Paper 284-2010

So You Need a Data Warehouse But Don't Have the Budget to Build One: What Now?

How we designed and built a health insurance data warehouse while delivering BI projects

Steve Morton, Applied System Knowledge Ltd, Henley-on-Thames, UK

ABSTRACT

Building a data warehouse successfully requires discipline and an architected approach. Designs need to work towards a clear objective – even if that evolves over time – so that the technology can be fully utilized to deliver value to the business users.

However, getting the budget to fully plan a comprehensive data warehouse can be a challenge in today's economy; the era of large-scale data warehouse deployments has gone the way of the dinosaur!

This is the story of how we built a data warehouse behind the scenes while delivering business intelligence data marts over a two year period using only the BI project budgets.

Overall it has been a successful process, but there are lessons we have learned, and things I would do differently if I were doing the same again. I aim to share with the audience a few of those lessons, as well as what worked for us, in the hope that others can benefit.

INTRODUCTION: SETTING THE SCENE

In the middle part of 2007 I became aware of a UK Health Insurer that was looking at proposals to buy a new corporate data warehouse. This was intended as a strategic move to rationalize and upgrade their future delivery of management information and business intelligence.

At that time I knew of this company as an established SAS® customer with an extensive set of data marts that had been successfully built and deployed over several years. However, as with many sites which have grown organically over time, the whole collection was becoming unwieldy and difficult to carry forward for the long term. The desire within the IT department was to create a new data warehouse to store data from all the operational systems, which would then provide a well-structured source for future SAS reporting, analysis and data marts.

There was one big problem: whichever way the proposal was estimated for cost it came out as a very large figure – and whilst the management board understood the need to modernize their Business Intelligence, this scale of investment would not be sanctioned as a single project. The SAS development team had been deferring major new work while this proposal was investigated, only developing a couple of new data marts while keeping the 'legacy' SAS environment going with small changes. Meanwhile the pressure was building to better utilize the new SAS 9 BI Server infrastructure to deliver more to the business.

When it became clear later in 2007 that the ideal solution of a new Healthcare data warehouse was not going to get agreement to proceed, the SAS team management moved to actively pursue mobilizing "plan B". This was when I became involved – through a conversation with senior members of the Healthcare SAS team at the SAS UK Users Forum that November, which revealed that we shared a view that, with the right approach, a data warehouse could be designed and built incrementally. In this way the required data warehouse might be built using more modest project budgets, rather than looking for a big up-front investment. The SAS development team was short of the design skills to do this, but that was where I could help!

So it was in January 2008 that I started working with the team, aiming for a design that should evolve into the data warehouse that everyone wanted – but without the high initial costs. It would be paid for by project budgets, delivering the required BI for each one but building reusable data integration rather than just stand-alone data marts.

MANAGEMENT POSITIONING

It was important to have management agreement for what we planned – we did not want to be seen as "counter-strategic" to the idea of a corporate data warehouse, with the potential objections that might generate. So the positioning was agreed thus:

- We would set out to create strategic data marts, for use with the SAS 9 reporting and BI tools
- At the same time, our design for data integration would aim for a reusable shared content layer
- At worst we would have a tactical integration layer that could be replaced by a data warehouse in future
- At best we might create a strategic data asset for management information from a project-by-project building approach

This was accepted as a good way forward, and low-risk as it would still deliver business requirements for MI and would not require large up-front investment. If a corporate data warehouse did get future approval we could switch our source to that, whilst offering continuity for data mart users.

THE DOWNSIDE

The downside of the project-based approach was that we would not get extra funding to do more than the projects themselves required – we would have to do it all on project budgets alone, without increasing the cost to the business.

As a result of that we knew there were risks; particularly that we would not have the resources to fully explore data and business rules in our source systems, which meant that we might have future re-work if we missed data or misunderstood some logical relationship due to incomplete understanding.

We would also not be able to consult a wider selection of business users – after all, there was no budget provision to do so and we would have to work within 'project boundaries'. That would mean depending on the team's own skills and experience to decide what would go into the data warehouse at each stage, while also meeting the business requirement. This was probably the biggest challenge: extrapolating from a few specific requirements to envisage some broader use of the data we were acquiring. We did have some access to knowledge about the data, based on the older 'legacy' data marts, but were warned to not rely on understanding how the older programs worked because of their age and limited documentation.

TECHNOLOGY LANDSCAPE

Before I go any further I'll give a bit of background to our environment: we have Enterprise SAS Data Integration Server on Windows 2003 servers and Windows XP desktops, with Enterprise SAS BI Server also on Windows server systems; we also have dfPowerStudio on the desktop for data profiling. Storage of the data warehouse tables is in good old SAS datasets – the volume of data for a health insurer is quite capable of being handled with this technology – and we utilize compression for all tables. Business users access their reports and data via a web portal, with several expert users having SAS Enterprise Guide as well.

