, and another for sustainability metrics.

The ADAPT Committee has a GitHub repository for source code, exercises transparent governance, hosts an email list for questions (adapt.feedback@aggateway.org), and accepts contributions from outside AgGateway. The scope of ADAPT includes self-propelled machines, non-mechanical processes, observations and measurements, and post-harvest traceability. The intention is for it to facilitate the growth of digital agriculture.

Keywords. international, ISO, software development, standards, ADAPT, AgGateway.

* Corresponding author. andres.ferreyra@agconnections.com, PO Box 978, Murray, KY 42071 USA

INTRODUCTION

Different brands of farm equipment and software currently collect and consume data in a variety of proprietary file formats. While this is a natural consequence of the industry's growth, it makes it hard for end-users to "connect the dots" and extract value from the data.

This lack of interoperability in agricultural field operations is not just a problem of a lack of common data formats or syntax. There has also been a lack of a shared understanding, or semantics, among the different industry actors involved in field operations. This can take the form of using multiple terms or codes to refer to the same concept, or using the same term to refer to multiple concepts. (Applegate et al., 2016)

A corollary of the aforementioned lack of common semantics is a lack of common Reference Data; i.e., a common set of controlled vocabularies, code lists and unique identifiers that can be used to identify resources such as crop inputs, farm machine, implement and sensor models, etc. in a consistent way understood by all.

Concurrently, growing public interest 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 these requirements in the data model of farm management information system (FMIS) software is very difficult, especially in an international context and given the realities of corporate information technology, where the release frequency of new software versions is constrained in multiple ways.

The keys to solving the aforementioned problems could be summarized as follows:

- Interoperate despite the multiple, often proprietary data formats used in the industry.
- Develop a framework to capture and express meaning in field operations
- Develop a framework for sharing Reference Data across the industry
- Decouple the infrequently- and frequently-changing aspects of FMIS data models.

AgGateway (www.aggateway.org), a nonprofit consortium of over 200 companies dedicated to the implementation of standards to advance digital agriculture, created its Precision Agriculture Council in 2011 to collaboratively tackle these interoperability problems. This led to the creation of the Agricultural Data Application Programming Toolkit (ADAPT) team, charged with implementing a common object model for field operations as well as a set of format conversion tools (AgGateway, 2016A).

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

The goal of this paper is to introduce ADAPT to the precision agriculture practitioner community. Its specific objectives:

- Describe precursor technologies to ADAPT, particularly the ISO 11783 standard;
- Describe the parts of ADAPT, and the current status thereof.
- Describe ADAPT's data-driven solution to the problem of incorporating geopolitical-context-dependent and other frequently-changing data into its generic field operations data model
- Describe the tooling put in place to support ADAPT users.
- Describe efforts to provide ADAPT with a framework of common meaning and Reference Data.
- Provide practitioners with an implementation example.

PREVIOUS EFFORTS

A common "language" to enable easy communication of data among a variety of farm equipment, software systems, and actors has long been a goal and need identified within the industry. Several previous initiatives achieved some success in this area and helped guide the ADAPT project.

One of the main antecedents to ADAPT is the ISO 11783 standard (ISO, 2017), also known as "ISOBUS". This standard was developed seeking to solve compatibility issues between tractors and implements from different manufacturers. This is often likened to the challenges farmers faced with hydraulic couplers: in order to run an implement of one brand with a tractor of another, a series of adapters and fittings were often required just to be able to get an implement to physically raise, lower, fold, or anything else that required hydraulic power from the tractor.

A similar situation occurred with electronics and terminals for controlling farm machinery. A grower often had a terminal in their tractor cab to control tractor operations such as transmission settings, hydraulic flows, and to monitor the general operation and performance of the machine. To use an implement in a field operation, they needed an additional monitor dedicated to that function; additional auxiliary control boxes for optional features, or add-on equipment may have been needed as well. These proprietary terminals required effort to install and set up, since many needed tractor information such as ground speed or GPS position that were not easily obtained in the proper format or scale.

The ISO11783 standard was meant to simplify this connection process so it would be more "plug-and-play", in line with consumer electronics. The ISOBUS standard includes several parts such as the physical layer (ISO, 2012), which standardized the actual connector plug and other base requirements for power and electronic communication for both the tractor and any implements or other devices connected to the communication network (ISOBUS) used by the connected devices. (These devices coupled by, and using, the ISOBUS are collectively known as the Mobile Implement Control System, or MICS.)

As the standard was developed, it also became apparent that the information being transferred over the ISOBUS had value, not only to help different devices connect to each other while performing field operations, but also to record what the different devices were doing, or supposed to be doing, and geo-referencing all this data.

This led to the creation of task files, which can be divided into two types: planned task files or *work orders*, that represent what is supposed to be done, and actual task files or *work records* - what was actually done. Work orders and work records correspond to two of five *Core Documents* with which AgGateway has modeled the documentation of principled decision-making in field operations (AgGateway, 2016). For reference, the other three are *Plan*, *Observations & Measurements*, and *Recommendation*.

While task files provide an important foundational piece to support field operations data exchange, there are some issues the ADAPT project hopes to resolve. An important one is that the ISO 11783 standard is very good at identifying and describing sensor values and parameters that can be logged by a machine or tractor-implement combination, but it does not define the necessary Reference Data very well.

A simple example will help introduce the critically-important Reference Data concept: The ISO 11783 standard clearly describes the per-area quantity of a chemical that applied in a given field operation, but the actual chemical of interest is often under-identified, to the extent of leaving it to the operator to manually enter the product name in the terminal. The standard cannot directly capture information such as the product's active ingredients, or the products that compose a tank mix, however. This can lead to problems after the field operation, where a software system like a Farm Management Information System (FMIS) needs to calculate total chemical/nutrient loading within a field, or a restricted-entry or pre-harvest interval that is dependent on the product used; if the product is identified inaccurately, these safety-related time intervals could be calculated incorrectly. This can have important regulatory implications for the grower, especially if the product in question is a restricted-use pesticide and its name is captured incorrectly. There is a gap in identification here; additional ISO 11783 gaps include (Applegate et al., 2016):

- The standard is limited to MICS-FMIS communication in the context of field operation execution, and does not support FMIS-FMIS transactions that may involve additional "documents" (e.g., a recommendation from an agronomist, or data needed to complete a regulatory report) that are not machine-specific and therefore not covered by the ISO 11783-10 format.
- It initially supported locally-scoped identifiers only.
- It is not well-suited for some field operations such as irrigation with center pivots, where irrigated areas have complex geometries.
- Its adoption is not yet universal, and there exists a multitude of proprietary formats for machinery and FMIS that are inherently incompatible with it.

Another precursor to ADAPT was the Field Operations Data Model (FODM), which sought to resolve another gap with the ISO11783 standard that was especially prevalent in the North American market in the early 2000s: most equipment manufacturers used proprietary file formats and did not follow the ISO standard. The FODM system used Field Operations Data Drivers, or FODDs (Mapshots, 2013) to convert between proprietary formats and the FODM. The FODDs managed the translation of data from proprietary (or ISO) formats through the FODM so FMISs were able to access the different file formats though one common interface. In most cases, equipment manufacturers provided their own FODD sand licensed them to different software systems.

This system works well within its scope. However, like the ISO11783 standard, FODM is focused on field operations that are carried out by tractor-implement combinations, with little if any consideration for irrigation operations and documents such as recommendations. Additionally, the proprietary nature of FODM makes any desired changes, and the governance thereof, reliant on the party controlling the FODM system.

