ASPOSE
Your File Format Experts

CASE STUDIES

Aspose.Cells for .NET Case Study

ModelSheet Authoring and Customizable Templates
Using Aspose.Cells to build and maintain complex spreadsheet models

August 6, 2009

Product Background / Overview

Our goal is to enhance the power of desktop modeling by augmenting, not replacing, existing tools such as spreadsheets. With this target, we have developed ModelSheet Authoring, an environment for building and maintaining model-based spreadsheets, and Customizable Templates.

ModelSheet Authoring

As models become larger and more complex, they become harder to author and maintain by conventional methods. ModelSheet Authoring addresses this issue on several fronts. It separates model logic from sheet layout, while using web workbooks (similar to Excel workbooks) to visualize the model during the authoring process, so modelers can focus on business logic. It captures model logic with named variables that have dimensions, time series and data types. Formulas are expressed with named variables; they apply to regions of cells, so there are far fewer formulas. Cell addresses don't exist anywhere. You can build and evolve your model in ModelSheet Authoring, and then export it to Excel where formulas are expressed with cell addresses. You can also re-import any values you change in exported Excel workbooks to update and refine your original model.

ModelSheet Customizable Templates

A customized template is a flexible model that you can adapt to your situation by filling in a simple form, without editing a spreadsheet or its formulas. For example, you can specify time range and time grain; number and names of items in a dimension (such as your products and product families); and include or exclude major features. The resulting spreadsheet matches your needs better than any standard template.

Requirements Scenario

Our requirements for an Excel spreadsheet library included: support for a wide range of Excel capabilities including formulas, formatting, sheet layout and named regions; the ability to handle (very) large spreadsheets efficiently; access to advanced features like graphics and pivot tables; full .NET support; flexible licensing terms for SaaS and desktop

deployment; and responsive technical support. After evaluating the leading candidates, Aspose.Cells came out on top.

Solution Implementation

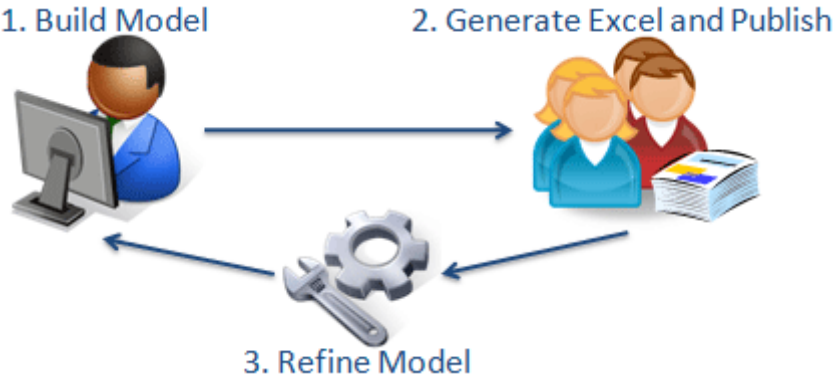
The main output of our product is a complex, fully functional Excel spreadsheet. Our solution is coded using C#, running as either an IIS ASP.NET web site or a Windows Forms application. We use both 32-bit and 64-bit operating systems (some of our processes require more than 4GB of managed memory.) The .NET version 2.0 Aspose.Cells DLL integrates perfectly into all of these scenarios. We use Aspose.Cells to create spreadsheets from scratch, filling in individual formulas, data and formats. We also merge existing worksheets into our output workbook, many of which contain charts and pivot tables, integrating them via named ranges. We use Aspose.Cells to do all of the reading and merging.

For more Information

If you wish to learn more about how ModelSheet uses Aspose.Cells, or more about their products please visit them at www.modelsheetsoft.com or contact them at info@modelsheetsoft.com.

Building a Model with ModelSheet Authoring

Using ModelSheet Authoring, the user builds models by working only with what's important – the pieces of information such as revenue and cost, and how they are related to each other. The user need not be concerned with how they are laid out on a spreadsheet. This enables the user to build large, reliable and expressive models quite simply. ModelSheet Authoring can then, using Aspose.Cells, generate Excel spreadsheets from these models and re-import any changes made to them as shown in steps 2 and 3.



1. Copy in data from various sources or type it in. Build and evolve your model using ModelSheet Authoring.

2. Export your model to Excel workbook(s) with a single click and distribute reports in the familiar and polished Excel format.