This SAS implementation infrastructure seems to be a fairly typical SAS 9 server configuration for an organization of this size.

As regards our source systems: new business systems are built on Siebel and supporting technologies, with Oracle and DB2 databases on midrange Unix servers; plus there are several major legacy IBM mainframe systems on DB2, VSAM etcetera which run the core of the older business processes and interact with the newer systems. Again, fairly typical of this type of business that has been around for many years and has been steadily modernizing systems.

We use CA AllFusion ERwin Data Modeler for data model design. Why? Because as a design grows it's important to be able to manage its complexity, so we need more than simple diagramming tools for the design. ERwin is one of the best-known data modeling packages around and has the advantage of being the only one I know that supports SAS as a target database – and hence can generate DDL to create or modify SAS tables, including labels, formats and informats. This is a great time-saver as it ensures the tables are set up in metadata to match the design with very little developer effort.

Now, on to the details of how the data warehouse design grew.

PREPARATION AND THINKING BIG WHILE STARTING SMALL

The team had already been thinking about the wide range of data that might eventually be included. A “cloud diagram” gave an impression of what could be included over time, and was already being used to try to stimulate management interest in the idea of a tactical data warehouse built to fill the gap until the hoped-for strategic data warehouse could be completed.

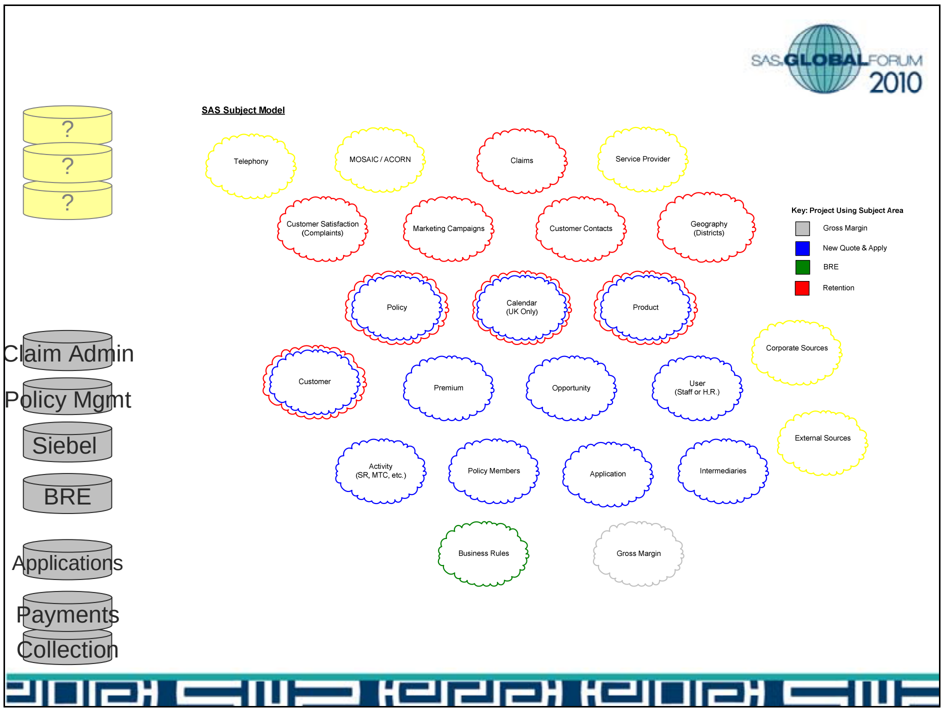


Figure 1 – “clouds” for everything

With a major business initiative for a new online Quote and Apply process, including new BI requirements to provide our first project for the new approach, we identified what data subjects would be involved and started to transition the diagram to a true “subject data model” as a precursor to designing for our first build.

As you can see in figure 2, there are still many clouds at this stage – we find this is a useful way to say to management that we recognize there is a lot more data out there!

