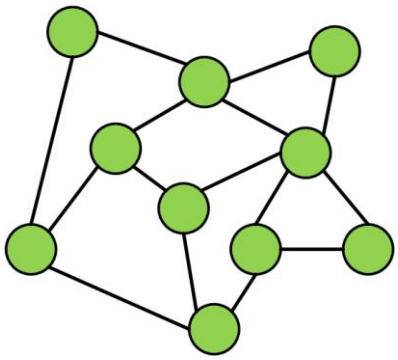
KnarrTek Data Sheet
MilramX Decision Support Software Platform

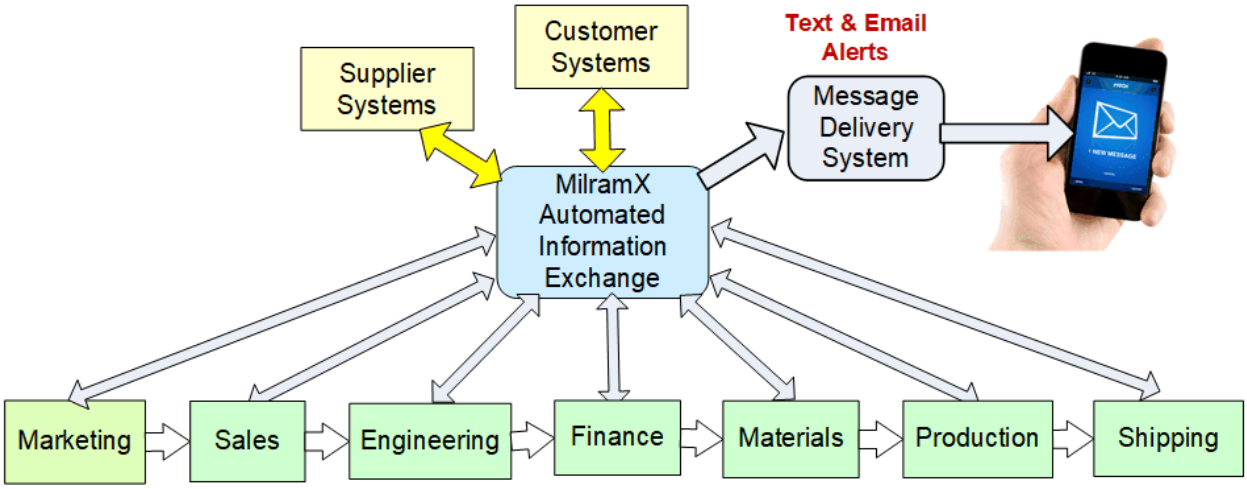
Overview



MilramX is a real time Artificial Intelligence (AI) software platform which uses the paradigm of interacting intelligent agents. It is used to "Make sure that everyone in each organization has the information they need to do their jobs, when they need that information, in a format that is most useful to them".

The primary application of MilramX is for materials management in the manufacturing, construction, and medical supply chains, where it is used in conjunction with the BellHawk real-time materials tracking software platform.

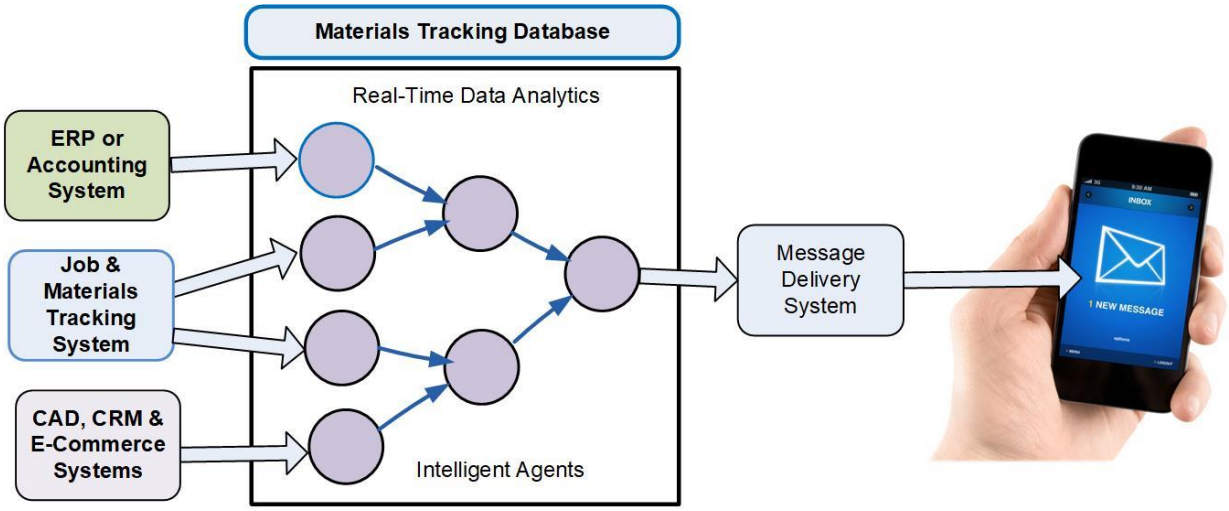
In most of these supply-chain decision support systems, MilramX provides most of the infrastructure code pre-built, minimizing the amount of code that needs to be developed and enabling rapid deployment. Typically, the MilramX platform provides over 90% of the code needed for most materials management applications.