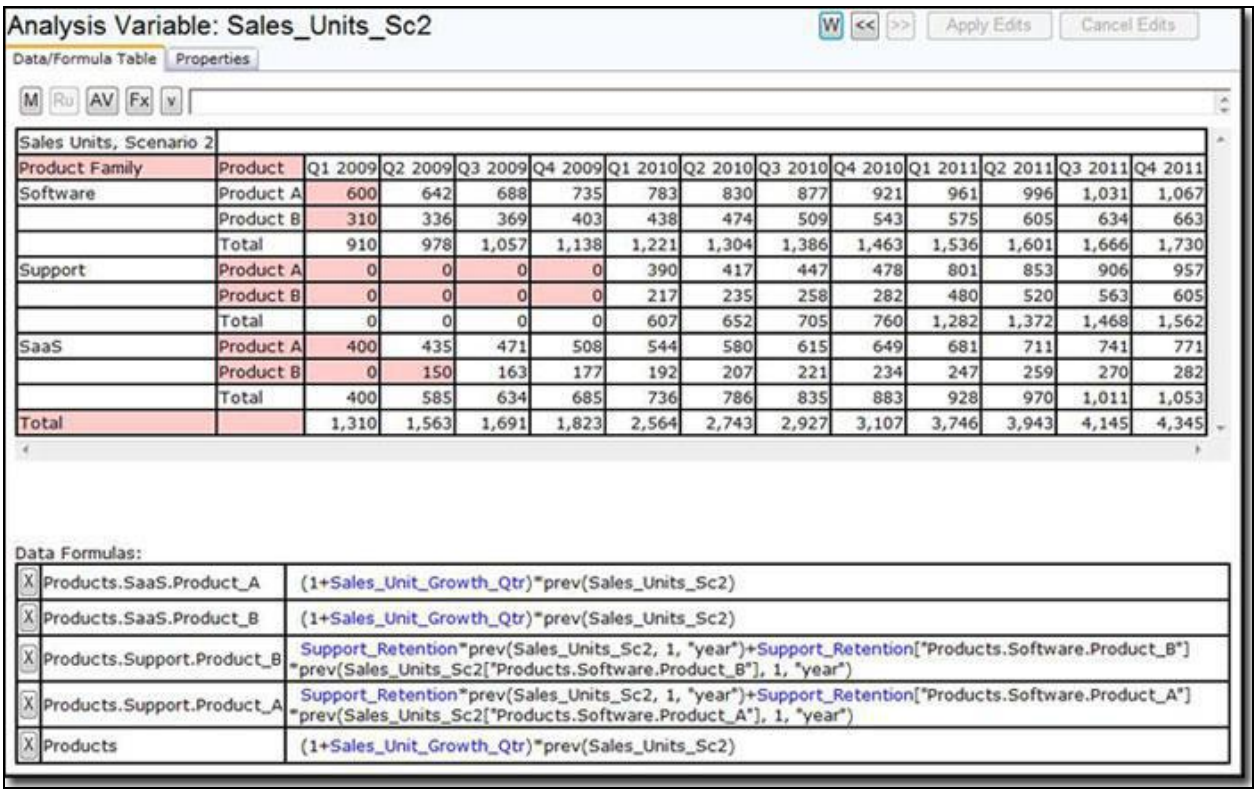
3. Capture changes made to data directly from the Excel reports. Refine and enhance model logic.

A Snapshot of a Part of a model in ModelSheet Authoring

This view shows the formulas for the sales forecast. Software sales for Products A and B are given in the first time period, and then grow by planned growth rates. Sales of support services are determined by planned retention rates of existing service customers, plus a planned percentage of new software license sales four quarters ago.

In this view, the pink cells contain the data and formulas (listed below the table) that define all values in the table.

Figure 1: Part of a model



Sales Units, Scenario 2

Product Family	Product	Q1 2009	Q2 2009	Q3 2009	Q4 2009	Q1 2010	Q2 2010	Q3 2010	Q4 2010	Q1 2011	Q2 2011	Q3 2011	Q4 2011
Software	Product A	600	642	688	735	783	830	877	921	961	996	1,031	1,067
	Product B	310	336	369	403	438	474	509	543	575	605	634	663
	Total	910	978	1,057	1,138	1,221	1,304	1,386	1,463	1,536	1,601	1,666	1,730
Support	Product A	0	0	0	0	390	417	447	478	801	853	906	957
	Product B	0	0	0	0	217	235	258	282	480	520	563	605
	Total	0	0	0	0	607	652	705	760	1,282	1,372	1,468	1,562
SaaS	Product A	400	435	471	508	544	580	615	649	681	711	741	771
	Product B	0	150	163	177	192	207	221	234	247	259	270	282
	Total	400	585	634	685	736	786	835	883	928	970	1,011	1,053
Total		1,310	1,563	1,691	1,823	2,564	2,743	2,927	3,107	3,746	3,943	4,145	4,345

