Precast Bridge Model-Based Automation

Authors: Lee Tanase
President, LEAP Software, Inc.

Dave Hansen
Corporate Software Architect, LEAP Software, Inc.

Jon Lea
Corporate SQA Engineer, LEAP Software, Inc.

Introduction

Automation models have become a common information technology artifact. They define the vocabulary and semantic structure of the engineering and design products within a particular industry domain, allowing this information to be exchanged, communicated and generated electronically.

However, most computer applications in the structural engineering field have been hindered by a common weakness since their adoption back in the 1960's: they lack the ability to readily share information with each other. Too often, the engineer finds himself entering geometric information to define a roadway, then later has to re-enter the same information in a separate product in order to analyze a bridge located on that same roadway. Or he finds himself typing spring coefficient values into a column design application that were themselves printed in the output of a footing analysis program.

The common approach traditionally used by developers is one where each application is implemented as a standalone product – a new island in an archipelago of products that comprise the structural engineering software field. What is needed is a way to link these islands together.

This paper reviews why there has been so little successful movement in this direction and examines several possible models designed to integrate applications in the bridge industry. An actual example of one of the software integration models will be made during the presentation.

Background

One of the earliest attempts at addressing this problem was also probably one of the most successful, though its data sharing goal was never achieved. The Integrated Civil Engineering System (ICES) developed at MIT in the early 1960's was an ambitious first effort. ICES was far ahead of its time technologically, and while it could not deliver data integration, it did produce COGO and STRUDL, two benchmark applications for our industry that are still widely used today.

Since then, there have been many efforts in the industry to integrate design and analysis applications. Thus far, none have been truly successful. The authors of this paper have been involved in some of these efforts, and as a result, were instinctively prone to draw parallels between application interoperability and perpetual motion when the project was first broached.

Why the slow progress?

One commonly held assumption why bridge design, engineering and construction have not truly utilized the knowledge-based integrated technology systems is due to its fragmentation. The industry has traditionally involved many small participants, where approximately 60% of the firms involved have less than 10 employees. Each firm carries out only a few steps of an increasingly complex process, and therefore many organizations came together to execute a project from beginning to end.

In addition, the very broad embedding of 2D drawings, combined with the fragmentation of commercial software applications creates a situation where no single participant has enough economic impact to justify the investment, especially without the participation of government agencies.

The Present Status

The Industry Foundation Classes (IFC), developed in 1998 in association with the International Alliance for Interoperability (IAI), stand out as a serious first attempt at a unified data modeling environment that would support multiple software applications. An unfortunate implementation decision to utilize a proprietary parser (EXPRESS) slowed initial adoption. EXPRESS was finally released as a public product, but by then XML (Extensible Markup Language) was finding rapid acceptance throughout the computer industry, raising questions about the long-term viability of IFC/EXPRES. Attempts are now underway to re-architect IFC using XML, but this seems only to have added to the confusion. The academic community has done most of the initial work on IFC, and most CAD providers in the AEC field also offer support, but the standard has made little headway elsewhere. Tekla Oy is one of the few companies that supports this standard.

Bentley further complicated IFC matters by announcing another standard, aecXML, in mid-1999. The product name is particularly expressive, but, unfortunately, does not fit the product definition that Bentley had in mind. The aecXML system was designed to describe the non-graphic data involved in the AEC field. To quote a Bentley spokesperson: "aecXML is for talking about things, not modeling them. We can use it to agree what 'door' means, but aecXML won't describe doors or model them." One might, therefore, think that aecXML has a place beside IFC, but recent efforts to extend IFC's support for non-graphic data

seem to have created an overlap. It appears that aecXML could be useful in areas such as product catalogues, scheduling, etc. As such, perhaps it has utility in the precast products industry, but it is difficult to envision a use for it in the current versions of bridge design applications.

