Utility Integration Bus (UIB) White Paper

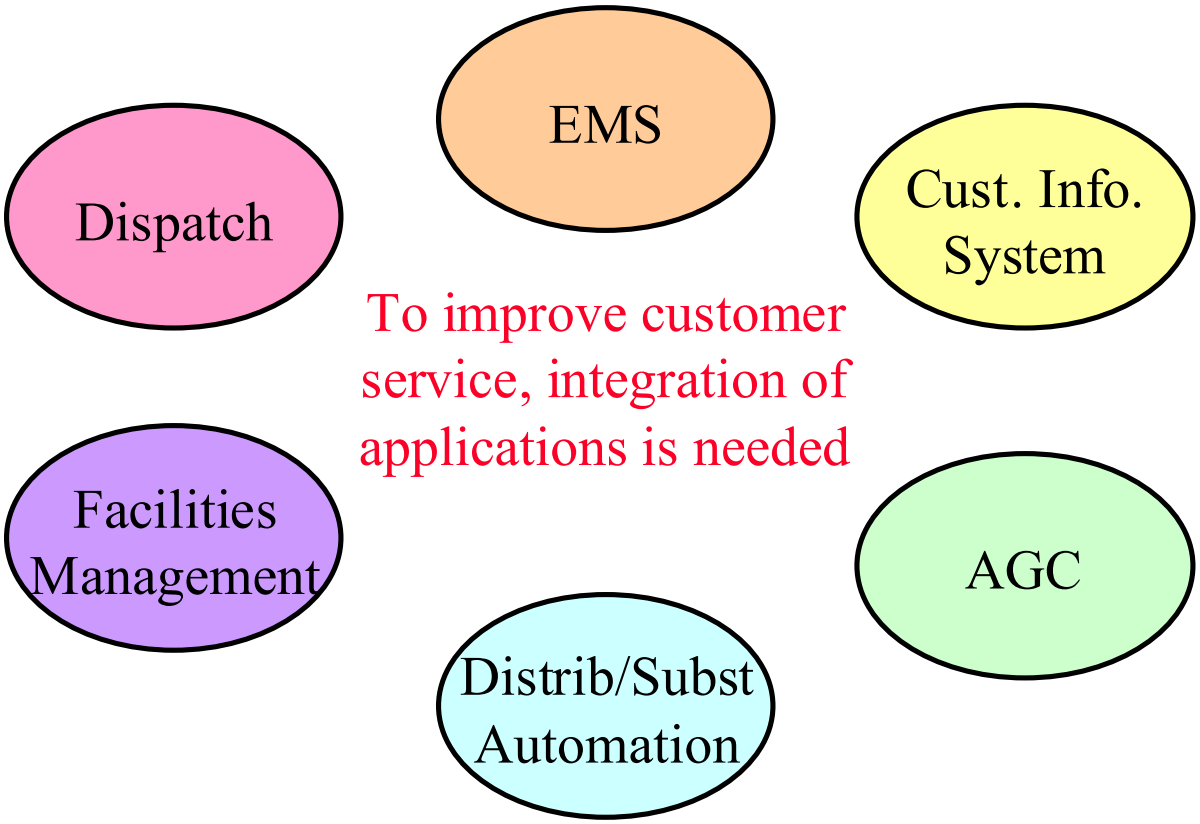
INTRODUCTION

Do you have a vision for a utility which can make decisions based on all the information available to it? Do you have a vision for a utility that is more responsive to the needs of customers, shareholders and employees? This vision has become a requirement in a world where deregulation, rapidly fluctuating energy markets and demands for more efficient operation force utilities to become more flexible. However, utilities can not adapt to changing business conditions without having complete control of operational systems and a flexible infrastructure that supports change. What is needed is a framework that enables data to be coalesced and transformed in to usable knowledge. This framework should be robust enough to that as new situations arise, information can be shared in ways not previously anticipated. SISCO's Utility Integration Bus (UIB) provides an integration framework which solves many of these issues while controlling complexity. This paper will discuss the principles and concepts of enterprise integration, the UIB, the relationship of the UIB with standardization efforts in EPRI and IEC.

