The Critical Next Step for Interoperability: Designing and Implementing Interfaces between Standards

Gary McNaughton
Cornice Engineering
Flagstaff, AZ 86001, USA
gmcnaughton@CorniceEngineering.com

Linda Rankin
QualityLogic
Portland, OR 97205, USA
lrankin@qualitylogic.com

James Mater
QualityLogic
Portland, OR, 97229, USA
jmater@qualitylogic.com

Keywords: Interoperability, Inter-standard mapping, MultiSpeak®, OpenADR, smart grid standards

Abstract

While conformance and interoperability of products adhering to a specific standard is a critical building block for smart grid systems, it is likely that deployment will require interfaces between products adhering to two or more standards. How such inter-standard interfaces are evaluated and the steps to ensure a standardized inter-standard interface is the subject of this paper. A recent project by the MultiSpeak® Initiative mapped the use cases and associated functions between the MultiSpeak [1] and OpenADR 2 (Open Automated Demand Response) standards based on business processes [3][1]. The project identified overlaps and gaps in the targeted functions and provides a guide for developers who are required to include such interfaces in their system implementations.

The development of the methodology used to map the demand response functions between two standards is consistent with other industry mapping efforts and contributes to the general methodology for undertaking such inter-standard interface analyses. This paper describes the methodology in depth, demonstrating its application and positive results.

An important result of this work is the knowledge that the functions, methods and data objects contained in MultiSpeak Version 4.1.5 are sufficient to send demand response and critical peak price events to an interface that implements the OpenADR 2.0a profile.

1. BACKGROUND

The MultiSpeak standard is an initiative of the National Rural Electric Cooperative Association (NRECA) that standardizes interfaces between enterprise applications commonly used in electric power utilities for distribution management. The OpenADR 2 standard was developed to facilitate automated demand response actions at the customer location including load shedding or load shifting. Management of electric load can be used to improve grid reliability and assist in the integration of renewable electricity generation sources (such as wind power). MultiSpeak provides standardized interfaces for load management applications within the utility domain; OpenADR 2 provides the methods and services for the utility to manage load in the consumer domain.

The starting point for the methodology is the understanding of each of the standards to a depth that permits appropriate business cases to be identified for mapping of the functions and data elements.

1.1. MultiSpeak Overview

In order to accomplish the exchange of data among enterprise application software commonly applied within utilities, the MultiSpeak Specification standardizes the interfaces between abstract software functions. These functions can then be combined to create various enterprise software applications. The MultiSpeak specification provides:

- **Definitions of common data semantics**. In MultiSpeak, data semantics are documented in the form of an extensible markup language (XML) schema.
- **Definitions of message structure (syntax)**. In MultiSpeak Version 4.1, the XML-formatted data payload is carried as part of a web services call for

[1] This material is based upon work supported by the Department of Energy under Award Number **DE-OE0000222**. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof

real time exchanges and as part of a batch file for off-line transfers. MultiSpeak messages consist of one or more of the following three parts: (i) one or more defined data objects (considered to be nouns), (ii) actions to be taken on those data objects (called data object verbs), and (iii) messaging components.

- **Definition of which messages are required to support specific business process steps**. Web services method calls are linked together to accomplish each potential step in a utility business process.

The MultiSpeak Specification in total consists of (i) a data model documented in Unified Modeling Language (UML) class model and Extensible Markup Language (XML) schema formats which includes data objects, interface definitions, and message structures, (ii) service definitions defined in Web Services Description Language (WSDL) contracts, (iii) schema documentation in hypertext markup language format which describes the schema, (iv) implementation guidelines documents, (v) use cases describing business processes addressed by MultiSpeak, and (vi) a specification document.

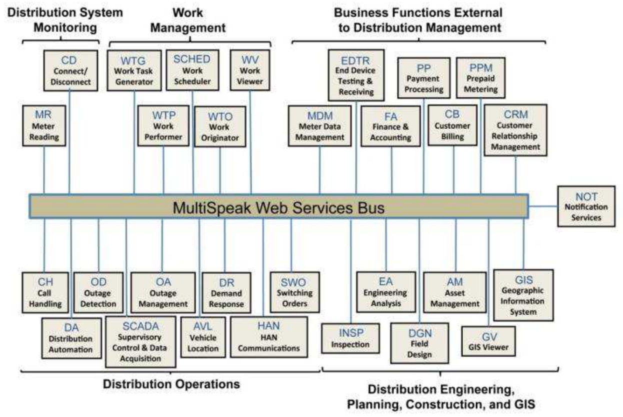

Figure 1 MultiSpeak Version 4.1.5 Reference Architecture showing supported abstract software functions.

