Milramco

Eliminating Duplicate Data Entry

Introduction

How much time is wasted and mistakes made in your organization due to the need to enter the same data multiple times in different systems?

If your organization is like most organizations, the answer is that a lot of time gets wasted, mistakes are made, information is not available when needed, and employees get frustrated. Even worse, this duplicate data entry can lead to mistakes that impact customers, such as shipping orders to the wrong address, due to delays in getting data into all systems.

Everyone's ideal is one system into which data is entered once and is then available for everyone else in the organization to use. Instead, we have separate systems, with each serving its own departmental silo, often supplemented by data maintained in Excel spreadsheets and ad-hoc databases plus information recorded on paper forms stored in a folder in some filing cabinet.

In this white paper, we examine the root causes of this problem. Then we suggest that rushing out and buying a new ERP systems at a total cost of ownership of several hundred thousand dollars is not a solution. Finally we present a technology solution that solves the fundamental problems and has worked for a number of industrial organizations.

Root Causes of the Problem

Before we can solve this problem, we need to understand the root causes.

First let us examine the issue of implementing one system that does all the functions needed to run an industrial business. There are two factors that work against this:

1. Every industrial business needs to offer unique products and services to fill market niches where it can command profitable margins and is not just competing on a commodity lowest-price basis. This implies that every business needs to use systems tailored to the way that it does business and cannot use a one-size-fits-all solution and still be competitive.

2. The optimum size of a software development team is 5 people, beyond this productivity goes down[1]. A team of 5 people can only develop and maintain a limited body of code that is typically limited to one solution domain, such as accounting, human resources, sales and marketing management, operations management or inventory tracking.

In the 1990's many Enterprise Resource Planning (ERP) software companies had the expressed goal of developing a single system that met all the needs of their customers. Unfortunately they needed to sell hundreds of thousands of copies of their software in order to meet their sales goals and to maintain their stock price or at least keep their Venture Capital backers happy. For the reasons detailed above the ERP companies failed to meet these objectives and ended up where they started, as combination accounting and materials requirements planning systems.

[1] "The Mythical Man-Month" Second Edition, Frederick Brooks, Addison Wesley 1995.

As a result, most industrial organizations implemented ad-hoc solutions comprised of many different systems, none of which typically talk to each other, supplemented by work-arounds in the form of spread sheets and ad-hoc databases to handle the special cases not covered by the standard applications they use.

Acceptance of Reality

The first thing to recognize is that there is no such thing as one-size fits all when it comes to computer applications to support business. Instead there are best-of-breed applications, which are best developed by small teams that really understand a specific business area such as accounting or operations tracking.

Different organizations have different needs in each area of their business. A privately owned, single plant manufacturing company may only need a simple accounting system whereas a multi-plant international company will need a sophisticated financial system. Yet the single manufacturing plant may need a more sophisticated operations tracking system than that used by a multi-plant organization due to the nature of its manufacturing operations.

So, the first step in finding a solution to this problem is to recognize that each organization should use systems that are the best of breed for its specific needs. Also, it is important to recognize that these systems may need to be customized to meet the needs of each individual business.

Often these systems already exist in the form of legacy systems, already used by the company, which have been highly modified over the years, supplemented by ad-hoc spreadsheets and database applications. Typically there is no financial benefit in replacing these systems, unless the underlying technology has become obsolete, as these systems already meet the operational and competitive needs of the business.

I sometimes find it hard to convince business owners and executives that there is no silver bullet and that just buying a new ERP system will not solve their business information problems. But that, instead, they should take a hard look at what the problem really is and solve that instead.

Solutions to the Duplicate Data Entry Problem

Most of the data that needs to be shared between systems exists in the form of a wide variety of databases and spreadsheets. These can be read, written and updated by computer software using standard data interface formats, such as ODBC and XML, so computer programs can be written to automatically move data between the different systems, thus solving the duplicate data entry problem.

The biggest problem with this solution is the time and cost it takes to develop these automated data exchange programs. The obvious answer would seem to be to develop standard interface programs between applications that could be implemented once and sold to many organizations at a reasonable cost.

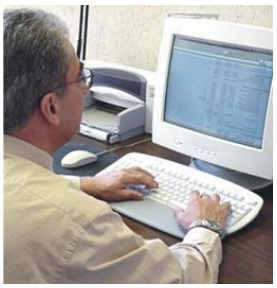
Unfortunately this does not work, except in a small number of cases. There are over 300 ERP and accounting systems in common use in industrial organizations in the USA, each which would have to have standard interfaces to hundreds of other applications, which may then have

to be interfaced to each other. It is not economical to have teams of people dedicated to the development and maintenance of all these interfaces, especially when each of these applications may be updated every six months or so and each one of these systems is in use in a variety of legacy versions.

Even where it is economical to have standard interfaces, it quickly becomes apparent that the data to be exchanged depends on how an individual company uses the systems that will exchange data. In implementing data exchange interfaces, I often find that critical data for the target system comes from user defined text fields in the source system, something that no standard interface would be expected to know about.

As a result, we have found that standard interfaces are not a solution except in a very limited set of cases and even then may be highly problematic due to the way the systems are used.

How We Previously Developed these Interfaces



When data was to be moved from one system to another, we would start with a clean sheet of paper and then design and implement a custom computer program to automatically move data between systems. These custom data exchange programs were expensive to develop because:

1. Data exists in different formats in different applications because the needs of each application are different. Accounting systems are focused on accounting transactions such as changes to the value of inventory whereas operations tracking systems are focused on the details of the movement of individual containers of material. These relate to each other but are stored in very different formats.

2. In most databases, data does not exist in the form of nicely defined business objects. Instead the storage of data is made efficient by using many levels of indirect references. This makes it difficult to directly relate data in one system to another and requires the writing of complex SQL statements with joins on many tables.