THE UTILITY INTEGRATION PROBLEM

Today's utility IT environment is truly heterogeneous. Some of the more significant features of this mix include:

- Many computing hardware platforms
- Many operating systems
- Mainframe/client-server/web-based systems
- Many component technologies (CORBA, DCOM, Enterprise Java Beans)



Furthermore, the utility of today has several sources of "real-time" process oriented information that allows the control and monitoring of key devices, measurements, and subsystems. These include: Supervisory Control and Data Acquisition (SCADA) systems, Energy Management Systems (EMS), Distribution Facilities Management Systems (DFMS), Automatic Meter Reading (AMR) systems, and other sources of "real-time" information. Several of these systems may need to exchange information with business applications and desktops or with other process oriented systems. For example within the control room, an EMS obtains power system information from the SCADA system and then commands the SCADA system to perform control on its behalf. Availability of process information outside the control room would allow further integration of operational data into business applications such as Work Management Systems (WMS). A WMS contains information about workorders or asbuilt designs that would be valuable to dispatchers in the control room. WMS and other systems like it typically employ databases that have been configured with customer and SCADA system information— any or all of this information may be valuable to another application.

Outage Management is another application that combines field device system connectivity, customer information, and real-time or historical information. Linking this application to SCADA is difficult enough. When one adds the requirement to link Outage Management with Automated Meter Reading (AMR), Distribution Automation (DA) devices and Substation

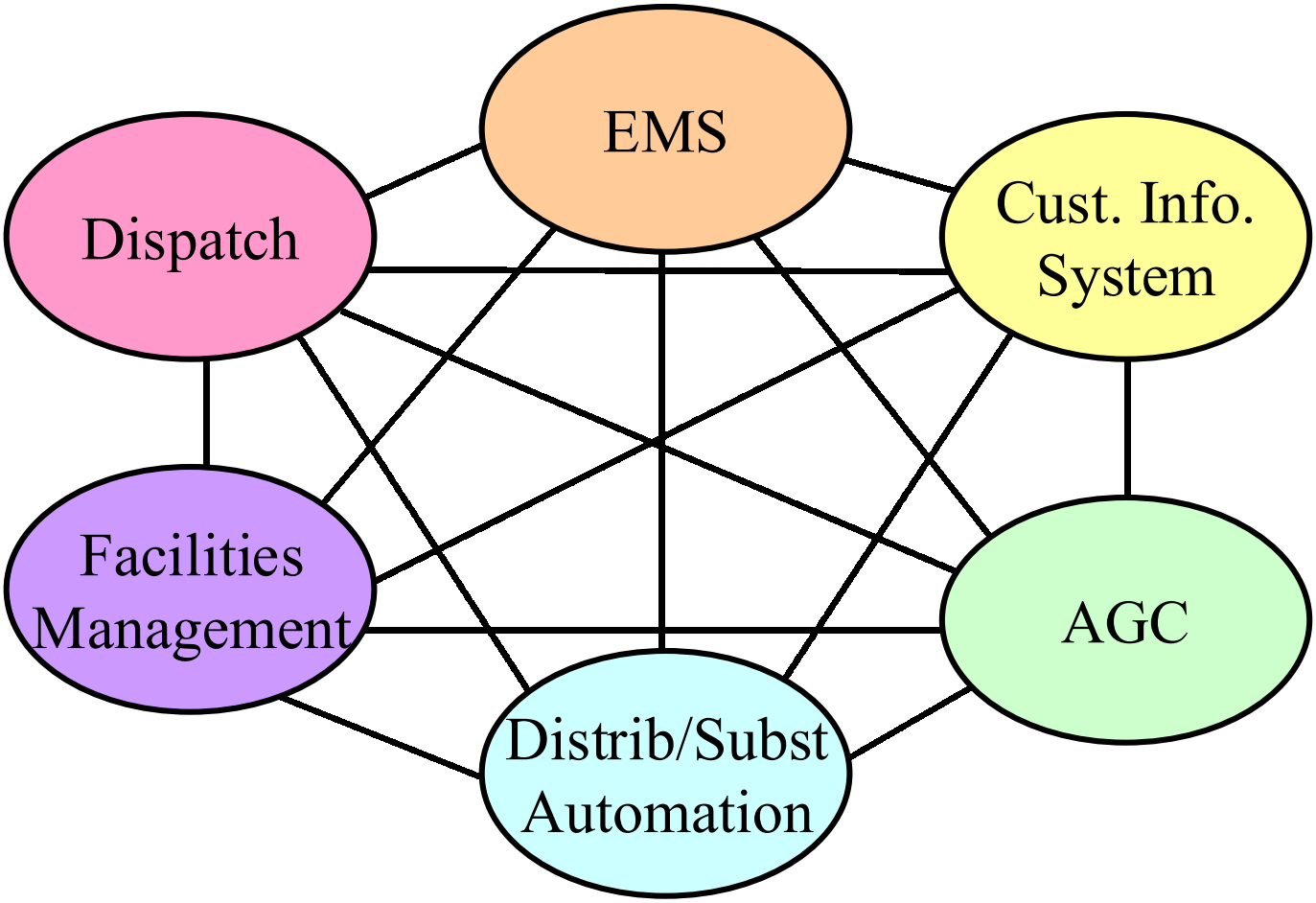
Automation (SA) devices, workorders and work history, the task is daunting. If a utility is to benefit fully from its investment in these technologies, the data from these systems must be made available.