One of the most recent efforts in the bridge software industry that aims to provide application interoperability is the Pontis/Virtis/Opis collection from the American Association of State Highway and Transportation Officials' (AASHTO). For close to 10 years, AASHTOWare has been working to deliver a set of products for participant state DOT's, and their progress has produced some results, especially in the development of a common database of bridge components and bringing some uniformity to the bridge inventory data collection. At present, some important design aspects such as substructure design, are missing, including a number of missing bridge data components – alignments, a world coordinate system, skews, support for box girder superstructures to name a few.

In summary, the bridge industry continues to use specialized design and analysis applications for the structural engineering user, but these applications generally have no ability to exchange data.

Model Scope

A critical early decision in all model based applications is to define the scope or domain of the integration model. What activities and data need to be shared within the model (*Internal activities*), and equally important, what data and activities need to be shared outside it (*External activities*)?

A very important third category of activities are *External activities that exchange information with internal ones*. It identifies activities that the model must support by exchanging model information with them, but which otherwise are not fully supported by the model itself. An example of these activities in the bridge domain might be "design" or "analysis" and "project scheduling". The activities need to exchange information, however the project scheduling would be external to the model itself.

Model Objective

In order to define a beneficial model, the model must describe the bridge industry's processes and must be as consistent as possible with its activities. High level activities and processes must be broken down in detail activities, and mutually accepted steps integrated efficiently.

The flow of information is then linked by their appropriate activities, and define channels of communication of information, such as tables, reports, drawings, etc.

Each activity or exchange process is distinguished by its information *input* and *output*.

Can The Present Technology Support This Objective?

Microsoft has fully embraced XML with Visual Studio .NET. XML is an outgrowth of HTML, but whereas HTML is designed to express appearance, XML is designed to express raw information absent any implied notion about how the data should be rendered. It's a simple language that is entirely text-based. Here's a snippet that might describe a 3-D coordinate point:

```
<Coord3D>
    <Id>PT0145</Id>
    <Units>Metric</Units>
    <X>1200.000</X>
    <Y>2000.000</Y>
    <Z>200.000</Z>
    <Desc>Pt at Northwest corner of Parcel 124</Desc>
</Coord3D>
```

There's really nothing inherently remarkable in this sample. But when this notation is combined with all the support tools already available to read, write, query, and validate data expressed in this form, the importance of XML cannot be denied.

It can provide a common data format for companies (and applications) that want to share data. It's used by web services to encode data in a platform-independent manner, and it's used to build web sites, where it serves as a tool to separate content from appearance.

But what really makes XML important is that fact that the entire computer industry has adopted it as a standard (practically overnight), and as such, there are numerous tools available for reading and writing XML. Visual Studio .NET not only provides numerous tools for these purposes, it utilizes these same tools internally for all kinds of activities, ranging from system registry to application internalization support.

Integration Models

Different formal methods can be identified to develop a robust integrated model able to support a variety of domains and component applications. Two very different options are evaluated below.

Option 1

Each pair of applications that wish to communicate define a private data structure representing the information to be shared, and read/write this data to an agreed upon disk file in order to exchange information (Fig. 1).

The file would be a disk image of the private data structure. Transfer would be initiated by the application originating the data, but the transfer would always be the result of a user request.

A private data structure could be defined for each transfer request, and would not have to be associative between any two applications – that is, the data structure to transfer pier data from APP-1 to APP-2 could be different than that used to transfer the same information from APP-2 back to APP-1.