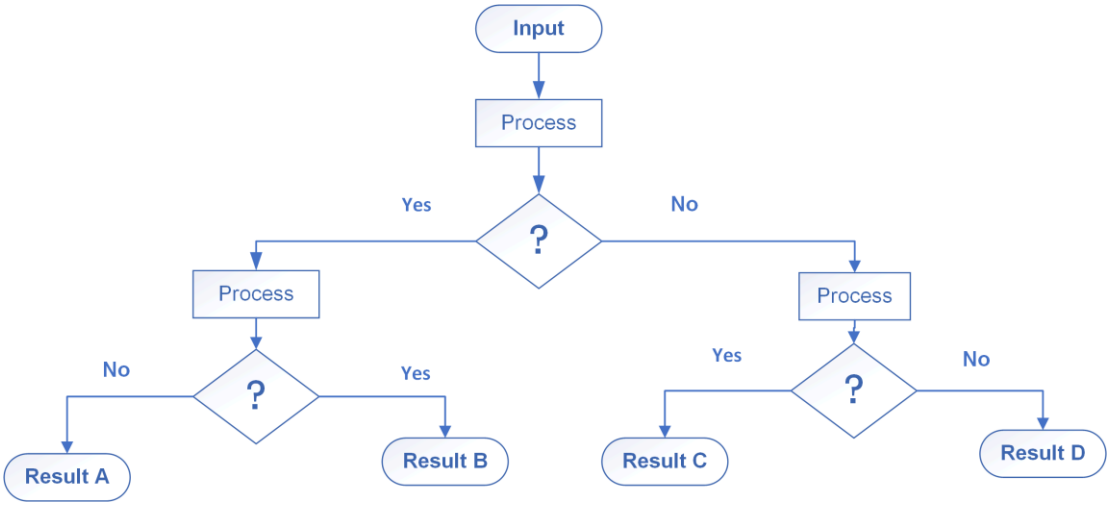
The intelligent agents within MilramX are independently scheduled code modules, which carry out a single function, such as retrieving the most recent updates from one system and sending an update to another system. These intelligent agents can also monitor changes in the databases of one or more systems and send Email or text message alerts to a list of people who need to be aware of the changes and/or to take action.

Because of the large number of different interactions that may take place asynchronously within an organization, the code modules within MilramX are scheduled independently rather than being organized as a hierarchy of subroutines, as in a decision support tree. This enables these intelligent agents, called Data Transfer Objects (DTOs) to be developed and tested independently, using agile development methods, rather that needing to develop and test one large body of monolithic code.

DTOs can interact by sending messages to each other and can act as a cascading chain, when needed.



This is in contrast to using hierarchy of subroutines to decide what data needs to be sent where and when, which requires the decision support system to be developed all at the same time.



With an intelligent agent approach, agents can be added and modified as the organizations needs change, without the need to modify the whole system.

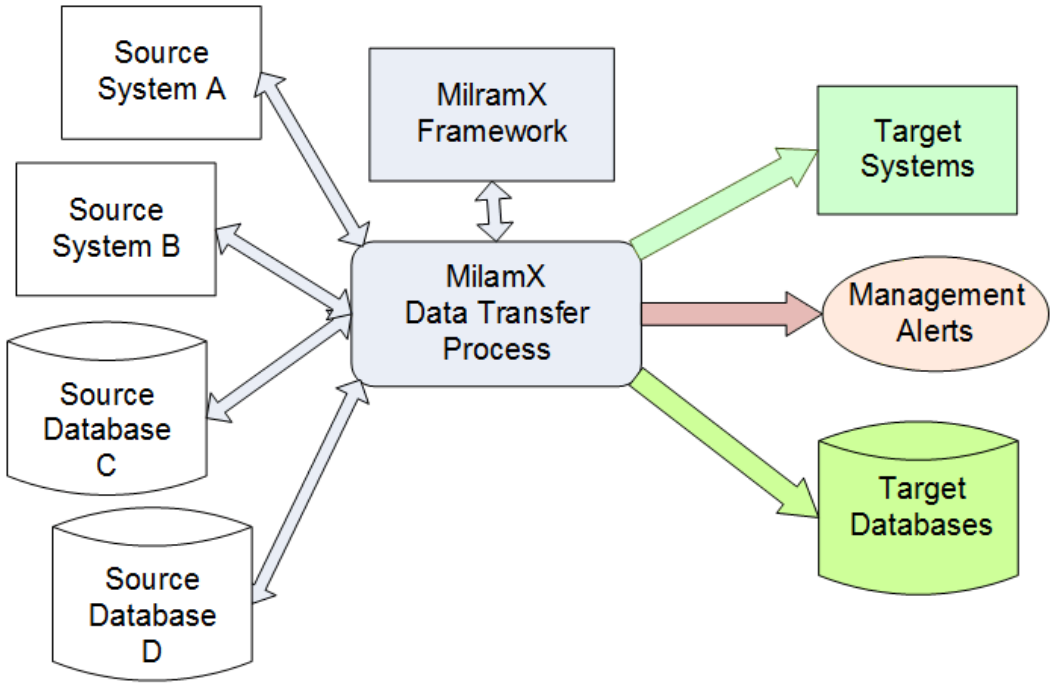
MilramX intelligent agent DTOs may contain rules or some decision-making algorithms. They may simply map the data from one system to another or use the data to learn the parameters of a model of a manufacturing plant, using neural network methods, to enable rapid prediction of future delivery times of materials or finished products.