Bill generation is another example of a business application where there is a potential need to merge process and non-process information. In this case, the customer database will need to be supplied with the process data made available from the AMR system.

Integration between systems are typically manual, and if not manual, are expensive (or labor intensive) to put in place. Each integration application in turn is different from all the other integration applications. Over time, the lack of standards results in a software management nightmare. While previously there has been no standard way to handle these types of integration problems, utilities have integrated applications. These non-standard methods include:

- The Buy Everything from One Source Approach - buy a system or subsystem from a single vendor with turn-key responsibility. The benefits of working with a single vendor are in minimizing the point of contact, less opportunity for miscommunications, and one source for accountability. The problem of working with only a single vendor is each vendor has it's own proprietary way of doing things; whereby, replacement, rather than upgrade, is the only option for system improvement at a later date. It is also rare that one vendor has the knowledge or experience in understanding all the components of a complex implementation and so will install component applications with little knowledge of the long-term ramifications.
- The "Kitchen Sink" Approach - everything federated into a single or multiple databases. There are many reasons to not store all data into a single database. First, not all data are efficiently stored in a single database. Process (real-time/temporal) data are not efficiently stored in a transactional (relational) database and model/configuration information is not stored efficiently in a temporal database. Second, specific users of differing types of data reside in a variety of groups around the organization. Users interested in Outage Management are not the same users who are make updates to map drawings. It does not make technical sense to have these groups working in the same database, however, it is important that the two group can access one another's data if needed.
- The Apply Glue as Needed Approach - development of point to point information links and gateways as needed. While this solves the short term problem of linking those particular "Islands of Automation", these types of solutions never establish a platform for obtaining an enterprise wide view of data. Integration techniques that do not facilitate future business intelligence or data ware house applications just create more "Islands of Automation".

Most frequently, utilities have chosen a mix of these short terms solution which as resulted in many point to point links as shown in the diagram below.



Successful integration of a utility's various systems requires a method that does not require existing applications to be disturbed. Recently, a new type of middleware has emerged whose sole function is to link legacy applications via the exchange of messages. This type of software requires building wrappers for existing applications and can provide a standardized way to perform integration. In general, they are responsible for exposing the functionality of the wrapped application in a way that is compatible with a common methodology. They also perform translation between object models. The most significant benefit of this approach is that it is technology neutral and that legacy applications do not need to be rewritten.

One problem with using type of approach is that adapters can become large. A significant amount of code is required to abstract away different calling conventions, design assumptions, data models, and mismatches in assumptions about business processes. As the size of adapters increases, so does the cost of writing and maintaining them.

