Profisee Group, Inc.

The MDS Desktop User Interface: Build or Buy?

Finding the best route to MDS value and productivity

Microsoft Master Data Services (MDS) is a powerful master data management platform, but it lacks good options for a user interface that is similarly "enterprise-grade". As a result, companies that want to use MDS as an enterprise MDM platform may be looking at a choice between building a custom UI on top of MDS (or getting a system integrator to build one for them), or buying an off-the-shelf solution, such as Master Data Maestro. This paper will help you understand what is involved in making a build-versus-buy decision when it comes to the MDS UI. It will walk you through important user experience considerations for the two primary data stewardship scenarios you'll need to support, and will present some of the key "build challenges" inherent in each. Finally, it presents factors to consider when calculating a likely return on investment for the build versus the buy scenario.

As an enterprise using Microsoft's Master Data Services (MDS), you already have a master data management database that is powerful, adaptable, secure and scalable. You can easily change schemas or models to meet your needs, you can rely on its solid security features, and, at the database level, you can build hundreds of millions of members in a single domain. In fact, it will scale as much as you want, as long as you provide the necessary hardware and SQL Server licenses. At the database level, MDS is truly "enterprise grade".

However, if you have looked at the MDS User Interface (UI) template, and the recommended Excel add-in, it's fairly easy to see the limitations of the MDS UI for any sort of robust, enterprise-grade MDM solution. Chances are you have already concluded that it isn't going to work for you, and now you are faced with the decision to either build or buy a better one. You may be thinking the solution is to build the UI yourself, or get a system integrator to build it for you. This paper will help you understand what is involved in such an undertaking, and introduce you to the available Master Data Maestro UI for MDS, and the Maestro SDK, as the fastest, most cost-effective route to deliver the full power of the MDS database, and provide an optimal user experience and superior productivity tools for your data stewards.

Beyond the Buy Decision: Case Studies from the UI Minefield

There are a number of case studies that illustrate the cost and time overruns and functional under-delivery inherent in the decision to build an MDS UI in-house, or similarly, to outsource the work to a systems integrator. For example, a global beverage manufacturer decided to go the Excel route, and engaged a leading consulting firm to build a UI that would enable their users to pull the necessary information from MDS into Excel, make changes in Excel, then pump it back into the database. In theory, it can be done. Certainly that is the implication of Microsoft's offering of the Excel add-in to enable data stewards to interact with the MDS database. And this particular corporation had what ought to be sufficient resources to make it work. In fact, they spent over $1.5 million to do so. Now, not only are they no closer to having a satisfactory UI, they are faced with having to pay an additional $25,000 for each change they need to make.

Another company, a world-leading semiconductor manufacturer, tried the in-house development route and had similarly disappointing results. When their data stewards found the MDS UI unusable in trying to manage their Product domain, the company decided to build their own. To begin with, they tried to use MDS web services, but found them too difficult to work with. Next, they spent a great deal of time trying to create a work-around using MDS subscriber views, but lost needed security features in the process. An average of four fulltime resources pursued these efforts for almost a year, with no successful outcome – they still did not have a usable UI for managing their Product data in MDS. In fact, they tabled the entire Product domain effort, and are now embarked on a Customer domain project in MDS – only this time, they are looking at buying the necessary UI functionality, based on their previous unsuccessful build experience.

Defining the User Experience

To begin with, let's look at what kind of user experience you want to support. There are two primary types of user scenarios, which we have defined as "workflow" data stewards and "power" data stewards, and chances are, you will need to support both of them. The desired user experiences are very different between the two scenarios, however, so we will look at the build-versus-buy implications of each one separately.

Workflow Data Stewards

The nature of the workflow data steward's role is one of infrequent interaction with the data, so they need direction; they need impetus whenever their input is required. For these users, the best option is to create a workflow-centric interface with very specific forms, web portals, and so on, to push specific tasks to specific people. For example, you may initiate a process with one data steward by sending a link to a form that says, "I need these three pieces of product information from you now," and then automatically routes the task to the next user to complete their part of the workflow, thereby orchestrating the tasks across users in multiple departments, while moving through the workflow as efficiently as possible, without gaps or delays. In this scenario, the master data member flows across many people who are performing discrete tasks – contributing, approving, reviewing – possibly on an individual record, or on a couple of related records; the opposite of the power steward scenario, where one person regularly performs mass changes on a large volume of data in a grid.

Power Data Stewards

The power steward spends a lot of time with the UI, working on a lot of records, handling frequent updates, massaging and entering data on a large scale. These users need personalized, reusable views of the data. They need the ability to do mass updates, imports, and edits. For the power steward, it's all about getting a lot of work done accurately and efficiently, so they need a more powerful, more productive general user experience.

Build vs. Buy Considerations

Both power steward and workflow steward user scenarios occur in any given enterprise and MDM implementation. Depending on your specific situation, you're going to have to provide supporting user experiences for one or both of these types of data stewards. Either way, your decision becomes whether to undertake the effort of developing what you need on top of MDS – build; or starting with a Master Data Maestro out-of-the box solution – buy. Let's examine the build-versus-buy considerations for each of these user scenarios.