Paradigm Shift

A major problem faced by many managers and their staff members in manufacturing and industrial distribution companies is having the right information, at the right time, to do their job.

Production managers often have to spend hours “walking-the-shop-floor” to tray to spot when something is going wrong, such as would result in later shipment of customer orders. Materials managers are often blindsided by stock-outs or late deliveries and both have to deal with a rapidly changing mix of customer orders. These managers and their staff often have to spend hours wading through stacks of reports or looking in multiple systems just to get the information they need to make good operational decisions.

MilramX was developed to enable the rapid implementation of automated information exchange systems to solve this problem. Instead of people traying to find when there are situations that they need to pay attention to, MilramX uses intelligent agents to monitor multiple sources of data and to alert people when there are events they need to pay attention to as well as transforming data from one system into information to be automatically inserted into another system.



MilramX uses an Intelligent Agent architecture to intelligently to fetch and interpret the latest updates to sources of data that it is monitoring. In this way MilramX is able to provide the information, in near real-time, that people in different parts of an organization need, rather than

just moving data between systems. This information transfer may be in the form of updates to the systems they use or by sending Emails or Text Messages.

In most information transfer applications, MilramX provides over 90% of the needed code, either automatically generated or supplied as code libraries. This enables these applications to be rapidly implemented at much lower cost and in a much shorter time than developing custom automated data/information exchange interfaces.

At its simplest, all the actions of MilramX can be controlled by user defined rules without code development. But as the data interpretation algorithms become more complex, developers can code their own Intelligent Agents in VB.Net or other .Net languages supported by the Microsoft .Net development environment.

MilramX was the outgrowth of a request, made 50 years ago, by the CEO of a high technology company: “Don’t give me a stack of daily reports but, instead, give me one sheet of paper with the six things I need to pay attention to today”. We have still not fully achieved this goal, but with MilramX we are coming close.

MilramX is based on the work for the US Air Force by Dr. Peter Green and his team on using the software paradigm of using collaborating intelligent agents to implement real-time decision support systems.

More about the Problems that MilramX Solves

ERP vendors, such as SAP and Oracle claim to provide systems that will meet all the IT (Information Technology) needs of most industrial organizations.

This has only proved true for the largest of manufacturing and distribution companies and, even then, these systems have needed massive amounts of expensive customization to meet the specific needs of these large organizations.

Small and mid-sized industrial organizations, by contrast, rely on separate systems, which have been selected by their users to meet the different functional needs of each department, such as finance, sales, marketing, operations management, and materials management.

Problems arise when data in one of these departmental systems is needed as information by people who use a different system. Some of the traditional solutions to this include:

- Duplicate keyboard data entry, typically based on paper forms generated by one department and sent to another department.

- Transferring data from one system to another in the form of an Excel spreadsheet. This typically requires modifying the layout of the exported spreadsheet to meet the needs of the target system before manually importing this into the second system.

These approaches have the following issues:

1. They are typically labor intensive and mistake prone due to the large number of keystrokes required.
2. Data is delayed in getting from one system to another, typically by at least one day and often more.
3. People needing the data have to go look for it and often have to use multiple systems to get their needed data, which is very cumbersome.

The obvious solution, if you are tech-savvy, and the system databases are easily accessible, is to write some SQL statements to extract data from one database and insert these into another database. Then you can wrap these SQL statements into a computer program, written in a language such as Visual Basic, which is run manually when needed.

This approach works very well, if you are simply trying to create custom reports from the source data and not trying to exchange data with another system. But, while this approach can be a considerable improvement over manual keyboard data entry or Excel transfers, it still has many problems, including:

1. The program you manually create needs to be run whenever there is new data to be transferred. This requires frequent manual intervention by the developer. Alternately the data transfer computer program can be coded to run as a system process on a scheduled basis, but now this has taken this interface from a simple code development task to one needing knowledge of the operating system's internals.
2. Required data transfers are often not as simple as transferring just one data object from a source system to a destination system. The data exchange between a BellHawk materials tracking system and a QuickBooks accounting system typically requires the transfer of dozen or more data objects and this can quickly exceed 20 or more data objects to be exchanged with an ERP system. As a result, the number of lines of code that needs to be developed, to implement a meaningful interface, rapidly escalates.
3. Because of the large amounts of data typically involved, only the latest updates from the source system should be transferred to the target system. This requires tracking when the latest instance of each data object in the source system was transferred, which further adds to the complexity. Also, overwriting already transferred data in the target system may have undesired side effects.
4. Data objects need to be transferred in a specific order. For example a new Purchase Order Line or Ship Order Line, may reference a newly entered Item object, which needs to be transferred first. If this ordering is built directly into the code, this quickly makes the addition of new data objects very complicated and error prone.

