Conversion from Corgent Diagram to Northwood GoDiagram

Consulting project for customer in Batch scheduling industry

Elmar Schumacher – aerobis Ltd.

elmar@aerobis.de

Background:

Recently I was faced with the task to create sophisticated Web diagrams for one of our customers. This customer is a company that distributes dedicated visualization tools for batch scheduling systems. Batch scheduling systems are still used by major companies (for example in the banking or insurance sector) to handle all the daily transactions. Although not very accessible or user-friendly, these host systems ensured fast and reliable transaction handling for years. This is the key factor for the companies that operate these systems and have created transaction-schemas with up to one million nodes!

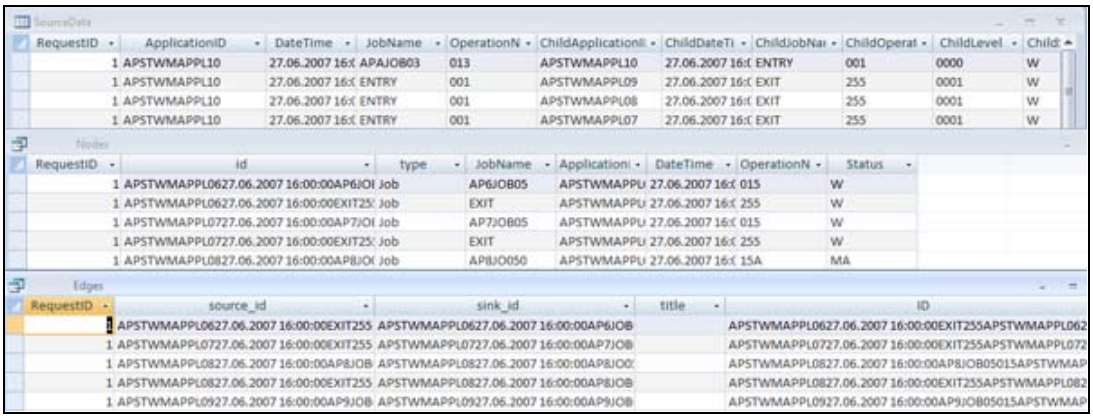
A lot of knowledge and experience has flown into these schemas to model the companies´ scheduling sequences over the years. But as it is a non-graphical environment - much like DOS - it is very hard to track errors or inconsistencies within these schemas.

My customer has a series of advanced tools makes all this scheduling information available on a Windows-based PC. The information is displayed graphical in various diagrams and can even be manipulated and uploaded back into the host system.

In addition to these desktop based tools the next stage is to make this scheduling data available on a company's intranet for everyone that has the corresponding rights to access the data. We originally chose Corgent as provider for the underlying diagram technology we wanted to use for our visualization. Although the complicated licensing was quite an issue for us we started to develop a prototype in cooperation with Corgent. Unfortunately Corgent decided to cease further development for their diagramming products and we had to find an alternative that helps us to leverage most of the development resources that we already invested.

Setup:

ASP.NET 2.0 based system that can be connected to a backend scheduling system. All data comes from that connection. The requested data is then parsed and buffered into a Microsoft Access database. Right now we are using Microsoft Access because it makes it considerably easier to convert the data at runtime and we don´t need to store any data across sessions. To make the data digestible for the diagramming component we convert the source data within the database with some procedures and views.



SourceData

RequestID	ApplicationID	DateTime	JobName	OperationN	ChildApplicationI	ChildDateTi	ChildJobNa	ChildOperat	ChildLevel	Child
1	APSTWMAPPL10	27.06.2007 16:(	APAJOB03	013	APSTWMAPPL10	27.06.2007 16:(	ENTRY	001	0000	W
1	APSTWMAPPL10	27.06.2007 16:(	ENTRY	001	APSTWMAPPL09	27.06.2007 16:(	EXIT	255	0001	W
1	APSTWMAPPL10	27.06.2007 16:(	ENTRY	001	APSTWMAPPL08	27.06.2007 16:(	EXIT	255	0001	W
1	APSTWMAPPL10	27.06.2007 16:(	ENTRY	001	APSTWMAPPL07	27.06.2007 16:(	EXIT	255	0001	W

Nodes

RequestID	id	type	JobName	Applicationi	DateTime	OperationN	Status
1	APSTWMAPPL0627.06.2007 16:00:00AP6JOI	Job	AP6JOB05	APSTWMAPPL	27.06.2007 16:(	015	W
1	APSTWMAPPL0627.06.2007 16:00:00EXIT25	Job	EXIT	APSTWMAPPL	27.06.2007 16:(	255	W
1	APSTWMAPPL0727.06.2007 16:00:00AP7JOI	Job	AP7JOB05	APSTWMAPPL	27.06.2007 16:(	015	W
1	APSTWMAPPL0727.06.2007 16:00:00EXIT25	Job	EXIT	APSTWMAPPL	27.06.2007 16:(	255	W
1	APSTWMAPPL0827.06.2007 16:00:00AP8JOI	Job	AP8JO050	APSTWMAPPL	27.06.2007 16:(	15A	MA