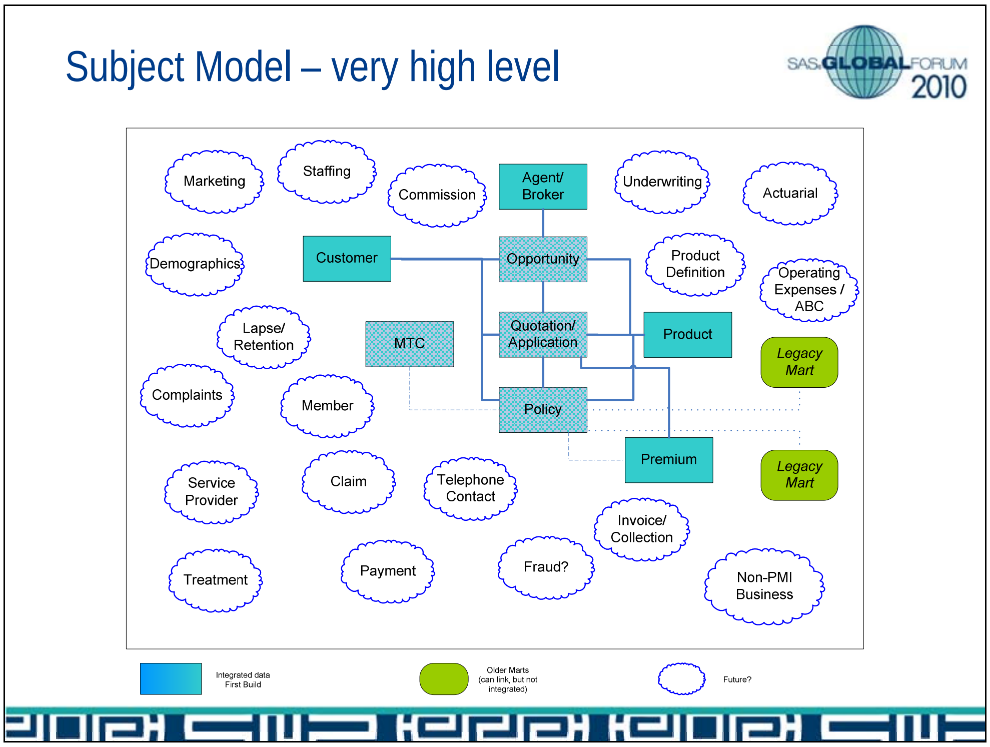
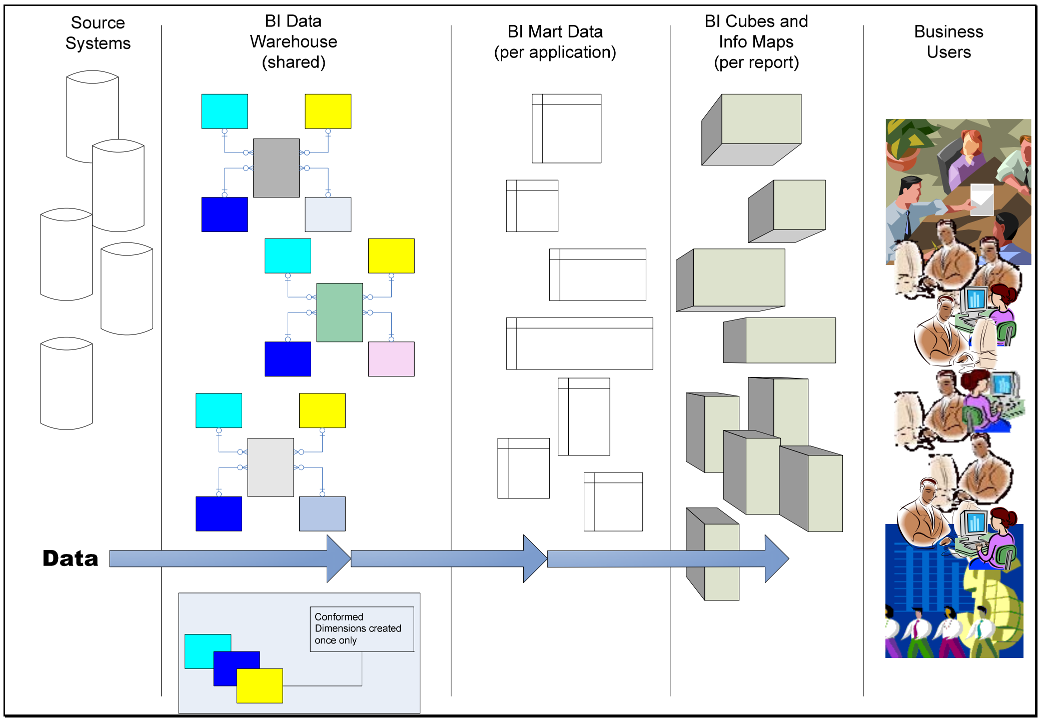


Figure 2 – “subjects and clouds” version 1

At the same time we defined standards for the environment, and a layered architecture that would enable the delivery we wanted to achieve. Most users would receive content through OLAP cubes and reports, but a few would also access tables directly – either at mart or data warehouse layer, as appropriate.