5. The fields in data objects to be transferred, often indirectly reference data in other tables, which requires the use of SQL statements with complex joins between tables. This can result in needing much more complex SQL code, which is often not the strength of people who are not professional programmers.

6. Data from the source system may have errors, especially if it has been in operating a long time. These need to be detected and corrected before the data is transferred to the target system. Code to detect and correct errors can often require an order of magnitude more lines of code than the code that actually transfers the data.

7. Data may need to be fetched from multiple sources in order to provide a meaningful data update to a target system. Also, instead of accessing a data base, data may need to be exchanged through a web-services interface. These can further dramatically increase code complexity.

8. Interfaces are not only needed between two systems. If you have 6 systems in common use then you could need 15 (5+4+3+2+1) different interfaces. These may need to be automatically scheduled at different times of the day.

9. Computers, databases, and networks involved in the data transfers all have limitations. It is therefore important to prioritize transfers to make sure that the most important data gets there first and does not negatively impact the performance of the systems involved.

10. Just transferring the data from one system to another is often not enough. It is essential that people get alerts by text message or Email when there are events happening which they need to pay attention to. Previously, action was often triggered by the arrival of a paper form. Now that we have done away with paper forms, we need to substitute electronic alerts.

As a result of all these complexities, developing a data exchange program between two systems can be a large and daunting programming task, let alone developing and maintaining multiple interfaces. Often these interfaces can run to tens or hundreds of thousands of lines of code, which is costly and can quickly become a maintenance nightmare, especially as the requirements for data exchange evolve over time.

MilramX is a software platform that is intended to provide 90% of the code, predeveloped or automatically generated, for implementing reliable automated data exchange interfaces between systems used by industrial organizations. The goal of MilramX is to substantially shorten the time and cost of developing these interfaces.

More importantly, MilramX provides a standard architecture to bring order out of the chaos that often results from these interfaces development efforts. MilramX enables interfaces to be implemented, modified, and maintained incrementally, under control, as requirements change over time, by different people.

MilramX solves the problem of having an interface developed and maintained by a single person, or even a small team, typically in the organization's IT department, who then move on to greater career opportunities rather than spend their lives maintaining the code "monster" they have created as the underlying databases, operating systems, and network architectures change over time.

The Myth of the Standard Interface

In the USA today, we are particularly prone to "Sound-Bite" myths, believing the latest advertising or sales message, rather than analyzing the facts and deciding what is best for our organization and all its stake holders.

One of the most common questions that I get, relative to the BellHawk materials tracking software is of the form: "Do you have a standard interface to my 1996 Podunk 5 ERP system?"

As there are at least 300 ERP systems in common use in the USA, the answer is inevitably "No". Which usually results in the reply that, as a result, the organization cannot possibly use the BellHawk software to automate their data collection and replace their use of paper forms and manual keyboard data entry as BellHawk does not have a standard interface to their ERP system.

At which point, the organization can do one of a number of things:

1. Purchase a new ERP system, typically at a cost of a few hundred thousand dollars for implementation, which the ERP sales person told them would handle all their real-time work-in-process and materials tracking needs. They then find that the ERP system does not meet these needs (because it was not designed to do so) and so they continue to use paper forms and manual keyboard data entry into their new, expensive, ERP system.

2. Continue to use paper forms and manual keyboard data entry for tracking their materials and work-in-process, as well as transferring needed data between systems, sometimes supplemented by the use of Excel spreadsheets. This typically results in unnecessary labor costs of between $50,000 and $150,000 per year as well as the inestimable costs of lost customers and expedited delivery costs due to late shipment of customer orders.

3. Attempt to customize their existing ERP or accounting system to meet their materials and work-in-process tracking needs. Even where successful, this then "freezes" the ERP or accounting system at its current version and so it cannot be upgraded when Tax or other financial requirements change, without paying to have the customizations moved to the latest update. As this results in organizations no longer buying software updates, most suppliers of ERP and accounting systems no longer make their source code available.

4. Attempt to develop a custom materials-tracking system that will work with their current ERP or accounting system or to develop their own custom interface to a system like BellHawk. Both these are expensive and time consuming and typically result in maintenance headaches and system abandonment when the developer leaves the organization for greener pastures.

5. Carefully analyze the situation and realize that in many cases, the investment in developing an interface, using a software platform like MilramX, can result in substantial cost savings and improved efficiency for the organization.

One of the facts that we have come to recognize is that, even if we had developed an interface to their Podunk 5 ERP system for another customer, then it would be unlikely to meet the requirements of the new prospective user of BellHawk, working out of the box.

