Improving Environmental Information Handling and Data Exchange within the Electronics Industry

Eric Simmon and John Messina, Semiconductor Division, National Institute of Standards and Technology

***Abstract*— Environmental regulations impacting the electronics industry are driving the need for new data management systems to track environmental data including material data. This paper describes efforts to take a holistic approach in managing this information by working with industry standards organizations and applying system design philosophies to the standards development process.**

***Index Terms*— Material content, information management, data exchange, supply chain**

I. Introduction

In the 21st century, managing the environmental impact of manufacturing has become an important aspect of electronics production. To protect both human health and the environment, governments worldwide have enacted legislation that places restrictions on the manufacture, use, and reclamation of products. Due to the complex material composition of electronics, as well as their high resource usage and short product lifecycle, many of these restrictions directly impact the electronics industry.

The EU Restriction of Hazardous Substances (RoHS) (2002/95/EC) [1] was the first major substance regulation affecting the electronics industry. The RoHS directive first went into effect July 1, 2006. It restricts the use of six substances used within electronics products: lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, and polybrominated diphenyls. While most companies have avoided immediate repercussions from RoHS, there are more environmental regulations on the way, each with new data requirements [2].

Complying with these new laws and regulations is no simple process and requires changes in product design, manufacturing processes, and in the management of environmental data. A major obstacle towards achieving these goals is the sheer number of environmental regulations. Many of these regulations cover the same territory but have subtly different requirements. This has created a situation in which companies are left struggling to determine what changes need to be made to their manufacturing supply chains in order to ensure compliance for their product in a particular market or country.

Another challenge is the lack of low cost solutions for small and medium sized businesses that are affected by these regulations. While the major enterprises at the end of the supply chain have extensive information management systems, the smaller companies that provide materials and parts often have little or no information management infrastructure and a correspondingly small budget to implement changes. To assist all stakeholders, data exchange standards should be designed to facilitate the creation of supporting software solutions.

Material composition information, one of the more important environmental product aspects, must be able to flow through the electronics supply chain in order for companies to be compliant. The diversity of the data exchange requirements for each new piece of environmental legislation, as well as the diverse business approaches used by companies in the supply chain, makes this a difficult task. Companies range from wanting to exchange only minimal material composition information (i.e., yes/no compliance) to wanting to exchange full material composition information. Facilitating this flow of information requires both the development of business-to-business data exchange standards and the development of underlying data management systems.

While some work has been done developing the material composition infrastructure, there has been no holistic approach to the problem. The result is a mismatch between the database management systems holding the product and environmental data and the data exchange specifications. This paper demonstrates a standards development process that naturally lends itself to the development of a corresponding data management system.

Certain commercial software is identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

E. Simmon. Author is with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (phone: 301-975-3956; fax: 301-975- 6021; e-mail: eric.simmon@nist.gov).

J. Messina. Author is with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (phone: 301-975-4284; fax 301-975- 6021; e-mail: john.messina@nist.gov).

II. Standards Design

One of the major problems encountered in ad-hoc standards development is that there is no method to ensure that all the required information is captured. Since the purpose of most information standards is to gather and exchange electronic data, the failure to identify key information can potentially render the standard useless. Failure to keep the project's domain information (data to be exchanged) and the software transportation layer (transfer mechanism) separate can also result in distorted or missing data. These situations come about because the developers are looking at the standard with slightly different perspectives and expectations.

Ideally, new data exchange standards can be developed in such a way that they can be easily implemented in a database management system. The Infrastructure for Integrated Electronic Design and Manufacturing (IIEDM) [3] project at the National Institute of Standards and Technology (NIST) [4] has been working with standards organizations to implement a more rigorous approach to the data exchange standards development. This approach, similar to ones used in software development, focuses on clearly defining the scope and requirements of the standard and then using that definition to generate both the data exchange standard and a compatible database structure. These specifications can then be used to generate well-integrated data management systems.

The task of designing data standards is complicated by the numerous players; all working together, each one with their own set of needs and priorities. Traditional standards committees often follow an ad-hoc approach, where the standard is developed piecemeal without following a development process specific to data exchange standards. This often leads to standards that are incomplete, cumbersome, or unworkable for their intended purpose. Structured methods are useful, both in building solid information standards that capture all necessary data and in integrating the standards into information management systems.