Any given piece of application software can implement one or more of the abstract functions shown in Figure 1 as appropriate. In some cases, such as a geographic information system (GIS), the application likely would implement only a single function, the GIS server. In other cases, an enterprise application might implement many abstract functional capabilities. For instance, an AMI system would implement a meter reading (MR) server and might also implement connect/disconnect (CD), outage detection (OD), demand response (DR), home area network communications (HAN), distribution automation (DA), and/or prepaid metering (PPM) servers.

Note that each physical software application, for instance AMI, would be a single actor, despite the fact that it might implement one or more abstract MultiSpeak software functions when represented in the MultiSpeak enterprise service bus representation.

Each of the software functions of an application is physically implemented using a Web service endpoint that uses the MultiSpeak-defined data objects and service definitions along with Web standards and protocols. Thus, a single application might implement one or more distinct Web service endpoints. This approach facilitates modular development. Each version of MultiSpeak is deployed in its own namespace, making it possible for a single application to implement interfaces that support a number of different versions of MultiSpeak.

Figure 1 shows the abstract representation of application interconnectivity, labeled “MultiSpeak Web Services Bus.” Physical implementations at a utility could be simply point-to-point interconnections between Web service endpoints, or could be a more complex middleware implementation such as an enterprise service bus, depending on the needs of each utility. If an application exposes a web service, it is available for any other application in the enterprise network to use. This provides much flexibility in adding applications that may be required for mapping from MultiSpeak to different standards and their associated protocols.

Finally, it is important to note that each of these functions might be instantiated by one or more applications in the enterprise. For instance, messages might be generated from a customer billing application, a critical peak alerting system, or another system seeking to get information to the customer. Each such system would need to exhibit the same services. As a result, abstract functional definitions have been developed that can be concretely implemented in numerous systems at the software design phase.

In addition to the methods defined specifically for each function, several generic web service methods are defined and are used for network management and discovery. For example *GetMethods* allows for an application to query another application’s web service to obtain a list of MultiSpeak-compliant methods that it supports.

1.2. OpenADR Overview

OpenADR is an application layer message exchange specification used for two-way communication of Demand Response (DR), price, and Distributed Energy Resource (DER) signals between the electricity service provider and its customers. The OpenADR Alliance is developing a number of profile subsets of the OASIS Energy Interoperability 1.0 standard; the first of these subsets is OpenADR 2.0a. This standard provides an open, standardized DR interface that allows communication of DR signals using a common XML-based payload on existing communication infrastructure, such as the internet. The OpenADR 2.0 standard is based on other Smart Grid standards:

- OASIS Energy Interoperability v1.0
- OASIS Energy Market Information Exchange v1.0
- OASIS WS-Calendar v1.0
- IEC Common Information Model

In the Service Provider/Aggregator domain where OpenADR is deployed[2], there are two main entity types that a particular device can represent: a Virtual Top Node (VTN) that can initiate a DR event, or a Virtual End Node (VEN) that can participate in a DR event. Generally in an interaction, the VTN acts as the server, providing information to the VEN, which in turn responds to the information. The response may be to reduce power to some devices, or it could also propagate the signal further downstream to other VENs. In this case, the VEN would become the VTN for the new interaction. OpenADR 2.0 allows for interconnection of these types of nodes in a connected network, but communication is always between VTNs and VENs (peer-to-peer communication is not supported).

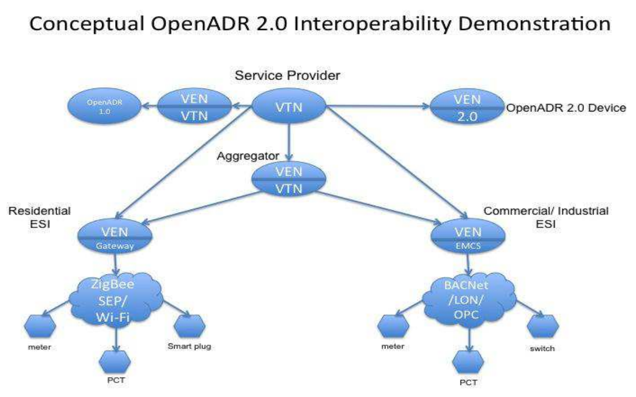

Figure 2 Conceptual Diagram of VEN/VTN node topology illustrating VEN/VTN relationships.