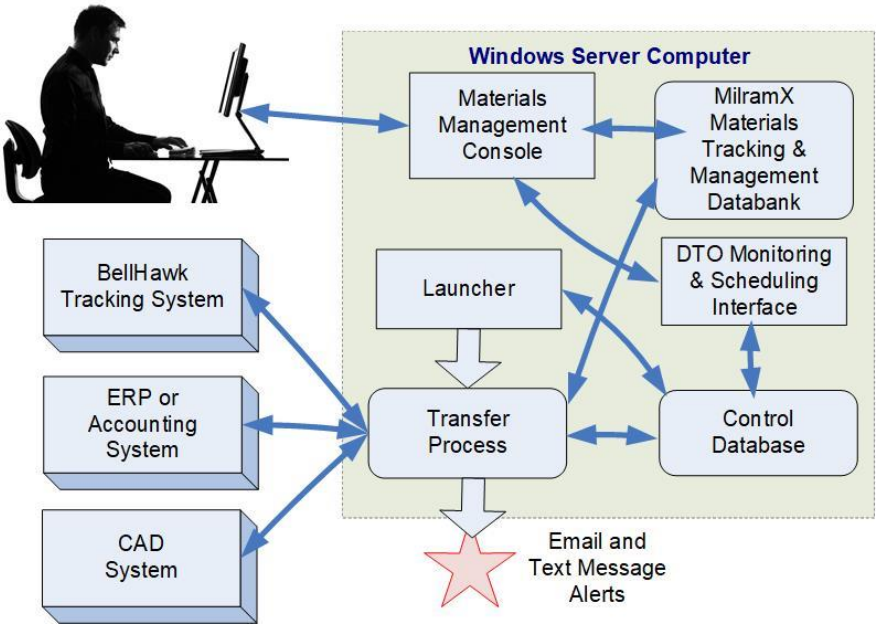
The reason for this is that, while the underlying database interfaces typically do not change from implementation to implementation, the way each organization uses its ERP system does.

One organization may want the inventory updated in its ERP system as soon as transactions are recorded in a material tracking system, like BellHawk. Another may want to update its raw materials inventory by "back-flushing" the inventory, based on expected Bills of Materials (BOMs) for products, after the products are made. This is done to avoid the "black-hole" problem of costs being incurred when raw materials are consumed, which may be days or weeks before the value of finished products are added back to the value of inventory.

Also, some organizations value WIP inventory, whereas others do not. Some organizations value certain materials at market or spot price whereas others value their inventory based on the purchase cost. Some organizations want to use customer sales orders from the ERP system and other want this data to come from an E-Commerce site. Some organizations need to send ASN data to their customers via EDI as soon as a shipment has left and others do not.

As you get into the needs of each individual organization, you quickly realize that there is no such thing as a standard interface, not even to a simple accounting system.

How MilramX Solves these Problems



MilramX runs on a Windows Server Computer and consists of the following main components:

1. A Transfer Process, which transfers data between different systems. It contains multiple subroutines called Data Transfer Objects (DTOs) that serve to transfer data, typically for one data object, such as a PO line record, from one system to another.

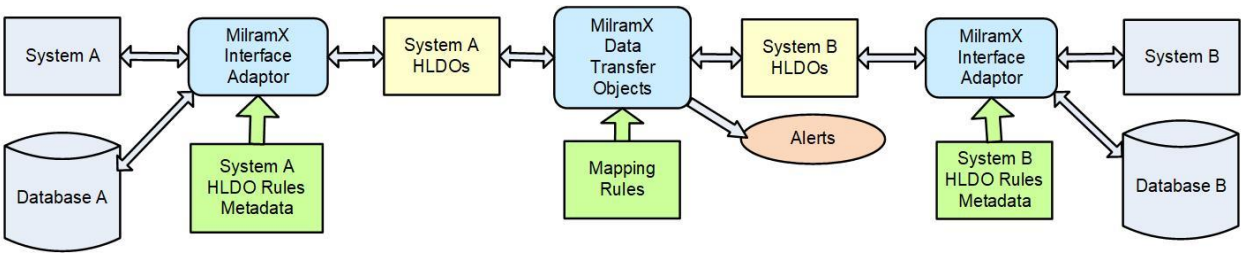
2. A Launcher, which runs as a system process and periodically runs the transfer process to transfer data objects using a specified DTO from a source system to a target system.

3. A SQL Server based Control database which stores all the scheduling and support information needed by the Launcher and Transfer processes. This also stores error reports and, if specified, a queue of the data objects which failed to transfer and can be retried, if needed, after editing.

4. A Materials Management Console. This interface, which can be remotely accessed through a web-browser, can be used to schedule the running of DTOs and to view and manage any errors which occurred during transfers. This management console is also used as the basis of custom interfaces to manage and visualize the state of projects, including creating purchase orders, pick orders, work orders, and ship orders in response to customer orders, if needed.