AGGATEWAY AND ITS FIELD OPERATIONS PROJECTS

Through 2004, agriculture-industry electronic-connectivity standards were fragmented across industry sectors. A number of industry leaders determined that an organization should be established to address this issue and promote industry-wide electronic connectivity; AgGateway was established in 2005 as a result. At the time, AgGateway supported seed, crop nutrition, crop protection, and feed companies, and executed projects for automating supply-chain management processes for those segments. Software service providers were engaged to implement standards support. Implementation of existing standards was favored over creating new ones. By 2018, AgGateway membership grew from a handful of companies to over 200.

In 2011, a set of influential companies in different segments (Equipment OEMs, FMIS companies, retailers, input manufacturers, etc.) of North American precision agriculture met in Des Moines, Iowa, United States to consider whether AgGateway's successful approach to supply chain standards and connectivity could be applied to field operations. They concluded that the approach could work well and set about establishing a Precision Agriculture Council within AgGateway. Shortly after establishment, the group began project planning for a multi-year phased project to implement field operations standards and launched SPADE1 in August 2012. A sister project, PAIL, was launched in late 2013 to focus on irrigation-related data and processes, as well as the Observations and Measurements core document, which is an agriculture-specific implementation of the ISO 19156 standard (ISO 2011A).

As SPADE1 transitioned to SPADE2, the project teams observed that the myriad field-operations data formats used in the industry were the primary connectivity issue facing growers. The team agreed that competitive advantage should be pursued at the level of doing things with data, and not at the data format level. With that principle in mind, the team (with PAIL input) began work on mapping out the scope, license, and processes for an open-source project to create a tool for enabling interoperability across various field-operations data formats; ADAPT is that tool.

ADAPT: WHAT IT IS, WHAT IT DOES, HOW IT WORKS

ADAPT is a set of tools for representing agricultural field operations data and converting it among different formats. It consists of three primary parts:

- A *common object model* for describing field operations, implemented as a collection of extensible business objects.
- A set of *plug-ins*, or external libraries, both open source and proprietary, which contain the data conversion logic.
- A *plug-in management framework*, i.e., a mechanism for discovering, enumerating, and invoking these data conversion libraries at runtime.

The ADAPT Object Model

An object model represents some part of the world that is of interest. Object models are composed of *classes* (which represent groups of related data; an object-oriented version of the classic concept of a *data type*) and the *relationships* among them. Figure 1 shows a small part of the ADAPT object model with five classes (boxes labeled Grower, Farm, etc.) and the relationships among them (the arrows). The data (*attributes*) contained in each class are listed inside its box; for example, a Grower (which represents the business entity, rather than the person, which is modeled using Person and PersonRole classes, not shown) has an attribute called Name, which is of the String class.

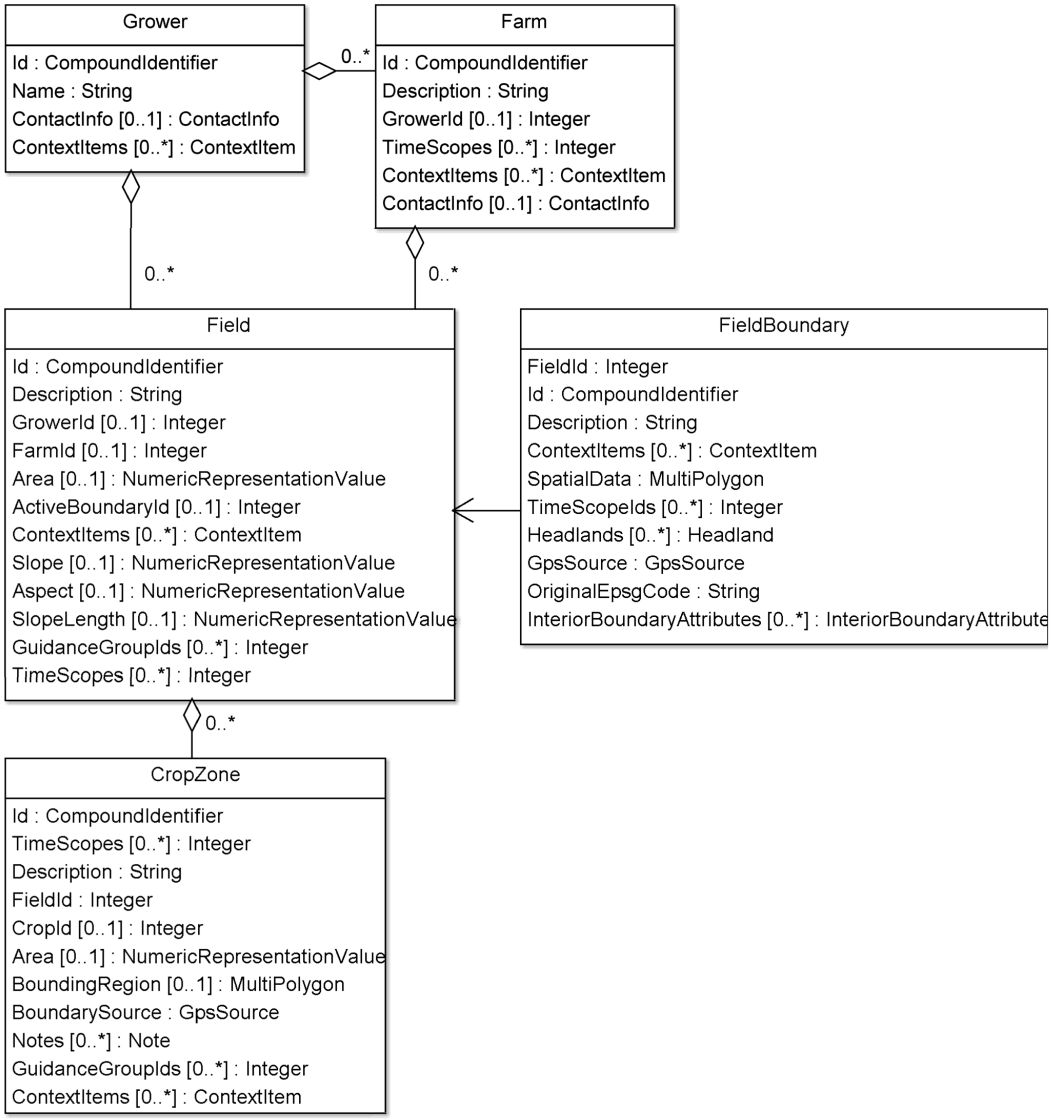

Fig. 1: A small subset of the ADAPT object model in the form of a UML class diagram (ISO, 2005) that shows some of the classes related to the concept of a Grower. Simple arrows represent associations (e.g., when an object contains references to another object). The arrows tipped with diamonds represent a specialized form of association called *aggregation*, where an object contains other objects (e.g., a farm can have multiple fields), but these contained objects can "stand alone", and have their own identity independently of the containing object. The way in which the ADAPT team implemented some of these concepts (e.g., fields being able to exist independently of farms) was a result of maintaining compatibility with the ISO 11783 standard, and with particular contributors' internal data models.