Source Systems
BI Data Warehouse (shared)
BI Mart Data (per application)
BI Cubes and Info Maps (per report)
Business Users
Data
Conformed Dimensions created once only
Figure 3 – layered architecture

We also set a principle that we would load all records from any source table we used – even if the initial requirement needed only some subset – and then do any sub-setting when creating the marts. In this way we should be somewhat prepared for new requirements from the same source in future.

We would also include all columns that seemed to be useful. This was actually the hardest part to achieve, due to constraints imposed by the project-by-project approach; when business analysts are focused on specified MI requirements only, it is difficult to investigate beyond those limits to understand what other the columns mean, and to look at secondary source tables.

DATA MODEL V1 – MAIN FEATURES

So the design started to grow with provision for Opportunities, Quotes and Applications in a set of dimensional models. We used the team's knowledge of insurance business generally, and the data available to us, to define dimensions that should conform - both within the context of our first build, and for future builds. Some of these decisions were obvious: there should be a Product dimension, a Customer dimension, Employee, Agent, Broker – and not surprisingly a Calendar dimension. We would also include event data from Service Requests and Activities to monitor the process workflow.

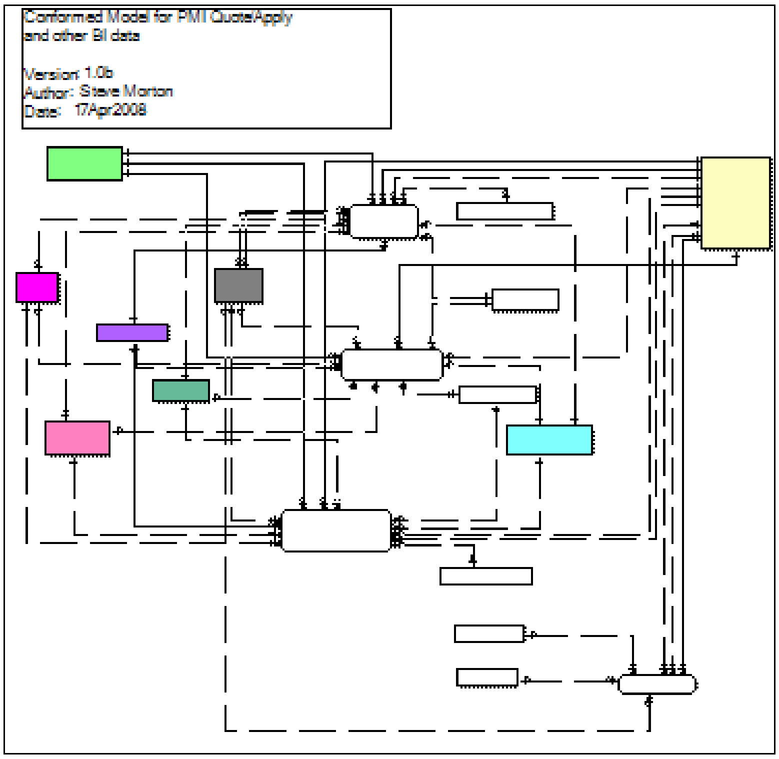


Figure 4 - Version 1 data model; four fact tables with conformed dimensions

Another important principle was to use dimensional models as far as possible, but be prepared to use normalized modeling when this seemed a better fit with the data. I'll comment on this again later.

THE BEST LAID PLANS…

So far so good – we were on track to build and deliver the new BI, with a data warehouse layer and data marts for user reporting that would meet the requirements. We also had new requirements lining up for the next project.

Then we got bad news: the new process we were supporting with the new BI would be going live later than originally planned. The implication was major for us, because the underlying database was changing at the same time. Although we were ready to go for July as intended, this meant we could not even put our new ETL into production, because the production database would not match our metadata! It also meant that business users would not be ready to perform User Acceptance Testing on our results, even if we could deliver them, because the whole schedule would be delayed and there would be no business user resourcing for UAT until the new operational system live dates.

This was a serious setback: we would have to delay going into production until the operational system update. On the positive side, the delay definitely wasn't our fault – and we could start to design and build version 2 of the data warehouse to include some new requirements. However we were taking a risk: if the operational system update were to be delayed further we could be left high and dry, unable to put anything into production, if our developments were too tightly integrated.

We decided the risk could be managed, because the next two projects involved stable, established operational data that would be able to go live in the data warehouse as soon as we were ready. If necessary we would partial-promote the second project first. So on we went…

SECONDS OUT, ROUND 2!