While a full-blown, rigorous approach to information design might be difficult to implement in the standards arena due to resource constraints, adopting even some of these tools can provide significant benefits to standards development. IPC (a global trade association) [5], working with assistance from NIST, used a simplified version of this approach in their IPC 2-18 subcommittee in creating the IPC 1750 series supplier declaration standards [6]. The IPC 1750 series was well suited to this as it is a data exchange standard that is designed as a modular framework that will support more declarations as needed (the IPC 1752 material declaration standard is one sectional).

The first step was to develop the business requirements that the standard must meet. These requirements were then used to develop use cases that show the specifics of how the standard would function and what information would be included.

The requirements and use case information were then used to develop a Unified Modeling Language (UML) class diagram. The UML is a modeling language composed of many different types of diagrams used to understand and structure the software design process. Of these diagrams, the class diagram (which describes data types and their relationships to one another) is one of the most widely used. Class diagrams can help mitigate some of the difficulties with information standards development by providing a visual representation. Figure 1 shows a simplified view of the IPC 1750 series class diagram showing the basic modules that make up the model. For the detailed class diagram, please contact IPC or the authors directly. In addition, this class diagram maps nicely onto both Extensible Markup Language (XML) definitions and database table definitions, and software tools can be used to automate the generation of both (see Figure 2).

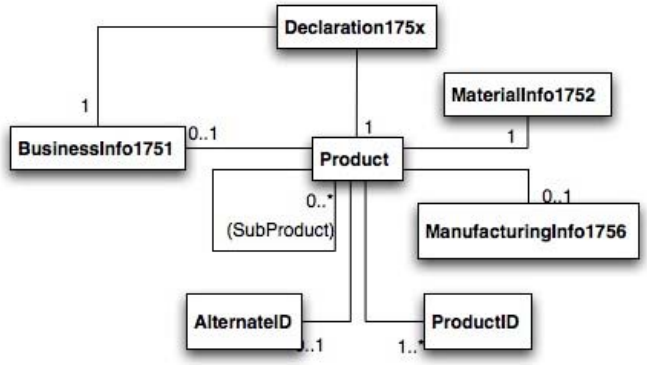

Fig 1 Simplified 1750 series class diagram

After the UML class diagram was finalized, the systems specialists used software tools to generate an XML schema and database schema directly from the model. The XML schema will be part of the published standard and will be released along with the rest of the standard, while the database schema will be provided as a supporting file.

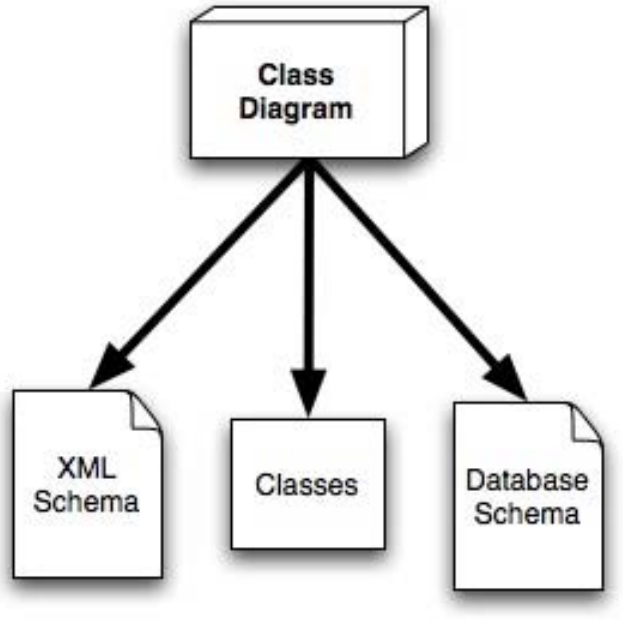

Fig. 2 UML Class diagram generates code

After the domain was defined and XML and database definitions were finished, the final step was to demonstrate how the new process could implemented in a data management system. For this purpose a set of software tools were created to handle data entry and editing. These tools were deliberately designed to help small and medium businesses implement the IPC 1750 standard and to provide a reference implementation for developers to build on.

III. Data Management System

A data management solution consists of several parts: the data format and data types, the programs that can access and manipulate that data, and the underlying data storage system. The data and relationships do not change regardless of whether the data resides in an XML file to be exchanged, loaded into memory by a program, or in a database table. This means we can use automated techniques to generate definitions for the XML file, programming code framework, and the database structure. The specifics of the relationship between the XML files, the data manipulation programs, and the database are described below.

The first goal is either to get data from an XML file into a program or to get data from a program and into an XML file. Figure 3 shows the underlying concept and describes how an XML schema can be used to generate programming code classes and how this programming code is then used to create an application that can hold the data from the XML file (the instantiated data is referred to as an 'object'). When the data is in memory it can be manipulated as needed. To get the data from a file into memory, parsing or marshalling/unmarshalling functions are used.