Scoping Development Resources

If you intend to build, you will need capable development resources with a solid understanding of the UI requirements associated with each type of data steward. For an in-house development team, this also likely means incurring the risk and cost to develop the skills necessary to work with the MDS API – typically, a fairly steep learning curve.

Facilitating Workflow

High-performance template web forms, such as those included with the Maestro SDK, have built into them the wisdom, skills and experience of many person-years of UI development in the MDS environment. They benefit from the knowledge of myriad techniques implemented under the surface that make the forms interact as lightly – as quickly and effectively – as possible, such as caching the right information only, posting back in the right situations only, refreshing data in exactly the right situations, in the most efficient way. For example, when the user clicks on a cell in the web form and a drop-down list is displayed, do you want to have the entire web page refresh? Probably not; technically speaking, you would probably want to have an Ajax call to refresh just the one affected object. These sorts of advanced UI factors require considerable knowledge and experience, especially when working in the MDS environment.

Doing these things right can make this kind of development very expensive to do well, but these techniques are what make it all perform effectively. From the beginning, you need to determine whether it makes sense to throw a lot of internal resources at developing – and then

Build Challenge: Web Forms

If you are building a web form from scratch on top of MDS, you will need to figure out how to cache certain information so that it is readily available and doesn't have to be retrieved each time a user session starts. Without a good understanding of what information to cache, and why and how, building a web form from scratch – even a very basic form, such as adding a new product SKU or customer name – can easily result in something unusable in terms of performance; for example, a three-minute page load/publish confirmation.

Build Challenge: Notifications

Beyond the web forms themselves, perhaps you're also looking at building your own notification framework. In that case, you will want to look at how you are going to escalate things that have taken too long. Will you build the escalation workflow, as well? How will you design it? Are you going to hard-code the whole flow of tasks, or do you want to have something generalized that allows you to design any given workflow, and modify it as the business needs change? How will you to deploy it? If you custom-code your notification framework, what will be your deployment mechanism to get it from development into production when you make minor changes?

maintaining and supporting – an MDS UI from scratch, or whether your development resources would be better utilized, and your MDM deployment objectives better served, through the faster, more efficient drag-and-drop workflow or form building options offered through Maestro. These types of very detailed performance optimization techniques, and the knowledge of how to apply them in the MDS environment, are already built into the template web forms, out-of-the-box workflow tools, and built-in web parts that are included in the Maestro SDK.

Powering Productivity

Beyond power and performance, you will need to look closely at productivity, a key success metric for your MDS UI. If you decide not to take advantage of an out-of-the-box UI, but rather to build something, it is likely to be built very rigidly, as building a more flexible interface from scratch will take considerably longer, and will be significantly more difficult for your in-house IT resources to maintain and support. With the time and resource constraints around building an MDS UI in-house, you may end up taking a page from Mr. Ford's book: "You can have any color you want, as long as it's black." Only in this case, unlike the color of a car, the restrictions can significantly limit your data stewards' productivity: "You can have any data you want, as long as it's the customer entity and these ten fields, in this order."

On the other hand, flexible, customizable views are easily achievable with the Maestro UI; each user can personalize what they want to see, when they want to see it, and how they want to see it. They can filter down to their relevant information, and further, they can save that view and reuse it. As you begin to look at your users' requirements for a customizable interface that enables them to leverage the power of MDS, you will find that it is neither easy nor affordable to build the necessary capabilities from scratch.

In addition to easily customizable views, your power stewards will need to be able to make changes to a number of

Build Challenge: Custom Views

Power data stewards will tend to need many specialized custom views to perform specific tasks. For example, a product pricing specialist may need a specific view of products and pricing-related columns for performing common mass updates, while a production manager wants a completely different view of selected product rows and columns, and a finance manager in accounts receivable wants a specific view of credit line per customer account. These specialized view requirements are in addition to the many common shared views that will likely be needed across the company. Therefore, if you choose to build your own MDS UI, you will have to build an underlying interface structure that has the necessary scope, flexibility and ease of use to accommodate the key scenario of self-service for user view creation, customization, reuse and distribution.

records at one time, for example, where a new sales representative takes over responsibility for all of the customers in a particular territory, or when product categories are realigned, and a large number of products need to be assigned to a different category. In these types of situations, the power steward will need to be able to efficiently change, for example, 100 records at a time; it is impractical and time-consuming to make these types of changes if each record has to be selected and the affected attributes updated, one at a time. Therefore, the UI for power stewards must support mass change capabilities.

In order to support this functionality, the UI should allow the user to select a range of cells, and select a value to apply across multiple rows within the selection, in order to change the same property across multiple records. Another important element of mass edit capabilities will be to allow the user to copy one value and easily apply it to multiple rows. The user making mass changes must be able to review those changes on screen prior to publishing, and have the ability to easily undo and redo changes. Each of these aspects of the mass change scenario – just one of many power user scenarios – has implications for the development of the MDS UI. While this is by no means an exhaustive list of user requirements, it gives a glimpse into the level of detailed development skill and expertise that is required to create a robust UI for your data stewards to effectively manage master data in the MDS environment.

