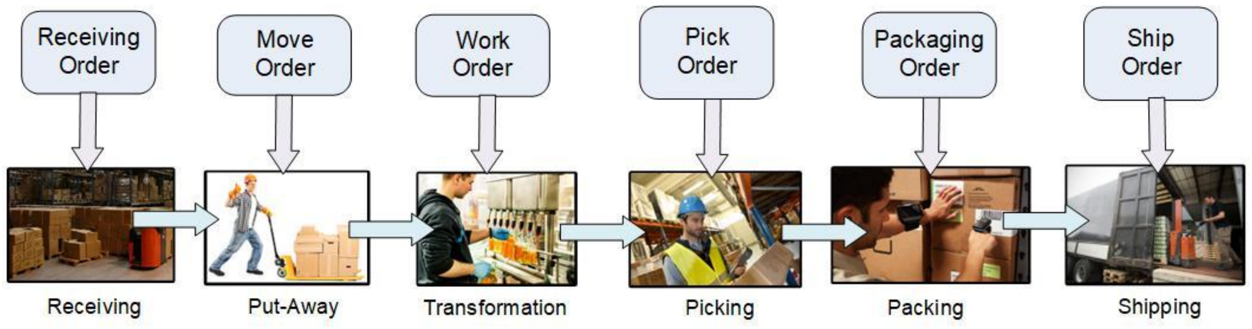
WIPtracker's Structured Customization Approach

Dr. Peter Green



WIPtracker is an Internet of Things (IOT) appliance that enables the real-time tracking and management of manufacturing and distribution operations within an industrial enterprise. It performs several functions, such as:

1. Real-time data capture using technologies such a mobile barcode scanning and RFID as well as automated data capture from weighing scales and on-demand custom barcode label and RFID tag printing.
2. Detection and prevention of operational errors such as using or shipping the wrong materials for an order or job or missing critical job steps.
3. Using Intelligent Agents to perform intelligent grunt work tasks such as preparing reports and exchanging data with a wide-range of other systems, such as ERP, CRM, and Business Intelligence systems.
4. Predicting when operational problems, such as an inventory shortage and a late shipment will occur and alerting managers and their staff.
5. Automatically scheduling jobs through a sequence of operations to help ensure customer orders get delivered or shipped on time.

Implementing the software to perform these functions involves many tens of thousands of lines of code, if not more, excluding the operating system, the SQL Server database system, and the web browser interface and web-services support system, such as IIS. WIPtracker, for example, with a dedicated development team has taken over a decade of development at a cost of several million dollars.

In an ideal world, one set of code would work for every manufacturing plant and distribution warehouse and could be developed once and used by many different organizations. Alas this is not true.

Every manufacturing plant has different needs because it uses different manufacturing processes that are the basis of its own unique competitive advantage. Every manufacturing plant and distribution warehouse has a continuously evolving customer base with continuously evolving requirements. Also, every plant and warehouse uses different equipment and uses a wide variety of different ERP systems in different ways.

So, how do we handle all these different requirements? Traditionally there have been several approaches used:

1. Write the operations tracking and management system “from scratch” as custom code to meet the specific needs of an organization. This is a time consuming and expensive process that is only practical for larger organizations. Even then, the organization has to pay for a team of expensive developers on an ongoing basis to maintain the software and to make changes as organizational needs change. Experience also shows that once the interesting work of developing a new project is complete, developers typically move on to newer and more exciting and career enhancing projects. All too often this leaves the custom software frozen into a form that no-one knows how to, or is available to, modify. Hence the system is used “as is” for several years before required changes necessitate its replacement.

2. Use an “Open Core” approach, now being promoted by all the major ERP systems, as they move to the Cloud, so they can have one body of software that is used in “multi-tenant” mode by thousands of different organizations. “Open Core” is a misnomer and should be labeled “Closed Core” as what it means is “use our software, as is, without any customizations”. Experience shows that while this may work for accounting and finance, with their standardized procedures, it totally fails to meet the needs of manufacturing, distribution, and other industrial organizations for a solution that need solutions tailored to their specific needs.

3. If available, use free open-source software maintained by an unpaid team of developers. While this may at first seem appealing as “something for nothing”, it suffers from the same problem as custom software development except that the developers are much more like to leave as they do not get paid. As a result, a company adopting open-source software for any mission critical application has to pay for their own developers to maintain software they did not even develop, which can be very challenging and not very exciting. As a result, the open-source software is typically used “as is” unless the organization pays for a “Freemium” version when consultants (typically the original developers) can be paid to modify the open-source software to the organizations specific needs.