In order to use the ADAPT data model (also called "application data model", or ADM), FMIS software must first create an instance of it, and then begin populating it with instances of objects as needed by the application at hand. For example, if a grower wishes to use ADAPT to send a Work Order for a spraying application to a retailer or custom applicator, their FMIS would follow several steps:

- Create an instance of the ApplicationDataModel, and its children the Catalog and Documents objects.
- Create objects for the *Reference Data* needed by the Work Order and put them inside the Catalog; for example, objects that describe the crop protection products that will be applied in the work order. Reference Data is meant to be universal, and independent of the specific grower.
- Create objects for the *Setup Data* needed by the Work Order and put them inside the Catalog. Setup Data refers to grower-specific data objects that describe resources being used by the document of interest (i.e., the Work Order), but that do not specifically describe the *state* of the resource. Examples include objects for the grower, farms, fields, cropzones, and boundaries that may be a target of the work order. If the work order was internal to the grower's own operation, it could also include references to specific machines, and to operators thereof.
- Create objects describing the desired field operation and put them inside the Documents object. In this case, it would entail creating instances of:
    - The Work Order document;
    - A WorkItem object, representing the desired pass over the field and referenced by the Work Order;
    - A WorkItemOperation object, that describes the spraying operation;
    - One of two possible Prescription objects (chosen based on the level of spatial detail desired to specify placement) that reference the Product objects created earlier, and that describe how much of each product to apply to the fields and cropzones represented by previously-created Setup Data.

To better understand specific classes (e.g. WorkItem), please refer to the source code located in https://github.com/ADAPT/ADAPT and the material listed in the Sample Code and Application Notes section, below.

An important aspect of ADAPT is how it manages identification. Figure 1 shows the use of a class called "CompoundIdentifier" to identify instances of objects. This class provides powerful functionality meant to reconcile different identification schemes used in the industry; its use was described in detail by AgGateway (2017). The Implementation Example section of this document shows how instantiating a CompoundIdentifier and various examples of Setup Data are actually implemented in code.

ADAPT Plug-Ins

Once the application data model has been instantiated, and the FMIS has populated it with objects of interest, the FMIS will invoke a plug-in to perform a conversion. Understanding how this works may be best accomplished using examples. Figure 2 below shows two:

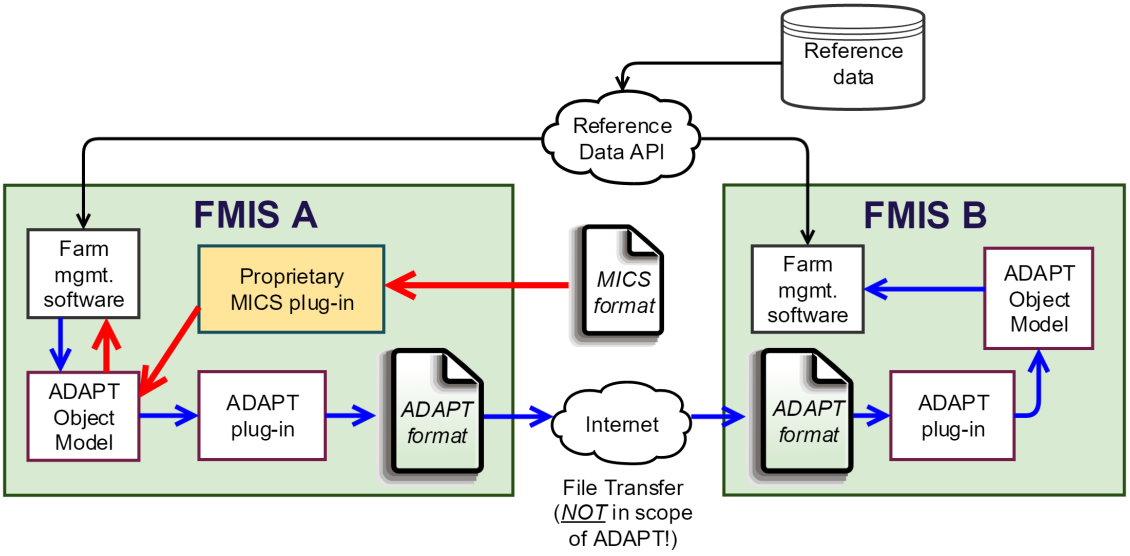

Fig. 2: Two examples of ADAPT-mediated data conversion. Fig. from AgGateway (2016), w. permission.

Incoming data from a Mobile Implement Control System (MICS, i.e., the controller in the cab of a tractor or other farm machine. Leftward data flow, shown in red): A proprietary-format data file coming from a controller in the field is converted by a manufacturer-specific plug-in into an instance of the object model; a farm management information system ("FMIS A") consumes the data.

Communication between Farm Management Information Systems (FMIS, i.e., farm management software. Rightward data flow, shown in blue): FMIS A creates an instance of the ADM, populates it with the data it wants to transmit, and uses the ADM ("ADAPT") plug-in to serialize it to a file. This file is transmitted to another FMIS using the Internet or another means. (File transport is currently out of scope of ADAPT, accommodating different solutions available in the industry.) FMIS B uses the ADM plug-in to convert the ADAPT-formatted file to an instance of the ADM, and then consumes the data.

Note how FMIS A and FMIS B are both supported by *Reference Data*, i.e., a distributed system of common unique identifiers for products to be shared across the industry by manufacturers and third-party data providers.

The ADAPT plug-in concept supports several scenarios along several dimensions: open-source plug-ins and closed-source plug-ins, freely-available plug-ins and for-fee plug-ins, liberally-licensed plug-ins and per-user licensed plug-ins. The plug-in ecosystem can be both market-driven and collaboration-driven as various plug-in providers consider what is best for their company and their customers. The Implementation Example section of this document demonstrates how to invoke the ADM plug-in (open-source, freely-available, liberally-licensed) for data export / import, but the example is extensible to any plug-in.

A few key points follow, related primarily to what ADAPT is *not*:

- ADAPT is not a stand-alone application: ADAPT is a set of libraries that are linked to, and used by, other software applications.
- ADAPT is not a file transfer mechanism: The scope of ADAPT is limited to invoking plug-ins. These were envisioned originally as format-conversion tools that produce a file on export. This file can then be transmitted as needed. Plug-in authors are free to add communication features to their plug-ins, however; there is nothing wrong with a plug-in invoking a web service to transmit data it has converted, for example.
- ADAPT is not an authentication & authorization mechanism for sharing data: Associated with the idea that file transfer is out of scope, the mechanisms for authenticating and authorizing users to access or transfer data is likewise out of scope.

ADAPT Plug-In Framework

Figure 3 shows how the ADAPT plug-in framework is organized. All plug-ins implement the IPlugin interface – a pattern of required data attributes and functions. This allows an instance of the PluginFactory class, created in an FMIS, to enumerate available plug-ins by inspecting the precompiled code libraries in a given subdirectory for the presence of "Import" and "Export" functions. Once enumerated, the FMIS can programmatically load the required plug-in and perform the necessary data transformation operation.

There remain some areas in which this implementation must mature. Often there are constraints on the MICS platform (usually on product name length, or the supported depth of the Grower/Farm/Field tree) that need to be communicated back to the FMIS prior to the creation and population of the ADM object. There is currently no automated mechanism for reporting these limitations, so the responsibility falls to the plug-in developer to document these issues appropriately.

Another current issue is how the ADAPT team communicates status of the plug-in ecosystem. The team maintains a list of those companies who have either provided a plug-in or committed to do so. Some plug-in providers have even gone further and provided details as to the status of their plug-in(s). While a good start, more data needs to be collected and shared regarding current plug-in status, future plans, and features supported. The information must also ultimately be expressed in a machine-readable way that can help farmers make equipment and software decisions.