The second major version of the data model would incorporate event-log data from a business rules engine (BRE) and some basic Claim payments data from the mainframe DB2 system and link them both to Service Request and Activity data that was already in the data warehouse design.

This would deliver a first cut of Claims Reporting, and particularly combine the results of the Claims TeleInterview process with subsequent payment details. It would thus be our first integration of multiple systems into the new data warehouse – an important step!

GOING LIVE

So it was that November 2008 saw us going into User Acceptance Test with a series of new data marts and reports, underpinned by several new data warehouse subjects. These covered our original Quote & Apply data, plus BRE analysis, TeleInterview and Claims Payment details. All of this duly went live in January 2009 – a significant milestone for us.

The Subject Model diagram in figure 5 illustrates our progress at this point – a few Subjects such as TeleInterview and Service Request which we considered reasonably "complete", others "partial" with a lot of useful content but still more work to do, and a significant number of clouds remaining!

It is worth noting that the term "complete" is relative; it is always possible that future work will uncover new data for a Subject which we had previously considered done. In the same way we may also add new clouds when we discover a previously unknown topic of data.

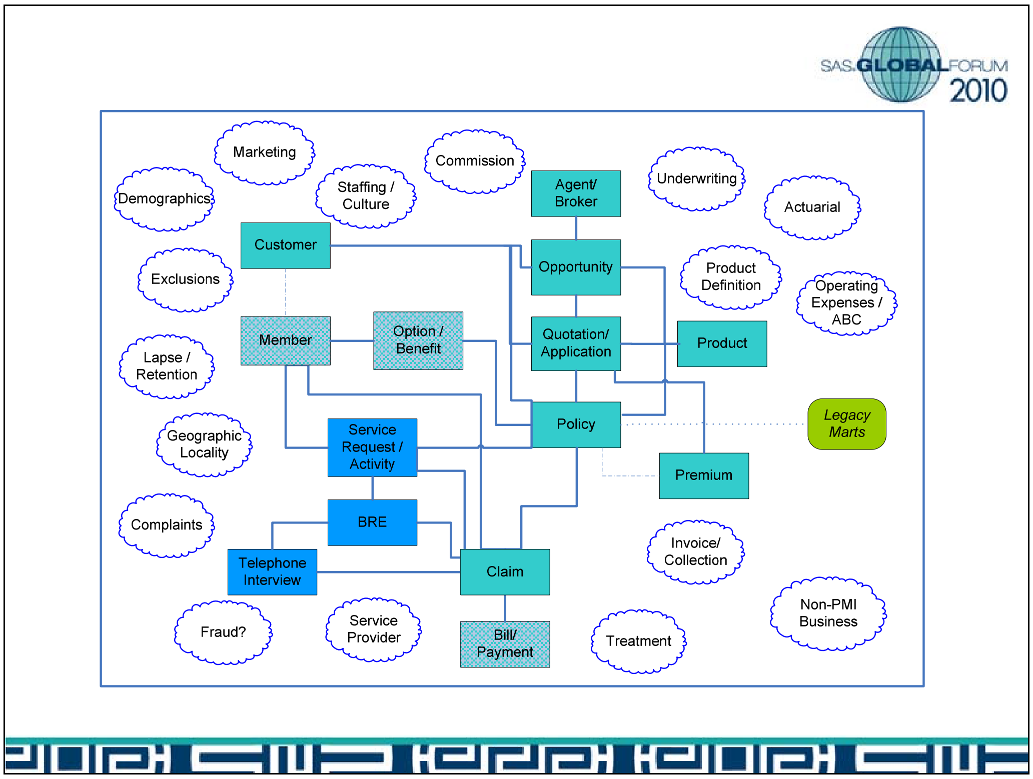


Figure 5 - Subject Model version 2

ONWARD AND UPWARD TOWARDS 2010

During 2009 we have continued this process of extending the data warehouse in stages. A third round of projects included more Policy, Member, Benefit and Exclusions details – this time from the legacy policy management system on the mainframe, since we learned that many of these details are not yet copied into the Siebel system. We also included Telephone Log data for Complaint handling, and made further use of Policy data we had already loaded. Then a further project for Claim analysis has added details of Claims Assessment and Treatments whilst extending the use of some of the earlier Claims payment data.

We ran into another setback, however, in relation to our provision of historical data. When testing with pre-production data, we found that older cancelled policies were not being fully detailed in the new systems, with only a 'skeleton' record copied in. This made good operational sense, as there would not be any further activity on these policies, but left us short of some detail for older cancellation history. To provide this we would have to get the older history direct from the legacy mainframe system – a process that has not yet been done, as it requires a separate budget to be agreed. In the interim, business users are combining data from the 'legacy' data marts with new data warehouse content to get the full picture, so the gap in our history is not critical, but we would prefer to eliminate this dependency.