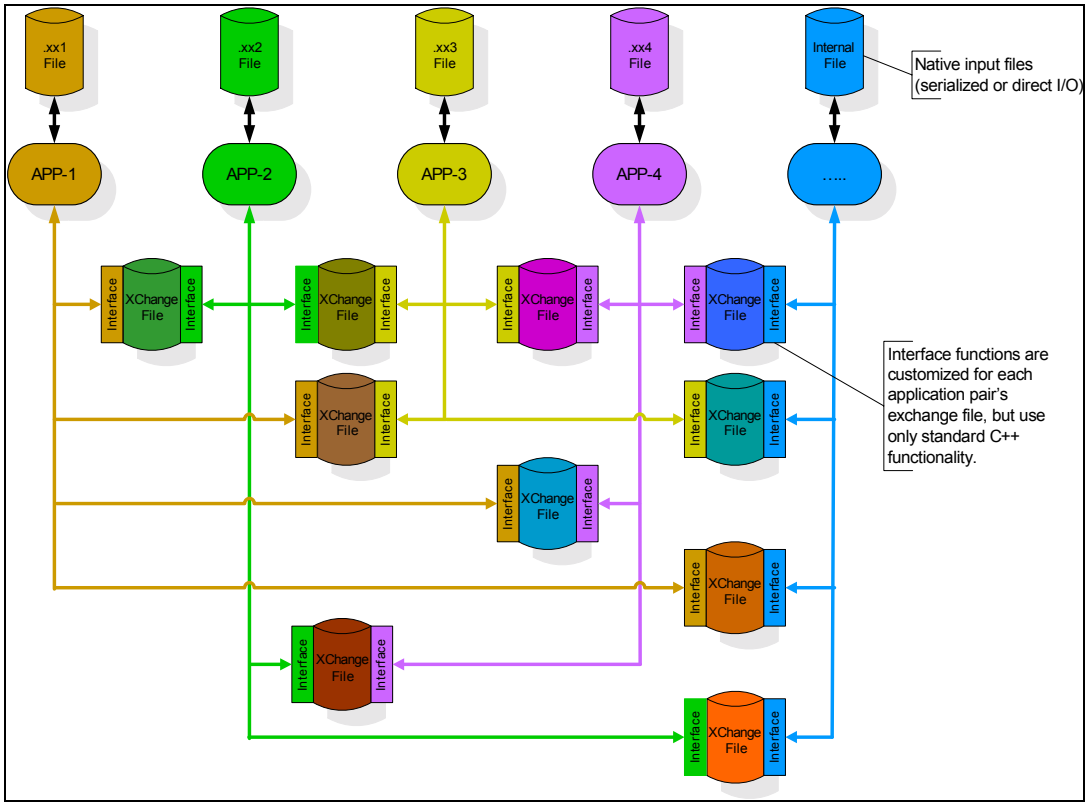

Fig. 1 Direct File Exchange Using Proprietary Data Format

Option 2

In this option, a data exchange software layer is developed to provide an API that can be invoked from any of the applications to be integrated (Fig.2).

Rather than model just the relevant data to transfer between the data originator and receiver, a set of data originator API functions and data

structures is defined, allowing the applications to make data queries. These functions should be customized to the specific data receiver functionality. Transfer would be initiated by the data receiver application, and could utilize COM, DCOM, or .NET to create the API.

The approach has the benefit of incremental implementation, and also would support data originator and receiver applications running over a network. Applications changes would be localized, and affect no other components. Benefits from changes to the Data Exchange Layer would only be realized if the application(s) that used the API change was also updated. Consequently, upgrades would minimally affect two applications at a time.

In addition, with an API approach, the model will also be able to expose the API to the user and any third-party developers.

At the application level, the user could now have to resolve just what information he wishes to use. This should generally be limited to a dialog enumerating the available components and asking the user to select the desired one for analysis. From there, the data exchange layer API can be called to retrieve the requested information.