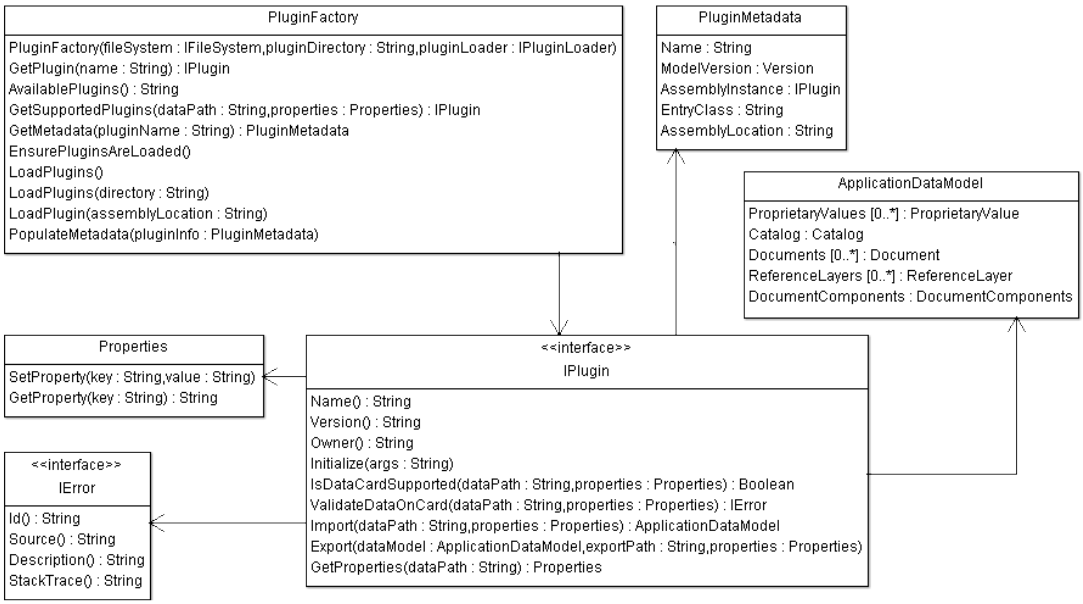

Fig. 3: Organization of the ADAPT Plug-In Framework, shown as a UML class diagram. (ISO, 2005).

DEALING WITH GEOPOLITICAL-CONTEXT-DEPENDENT DATA

As mentioned earlier, farm management information systems are inextricably linked to the grower's business processes, which in turn tend to involve geopolitical-context-dependent data; for example:

- The EPA number in the description of a crop protection product in the United States.
- The German Bundessortenamt code used to designate crop varieties.
- The FSA Farm Number, Field Number, and Tract Number, used to identify areas for participation in US government programs.

These attributes are specific to a particular country or region and can change along with the laws and regulations that mandate their use. It is of great importance to an FMIS to include this information (in order to be relevant in the context of the grower's business processes), but geopolitical-context-specificity presents a formidable scalability and versioning problem when the FMIS is used internationally. Additionally, the prospect of temporal change creates problems for users because object model changes that would accompany the addition or deprecation of a particular attribute would necessarily require the release of a new version of the software, with deployment, training and version maintenance costs for all involved.

These challenges are generally inconsistent with the use of standards such as ISO 11783, which focus on representing "universal" concepts such as application rates by mass, volume or units per area, distance or time; yields by mass or volume per unit area; distances; and so forth. Making changes in a standard usually follows a multi-year timeline, which can't accompany the rate at which variables may enter and exit a grower's regulatory context; ISO 11783-11 (ISO, 2011) targeted the timeline problem by defining a data dictionary for the variables being collected in ISO 11783-10 task files that can be added to independently of the standard's revision cycle; however, the data dictionary entities (usually called "DDIs") are very narrowly-scoped to be machine- and implement-specific; furthermore, a cursory examination of their content (ISO, 2018) shows that the underlying intent is to capture "universals" that are independent of any particular geopolitical context.

An FMIS object model should thus simultaneously reconcile:

- Simple/generic (as a typical internationally-usable standard that is free of regional clutter) vs comprehensive/specific (to enable the capture of geopolitical-context-dependent data), and
- Static (based on controlled vocabularies the meaning of which can be agreed upon by all users) vs dynamic (to enable rapid extensibility of said vocabularies to accompany change).

The ADAPT, SPADE and PAIL teams solved this problem using a flexible data-driven approach: the ContextItem System. This method was described in detail by Daggett et al. (2016), but essentially implements the following key ideas (Illustrated in Figure 4):

- ContextItems are defined as key-value pairs with optional additional metadata (such as units of measure and timestamp information).
- The keys (called "Code" in the model) convey the meaning of the value.
- This meaning is further documented using a class called ContextItemDefinition, available using an API at https://api.contextitem.org that can be accessed by FMIS or other systems that encounter (or wish to encode) a given ContextItem in a data file.
- ContextItemDefinitions can provide strings in various languages (Lexicalizations) to communicate to a user what the underlying variable means. They can also encode the geopolitical context (using controlled vocabularies such as the ISO 3166-2 codes) and the class(es) within the data model that the given ContextItem can be used in. This enables class- and geopolitical-context-specific searches for ContextItemDefinitions in the API.

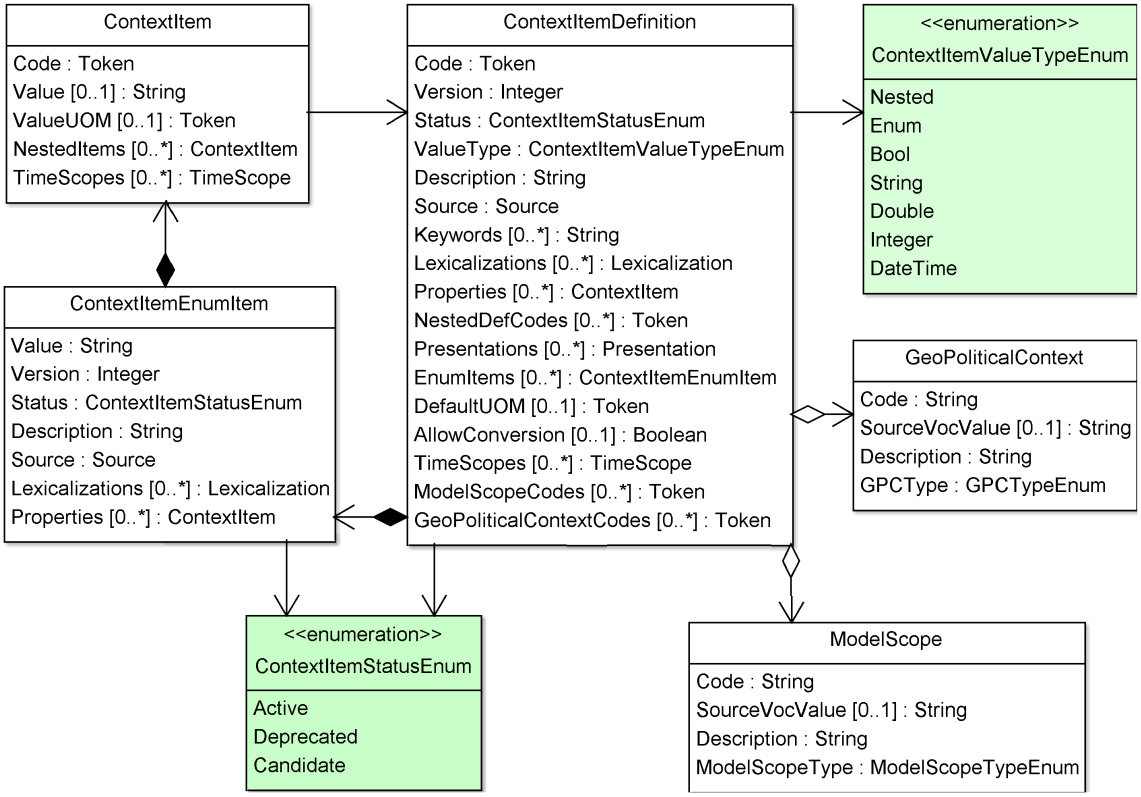

Figure 4: The ContextItem system data model. Several classes (CompoundIdentifier, TimeScope, Lexicalization, GPCTypeEnum,etc.) are not shown, for clarity.