5. The Materials Management Databank, which is a database that contains the current status of materials as well as purchase orders, pick orders, work orders, and ship orders. This is maintained automatically by MilramX based on updates fetched from multiple systems and is used for activities, such as materials management, visualization, planning and scheduling.

There are also other components such as a daily log file of errors and a mechanism to notify an IT person that an error occurred in a data transfer. These enable IT staff members to remotely manage the transfers and to track down the source of errors and correct them, if needed.



Inside the transfer function there are:

- Adaptors which translate the fetching of data objects from complex database or web-services interfaces into simple High Level Data Objects (HLDOs). These Adaptors manage fetching just those data objects which have been added or changed since the last updates were fetched. They also manage the translation of HLDOs to be stored in the target system(s) into ODBC calls to store data into databases or web-services calls to store new data in external Cloud-based systems.

- Data Transfer Objects (DTOs) which:

    1. Fetch HLDO data from one or more source systems using the appropriate adaptors

    2. Translate this data into HLDOs to be sent to one or more target systems or into text or Email alerts, which are sent by the transfer function.

    3. Store one or more HLDOs in target systems using appropriate Adaptors.

- Metadata rules which control the actions of the interface adaptors.

- Mapping rules for 1:1 mapping between HLDO parameters.
- A mechanism to execute a Sequence of DTOs to make sure that different data objects are transferred in the correct sequence from source to target system.
- Error detection rules and code, which detect unacceptable characters fetched from a source system and has the ability to translate these characters into acceptable character strings for the destination system.
- An Alert mechanism which sends messages by Email or Text Messages to a list of recipients when an alert event is detected by a DTO.
- An FTP mechanism which will send files created by DTOs to destination FTP sites.

An HLDO consists of a keyword, such as “ITEM” and a set of parameter-name:parameter-value pairs of strings, such as “ItemNumber”:”ABC123”. Because parameter values are expressed as strings, they are independent of the data representation used in the underlying databases or interfaces. By convention we use “CamelBack” notation for parameter names, with no spaces (to avoid mistakes). This is similar to how JSON strings represent data.

Each HLDO has an XML metadata description, and corresponding Excel representation (for setup purposes) that describe how the high-level data object parameters relate to the schema used by the underlying database or web services interface. This metadata enables:

1. MilramX to automatically generate SQL code to translate between underlying databases and HLDOs exchanged between systems.
2. Verifying that the data being imported or exporting is in an acceptable format and does not contain any characters (such as control characters) that would be unacceptable to a target system or to code that is processing the HLDO data.
3. Setting of defaults for unknown values when data is stored into a target system.
4. Providing documentation of how HLDOs relate to underlying database, SDK, or web-services interfaces.

This use of HLDOs hides much of the complexity of the underlying data structures or web-services interface from the developer of the DTOs, which are typically developed using VB.Net but equally could be developed using C##.Net. This enables DTOs to have a very simple canonical form, which is as follows:

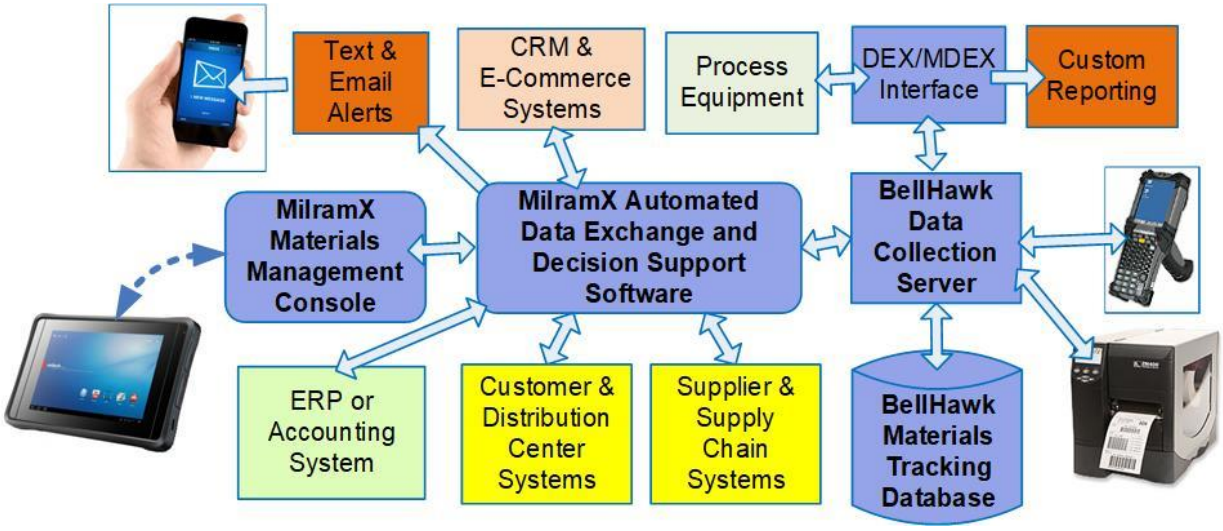
1. Use Fetch function to get latest updates to a specified data object from source system by simply providing the adaptor and HLDO name.
2. Use GetNextRecord function to get HLDOs resulting from Fetch one at a time.
3. Create Output HLDO from input HLDO
4. Use Store function to output created HLDO into the target system.
5. Repeat until there are no more Fetched records to process.