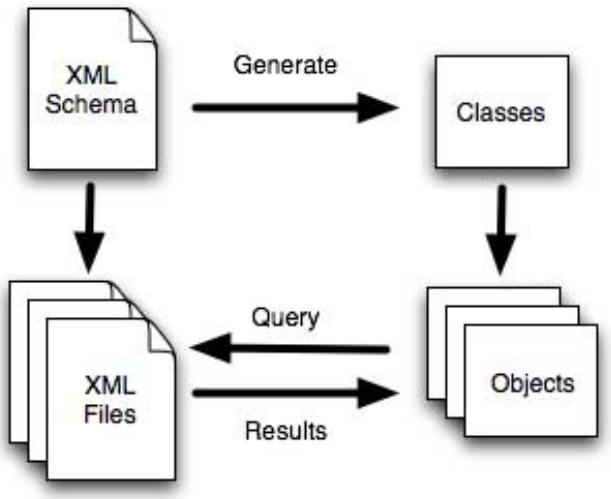

Fig. 3 Relationship between XML schema, classes and XML instances

After the tool has converted documents into in-memory objects, the next step is to import the data into a database. This will require the tool to have the ability to process in-memory objects and make transaction and query based calls against the database. (see Figure 4).

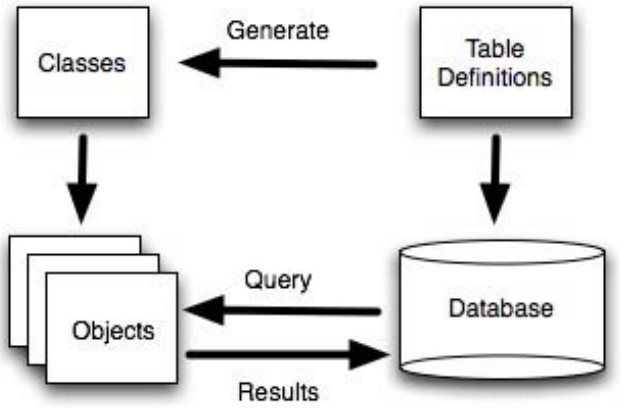

Fig. 4 Relationship between database schema, classes and data objects

Once the theoretical structure of the system is understood, the next step is to implement the data management solution. We chose to implement our applications using Java and Java toolkits [7] as Java is a freely available programming language with many open source toolkits. In theory, this approach could be implemented in any programming language (although it does lend itself to an object oriented programming approach).

The two main toolkits used to generate the Java classes and the parsers for the database and XML are Java Persistence API (JPA) [8], for the database side, and Java Architecture for XML Binding (JAXB) [9], for the XML side.

A. Mapping Java into XML

The JAXB application programming interface (API) is a set of interfaces and classes through which applications communicate with code generated from a schema. It provides an Unmarshaller class that converts XML into Java and optionally validates the XML and a Marshaller class that converts an object into XML data and optionally validates it. Aditionally it creates classes that can contain the data in the XML file.

To create a tool using JAXB, a collection of Java classes is created from the XML schema using the JAXB schema compiler. The schema compiler takes XML schemas as input and generates a package of Java classes and interfaces that reflect the rules defined in the source schema.

B. Mapping Java into a Database

The Java Persistence API (JPA) provides similar functionality to JAXB but provides the connection between the database and the programming code. It provides a mapping of Java classes to a relational database.

C. Scriba Editing Tool

Scriba is an editing tool that was developed by NIST as the reference implementation of IPC 1750 that businesses can use as-is or modify to suit their specific needs. In addition to the editing functions, it implements the marshalling functions discussed above to allow the importing and exporting of XML files. It is written in Java and uses readily available Java APIs; it runs on multiple computing platforms. Scriba has a graphical user interface to facilitate the generation and editing of XML documents that comply with the IPC 1750 Standard. Scriba allows the user to create IPC 1750 compliant files, validate XML files against the IPC 1750 XML schema, modify existing IPC 1750 files, and sign and verify signatures according to the IPC 1750 standard. Previous versions of the IPC 1750 reference implementations had relied on a dynamic PDF [10] form. However, the limitations of an input tool that mimics a paper form became readily apparent when trying to add support for multiple products in a single declaration. Scriba's interface is designed to be similar to the IPC 1750 version 1 form, but with enhancements designed to support multiple products.

Scriba's editing interface allows the user to view each element of the XML document, edit it and modify it by following the structure of the IPC schema (as show in figure 4).