PROGRESS TO DATE

Implementation of the Framework and Data Model

The data model has been released and the source code of its current release version (1.2), that all known plug-ins are compatible with, is available at https://github.com/ADAPT/ADAPT/releases/tag/v1.2.0.

This object model is currently in use in multiple production systems. Being an open-source project, it will continue to evolve with the changing needs of the participating companies. The latest versions of the data model expressed as UML class diagrams, available at https://aggateway.atlassian.net/wiki/x/RQDbAg, offer a glimpse of what may be implemented in the future.

Governance

AgGateway established a standing ADAPT Committee, divided in turn into Business and Technical Teams, which provide day-to-day oversight for the project. The teams meet periodically: the Business Team establishes the general project direction, manages resources, and enacts a communication strategy; the Technical Team reviews code contributions for consistency with the existing code and relevance to the overall goals of ADAPT. Code contributions follow a documented path to approval on Github (github.com, 2016), allowing for transparency and maintainability of the source code. While most contributions come from AgGateway members, the project welcomes all interested parties. There is an email address to direct questions to, adapt.feedback@aggateway.org.

Code Versioning and Branching

An important aspect of governance of a multi-stakeholder, open-source project like ADAPT is how to manage successive versions of the code and the naming thereof ("versioning"). Another important aspect is how code submitted by the different contributors gets incorporated into these versions ("branching policy").

The ADAPT team chose a versioning strategy called Semantic Versioning 2.0 (Preston-Werner 2013). It is based on a version naming scheme of the form A.B.C where A is the *major version number*, B is the *minor version number*, and C is the *patch number*. The major version is incremented when the new version includes a "breaking change", that will result in a new version being incompatible with the previous one. The minor version is incremented when there is a significant, albeit non-breaking, addition. The patch number changes when there is a revision to a minor version, typically as a result of a (nonbreaking) bug fix.

Regarding branching policy, ADAPT follows a method described by Driessen (2010). First, a definition: a *branch* is a copy of the source code that incorporates a series of changes made by one or more developers. ADAPT keeps two branches, *Develop* and *Master*. The *Master* branch reflects the latest released version, and is the source code used for deployment. The *Develop* branch contains the code staged for the next release. Developers wishing to contribute must make their own *fork* (i.e., personal copy) of the *Develop* branch, modify it, and request that it be merged back into *Develop* (a *pull request*, Johnson, 2013). These requests, and the approval thereof, are managed by the ADAPT Technical on GitHub (github.com, 2016).

Proprietary Plug-ins and FMIS Adoption

As with any project of this nature, initial adoption has been a function of market forces. Many MICS and FMIS companies were hesitant to commit their limited development resources to the task of integrating ADAPT until there was both verified demand by their customer base and sufficient confidence that the collaborative nature of ADAPT was sustainable.

Despite this behavior there have been several MICS companies that have created ADAPT plug-ins for their proprietary formats. John Deere was the first to release plugins for three different MICS models (Greenstar 2600, 2630, and the 4600). Topcon provided engineering help to get the ISO 11783 plug-in to a production state in early 2018, and ISOXML-using companies are expected to use this work as a basis for their own ISOXML implementations. In June of 2018 Trimble Inc. announced to the ADAPT community the availability of two plug-ins to support four of their MICS models. CNH Industrial and two other major OEMs are working to develop plug-ins to support their formats of interest; these plug-ins should be released by the end of 2019. The page at https://adaptframework.org/companies-supporting-adapt/ provides more information.

On the FMIS front, several companies have integrated the ADAPT framework to leverage the commercially-available plug-ins from John Deere, as well as the ISO plugin. There have been efforts to use the "ADM plug-in" to serialize data for exchange between FMIS companies, and a pilot project is underway to use RESTful APIs for exchange (allowing for more "transactional" exchange of data using ADAPT objects). Many companies appear to be embracing a philosophy of opportunistic adoption by leveraging existing engineering priorities and upcoming projects as opportunities to integrate ADAPT.

Community-Supported Material

The ISO 11783 Plug-In

One of the first plug-ins developed for ADAPT was the "ISO Plug-in", meant to provide support for data files that follow the ISO11783 standard. At the time of this writing the ISO plug-in is on its second revision. The plug-in is made available under the same open-source license as the ADAPT Framework; this enabled a wide community to contribute to its creation and improvement.

The ISO 11783 plug-in serves several purposes; first, *enabling ADAPT compatibility with the numerous ISO-11783-compliant machines and terminals in the marketplace*. This is important given the growing number of 11783-compatible devices worldwide, but is especially significant for the European marketplace, where adoption of the ISO standard is widespread. The ADAPT ISO plug-in thus enables FMIS software to read/write to many different ISO-compliant OEMs, thus increasing the value that integrators receive for implementing the ADAPT framework, even before many OEMs with proprietary formats have finished their respective plug-ins.

A second purpose of making the ISO plug-in available and making it open-source was *to give other OEM's a starting point*. Without some open-source material to draw from, OEMs with their own proprietary formats had an uphill climb to create their own plug-in from scratch. The ISO plug-in can be used as a guide on how the ADAPT team would like to handle different edge cases as well as base to build from and tweak to create a plug-in for a proprietary format.

All plug-ins serve basically the same function, to map data from one format into, or out of, the ADAPT model; this provides FMIS and other software companies a common interface for all file formats. Using the open source ISO plug-in allows OEMs to have a functional plug-in to look at to see how it works, but since it is under the open source license they also have the potential to reuse some of the code where applicable reducing their overall development effort.

A third purpose of the ISO plug-in was to *drive more consistent implementation of the ISO 11783 standard itself.* As with most standards, the purpose is to get all parties to do things in the same way where possible while not inhibiting competition. However, standards are created by parties who often have very different approaches and goals; they come together to create a common understanding. This generally leads to some grey areas in the standard where the parties could not come to full agreement, and ISO 11783 standard is no exception. In an effort to not limit the viability of different solutions, the standard had to be flexible enough to support very different use cases and solutions. This, however, leads to slightly different implementations that can be confusing and cause incompatibility between different systems that can both claim to follow the standard. The Agricultural industry Electronics Foundation (AEF) has worked to develop conformance tests for the ISO11783 standard to resolve this issue. The conformance tests are created to drive a common implementation of the standard to increase compatibility between systems and minimize the differences in implementation. The AEF's efforts have predominately focused on machine to machine interface (tractor-implement) portions of the standard, with not as much focus put toward the compatibility of field computers and FMIS systems. The ADAPT team worked closely with the AEF FMIS conformance test team during the development of the ISO plug-in to help insure it would follow the FMIS conformance test requirements. The hope being that the ADAPT plug-in could be used as a means to implement the conformance test requirements, almost automatically to an FMIS that implements the framework. This would also have the effect of driving OEM's to ensure their field computers were compatible with the ADAPT ISO plug-in as a kind of conformance test. If the OEM did not agree with the implementation within the ISO plug-in or was unable to change their implementation, they would then have the option to develop their own, essentially proprietary ISO plug-in to support their specific implementation of the standard using the original ADAPT plug-in as the open source base for their development.