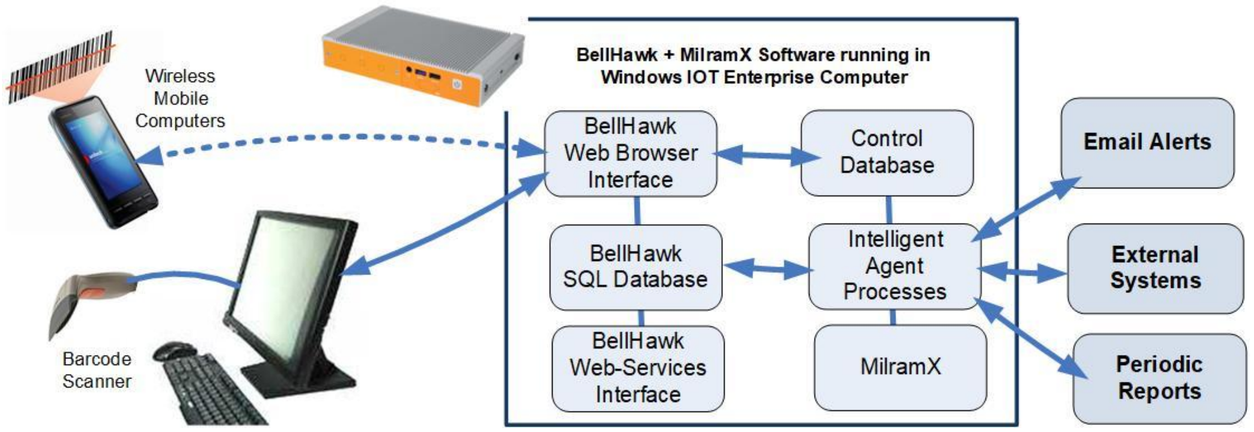
4. If available, purchase a source code license for software that meets most of your requirements. This overcomes the some of the disadvantages of open-source software but can incur substantial cost for licensing and training your own people so that they can modify the source code. The biggest issue here is that once your own people or consultants that you hire modify the source code, you are responsible for maintaining the modified version of the software, which can run to tens or even hundreds of thousands of lines of code. With a source code license, you can contract with the original developers to modify the source code for you, but that defeats the whole purpose of buying a source code license. A source code license only makes sense if you are integrating something like WIPtracker into your own product line to resell in quantity as part of your product line.

5. Use a “Structured Customization” approach, which is what WIPtracker uses. This is based on the concept that, for almost all operations tracking and management applications, over 90% of the code is the same. But there is always about 10% that needs to be customized for each application.

Our goal with WIPtracker, and the BellHawk and MilramX software contained in the WIPtracker Server IOT computer, is understand what customizations are typically needed across a wide range of applications and then to make it as easy as possible for business analysts, manufacturing engineers, and IT people to make these changes themselves.

In the rest of this white paper, I describe the different approaches taken in WIPtracker to make these customizations as easy as possible.

Structured Customization of WIPtracker



A WIPtracker server contains a BellHawk SQL database, which is used to capture operational tracking data and history about jobs and containers of material. Data can be captured in this database using a variety of barcode scanning equipped devices through the BellHawk web-browser interface. Data can also be exchanged with this database through a web-services interface.

In WIPtracker, the capture of operations tracking data is controlled by a set of expert systems rules whose parameters can be configured by importing these parameters in the form of Excel spreadsheets and then storing these parameters in the BellHawk database. These rules cover most of the industrial data capture situations that I have encountered in hundreds of applications over the past 25 years.

In this case, data collection customization is made much easier by the use of a web-browser interface. This avoids loading data capture software on each PC and mobile computing device. Thus, all data capture customizations can be constrained to the WIPtracker Server computer itself where the rules are much easier to change.

Also, the data capture capabilities can be extended by enabling users to define their own data capture parameters, again via Excel spreadsheets, while still retaining the ability to fully validate data before it is stored in the database.