A final cluster of projects for 2009 addressed Marketing Campaigns, Offers, Prospects, Responses and Quote details for direct sales-force, telesales and internet. This was our most ambitious build yet, adding more than 20 new data warehouse tables, and accessing three new source systems and several new Siebel tables. As we have gone on we have become more confident, as well as more efficient in managing the process, and we are seeing the re-use of data warehouse assets from earlier work with very little rework – just as we hoped when we started. To give some idea of scale on this, out of 80 existing tables in the data warehouse only 5 of them required changes for this major step in development.

WHERE ARE WE NOW?

Version 6 of the data warehouse model was completed in January 2010, with a good level of data coverage for major data subjects (see Figure 6, the version 6 subject model).

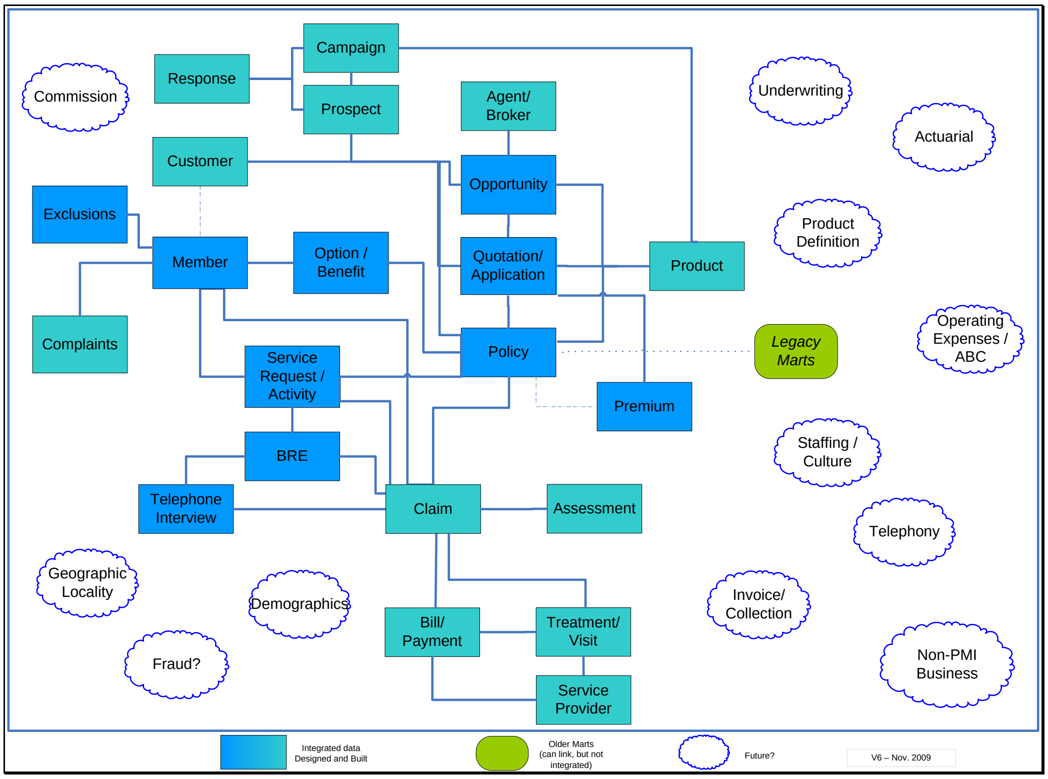


Figure 6 - Version 6 Subject Model (while under development)

We still need to populate older policy & member history by drawing from the legacy mainframe systems, an issue we did not expect initially as we had been assured that history for all policies would be loaded. Changing plans in the operational systems can have unintended consequences - another difficult lesson - even when the reasons are completely justified.

However the good news is that we are now delivering plenty of routine BI and data for analysis on the most important products, and hence providing real business value from the SAS data warehouse. We have achieved what we set out to do: an effective data resource for management information that is providing a platform for further development.

It has been recognized within the company that this is a successful way to proceed in an organization that prefers a stepwise approach to funding rather than ‘big ticket’ large scale plans. As a result we now have management support for more subject-based work in future and hence the additional business analyst and source system owner support we need to improve the design and build process.

As we move into 2010 our new projects include renovating the management dashboard delivery, adding strategic key result indicators to the tactical KPIs, reporting cubes and detailed analytical data we have delivered so far.