Another problem with this type of software is that it only provides tools for inter-application communication and does not facilitate the creation and management of common data semantics. Thus while integrating existing applications using these tools solves operational problems, in fact the real value of doing integration can be that it facilitates the creation of higher level data analysis applications. One needs to not only create an communication infrastructure to automate

the exchange of data, but also to establish common data semantics. In this way data in existing systems can be can become shared knowledge.

UIB DEVELOPMENT

Several initiatives started in the mid-1990's have begun to address the need for a standardized method to exchange information. A couple of these are: the Electric Power Research Institute's (EPRI's) Control Center Application Programming Interface (CCAPI) project which includes the development of the Common Information Model (CIM); a National Rural Electrical Cooperative Association (NRECA) Inventory Management system, the EPRI Customer Interface project known as CS2000, and the Kansas City Power and Light (KCP&L) control center integration project. Fundamentally, these initiatives have focused on the same basic issue of integrating process information with corporate information using standardized and extendable methodologies.

While these US projects have progressed, the need for a standard integration technique has also been acknowledged within international standards bodies such as the International Electrotechnical Committee (IEC) Technical Committee (TC) 57, and the Object Management Group (OMG). The UIB is being designed for submission to the CCAPI working group, IEC, and OMG groups for consideration. As will be discussed later, OMG's Common Object Request Broker (CORBA) and Microsoft's Distributed Common Object Model (DCOM) compliant enterprise products provide standard middleware technologies, which are the underlying framework upon which the UIB operates. The UIB does not propose to replace these technologies, but extend these technologies with specific functionality needed by utilities alone.

THE UTILITY INTEGRATION BUS

Fundamentally, application integration with the UIB involves looking at the big picture. However, an integration project may encompass data from a large or small set of applications. One does not need to undertake a major project that requires many months to complete. The issue here is the development of a long term enterprise wide integration strategy so that a small integration project does not become just another slightly larger island of automation. Thinking at the enterprise level while integrating at the department level minimizes risk and maximizes the chances for long term success. Part of this enterprise view is the understanding of enterprise data semantics and the business decision making process.

The understanding of data semantics requires a unified data model. The coalescing of an enterprise's many data models into a smaller more rational set whose purpose is to enable decision making is often called data integration. Data integration is some what different from most programming tasks in that the goal is not necessarily to add new features, but rather to link and expose existing data while minimizing reprogramming. The creation of an integration bus is inextricably linked to the creation of shared data models.

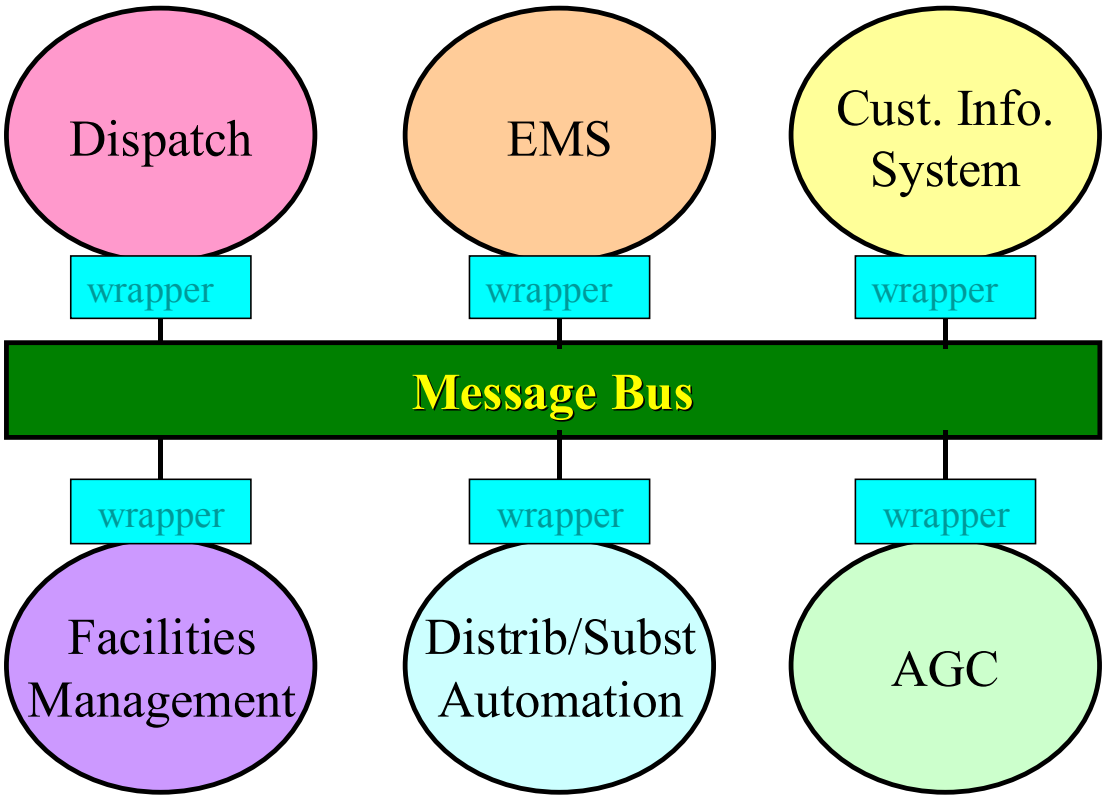
In order for a data model to be used by multiple applications, its semantics must be understood and managed. The commonly accepted way to manage data semantics is by describing what, where, and how data is used in a metadata repository. Metadata is data about data. A metadata repository serves as a central point of control for data semantics, providing a single place of record about information assets across the enterprise. It documents where the data is located, who created and maintains the data, what application processes it drives, what relationship it has with other data, and how it should be translated and transformed. This provides users with the