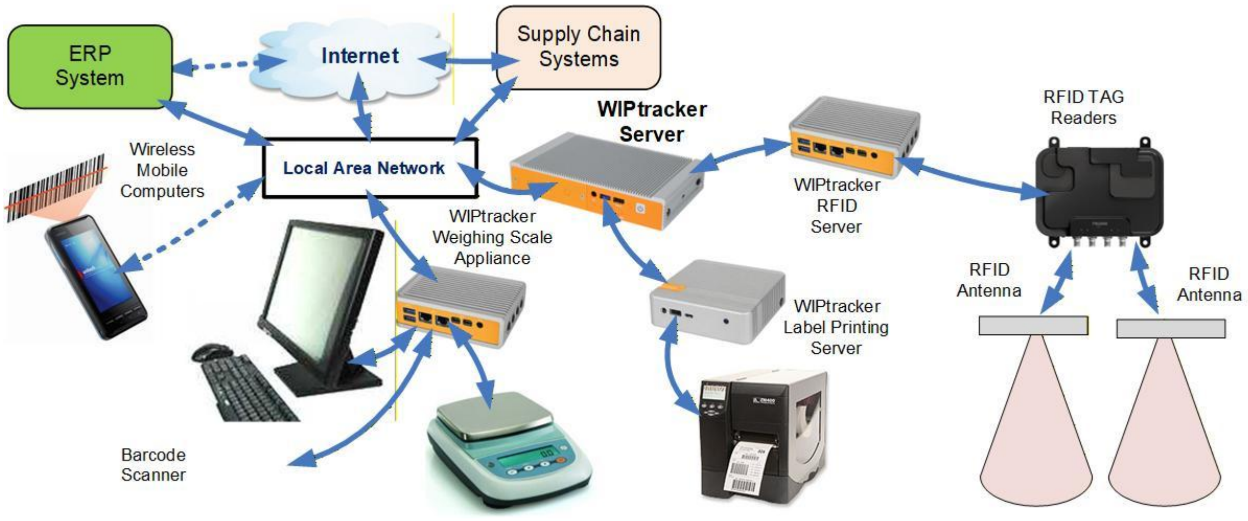
The BellHawk database itself, and its ODBC and web-services interfaces, can be customized and configured by rules that are imported as Excel spreadsheets and stored as XML metadata on the WIPtracker server.

This approach has proven to be very successful in customizing WIPtracker’s barcode data capture and operations tracking engine for a wide variety of applications, without the need to modify a line of code, although implementers are in-fact coding in a specialized expert-systems rules-based language, often without being aware of this fact.

WIPtracker also includes the MilramX Intelligent Agent mechanism, which is used for alerting, custom reporting, and data exchange with other systems, primarily through their REST interfaces. Here a different approach, has been taken with the Intelligent Agent Processes (IAPs) consisting of a Dynamic Link Libraries (DLLs) which contain large amount of standard .Net code and Python scripts which are used by system implementers as modifiable “Glue” code to customize the actions of each IAP.

By carefully choosing the interface between the Python control scripts and the underlying DLLs, we are able to reduce the control of each IAP to about a hundred lines of Python script code which can be written or modified by business analysts, manufacturing engineers, and IT people. This avoids the need for a “full stack” .net programmer to write and maintain custom code for each situation. Here MilramX provides 90% of the needed code as standard or automatically generated and the Python code provides the other10%, which varies from application to application.

For applications needing RFID scanning and interfaces to weighing scales, as well as barcode label and RFID tag printing/encoding we use additional IOT computers, which communicate with the WIPtracker server over its web-services interface.



In each case, we use a similar approach with 90% of the code being provided by efficient .net DLLs and the other 10% being provided by user modifiable Python code.

For details, please see the appropriate WIPtracker, BellHawk and MilramX documentation.

Commentary

This approach enables the rapid deployment of WIPtracker based systems across a wide range of applications. These initial deployments are often done by systems integration partners who are experts in WIPtracker. But then the client's own business analysts, manufacturing engineers, and IT people can modify the system, as needs change and expand, with only minimum involvement of the initial systems integration team.

Is this approach perfect? No, we still find situations where the underlying 90% standard code needs expanding to meet a specific need. Currently this has to be done by the development team that is in charge of WIPtracker. However, we regard these issues as opportunities to expand the underlying system and the "handles" we provide to implementers to enable them to do their own customization.

This approach has proven its worth in longevity of systems deployment, with some organizations using BellHawk and MilramX on their own servers for over 20 years before Microsoft forced an upgrade to the latest WIPtracker system. But, even then, we were able to migrate their BellHawk database and so "rescue" all that historical tracking and management data.

This lack of system "churn", even though several ERP systems came and went, has enabled stability of training and processes on the production floor and in the warehouse, with changes only being made to operational tracking processes, when needed, typically in response to the needs of their customers.

Author

Peter Green

Dr. Peter Green serves as the Technical Director of KnarrTek Inc. and Smart Operations Management LLC. Dr Green obtained his BSC (Hons) in Electrical Engineering and his Ph.D. Degrees in Electronics and Computer Science from Leeds University in England. Subsequently Dr. Green was a senior member of technical staff at Massachusetts Institute of Technology and a Professor of Computer Engineering at Worcester Polytechnic Institute.

Dr Green is a systems architect who is an expert in technology solutions for materials tracking and traceability in the food, medical, industrial, construction and defense supply chains. He has led the implementation of over 100 such systems over the past decade. Dr Green also led the team which developed the BellHawk barcode tracking, labeling, materials tracking and traceability software as well as the MilramX decision support and supply chain information integration software platform.

For further discussion, or to send comments, please contact pgreen@SmartOpsMgt.com.

Copyright