The ADM Plug-In

Plug-ins made for MICS-to-FMIS communication such as the ISO 11783 plug-in, or specific manufacturers' MICS plug-ins, are inherently *lossy*. This means that the ADAPT data model can express more data than the MICS (for example, a strategic document such as a crop Plan, or the Observations and Measurements taken in the field that gave rise to a Work Order), and that some data will be lost in FMIS-to-MICS translation (though likely not in the opposite direction). This is a natural consequence of the data

model on the machinery being specific to managing the field operation itself. There are, however, many situations that call for lossless communication of data between two parties; FMIS-to-FMIS data exchange is the prime example thereof.

The ability for two different manufacturers' FMIS to exchange data beyond the scope of a field operations task file has eluded the industry to date. The ADAPT Team created a special plug-in to enable the necessary lossless serialization (i.e., conversion of data in an object model to a stream of bytes that can be exchanged): the ADM plug-in. It uses a mix of JSON (IETF, 2017) and ProtoBuf (Google, 2017) to serialize the entire common object model for persisting to storage or packaging for transport. While ADAPT does not currently make provisions for the movement of the resulting file among data exchange partners, as of this writing, companies are either using RESTful application programming interfaces (APIs) to exchange data or building communication features into their plug-ins to do the same. (M. Stelford, Pers. Comm.)

Visualizer

As the ADAPT data model took shape among the development team, it became apparent that the team needed a way to visualize the data model. So, the team develop the aptly-named utility called "The Visualizer". This program, located at https://github.com/ADAPT/ADAPT-Visualizer, represents the data model in a tree structure with special support for crop field views. Not only does it help the team during development, it has proven a critical learning tool for new implementers.

Sample Code and Application Notes

The team put together a group of code samples, found at https://github.com/ADAPT/ADAPT-Samples, that illustrate various aspects of integrating the ADAPT framework into FMIS and other software. Another important deliverable was a set of application notes that cover various fundamental aspects of using ADAPT. These application notes feature excerpts from the sample code; they can be found at http://aggateway.org/eConnectivityActivities/Committees/ADAPTOversight.aspx.

Other Tooling

Early in the development effort the ADAPT team selected Git (Chacon and Straub, 2014) as its version control system and GitHub (Github.com, 2016) as its primary repository service. In March 2018, the team connected GitHub with the project-management service called JIRA (Atlassian, 2015A) to provide for sophisticated workflow management. Also, in March 2018, a participant implemented continuous integration support using Travis CI, a hosted, distributed continuous integration service, free of charge for open-source projects (Travis CI, 2018). This enables the team to go from accepting a pull request to compilation, to unit testing, to NuGet-package production, to NuGet-package deployment, where NuGet (nuget.org, 2018) is a manager for dynamic-link libraries, a preferred distribution mechanism for the ADAPT framework and its plug-ins.

Community

ADAPT is an open-source project. The Australian organizational-productivity company Atlassian graciously provided its best-in-class wiki and project-management cloud services, Confluence (Atlassian, 2015) and JIRA (Atlassian, 2015A) respectively, to support ADAPT's development, given its open-source status. Anyone in the world with an internet connection can thus create a wiki account and a GitHub account and begin contributing. AgGateway funds project management and marketing efforts. ADAPT's core team is composed of representatives from AgGateway-member companies. Team process definitions were informed by an open-source community expert that AgGateway hired to help avoid pitfalls of previous failed open-source projects, as well as implement proven successful practices.

Licensing

The ADAPT Oversight Committee worked with all interested AgGateway Member Companies to compile a list of key requirements for the Open Source Software (OSS) License to select for recommendation to AgGateway Leadership for providing access to the ADAPT software assets to the global community. These requirements are listed in Table 1, below.

Table 1: Stakeholders and their interests

Stakeholder	Interests
AgGateway (ADAPT's Steward)	• The license is usable. • The license is well-received internationally. • The license is viewed as an integral part of promoting community and preventing fragmentation. • The license encourages integrators to keep current with the latest stable ADAPT releases.
End Users	• ADAPT performs as promised
Software (MICS and FMIS) Companies	• The license permits ADAPT-compatible extensions to the data model and plug-in source code • The license permits providing ADAPT-compatible plug-ins under company-specified license terms, without restrictions on those terms. • The license permits releasing plug-ins in source code form, binary form, or both. (In other words, source code disclosure is not required.) • The license expressly states that companies are not required to support any unofficial "versions" of ADAPT (i.e., "forks).
Software Developers	• The license permits building proprietary software that leverages the ADAPT data model. • The license permits running multiple concurrent ADAPT instances. • The license permits use of supporting resources, such as wiki content, test data, access to additional support, etc. • The license permits developing plug-ins for formats other than those over which the developer has control (e.g., AEMP, USDA) and provide those plug-ins to other companies, provided that the identity of the software provider is clear to plug-in users.

Through communication with the Eclipse Foundation, the ADAPT team identified an OSS community expert to help guide the process of selecting the best OSS license to fulfill ADAPT's requirements. The member companies provided funding to engage the expert on a consulting basis.

Although not an attorney, the consultant had a lot of practical knowledge as well as a great ability to explain OSS complexities to stakeholders unfamiliar with the domain. This helped to speed up the process of selecting a license. Stakeholder companies were engaged to ensure their businesses would be supportive of the selected license to recommend for consideration to AgGateway leadership. Attorneys from several companies weighed in with questions/perspectives which helped the ADAPT Team gain additional confidence in the recommended license.

In the end the ADAPT Committee recommended the Eclipse Public License version 1.0 OSS license (Eclipse Foundation, 2003), which the AgGateway Leadership team adopted. Although this process took much longer than most stakeholders expected, it was an important foundational step to ensure strong support by a wide array of companies worldwide.

A significant aspect of ADAPT is that, despite its governance residing within an AgGateway committee, AgGateway does not require that licensees of, and/or contributors to, the ADAPT framework be members of AgGateway.

IMPLEMENTATION EXAMPLE

This section guides the reader through a simple implementation of ADAPT concepts associated with the second example in the above "How it Works" section (FMIS-to-FMIS communication). It assumes the reader has a basic familiarity with the C# language and object-oriented software concepts, and can access the code and data model at www.adaptframework.org.