Communication between the VEN/VTN uses standard internet protocols such as HTTP, and a common data model is described using XML schema. The VTN can be a service provider such as a utility, and the VEN could be a gateway to a HAN or Energy Management Control System.

OpenADR was initially developed to reduce peak loads in response to "event-based" signals. This Demand Response (DR) goal is represented by the "simple" or "OpenADR 2.0a" profile that is targeted towards low power devices. Extensions to the standard to support more robust devices and the wholesale space (ISOs) will result in "2.0b" profile. The OpenADR 2.0 profiles are a subset of the OASIS Energy Interoperability Standard.

It should be noted that OpenADR was originally designed considering building HVAC and lighting controls. As a result OpenADR assumed an intelligent energy controller (energy management system) would be present and would allow, for example, for ramp-up/ramp-down capability in demand response profiles. In contrast neither MultiSpeak (nor the ZigBee Smart Energy Profile (SEP)) take this requirement into account, and both make the assumption that demand-responsive loads are on/off loads with limited local intelligent control in place. In the case of HVAC loads, for example, demand response in SEP and MultiSpeak are of the type "set the thermostat back 10 degrees" rather than "go to load profile ABC".

2. METHODOLOGY

The focus of the project was to develop a comprehensive mapping between the MultiSpeak Version 4.1.5 Specification and OpenADR (Open Automated Demand Response) 2.0a. The mapping of functions was developed by:

- Identifying the common business processes supported by MultiSpeak and OpenADR.
- Specifying the use cases needed to achieve the goals of these business processes.
- Identifying the available MultiSpeak messages and payloads for those messages to provide the data exchanges needed by OpenADR-enabled applications.
- Identifying the corresponding OpenADR services and payloads.
- Identifying the overlaps and gaps in the functionality between MultiSpeak and OpenADR.

The analysis was based on the following assumptions:

- An adaptor application, DRMS, maintains state regarding the active and pending events for the OpenADR domain. This state is updated and managed through the MultiSpeak methods for initiating and cancelling events. The DRMS, for example, may be an OpenADR application that has implemented an interface to MultiSpeak compliant applications.
- The events originate in the MultiSpeak domain and that the DRMS will operate as a VTN for the downstream traffic.

[2] This is referred to as the OpenADR domain in the remainder of this paper.

Mapping of functions and data flows show how both the Initiate and Cancel methods of MultiSpeak for demand response and critical peak price events can be mapped to corresponding request/response flows in OpenADR. Flows for both the OpenADR PUSH and PULL methods were developed.

3. DETAILED METHODOLOGY

The following sections of this paper detail the methodology used and demonstrate its application to the MultiSpeak-OpenADR interface.

3.1. Business Processes

The mapping of MultiSpeak to OpenADR is based on identifying the end-to-end business processes supported by the functionality of both standards. Business processes can then be further expanded to individual flows or use cases that are needed to implement them.

MultiSpeak provides standardized interfaces for load management applications within the utility enterprise (both distribution utilities and vertically integrated utilities); OpenADR provides the methods and services for the utility to manage consumer loads for integration of renewable energy, grid reliability and energy savings. In this mapping project, the utility business processes that would be enabled by systems where OpenADR is deployed are:

1. The utility manages its demand response resources by distributing events to customers with responsive assets. Customers may choose to participate in one or more events.
2. The utility distributes/updates critical peak price events to customers enrolled in a program. Customers may choose to participate in one or more events.

3.1.1. Actors and Domains

Actors and domains are abstractions that are used to illustrate the business process use cases. The eventual implementation may vary, although the flow of data is expected to be the same.

MultiSpeak is used within the utility enterprise as a means to standardize the interface between utility enterprise applications. OpenADR is a protocol that is used to communicate demand response events between the utility and the customer. As such, each standard is used to interface to different applications in a separate operational domain. In the MultiSpeak domain, the actor selected for these examples is a demand management application that is responsible for managing the demand response programs within the utility.

