**Challenge:**

All current Symantec Workflow Projects no longer function when upgrading BuilderMT WMS to 5.8. WMS 5.8 brings many fundamental changes to the database engine including schema and security changes that require a new BMT API(v3) to interface with. Symantec Workflow is no longer a viable product due to Symantec as a company no longer in existence and end of life as a result. The current system is rigid and not scalable nor dynamic with the ebb and flow of the business.

Our current version WMS 5.7 is end-of-life and no longer supported by the vendor. This exposes the business to risk in the event a major issue arises and requires vendor assistance.

The BPMs have been identified as integral to our daily business processes and require upgrading. The inter dependence of WMS (Erp), Accounting (SAGE 300 CRE), and Estimating (SAGE Estimating) requires us to first update the BPMs to a state where they are version agnostic before we can continue with updating the other products in lock-step due to compatibility.

**Terms:**

BPM = Business Process Management App

SQL=Server Query Language, SQL Server

SPROC=SQL Stored Procedure

View=SQL Query stored as a premade object on a SQL Server.

GUI=Graphical User Interface

DLL=Dynamic Link Library

AD/LDAP=Active Directory

ACL=Access Control List

XML=eXtensive Markup Language

BMT=BuilderMT

API=Application Programming Interface

SSX=Sales Simplicity 10 Product

WMS= BuilderMT’s Workflow Management System Product (ERP)

SAGE=SAGE 300 CRE (Accounting and Job Costing)

SSMS=SQL Server Management Studio

PLM=Punch List Manager

**Design Philosophy Requirements for all Process:**

Typical terms; Robust and Scalable.

Ability to make changes to the app (email addresses, names, called reports, data sources) while not requiring entry to the design aspect and ultimate republish/recompile requirement of a project (reference Symantec Workflow) when changes are needed. Remove any static element within the application and create a dynamic approach that is processed at app runtime.

Instead of variables stored globally in the Process, rely on entries outside the flow that is parsed at run time so changes can be made on the fly such as network share locations, server names etc.

Application is data driven and dynamic to the data. Example: multiple divisions are determined by a value in a column of a SQL query as opposed to creating an application per division, and any new divisions do not require modify the front-end interface.

Control configuration of apps via an external configuration file or SQL data model.

Rely on a dedicated control database/table schema for tracking, history, auditing, control. Potential parallel/modular design that handles i/o between various Applications and/or databases where a user interface is a simple i/o between a SQL database like a shell. As much server-side processing and data manipulation as possible (could end up being more complex but removes the i/o from the user-run interfaces). Effectively creating a set of API’s instead of hard coding the GUI for data manipulation, triggers etc.

Modular; common components among all apps to be made into their own app with in out. Effectively an API.

Wrapped around SPROC if possible, where any GUI display is dynamic based on Columns returned. Allows for future dynamic updating of SPROC results without modifying the GUI (Forms, Variables/Arrays etc.).

Avoid 'DLL Hell' if possible. Crystal Report runtime specifically (JeffNET could still be a viable product as it is runtime independent, XML based and batch by design). Could replace the never-changing print-to-PDF requirements with static HTML. Crystal Reports is preferred as it handles data between SQL and Report and can be run independently outside of an application if required. Crystal Reports is also generally more universal and is backed by a large enterprise corporation with many support tools available. Crystal Reports is also capable as a no code report solution as it requires basic trainable knowledge to end users without any knowledge of SQL or other code or even data modeling. It is dynamic at handling different data sources, pagination and can be wrapped around SQL stored procedures with input variables to pre filter the data on the report.

Better Logging/Auditing to a SQL table.

Error-handling. Must not ‘stop’ the application if an error occurs and must provide meaningful error descriptions to better troubleshoot or fix the issue. Error logging to a data table.

Better Security - Active Directory/LDAP on IIS and AD ACL via IIS. Perhaps a table that maps SSX or SAGE logins to an AD login via a SQL table. Remove the requirement to code for security in any of the Apps and gatekeep via IIS and AD groups. Would require login when prompted. User LogonPersistent in the app as to avoid frequent prompts for login if this ends up happening.

If possible. remove the session-based flows where an email goes to a user presenting a link to a new entry point in a workflow; instead, a user receives an email with a link to log in where they are presented a list of action items with various batch options depending on the context of the application.

Platform agnostic and as best practice not too reliant on third party or obscure frameworks; HTML 5 as much as possible and framework (as common supported frameworks such as .NET. JavaScript) as best as possible.

An application-specific process/app management database that runs on the IIS server (in this case MORIIS) that only stores data for the configuration/control of the App (not user data). User data is to be stored on MORSQL. An example of user data is OE data (a record that represents the OE request).