ability to utilize data that was previously inaccessible or incomprehensible. A metadata repository also allows all applications to be integrated to communicate using a unified data model instead of creating multiple point to point links. Lastly, a central location for the control of metadata ensures consistency and accuracy of information, providing users with repeatable, reliable results and organizations with a competitive advantage.

Besides providing a way to manage common data models such as the CIM, the UIB provides a common way to automate communication. The UIB acommplishes this by automating the exchange of eXtensible Markup Language (XML) formatted messages. Just as HyperText Markup Language (HTML) has become the universal language of the Web, businesses have sought a similar language for describing business data. XML has been adopted by the World-Wide Web Consortium (W3C) and is rapidly becoming the preferred format for exchanging complex businesses data internally as well as between E-Commerce applications. Similar to HTML, XML allows the designer to create custom tags and describe how message data is used and thus provides the facilities to create self describing messages. This capability is independent of transport mechanisms, calling conventions (the order in which parameters are passed as well as how data is returned), and data formats. This significantly reduces the size and complexity of legacy application wrappers. The UIB uses XML messages so that E-Commerce can be more easily integrated with back end systems as well as to take advantage of XML standardization and extensibility.

In order to reliably exchange XML packages a communications infrastructure is also needed. One could use CORBA or DCOM to link legacy applications, but these technologies require a common security domain context, function calling convention, binary data types, and way of locating and activating remote applications. Additionally, CORBA and DCOM require that server applications must be ready to service a request when the client wishes. Thus CORBA and DCOM are better suited to assembly of tightly coupled components. Existing applications require a looser coupling as they might not even be running at the same time. To use a post office analogy, no one waits at the front door for the postman to arrive before mailing a package. Mailboxes provide a convenient method for storing letters until a mail truck comes along to pick up the mail and deposit the received mail. One could use email, but email has not been designed for efficient automation. As mentioned above, existing message oriented middleware products help link applications. In general, these software products include a message broker. With message broker technology, a business application can send business messages to a broker message queue for later delivery. The message broker can also dispatch messages to other internal or external applications based on predefined criteria. Message brokers facilitate location and technology independence and have proven to be the best way to link loosely coupled legacy applications

In addition to message brokering, these middleware products often include the ability to automate business processes (data transformation and workflow). In doing so, they provide a single point of control for managing information flow across multiple applications. For example, the generation of a bill can be automated by creating a script that first collects meter and customer data, then sends the information to a billing application and lastly routes the bill to an application for presentment to the customer. In between these steps, message data may be manipulated so that it matches the internal data model of these applications. The UIB is built on top of this technology and provides additional tools to create a centralized maintainable enterprise view of data processing as well as a utility specific way of maintaining the data model.