The actor in the OpenADR domain would be a system that presents a VEN interface to the utility. This application may be one implemented by an aggregator (that in turn manages downstream events as a VTN), or it may be the application used by a gateway to a Home Area Network (HAN) or an Energy Management Control System. This actor is the OpenADR client, or VEN, and is completely contained within the OpenADR domain.

To bridge between the two domains, a third actor needs to be introduced; an adaptor application called the Demand Response Management System (DRMS). This application interfaces to the demand management application in the MultiSpeak domain, and then acts as a VTN in the OpenADR domain. This actor contains all of the communication and transport layer functionality to manage each interface independently as well as the ability to transform the data and information from MultiSpeak to OpenADR (and vice versa).

Figure 3 provides a schematic representation of the actors and domains of interest.

It is also assumed that the DRMS maintains the state information needed in order to map the services and functionality from one domain to the other. For example, in the MultiSpeak domain, demand response events are *initiated* or *canceled*. In the OpenADR domain, once an event has been created, it remains *active* or *pending* until it has expired or it is *canceled*. The DRMS/VTN is the entity that maintains relevant state information regarding events in both domains as well as performing the logical translation between the two.

This model assumes that events originate in the MultiSpeak domain, and that the utility is not acting as an OpenADR aggregator and receiving events from an upstream VTN. This is consistent with the usage model for the OpenADR 2.0a profile.

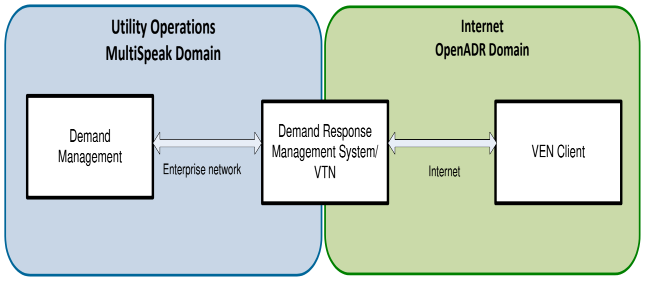

Figure 3 MultiSpeak/OpenADR domains and primary actors

To summarize, the analysis is based on the following assumptions:

- An adaptor application, DRMS, maintains state information regarding the active and pending events for the OpenADR domain. This state is updated and managed through the MultiSpeak methods for initiating and canceling events. The DRMS, for example, may be an OpenADR application that has implemented an interface to MultiSpeak compliant applications.

- The events originate in the MultiSpeak domain and the DRMS will operate as a VTN for the downstream traffic.

The challenge then is to ensure that MultiSpeak contains sufficient capability to provide the data required by OpenADR-enabled applications to implement their desired functionality.

3.2. Use Cases and Sequence Diagrams

For each business process a set of use cases and their corresponding sequence diagrams have been developed to illustrate the operations and transfer of information between each of the actors in each domain. Along with the mapping of business processes to use cases, the MultiSpeak methods and objects and OpenADR services and payloads that would be used in implementing the use case were developed. These provide the framework for the flow diagrams along with the XML mapping tables for each of the use cases.

More tables and detail is provided in the QualityLogic report published by the MultiSpeak Initiative [3]. The Report includes:

- Sequence diagrams for the use cases that show the flow of information and data from one domain to the other.
- For each object used in the flows, an XML mapping between the elements of one domain to the other is shown. The criterion for mapping an object is if the information in the payload is relevant to the functionality being performed in the domain being mapped. For example, a mapping of the oadrRequest event payload to a MultiSpeak object was not required because it is handled solely by the VTN and does not require information or data to be exchanged directly with an application in the MultiSpeak domain.
- XML diagrams showing the OpenADR objects that are being mapped to are in Appendix A, and the MultiSpeak objects are shown in Appendix B of the report.

3.2.1. Use Cases

In the investigation of the two business processes that are facilitated by mapping of MultiSpeak to OpenADR, the following Use Cases were identified for each:

Utility manages demand response event to customer demand response resource(s).

- Utility issues demand response event to customer demand response resource(s) (PUSH Method)
- Utility cancels active or future demand response event (PUSH Method)
- Utility modifies demand response event (PUSH Method)
- VEN requests list of active events (PULL model)

The utility manages price signals to customers, who decide how to respond using their demand resources.