3. Many legacy databases contain data that is not acceptable to the target system to which data is being moved. This may consist of control characters accidentally typed into text strings or text strings that contain special characters that are unacceptable to the target system. This requires writing a lot of error checking code.

4. Sequencing of data movement is important. It is important to process transfers of the entry of inventory into a warehouse before the records relating to its shipment are processed. Some data movement may need to be processed every few seconds, whereas other data objects may only be moved once a day.

5. Handling of rejected transfers is important. Updates to databases are often done using stored procedures. These stored procedures may return errors because of data inconsistencies. For example, a transaction may be passed to an accounting system with a date for an accounting period that has already closed, thus resulting in the accounting system rejecting the transfer.

What we learned was that, while we were able to write the actual data transfer code relatively quickly, writing all the error detection and correction code took many weeks or even months.

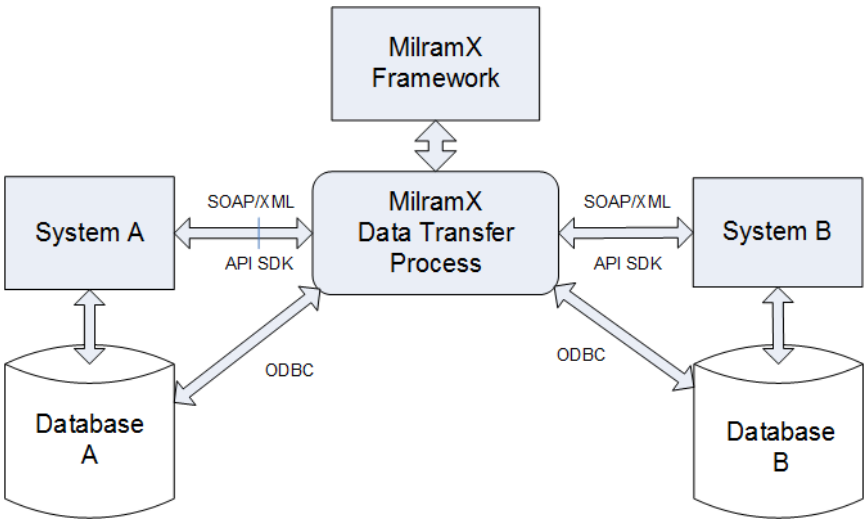
Solution to the Data Exchange Interface Development Costs

After a number of years we came to recognize that we were writing similar code over and over again. So we started on a research and development project to provide most of the needed code in the form of pre-developed libraries or automated code generation tools. After some evolution, the resultant body of code became the MilramX Framework and Software Development Tool Kit (SDK).

MilramX typically provides over 90% of the code we need to develop robust data exchange interfaces between systems, leaving the interface development team only the remaining 10% or less of the code to write and test. More important MilramX has evolved to provide standard methods for implementing these interfaces including providing a methodology for experts in the different systems to collaborate in interface development. This has resulted in a reduction of the development time from weeks to days and has shown that it can cut the development costs by an order of magnitude.

It still can cost between $5,000 and $15,000 of labor to develop and test an automated data exchange interface between two major systems, used in different functional areas within a business. But when this is compared with the average loaded labor cost of over $50,000 a year for typical employees whose labor will be saved, this now makes economical sense as a solution to the problem of duplicate data entry.

What the MilramX Provides



The MilramX software provides the following:

1. A framework to control and monitor the transfer of data objects between two systems. This includes a web-browser based interface so that this can be used remotely.
2. A variety of standard interfaces including ODBC/SQL, SOAP/XML web services, Excel spreadsheet import and export, and a .Net SDK that enable the implementation of interfaces with a wide variety of systems.
3. Mechanisms for relating top level business objects to underlying database or web-services data structures. These named business objects are defined as an array of name**:**value

parameters. The business objects, and their relationship to the underlying databases, can be defined using Excel spreadsheets. These spreadsheets are then translated to XML metadata definitions, which are used to automatically generate SQL/ODBC code or call web-services proxy methods.

4. A .Net based framework for the data transfer process that enables the translation between business objects for different systems to be coded in VB.Net or C# in the form of individually scheduled data transfer objects.
5. Built-in error checking to prevent bad data from one system being relayed to another system. This includes an error logging system and notification of the monitoring framework.
6. A mechanism for saving failed exports and then for subsequently editing and retransmitting the failed business objects.

For more details, please see www.MilramX.com .

Conclusions

In a number of industrial plants we have successfully demonstrated that duplicate data entry can be eliminated using automated data transfer software. This not only saves the cost of the labor previously wasted in duplicate data entry and significantly reduces the mistakes but also improves the efficiency of operations by making new data rapidly available to everyone who needs the information through the systems they use on a regular basis.

We have found, based on a lot of practical experience, that a framework and software development tool kit, such as MilramX, is essential if the development of these interfaces is to be done quickly and at an economical cost.

We have also found that software like MilramX is not something that is developed once and then used thereafter. MilramX is continuously evolving as we add new features to make it easier to develop interfaces and as we meet new challenges with interfacing to both new legacy and cloud-based systems.

One big area of current interest is the exchange of data between E-Commerce websites and systems used to manage operations within industrial organizations. Another is the collection of data from mobile devices and the display of data, especially from legacy systems, on these devices. Yet a third is the collection and aggregation of data in a form suitable for “Artificial Intelligence” analysis, reporting, and for alerting managers when problems arise.

Author

This paper was written by Dr. Peter Green who is the Chief Technology Officer of Milramco. Dr. Green can be reached at PGreen@Milramco.com. Also please see www.MilramX.com for more details on the MilramX software.