Instantiating and Populating the Application Data Model

The ApplicationDataModel is the root object in ADAPT

```
ApplicationDataModel export = new ApplicationDataModel();
```

The Catalog object (inside the ApplicationDataModel) holds all the items one would expect to find in a "pick list". Alternatively, it could be seen as the place to put everything used "by reference" (as opposed to “by value”) in any of the Documents being built.

```
export.Catalog = new Catalog();
```

The Documents object (inside the ApplicationDataModel) is built to house the *Core Documents* that represent field operations (AgGateway, 2016), and holds all the Plans, Recommendations, WorkOrders, WorkRecords, and their respective component parts. For clarity, this example does not include documents.

```
export.Documents = new Documents();
```

FMIS applications often put field operations in the context of a crop year. What follows creates a "crop year" TimeScope object to tag objects with.

```
TimeScope cropYear = new TimeScope();
cropYear.Description = "2018";
cropYear.DateContext = DateContextEnum.CropSeason;
export.Catalog.TimeScopes.Add(cropYear);
```

The next step is to create the Grower object. The constructor will automatically create the Id property and assign the next available ReferenceId integer. Note that ReferenceId must be unique in the scope of any instance of the data model; this is ensured by having the framework create it automatically.

```
Grower adaptGrower = new Grower();
```

ADAPT allows systems to associate internal, unique identifier to objects. In this case, a Universally Unique Identifier or UUID (Leach et al., 2005) is associated with the Grower object by creating a UniqueId object and adding it to the Grower object's CompoundIdentifier.

```
UniqueId ourId = new UniqueId();
ourId.Id = "7d2253f0-fce6-4740-b3c3-f9c8ab92bfaa";
```

Note the available IdTypeEnum choices. Not everybody uses the same way of identifying things in their system. As a result, ADAPT supports a number of identification schemes.

```
ourId.IdType = IdTypeEnum.UUID;
```

Almost as important as the identifier is knowing who created it (or where it came from).

```
ourId.Source = "www.some-company.com";
ourId.SourceType = IdSourceTypeEnum.URI;
```

Each ADAPT CompoundIdentifier can thus have multiple unique identifiers associated with it. This has powerful implications: not only can an organization’s identifier for something be persisted, but so can the identifiers that the organization’s data exchange partners assign to the same object. *Users are strongly encouraged to persist and return identifiers passed to them;* this has the potential to result in a "frictionless" conversation once the initial mapping is done, but the benefit only emerges if all participants are "good neighbors".

```
adaptGrower.Id.UniqueIds.Add(ourId);
```

Many of the objects in ADAPT have a minimal number of properties. Developers shouldn’t be overly worried if there isn’t an obvious place to put all their data. It may be in an associated object or intended to be expressed as a ContextItem (See the section below on Geopolitical-Context Dependent Data)

```
adaptGrower.Name = "Some Random Grower";
```

Adding the Grower object to the Catalog:

```
export.Catalog.Growers.Add(adaptGrower);
```

Having created the Grower, creating the Farm object is next. The constructor will automatically create the Id property and assign the next available ReferenceId integer.

```
Farm adaptFarm = new Farm();
ourId = new UniqueId();
ourId.Id = "c4618a14-4c60-4873-964f-52cc804f1856";
ourId.IdType = IdTypeEnum.UUID;
ourId.Source = "www.some-company.com";
ourId.SourceType = IdSourceTypeEnum.URI;
adaptFarm.Id.UniqueIds.Add(ourId);
adaptFarm.Description = "Some Random Farm";
```

This farm object is now linked to the grower. Note that this is the ADAPT-generated integer (ReferenceId) in the Grower's CompoundIdentifier object, and *not* the FMIS-generated UUID.

```
adaptFarm.GrowerId = adaptGrower.Id.ReferenceId;
```

Add the Farm object to the Catalog.

```
export.Catalog.Farms.Add(adaptFarm);
```

The important takeaway here is that as part of integrating ADAPT, the FMIS must include logic that maps the data from its internal business objects (Grower, Farm, Field, Product, etc) into the corresponding business objects in ADAPT. In the same fashion, plug-in creators must map the business objects contained in their targeted formats to ADAPT objects.

Exporting Data with the ADM Plug-In

The PluginFactory looks at all the DLLs in the target directory to find any that implement the IPlugin interface.

```
var pluginFactory = new PluginFactory(pluginPath);
```

Only the ADMPlugin (i.e., lossless serialization for FMIS-to-FMIS communication) is of interest in this example, so the code below addresses it directly instead of looking through all the available plug-ins that the PluginFactory found.

```
var admPlugin = pluginFactory.GetPlugin("ADMPlugin");
```

The ADMPlugin doesn't require any initialization parameters.

```
admPlugin.Initialize();
```

Export to a local directory using the ADMPlugin

```
admPlugin.Export(export, outputPath);
```

The ADMPlugin creates an "adm" subdirectory in the indicated local directory that contains:

- An additional "documents" subdirectory that contains document files with data encoded using protocol buffers or “protobufs” (Google, 2017).
- An `AdmVersion.info` file that contains version information.
- A `ProprietaryValues.adm` file
- A `Catalog.adm` file that contains the zipped JSON serialization of the `ApplicationDataModel.Catalog` object.

Importing Data with the ADM Plug-In

What follows is logic to import (presumably in another system) the same data from the "adm" subdirectory just created; the corresponding ADP instances can be easily compared in the debugger.

```
var pluginFactory2 = new PluginFactory(applicationPath);
var admPlugin2 = pluginFactory.GetPlugin("ADMPlugin");
admPlugin2.Initialize();
```

Note that when a plug-in imports, the returned object is a list of ApplicationDataModel objects.

```
var imports = admPlugin2.Import(outputPath);
```

DISCUSSION

ADAPT's power lies in its common object model

The ADAPT common object model is the direct result of requirements elicited from the SPADE and PAIL projects, as well as other participating stakeholders. This has allowed it to be representative of the data requirements of real-world field operations. Very early versions of the model were large and skewed heavily towards North America, but the team refactored the data model in a more data-driven way (using ContextItems, for example) that resulted in a leaner, multiple-geography-friendly model.

ADAPT's plug-in architecture removes barriers to adoption

The ADAPT plug-in licensing policy enables the authors of the plug-ins (format conversion libraries) to license and distribute (or not) their plug-ins (and the source code thereof) as they see fit. This was designed with the intent to remove barriers to adoption of ADAPT.

Authors who wish to solve a format conversion problem and subsequently wish to make their plug-ins community-supported (i.e., open-source) are encouraged to do so, and encouraged to use the Eclipse Public License used for the ADAPT data model and plug-in framework.

However, authors who wish to retain ownership of their source code, and/or want to distribute binaries (i.e., compiled plug-ins) only, are welcome to do so as well. This enables them to protect intellectual property, while at the same time contributing to the solution of the industry's interoperability problem.

ADAPT serialization enables lossless FMIS-to-FMIS transfer

Plug-ins created for communication with machinery are inherently "lossy", because ADAPT's object model is a comprehensive superset of individual companies' contributions, and therefore more comprehensive than the data model of the MICS. In other words, ADAPT's data model can likely contain data that does not fit into a controller.

One of the use cases ADAPT was meant to address is lossless data transfer; i.e., where the sending end *and* the receiving end of a data transfer are using the ADAPT data model. As mentioned earlier, enabling this use case required the creation of the ADM plug-in, which serializes everything in an ADM to JSON and Protobufs files. While this is very valuable and revolutionary by itself—FMIS of different brands had heretofore never been able to exchange field operations business data such as recommendations, work orders, etc.—the ADM plug-in has another valuable purpose: to provide a starting point for plug-in writers. Since it is designed to serialize all the classes in the data model, developers can tailor it to their needs, primarily by removing unnecessary parts.