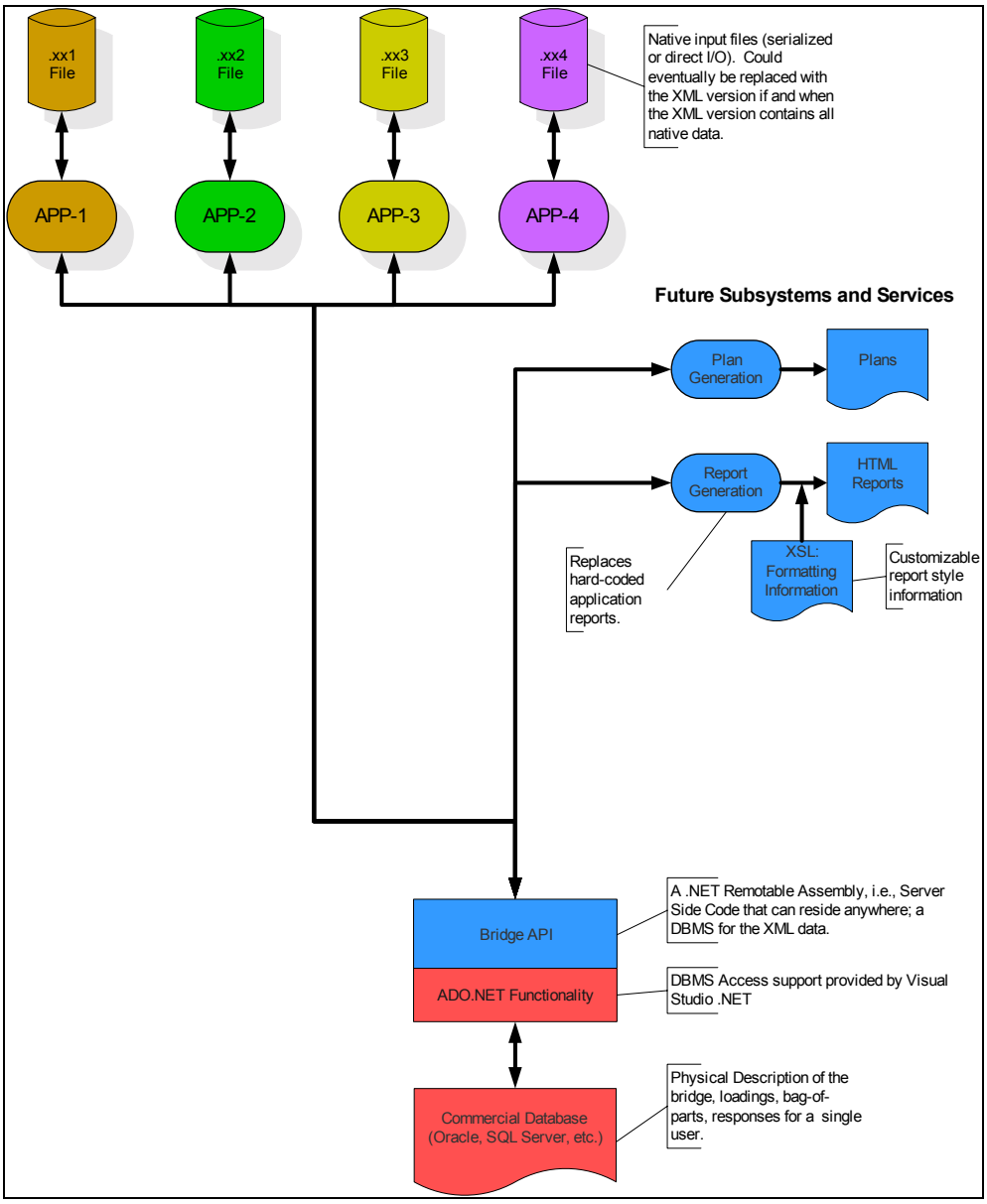

Fig. 2 – Open Standards Data Pool Shared By All Applications

Summary

We have evaluated the concept of bridge model based automation, designed to support the integration of existing component based applications, and outlined two suitable methods for the implementation of an industry wide of a model.

The authors believe that the potential to derive benefit from knowledge-based integrated bridge modeling is tremendous, and propose incremental strategies for achieving the benefits provided today by advanced technology, with each step providing increased payoffs. Assuming that additional industry sectors could

follow the lead of the precast bridge industry, multiple critical masses may be reached with increased benefits for all the participants.

References:

1. This work is partially supported by research done by the authors as part of the Technical Committee work related to an industry consortium of producers of precast concrete that is working to develop a model for their industry, the Precast Concrete Software Consortium (PSCC).
2. Eastman, C. M., Sacks, R., Lee, G. (2002) "Strategies for Realizing the Benefits of 3D Integrated Modeling of Buildings for the AEC Industry". *ISARC – 19th International Symposium on Automation and Robotics in Construction, Washington DC.*
3. P. Teicholz, M. Fisher (1994) "Strategy for Computer Integrated Construction Technology". *ASCE Journal of Construction Engineering and Management.*