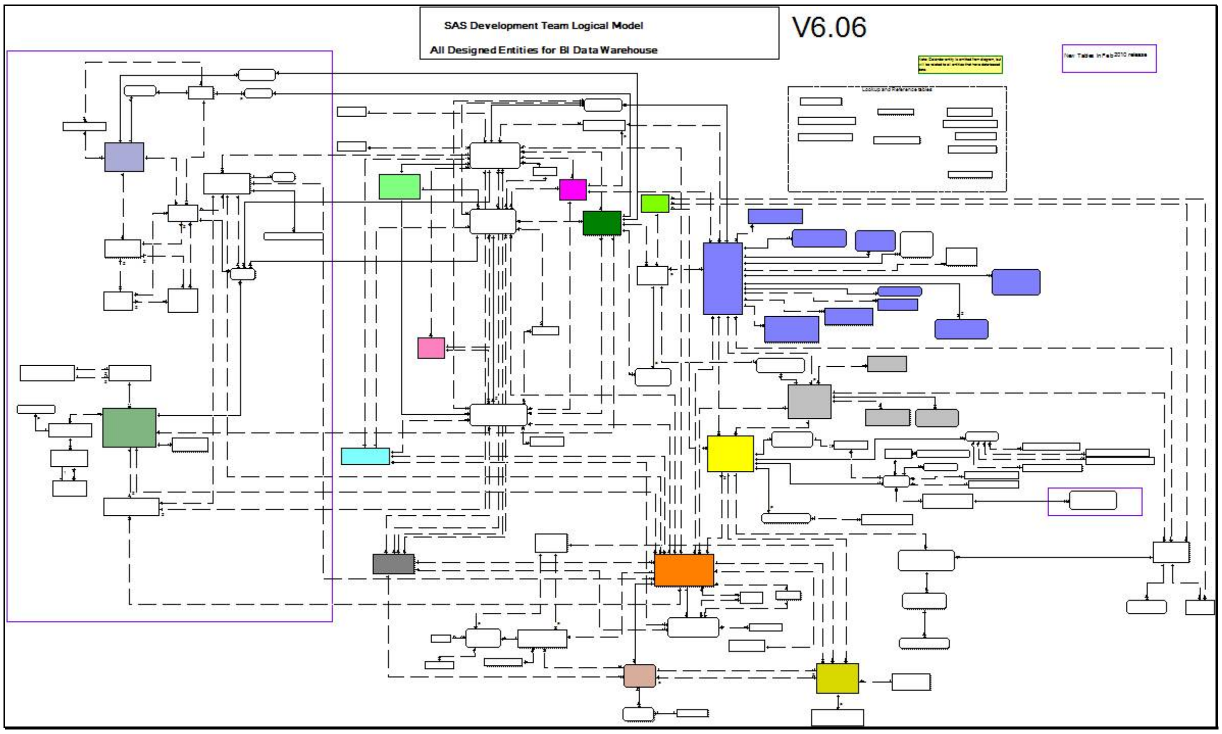

SAS Development Team Logical Model
All Designed Entities for BI Data Warehouse
V6.06

Figure 7 - Data Warehouse model version 6

Figure 7 should give you an idea of the range and complexity of what we have built so far. You should see from the diagram that the model is making use of star schemas wherever possible, but also has normalized and semi-normalized elements where dimensional models are less suitable. This gives us maximum flexibility in building, operating and extending the data warehouse. It also allows us to consider offering 'expert user' access to the data warehouse itself, rather than just the data marts. This approach to design is encouraged by Imhof, Galemmo and Geiger in their 2003 book: *Mastering Data Warehouse Design* and I have found it a good way to deal with a growing data warehouse.

You may also see that the latest additions are outlined on the diagram. This puts the new work in context, but I also provide sub-diagrams for each development phase which focus just on the new tables being developed in that round of effort. In this way the development team can focus on the new build, while we extend the overall span of data.

WHAT WORKED?

Some parts of what we did have worked very well.

- Flexibility over the use of Dimensional and Normalized modeling has worked well for the design, which has proven to be extensible with very little rework. We accept a small amount of redundancy as being an acceptable way to minimize rework.
- We have an effective data warehouse that supplies several new data marts for reporting and drill-down analysis. Daily updates ensure that these are timely and efficient for management information.
- We now find that when addressing new requirements we are very often able to say "we have that data already" and speed up the development or include requirements the user department thought would have to be excluded.
- We also found the Subject Model diagram ("boxes and clouds") very effective to describe our progress to management, and as a way to introduce newcomers to the scope of the data warehouse.

As a result of our successes to date we now have an official remit to take a systematic approach to data for a given subject when adding it to meet new requirements; this empowers us to work with a wider user community in a more

pro-active way. We should also be able to revisit some existing subjects to make them more comprehensive.
Most importantly we have proved that an incremental design can work, and that a 'grand plan' data warehouse is not the only way to deliver good BI. In these times of constrained budgets and careful cost management that is extremely valuable!