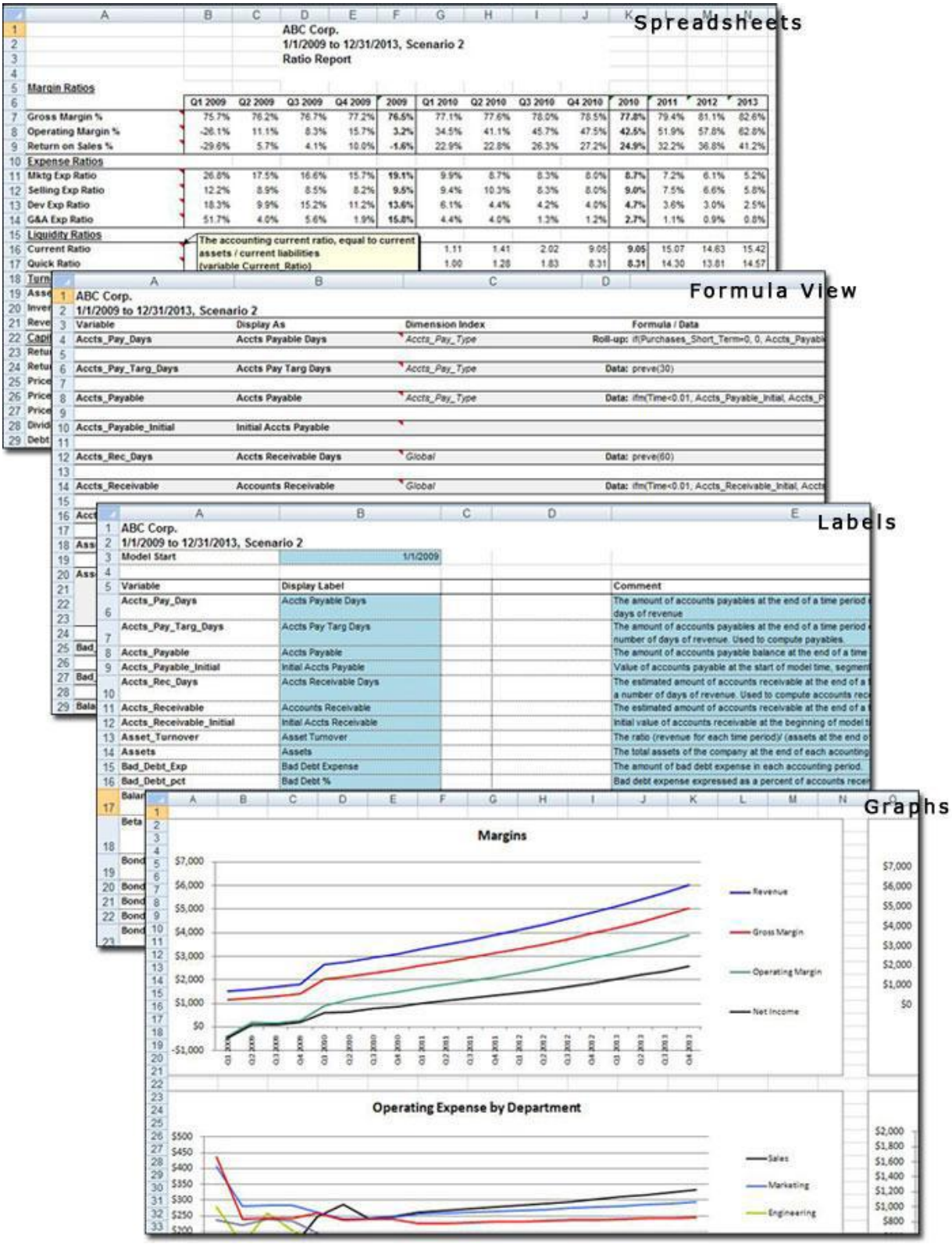
Data Formulas:

Products.SaaS.Product_A	(1+Sales_Unit_Growth_Qtr)*prev(Sales_Units_Sc2)
Products.SaaS.Product_B	(1+Sales_Unit_Growth_Qtr)*prev(Sales_Units_Sc2)
Products.Support.Product_B	Support_Retention*prev(Sales_Units_Sc2, 1, "year")+Support_Retention["Products.Software.Product_B"] *prev(Sales_Units_Sc2["Products.Software.Product_B"], 1, "year")
Products.Support.Product_A	Support_Retention*prev(Sales_Units_Sc2, 1, "year")+Support_Retention["Products.Software.Product_A"] *prev(Sales_Units_Sc2["Products.Software.Product_A"], 1, "year")
Products	(1+Sales_Unit_Growth_Qtr)*prev(Sales_Units_Sc2)

A Snapshot of Parts of an Exported Excel Workbook

ModelSheet Authoring models are exported to Excel using Aspose.Cells resulting in workbooks that are complex and detailed, making them ideal for making key inferences and decisions that can help users take action to maximize outcomes.

Figure 2: Exported Excel workbook



Benefits

In our situation, using Excel automation to read and write workbooks wasn't possible: it was far too slow and didn't work in a server-based environment. And, with our small team, custom-coding reading and writing of Excel didn't make sense. Aspose.Cells provides an off-the-shelf, cost-effective, well-supported solution that frees us to spend time on our core technology. Also, since we're using spreadsheet technology in a novel way, we fully expected we'd need additional support and features. We've found Aspose's quick response to be a real benefit and a critical factor in our success.

Future Implementations

We will likely enable ModelSheet to produce workbooks for Excel 2007 in the near future and for Excel 2010 in a year or so. We also expect to be generating charts and pivot tables from scratch.

Conclusion

Aspose.Cells has become an integral part of ModelSheet technology and products. It has worked well and enabled us to focus on our core competencies. The excellent technical support, especially the willingness to quickly add new features in response to our needs, has been critical to our success.