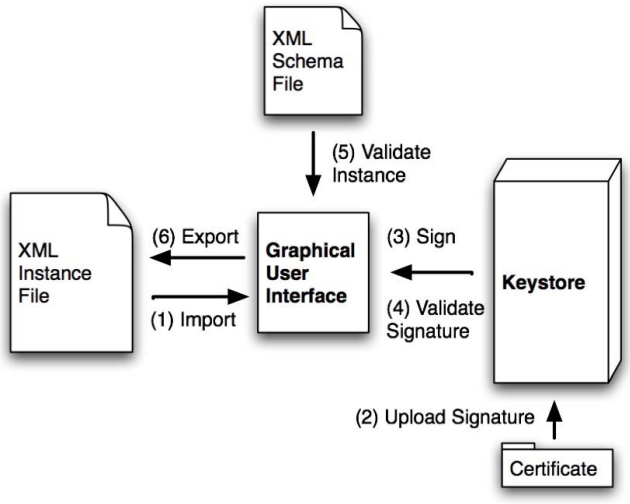

Fig. 5. General steps in editing an XML data file using Scriba

Scriba consists of the following basic components:

1. XML Import; Parse the XML instance and show its contents in the GUI. Automatically validate the instance according to the schema, if it's valid upload it, otherwise notify the user.
2. Import Certificate; Before signing a document, a valid certificate must be uploaded to the Scriba keystore.
3. Sign XML; The user signs the file using certificates stored locally in the keystore (a keystore is database file that stores encryption keys).
4. Validate Signature; To ensure the file was signed correctly, the signature is verified after the document is signed.
5. Validate XML instance; The user can also validate the XML against the schema while editing.
6. XML Export; The data is marshalled from the Java object into an XML instance and written to disk.

IV. Security

An important feature added to IPC 1750 version 2.0 is the ability to electronically sign a document using an XML signature. Scriba provides this functionality and interacts with the security module to provide the user with an option to sign the document. This option uses a certificate of authenticity provided by a third party and the tool provides a means of pointing to the certificate, which is then used to sign the document. In addition to being able to sign a document, the tool can also validate a signature, and provides an icon that identifies an authentic signed document.

V. Conclusion

The new standards development process (based on system design and software development techniques) was successfully demonstrated during the process of developing the supplier declaration data exchange standard IPC 1750. Committee experts were consulted to define the scope and requirements needed to understand the process of exchanging material composition data. From this data, a UML Class Diagram was created which in turn was used to generate both an XML schema as well as a database definition. Using freely available software toolkits, the project team was able to create a prototype data management system and demonstrate how it could be used with in tandem with a data entry application. This process successfully demonstrated that the proposed development process can help developers create new data exchange standards and underlying data management systems at the same time. The next step for the project includes improving and streamlining the requirements development process.

Acknowledgment

The authors would like to thank Fatima El Osbi for her work developing Scriba and the database tools and Matt Aronoff for his perceptive insight in constructing the XML schema.

References

[1] European Union's Restriction of Hazardous Substances Directive. Available at: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0095:EN:HTML. Accessed on: Feb. 20, 2008.

[2] J. Messina, E. D. Simmon, M. Aronoff, "Environmental Regulations Impose New Product Lifecycle Information Requirements,"*Complex Systems Concurrent Engineering; Collaboration, Technology Innovation and Sustainability,* London: Springer-Verlag2007, pp. 373-381.

[3] Infrastructure for Integrated Electronic Design and Manufacturing (IIEDM) project web page. Available at: http://www.eeel.nist.gov/812/IIEDM/. Accessed on: Feb. 20, 2008.

[4] National Institute of Standards and Technology (NIST) web page. Available at: http://www.nist.gov. Accessed on: Feb. 20, 2008.

[5] IPC web page. Available at: http://www.ipc.org. Accessed on: Feb. 20, 2008.

[6] IPC 1750 standard web page. Available at: http://www.ipc.org/. Accessed on: Feb. 20, 2008.

[7] Java website. Available at: http://www.sun.com/java/. Accessed on: Feb. 20, 2008.

[8] Java Persistence API website. Available at: http://java.sun.com/javaee/overview/faq/persistence.jsp. Accessed on: Feb. 20, 2008.

[9] Java Architecture for XML Binding web page. Available at: https://jaxb.dev.java.net/. Accessed on: Feb. 20, 2008.

[10] Adobe Portable Document Format (PDF) web page. Available at: http://www.adobe.com/products/acrobat/adobepdf.html. Accessed on: Feb. 20, 2008.