LESSONS LEARNED

Positive lessons:

- Incremental design can and does work when informed by solid experience in data warehouse development and access to good information about the source systems and the data in general.
- It's better to rely on a combination of established staff who know the source systems already working alongside contract resources, rather than staff new development projects with contractors alone – even when you need to strengthen the team with a few people who already have the required skills and experience.
- When source system experts understand what we are trying to achieve by accessing their data they can be very helpful. Learning how to correctly access and interpret their data is much more effective than having them provide just an example query that delivers only what the specific BI requirement demands – especially with complex, highly-normalized systems such as Siebel.

Cautionary lessons:

- Plans for operational systems can – and do – change, and the impacts for data warehouse and BI are not always appreciated by operational system managers making decisions. It is essential for the data warehouse team to be "in the loop" when decisions are made – not necessarily to alter the decision (although some influence can be helpful when choices are evaluated), but at least to ensure that the implications are picked up early and acted upon.
- A changing source system environment may have costly impacts when project staff are mainly contractors. Whereas permanent staff can usually be reassigned to another task when a delay occurs until the cause is resolved, there is an ongoing cost for contract developers who are usually brought in for specific project work. It really is important to keep most of the developers in place until UAT is complete, and this becomes difficult when the budget has mainly been spent!
- Operational test data is not assured to be useful for BI test data; the objectives are different, and if the data warehouse team relies on the operational test cases alone there will probably be insufficient BI test data. It is also likely that data warehouse and BI development will lag behind operational system live-dates for new business processes, so ensuring that the operational UAT databases remain online for data warehouse ETL testing and may require some strong negotiation by project managers!

WHAT WOULD I DO DIFFERENTLY?

There are a few things which I might do differently if I had the chance to start again.

- Try at an early stage to get a business analyst assigned to the team and ready to investigate "data subjects" rather than just "business requirements". This should allow the subject-oriented design to be better informed and reduce the risk of later re-work.
- Choose a first project using stable, established data in order to avoid developing the first section of data warehouse against new business processes that involve database changes, as this locks delivery dates into step with other teams' schedules. Doing this would have allowed us to get new material into production more quickly and reduced our stress levels.
- Ensure access to a well-populated 'UAT' version of the key operational systems at the outset, with the ability to create our own test case records. This also requires business analyst support to explain details of the real business processes and how the system is actually used to enable us to create meaningful test records for BI, and to confirm our understanding of data.

CONCLUSIONS

An incremental design for a data warehouse is a practical solution to budget constraints, provided that managers accept that there will be some rework in later phases. The risk can be minimized by good design practice, and will be lowest if a truly "subject oriented" approach is used, where each new part of the source data can be fully investigated and incorporated. However this requires consistent business analyst support and sympathetic help from the source system owners, as well as development managers who believe it can be done and who encourage the approach.

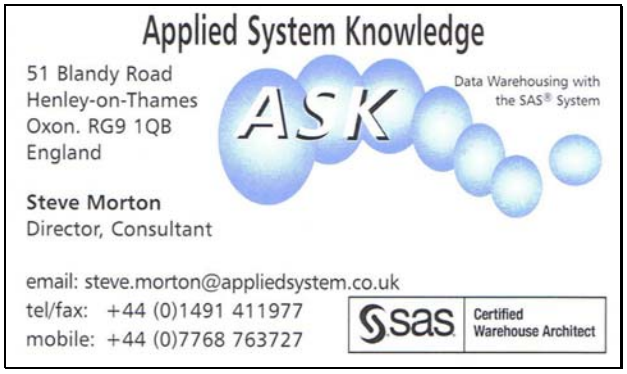
The author would like to thank past and current management of the SAS development team at this site, in particular Anthony Curtis and Guy Garrett for believing it could be done and getting this started at the end of 2007, and Paul Stephenson for his ongoing positive support and carrying the message to senior management to stimulate their approval for our efforts in 2009 and 2010.

REFERENCES

Mastering Data Warehouse Design by Claudia Imhoff, Nicholas Galemmo & Jonathan G. Geiger. © 2003 Wiley Publishing Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:



Applied System Knowledge
51 Blandy Road
Henley-on-Thames
Oxon. RG9 1QB
England

Data Warehousing with the SAS® System

Steve Morton
Director, Consultant

email: steve.morton@appliedsystem.co.uk
tel/fax: +44 (0)1491 411977
mobile: +44 (0)7768 763727

Certified Warehouse Architect

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.