The only code that typically has to be created is that for translating the parameters of the source HLDO into those of the target HLDO, a sample for which is shown below.

```
DX_Receipts("SupplierNumber") = BH_Receipts("SupplierNumber")
DX_Receipts("PO") = BH_Receipts("PONumber")
DX_Receipts("LineNumber") = BH_Receipts("LineNumber")
DX_Receipts("ItemNumber") = BH_Receipts("ItemNumber")
DX_Receipts("UOM") = BH_Receipts("UOM")
DX_Receipts("Quantity") = BH_Receipts("Qty")
DX_Receipts("UnitCost") = BH_Receipts("UnitCost")
DX_Receipts("ExportDate") = BH_Receipts("ExportDate")
DX_Receipts("RefNumber") = strRefNumber
DX_Receipts("IsTransferred") = "N"
DX_Receipts("DateLastMod") = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")
blnSuccess = DX_Receipts.Store("C")
```

Once the HLDO's are defined, using Excel spreadsheets, and imported into MilramX through its web-browser interface then the DTO coding becomes very simple and yet retains all the flexibility of using a programming language like VB.Net for more complex algorithms.

MilramX and BellHawk Integration



One of the principal-uses for MilramX is in implementing data exchange interfaces between BellHawk and a wide variety of ERP and accounting systems as well as providing mechanisms for supply chain integration. It is also used to generate text and Email alerts to notify people when events occur in BellHawk that they need to pay attention to.

To facilitate this MilramX comes with pre-defined HLDOs for BellHawk and interface adaptors enabling data exchange directly with the BellHawk database as well as to exchange data with BellHawk through its web-services interfaces. From a MilramX DTO view point, these interfaces are identical and can be used interchangeably.

This enables MilramX to be installed on the same Windows Server as BellHawk and take advantage of the higher speed disk access to the BellHawk database. Or it can be installed on a

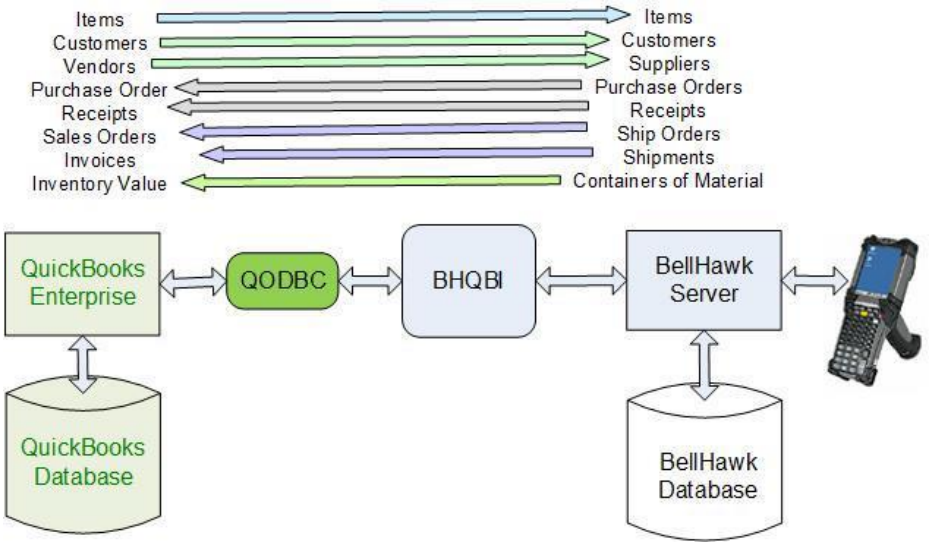
Windows Server at a remote location and communicate with BellHawk using BellHawk's web services interface over the Internet. In either case, the DTOs are identical.

A set of DTOs is available with MilramX which enable communications between the Materials Management Databank and a remote version of BellHawk. This enables the databank to be used as the basis of producing custom SSRS (SQL Server Reporting Services) reports, based on data captured in BellHawk, on a 24x7 basis.

Many of the mechanisms from MilramX are incorporated into the BellHawk DEX interface which automatically exchanges data between a SQL Server Express database installed on a local PC in a plant or warehouse and a remote copy of BellHawk through its web-services interface. The difference here is that the DEX interface program is run when needed to transfer data rather than transfers being made on a predetermined schedule, as is the case with the full MilramX implementation.

Interface to QuickBooks

To every rule, there is an exception (sort of) in that there exists a set of DTOs and associated HLOs for interfacing with QuickBooks Enterprise. These DTOs, referred collectively as the BellHawk-QuickBooks Interface (BHQBI) perform the data transfers shown below:



The QuickBooks adaptor for MilramX use the QODBC read/write Desktop interface for exchanging data with the desktop version of QuickBooks Enterprise or the On-Line version of QODBC for exchanging data with the Cloud-based version of QuickBooks Enterprise.