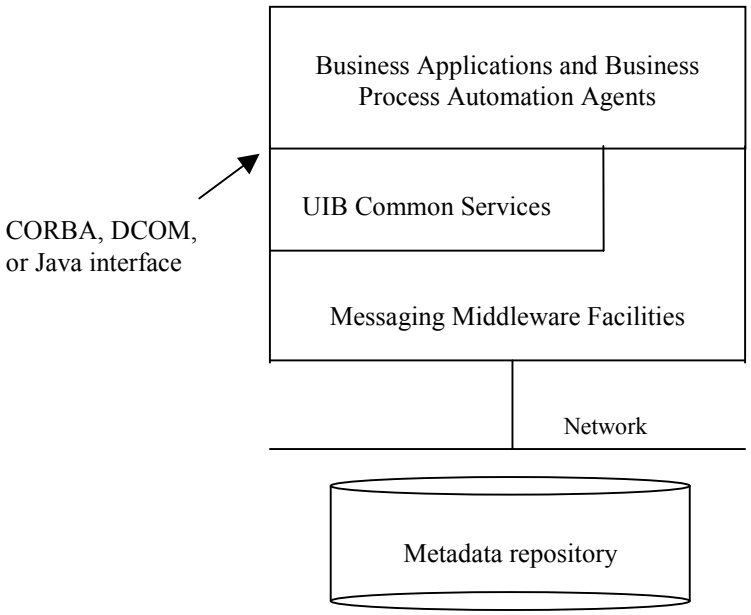


Using XML messages routed via a workflow enabled message broker vastly simplifies application wrapper complexity. While wrappers still need to perform potentially complex translation between object models, this too can be greatly simplified. Ideally, applications would export their metadata into the metadata repository using a standardized mechanism so that the actual transformation code could be automatically generated using a visual tool. In fact, several standards groups are working on defining common ways to export metadata via XML. Companies such as SAP, Baan, Peoplesoft, and JD Edwards have committed to adding standardized metadata export functionality. While at this point, the UIB does not include this tool, even a manual process is greatly eased by a standard metadata export file format. Lastly, as the CIM becomes more widespread as the standard object model for control center application integration, it is likely that utility specific vendors will provide standard mapping between their proprietary object model and the CIM.

The infrastructure described above system can extensively use a publish/subscribe message passing paradigm. In this case, data consumers may not know who is the data providers are and visa versa. Consequently, there must be some component that knows if all events are being processed. The UIB maintains the status of all UIB application components. Additionally, since the UIB is aware of where data originates, the UIB can monitor data availability. Thus the UIB provides a robust platform for performing real time monitoring and analysis of information availability and flow.

UIB ARCHITECTURE

The diagram below illustrates a business application or automation agent with two different sets of interfaces available to it. The first set of interface definitions present how a business application is to invoke one or more of the UIB's common services. The second set of interfaces is provided by a middleware provider. The middleware is comprised of two parts: A messaging service provider such as IBM's MQ Series and a business process (data transformation and workflow) engine. Although the actual interfaces depend on the component technology used (CORBA, DCOM, Enterprise Java Beans), the functionality of UIB services is independent of component technologies and underlying messaging middleware. Since the UIB encapsulates the underlying middleware features business applications are free to use either interface. Business applications access UIB services as a CORBA, DCOM, or Java component.



CONCLUSION

In the past utilities were highly regulated companies producing a stable stream of earnings, but not rapid growth. The reorganization of utility business entities makes this scenario much less certain. Furthermore, the combination of more dynamic energy markets, and the growing importance of E-Commerce is driving the greater use of Decision Support Systems and application integration tools. Key to successful implementation is an enterprise level architecture describing how information is shared. It is critical that IT managers to understand the issues at hand as they adopt integration strategies and that they be in a position to evaluate what set of vendor products are appropriate to their situation. The UIB has been created to maximally leverage existing standards and products. Today the UIB is the only product to fully take advantage of all of these technologies and provide the crucial missing pieces.