ADAPT leverages the use of data type definitions and controlled vocabularies

ADAPT has mechanisms for defining variables of different types in a data-driven way. The ContextItem system described above enables defining variables for use in various geopolitical contexts, and the Representation System, analogous to the data dictionary of ISO 11783-11, allows defining universal variables such as yield. A third mechanism for defining data types specifically for use in Observations and Measurements, the *OMCodes System*, is under development.

These three *data type registries* provide FMIS and other systems the capability of using controlled vocabularies (e.g., pest lists, phenological scales, and/or other products of university and government research and extension) without a lot of overhead: define the variable in data (as opposed to code), put it on the shared infrastructure (AgGateway provides a simple, transparent governance process for doing this), and *all* of the ADAPT-aware systems throughout the world become instantly able to collect and display this information. Hopefully this will translate into greater operational use of research work, and greater collaboration between academia, government, and the private sector.

ADAPT is a Cooperative Endeavor

In addition to the collaborative work of multiple companies within ADAPT, the ADAPT team operates in the context of continuing cooperation between AgGateway and relevant standards organizations such as the Agricultural Industry Electronic Foundation (AEF), which implements the ISO 11783 standard, and the American Society of Agricultural and Biological Engineers, which holds the United States' *TAG* (Technical Advisory Group) role for the ISO 11783 standard; close collaboration with these organizations allows the ADAPT team to provide feedback to ISO 11783's ongoing development process.

Collaboration is not restricted to North America: there are multiple European contributors to ADAPT, some of whom are using it in the context of the European Union's project of Internet of Food and Farm, IOF2020 (www.iof2020.eu). There are also active users in Japan, and expressions of interest from other parts of the world.

CONCLUSIONS

The agriculture industry's need for greater data interoperability motivated AgGateway field operations projects such as SPADE and PAIL. These projects elicited knowledge from subject matter experts, and provided a set of requirements that in turn motivated the creation of an open-source project, the Agricultural Data Applications Programming Toolkit (ADAPT).

ADAPT consists of a common object model and tools (software libraries called "plug-ins") to interconvert between that model and other formats. Plug-ins can be licensed and distributed as needed by their authors. The common object model is distributed under the Eclipse public license, as are two community-supported plug-ins, one used for ISO 11783-10 format conversion, and the other to enable lossless FMIS-to-FMIS communication.

ADAPT uses established best practices for managing versioning and code branching, as well as transparent governance. Although the governing body is a committee within AgGateway, code contributions are welcome from AgGateway members and nonmembers alike.

ADAPT is now in the hands of the agriculture industry and open-source community. The direction and growth of the code will in large part depend on enhancements contributed by participants, driven by their individual business needs. Contributions that serve the needs of one company often serve the needs of many, and project participants have welcomed the contributions of their peers.

ADAPT's future is bright. The broad industry support, vibrant community, comprehensive data model, marketing and communications activity, and growing global awareness suggest that the tipping point was reached some time ago. That said, there is still much work to do.

REFERENCES

AgGateway (2018). *ADAPT Branching and Versioning Policy*. https://aggateway.atlassian.net/wiki/x/WZj6CQ Accessed 14 May 2018.

AgGateway (2017*). Identification and Record Linkage* http://aggateway.org/eConnectivityActivities/Committees/ADAPTOversight/IdentificationandRecordLinkage.aspx Accessed 14 May 2018.

AgGateway (2016). *Core Documents for Field Operations: The Foundation for Efficient Communication in Precision Agriculture*. http://s3.amazonaws.com/aggateway_public/AgGatewayWeb/About%20Us/CommunicationsKit/AgGateway_core_documents_72616.pdf Accessed 14 May 2018.

AgGateway (2016A). *The ADAPT Toolkit: Implementing Interoperability in Precision Agriculture*. https://s3.amazonaws.com/aggateway_public/AgGatewayWeb/News/CommunicationsKit/AgGateway_ADAPT_Toolkit_103117.pdf Accessed 14 May 2018.

Applegate, D.B., Berger A.W., Berne, D.T., Bullock, R., Craker, B.E., Daggett, D.G. (2016). *Toward Geopolitical-Context-Enabled Interoperability in Precision Agriculture: AgGateway's SPADE, PAIL, WAVE, CART and ADAPT*. A paper from the Proceedings of the 13th International Conference on Precision Agriculture, July 31 – August 4, 2016, St. Louis, Missouri, USA

Atlassian (2015). *Confluence 101: Getting started in Confluence*. https://www.atlassian.com/dam/jcr:438a8cda-b614-4af8-a6c5-2a3fc949b4a6/confluence-101-getting-started-in-confluence.pdf Accessed 14 May 2018.

Atlassian (2015A). *Jira 101*. https://confluence.atlassian.com/jira064/jira-101-720412861.html Accessed 14 May 2018.

Chacon, S. and Straub, B. (2014). *Pro Git, 2nd ed.* https://git-scm.com/book/en/v2 Accessed 14 May 2018.

Daggett, D.G., Ferreyra, R.A., Reddy, L.T., Rhea, S.T., and Tevis, J.W. (2016). *Filling in the blanks with ContextItems: a lightweight method for extending field operations object models*. ASABE Paper No.2462418. ASABE, St. Joseph, MI.

Driessen, V. (2010). A successful Git branching model. http://nvie.com/posts/a-successful-git-branching-model/ Accessed 14 May 2018.

Eclipse Foundation (2003). *Eclipse Public License – v 1.0*. http://www.eclipse.org/legal/epl-v10.html Accessed 14 May 2018.

Github.com (2016) *Hello World*. https://guides.github.com/activities/hello-world/ Accessed 14 May 2018.

Google (2017). *Protocol Buffers Developers Guide*. https://developers.google.com/protocol-buffers/docs/overview Accessed 14 May 2018.

IETF (2017). *The JavaScript Object Notation (JSON) Data Interchange Format*. https://tools.ietf.org/html/rfc8259 Accessed 14 May 2018.

International Organization for Standardization. (2018). *Snapshot of the ISO 11783-11 online data base*. https://www.isobus.net/isobus/site/exports?view=export Accessed 14 May 2018.

International Organization for Standardization. (2017). *ISO 11783-1:2017 Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 1: General standard for mobile data communication*. International Organization for Standardization, Geneva, Switzerland.

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.

International Organization for Standardization. (2012). *ISO 11783-2:2012 Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 2: Physical Layer*. International Organization for Standardization, Geneva, Switzerland.

International Organization for Standardization. (2011). *ISO 11783-11:2011 Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 11: Mobile Data Element Dictionary.* Organization for Standardization, Geneva, Switzerland.

International Organization for Standardization. (2011A). *ISO 19156:2011 Geographic information -- Observations and measurements*. International Organization for Standardization, Geneva, Switzerland.

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.

Johnson, Mark (2013). *What is a pull request*?. http://oss-watch.ac.uk/resources/pullrequest Accessed 14 May 2018.

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 Accessed 14 May 2018.

Mapshots (2013) *Field Operation Device Drivers*. https://www.mapshots.com/_fodm/#CNH Accessed 14 May 2018.

Nuget.org (2018). *An introduction to NuGet.* https://docs.microsoft.com/en-us/nuget/what-is-nuget Accessed 14 May 2018

Preston-Werner, T. (2013) *Semantic Versioning 2.0.0* http://semver.org/spec/v2.0.0.html Accessed 14 May 2018.

Travis CI (2018). *Getting started*. https://docs.travis-ci.com/user/getting-started/ Accessed 14 May 2018.