- Utility issues critical peak price event (PUSH Method)
- Utility cancels active or future critical peak price event (PUSH Method)
- Utility modifies critical peak price event (PUSH Method)
- VEN requests list of active events (PULL model)

3.2.2. Messages, Objects, Services and Payloads

For each use case, the specific MultiSpeak messages and OpenADR services used to implement the needed functionality in each domain have been identified. For example, in the use case where the utility issues a demand response event to a customer, a DemandResponseNotification message originates in the MultiSpeak domain, and is transformed into a corresponding oadrDistributeEvent message using the EiEvent service in the OpenADR domain. The information that is exchanged between the two domains is also determined by the MultiSpeak objects used in the messages, and the OpenADR payloads. For instance, the MultiSpeak messages/objects and OpenADR services/payloads for the use case

"*Utility issues demand response event to customer demand response resource(s) (PUSH Method)"* are*:*

- MultiSpeak Message: DemandResponseEventNotification; InitiateDemandResponseEvent; InitiateDemandResponseEventToGroup
- MultiSpeak Object used in the Message Payload: demandResponseEvent; demandResponseEventStatus
- OpenADR Service: EiEvent
- OpenADR Payload: oadrDistributeEvent; oadrCreatedEvent

Similar Messages, Objects, Services and Payloads have been identified for the other use cases.

3.2.3. Mappings of Data Elements

The next step was to develop mapping tables between the XML data elements of the MultiSpeak objects and OpenADR payloads. Mapping from one standard to another is not always direct, and a set of mapping descriptors were defined for the tables. The following mapping terms were defined:

- Direct: A mapping can be done directly from one schema element to the other. A transformation from one format to another may need to be performed (i.e., integer versus floating point).
- Derived: The value or content of the destination element can be derived from other information, such as the function name, other elements, etc. Examples of recommended values or algorithms may be provided.
- NA: The element being mapped is specific to the standard, such as protocols used in services for tracking messages, or does not apply to the specific use case.
- Computed: The data element can be computed (elements such as timestamps would fall under this category). UsageGap: Required to implement the functionality in the standard that is being mapped if it is not directly available and cannot be computed or derived.
- ExtGap: The standard being mapped has additional features that are not currently supported by MultiSpeak usage models or data objects and cannot be computed or derived.

As a result, the mapping table analysis shows how elements are mapped, along with identifying current gaps in data content or areas of future development.

For instance, in the mapping of DemandResponseEvent to oadrDistributeEvent.oadrEvent.eiEvent.eventDescriptor, the <objectID> element in the MultiSpeak message could be used directly for the <eventID> in the OpenADR eventDescriptor. In the mapping table, the mapping field for these elements would indicate "Direct".

As another example, the OpenADR <eventStatus> is used to indicate if the event is far, near, active or canceled. In the mapping table, this element is derived from the MultiSpeak element <eventStartTime>, and if the MultiSpeak message is used for initiating or canceling an event. In this case, the mapping is "derived" and the algorithm is described.

4. GAP ANALYSIS

There are two ways to categorize gaps in this analysis. The first is identifying those gaps where *existing* MultiSpeak methods and data objects are not sufficient in providing the information such that the same functionality can be supported using OpenADR. These are called *Usage Gaps*.

The second set of gaps are those that have been identified when the objective is to determine if OpenADR has additional functionality or content that would be of value to import to MultiSpeak. Addressing this set of gaps would provide the functionality such that MultiSpeak could support the full range of capabilities in OpenADR. These are called *Extension Gaps.*

Usage Gaps

The functions, methods and data objects in MultiSpeak Version 4.1.5 are sufficient in sending demand response and critical peak price events to an interface that implements the OpenADR 2.0a profile. No critical usage gaps were identified. However, some of the elements that are present in the MultiSpeak demand response event object have no counterpart in OpenADR.

Extension Gaps

There were several extension gaps identified in the mapping of the MultiSpeak data objects and functions to the OpenADR eiEvent services.

OpenADR supports the ability to "modify" events. MultiSpeak currently only supports the verbs of "Initiate" and "Cancel" for the operations on the data objects. A flow was described that allows for modification of events using the capabilities present in MultiSpeak Version 4.1.5.

OpenADR event signals support more than one interval. This allows more control points over the load. For example, a price signal may have three intervals, one where the value is low, medium and then high. The current definition of DR and critical peak price events in MultiSpeak are limited to one interval and therefore one value over the duration of the event.

OpenADR eiEvent includes attributes that define notification duration; ramp-up and recovery times for the event for examples as to how they relate to the event. These periods do not exist in the MultiSpeak event objects.