Must be able to work within a web browser (Chrome, Edge, Safari, Firefox) without any third-party plugins and ideally out-of-the-box configuration common amongst all browsers. For example, nothing chromium-specific and HTML 5 (or newer) if possible, across the board. Sany function of any app needs to work on windows/android and mac osx/ios. Ideally nothing architecture dependent (specific to x86 or ARM in frameworks)

**Software Requirements on MORIIS**

BMT API v3 installation on MORIIS by BMT Support

WMS version 5.8.x

Sales Simplicity 20.x or newer - MORSQL instance, ssnet database

MS Visual Studio for development

3rd party: Crystal Reports runtime for Visual Studio (SP29 or newer) https://help.sap.com/docs/SUPPORT\_CONTENT/crystalreports/3354091173.html

SAGE 300 CRE - SQL instance to bridge SAGE to SQL via SQL 2014 32bit as VMORTL\SQLEXPRESS instance. SAGE ODBC has a username/password schema controlled by SAGE 300 CRE security that is not accessible outside of SAGE. SAGE cannot be written to without its batch processor and use of per module (JC/AP/AR) macros which are created in SAGE itself.

**Operating Expense (OperatingPurchaseRequest)**

1. Redesign to be quicker/simpler; data i/o via SQL table schema.
2. The user logs in and creates a singular request.
3. Email notification of a new Approval Request requirement to approver based on SQL query that determines Approver (gate keeper determined in SAGE G/L); approver logs into app and is presented with a list of outstanding approvals, and ability to see historical requests and re-add to SAGE import file if needed.
4. Approve All option from the list of outstanding requests with the ability to click to see the detail or modify the detail of the request.
5. Email notification to the original requester of approval or denial with reason.
6. Upload Invoice and attach ‘OE’ entry
7. Print OE to PDF to satisfy the existing physical requirement of original invoice plus stapled OE print submission to Accounting AP clerk.
8. The output of data into a SAGE specific format .csv file, appended to with each subsequent Approval. A BATCH file is run daily as Scheduled Task run the SAGE MACRO via command line that ultimately imports the OE into SAGE for payment processing.
9. All SQL queries to be based on premade SQL Views or Stored Procedures (or Functions) in the database schema when possible. This allows data manipulation server side without recompiling the front end. Example, new data sources and logic to filter data being fed to the front end.
10. Morrison Homes branding logo top left of interface page for all apps.

**Document Management (DocMan\_MainMenu)**

1. Redesign based on Design Philosophy requirements.
2. Same concept and design as the existing process; Ability to upload, download or replace a document to a specific Job using the BMT API v3.

**Trade Notification (SSSCalgaryNotification)**

1. Redesign to align with Design Philosophy requirements.
2. All drop lists based on a SQL query as per Design Philosophy outline.
3. History Tracking. Ability to resend as original batch (what vendors were selected etc.). Ability to override target email addresses attached to trades on the fly; add runtime only email addresses as targets.
4. Remove any requirement of login but a job number, the user presented a list of qualifying job's to select from that have not been notified but are ready to be notified (as in email are sent to specific trades as part of the job with the PDF's that are generated as per existing workflow).
5. All divisions (Calgary and Edmonton) and dynamic to that effect. Will require the use of SQL unions, with all SQL statements stored server-side as views or sprocs.
6. Still require the use of a Crystal Report to generate the PDF, either by JeffNET or Crystal Reports DLL wrapper (could be a created API or purchased API from SAP).

**CustomerLetters**

1. Redesign to match Design Philosophy requirements.
2. Fundamental must perform the same actions as the original app as per its logic flow in Symantec Workflow; determine stage of construction based on SQL query, determine customer contact information (email) based on Job, email customer a PDF printed version of the report and insert a date to a specific activity in WMs using the BMT APIv3.
3. Ideally to use existing Crystal Report reports (with subreports). They do most of the heavy lifting in that a process variable is fed as an input variable to the Report where the report handles the data retrieval based on a stored procedure. Could be a created API or Jeff-Net called report.
4. Reinstroduce Calgary to the app and remove manual process from Warranty Staff and PLM.

**ScheduleLookup**

1. Redesign to match Design Philosophy requirements.
2. Retain same functionality and purpose as original.

**Construction Job Sheets (JobSheets)**

1. Redesign to match Design Philosophy requirements.
2. Retain same functionality and purpose as original.

**Deposit Posting (SSSDepositPosting)**

1. Redesign to match Design Philosophy requirements.
2. Retain same functionality and purpose as original.
3. Never worked for Calgary (worked in Edmonton). Used almost exclusively by Shannon.

**Trade Push (TPortal\_Assign\_Trade)**

1. Redesign to match Design Philosophy requirements.
2. Retain same functionality and purpose as original.
3. May require further interviews with Derek Fiorante and Amy Hemminger as it may no be required any longer. Works in conjunction with Trade Vendor Reporting.

**Trade List (TradeList)**

1. Redesign to match Design Philosophy requirements.
2. Retain same functionality and purpose as original.
3. May not be required if an alternative is determined to lookup Trade contact information including how to store information. An alternative could include entering contacts into Exchange. Interdependent on the Trade Database.

**Update WMS from SSX customer data (UpdateWMSJobInfo2)**

1. Replaced by a SQL update query that would run server side on SQL sever daily.

**Construction Database Contact Management (Replace Access Database interface)**

1. Design to match Design Philosophy requirements
2. Create an interface to manage the Trade Database as per original MS Access front shell.

**SSX to SAGE (Replaces Access Database interface)**

1. May not work outside of ODBC due to username/password prompt and proprietary ODBC driver/interface
2. May be able to create macro for data import with .csv, similar to other SAGE import/ETL. Requires further exploration.