Build Challenge: Mass Changes from Import

In making mass changes, the power steward may not be keying the changes directly in the UI; instead, for example, they may be importing changes from a spreadsheet. For this, they need bulk import and merge capabilities that allow them to map the new information to the data model, and then apply changes all at once. The merging aspect of this requirement is key. The UI must be able to find and update existing records, versus creating new ones, and be able to set up a relationship; for example, when importing new products that will be assigned to a new category, as well as assigning existing products to the new category.

Build Challenge: Paging/Filtering/Sorting

In a typical power steward scenario where the user needs to change dozens of records from a dataset of millions, the UI will determine how quickly and easily those specific records are displayed on the screen. Or, where even more records are returned, if – as may well be the case with power steward tasks – thousands of records meet the selection criteria, the user should not have to download the whole list. The UI should provide an option to download, for example, the first 100 records, and then the next 100, and so on, so that the user can see what needs to be changed and act quickly, without having to wait each time for the system to retrieve the entire list again – a task that, if not properly optimized, could take up to five minutes per page-refresh. The user should also have the option to sort specific records to the top of the list, and handle changes in multiple different page sets of records.

Calculating ROI

With the considerations presented here, it is clear that a build-your-own UI effort is not a trivial undertaking. It requires significant knowledge and expertise around development in the MDS environment; a thorough grounding in master data management practice and principles; and a solid understanding of the requirements of your users. All of this adds up to a substantial investment of money, time and resources, but without a robust, reliable and flexible UI, you may well find – as did the companies in our "Case Studies from the UI Minefield" – that your MDM implementation never really gets off the ground. In that case you have lost on both your development investment, as well as the lost potential for return on your investment in MDS as an MDM foundation.

An investment in Maestro, on the other hand, returns the power, performance and productivity of a proven enterprise-grade UI that not only ensures a solid return on your MDS investment, it adds value above and beyond the practical scope of an in-house development effort, with features such as:

- Advanced master data modeling capabilities;
- Matching and survivorship functionality;
- Data cleansing and quality services;
- System adapters; and
- Predefined industry models.

Maestro also offers a level of ongoing training and support unlikely to be available from in-house development resources, and equally unlikely to be affordable from a system integrator. In the final analysis, even if build-versus-buy costs on the UI itself were breakeven; even if you could build, or have built, an MDS UI that could equal the usability and functionality of Maestro's; these added features clearly tip the value scale in favor of Maestro.

Appendix A: Delivering an Enterprise-Grade Data Stewardship Interface

The key to delivering an enterprise-grade MDS UI lies in being able to put the power of the database in the hands of your MDM users at the enterprise level; especially your hard-core data stewards. These could be IT professionals or business people within your organization. They are probably responsible for one or more domains of your MDM implementation – responsible for the totality of the domain in the enterprise – responsible for large subsets of attributes, with large subsets of hierarchies and roll-ups. In a midsize organization with perhaps a couple thousand products, this might be one person who is in marketing, who is the ultimate product management guru – the power steward – for the business. A larger organization might have dozens of such power stewards in different geographies around the world, each responsible for big segments of a very large domain of product, managing a model with tens of millions of members. And it is here, in the interface between the database and these power users, that MDS falls down.

The MDS User Interface (UI) template, and the recommended Excel add-in, simply can't deliver the necessary power at the kinds of database thresholds where these users work. They weren't designed to handle the volume, and as a result, using them is like trying to squeeze the water gushing from a fire hydrant – MDS model on the back end – through a straw – the MDS front end. For example, in the MDS UI, there is no ability to edit more than one member at once, which is completely implausible for anyone managing millions of members. There is no ability to clone members, there is no ability to make mass deletes, there is no ability to drag and drop. And Microsoft's offered solution, the Excel add-in, is equally insufficient to the task.

Excel: Not a Winning Formula

Excel is a spreadsheet program; nothing in it has ever been built specifically to support the data stewardship function, or the needed personalization required by the power steward. This personalization enables the data steward to create whatever view is needed, as many views as needed, for as many domains as needed, filtering the data in any way desired, so that stewards can create their own custom tabs, and their own sets of attributes. Further, they need to be able to share and publish these, in order to be able to work collaboratively more effectively. None of this happens in the Excel workbook environment. There, people who are not actually connected to the database server all the time are copying the workbook, thereby creating multiple instances of it, editing the data in whatever way each sees fit, and then copying and reproducing it, without any effective way to standardize or integrate their changes. All of this makes it virtually impossible to use Excel in a power data-stewardship environment.

It quickly becomes clear that the UI options provided with MDS are insufficient for an enterprise MDM solution, and you are left with the decision of whether to buy or to build the robust and flexible UI to meet your needs. This paper explores those options in context of delivering an enterprise-grade user experience on top of MDS. It identifies two types of data stewards, and touches on some of the key UI requirements that characterize each, and the build challenges associated with meeting those requirements.