A critical peak price event by definition implies that the price for the event is at the highest level. The EiEvent <signalPayload> attribute allows for events to have 4 values (normal, moderate, high or special). The event objects in MultiSpeak do not provide for representing other values for price besides "peak".

5. SUMMARY AND CONCLUSIONS

The mapping of functions between the MultiSpeak and OpenADR 2 standards are analyzed and presented using a use case methodology based on business processes. Use cases and their corresponding sequence diagrams provide a means to represent detailed and complex flows required to implement a specific business process. Where gaps in MultiSpeak coverage are identified, they are documented for future action. The result is that a comprehensive description of how MultiSpeak maps to OpenADR 2 as well as a quick reference and guide to the relevant MultiSpeak methods within the MultiSpeak specification are now available to developers and system implementers.

The results of this work indicate that the functions, methods and data objects contained in MultiSpeak Version 4.1.5 are sufficient to send demand response and critical peak price events to an interface that implements the OpenADR 2.0a profile. No critical usage gaps (gaps where a particular OpenADR functionality cannot be supported by MultiSpeak) were identified. Two minor usage gaps and several extension gaps (capabilities in OpenADR that would enrich MultiSpeak if introduced into the MultiSpeak specification) were noted. It was determined that both the minor usage gaps and all extension gaps can be handled through the use of existing extension mechanisms in MultiSpeak Version 4.1.5 and dealt with by minor changes to the functionality of MultiSpeak in future Versions.

The methodology developed for this project can serve as a model for other mapping efforts.

5.1. References

[1]. Additional information about the MultiSpeak Initiative and Specification is available at http://www.multispeak.org.

[2] Additional information about OpenADR is available at http://www.openadr.org/.

[3] "Function and Data Mapping: MultiSpeak® and OpenADR 2.0", DRAFT, Version 1.0, September 24, 2012, prepared by QualityLogic for NRECA CRN Smart Grid Regional Demonstration Project and National Rural Electric Cooperative Association.

6. BIOGRAPHY

Gary A. McNaughton is the Vice President and Principal Engineer for Cornice Engineering, Inc. He received a B.S.E.E. degree from Kansas State University in 1976 and an M.S.E.E. degree from the University of Colorado in 1980. Prior to joining Cornice in 1995 he worked as a Plant Electrical Engineer for Union Carbide, at the Oak Ridge Gaseous Diffusion Plant, at Oak Ridge, TN, as a Transmission Planning and Protection Engineer for Colorado-Ute Electric Association, a generation and transmission cooperative, located in Montrose, CO, and as Staff Engineer, Manager of Engineering, and Assistant General Manager for Engineering and Operations for La Plata Electric Association, in Durango, CO. Mr. McNaughton currently serves as the Technical Coordinator for NRECA's MultiSpeak® Initiative. Mr. McNaughton is a registered professional engineer in the States of Colorado and Arizona.

James Mater founded and has held several executive positions at QualityLogic Inc. from June 1994 to present. He is currently Co-Founder and General Manager, Smart Grid, working on QualityLogic's Smart Grid strategy, including work with GWAC, the Pacific Northwest Smart Grid Demonstration Project, and giving papers and presentations on interoperability. From 2001 to October, 2008, James oversaw the company as President and CEO. From 1994 to 1999 he founded and built Revision Labs which was merged with Genoa Technologies in 1999 to become QualityLogic. Prior to QualityLogic, James held Product Management roles at Tektronix, Floating Point Systems, Sidereal and Solar Division of International Harvester. He is a graduate of Reed College and Wharton School, University of Pennsylvania.

Linda Rankin is a Senior Test Architect for QualityLogic and is the Company's technical lead for the Pacific NW Smart Grid Demonstration Project. She has been an Assistant Professor and Research Scientist at Maseeh College of Engineering and Computer Science of Portland State University, teaching and developing curricula pertaining to Smart Grid digital technologies. Until 2008, she was a Principal Engineer at Intel Corporation and has more than 20 years experience as a system architect working applied research and advanced product development in the areas of networking, parallel processing, server platforms, and microprocessors. She is a graduate Lewis & Clark College, and Oregon Graduate Institute. Linda has authored papers for technical journals, is a Senior member of IEEE, and holds more than 25 patents or pending patents, many of which are international.