While these BHQBI DTOs have been proven to perform the data exchange tasks needed by a significant number of organizations, please be aware that, in almost every case, the DTO code needed changing to meet the specific needs of each organization.

Supply Chain Integration

One of the uses of MilramX is supply chain integration. When used with BellHawk, a MilramX DTO can use the BellHawk database records of shipped containers, together with related ship order and customer order data, to create Advanced Shipment Notice (ASN) data. MilramX then transfers this ASN data to third party EDI (Electronic Data Interchange) or FTP (File Transfer Protocol) systems for delivery to the warehouse receiving the shipment.

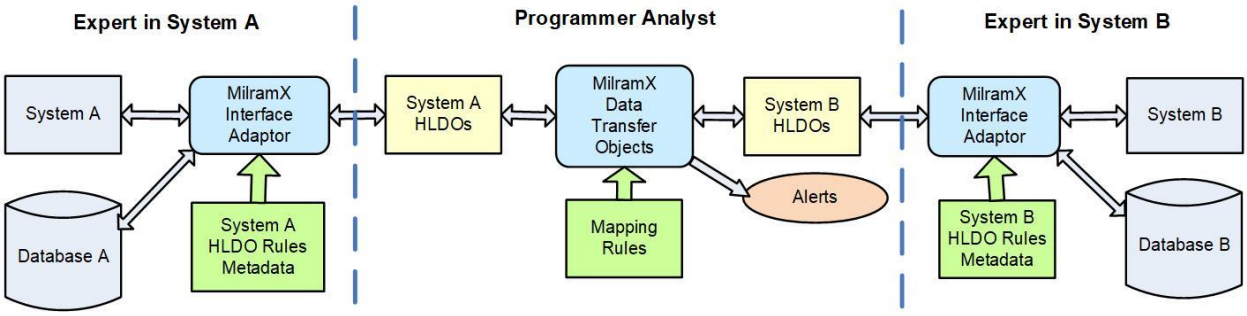
This ASN data can then be used by the receiving organization to quickly receive pallets of material by scanning the tracking barcode on the outside of the pallet and using the ASN data to receive its contents into inventory. Equally, MilramX can be used to receive ASN data from suppliers and to insert it into the BellHawk database, ready for the receipt of the pallets by scanning their tracking barcodes.

For products such as Pharmaceuticals, MilramX can also be used to send the traceability data captured by BellHawk to on-line repositories which can be rapidly interrogated by other systems to make sure that specific products are safe to use.

Interface Development

MilramX is a powerful tool which can be used to substantially reduce the time and cost of implementing automated data exchange interfaces within an industrial enterprise. But implementing these interfaces does need close collaboration between people who understand how each of the systems work and the details of their interfaces and the systems analysts who understand what information needs to be exchanged as well as what text message or Email alerts need to be generated.

This is facilitated by the structure of MilramX, where different people, who are experts in specific systems, can work on the HLDO rules for the different adaptors. Then these can then be integrated into DTOs by a programmer analyst who understands the information transfer requirements of the organization.



This solves the problem of finding one person who is an expert in the inner working of the two systems to exchange data, as well as being an expert in developing the data exchange code.

When interfacing to BellHawk, all the HLDO's and adaptor rules for BellHawk have already been defined but there is still a significant consulting role that the organization supporting the

BellHawk system need to play in implementing the interface. This is typically required to explain the nuance of the meanings of the predefined HLDO parameters.

It is best to use agile incremental methods to develop these interfaces, with one DTO being developed and tested at a time. This includes developing the HLDOs and the adaptor rules, needed to support the DTO. With BellHawk, this can often be done in parallel with its incremental deployment.

DTO Development

For those organizations without internal software development resources, the KnarrTek team can implement and maintain the needed DTOs. For those organizations with their own .Net developers a MilramX development kit is available. This kit consists of code libraries, as well as sample databases, a web-interface, and other tools, ready for inclusion in the Visual Studio IDE.

Please note, however, that, while MilramX is reasonably well documented, any and all support from the MilramX technical team in using the MilramX developers kit must be paid for in the form of prepaid Support Service Bundles.

Commentary

While MilramX can significantly reduce the time to develop the information exchange interface code, these interfaces, and especially their pre-deployment testing, can take a substantial amount of time, if done internally by an organization's own IT staff or expense, if an external interface development and testing team is used.

The resultant pay-off, however, can be dramatic in terms of improved operational efficiency, elimination of time taken for duplicate data entry, elimination of mistakes, and making sure that customer orders get out of the door on time. Also, the development of custom DTO’s to monitor operations and send alerts can take a significant amount of time if the monitoring task is complex, such as deciding whether the delivery of a customer order many be late, if there is no intervention.

The biggest advantage of MilramX is that additional DTO intelligent agents can be developed independently and added to the system, without the need to disrupt existing transfers. This enables the use of agile iterative development processes. It also helps ensure that the system can easily evolve to remain of great value to the organization even though requirements may evolve.