Edges

RequestID	source_id	sink_id	title	ID
1	APSTWMAPPL0627.06.2007 16:00:00EXIT255	APSTWMAPPL0627.06.2007 16:00:00AP6JOB		APSTWMAPPL0627.06.2007 16:00:00EXIT255APSTWMAPPL062
1	APSTWMAPPL0727.06.2007 16:00:00EXIT255	APSTWMAPPL0727.06.2007 16:00:00AP7JOB		APSTWMAPPL0727.06.2007 16:00:00EXIT255APSTWMAPPL072
1	APSTWMAPPL0827.06.2007 16:00:00AP8JOB	APSTWMAPPL0827.06.2007 16:00:00AP8JO0		APSTWMAPPL0827.06.2007 16:00:00AP8JOB05015APSTWMAP
1	APSTWMAPPL0827.06.2007 16:00:00EXIT255	APSTWMAPPL0827.06.2007 16:00:00AP8JOB		APSTWMAPPL0827.06.2007 16:00:00EXIT255APSTWMAPPL082
1	APSTWMAPPL0927.06.2007 16:00:00AP9JOB	APSTWMAPPL0927.06.2007 16:00:00AP9JOB		APSTWMAPPL0927.06.2007 16:00:00AP9JOB05015APSTWMAP

underlying data-structure

GoDiagram solution:

After looking at some alternatives we quickly realized that Northwood's GoDiagram would help us to accomplish everything we wanted to do – and more. The big question was how to accomplish a smooth transfer of your existing code to their product. Fortunately Northwood provided us with a lot of hands-on examples and some guidance in the conversion process and it turned out that we did not have to change anything in the underlying data structure. We actually took one of their examples that come with every GoDiagram installation as a basis for our solution.

To accomplish the transfer we simply created a set of GoNode and GoLink subclasses that had very similar appearance and behavior as those we had implemented using Corgent Diagram. This is simply a matter of choosing existing GoNode subclasses that are close to the desired appearance and behavior and then specifying the necessary properties or overriding the necessary methods to customize it further. With the large number of pre-existing node classes and the ability to override almost any method, this process is not too difficult and can accommodate a wide variety of diagrams.

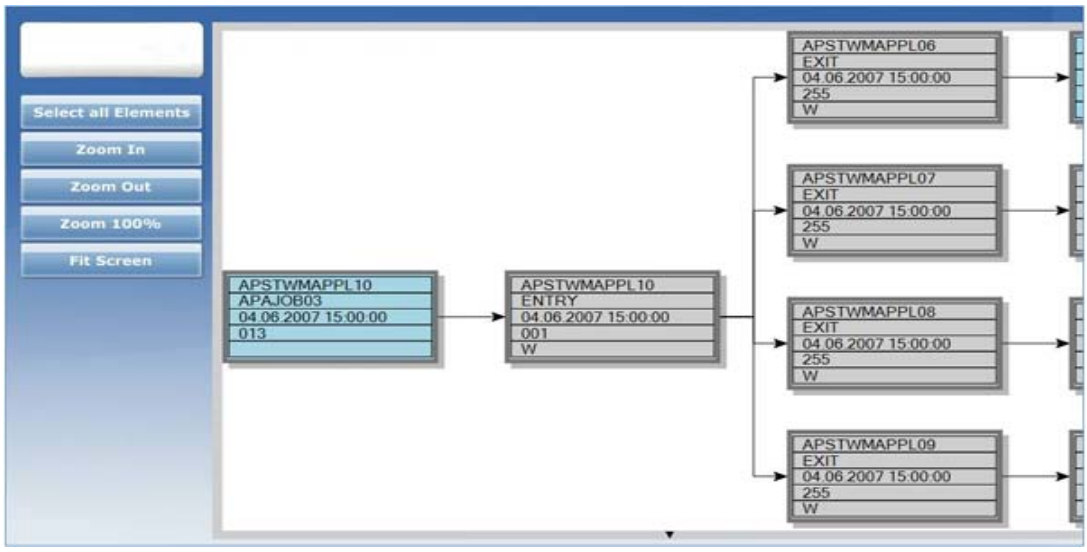
As mentioned above, we did not need to change our underlying data structure. GoDiagram is agnostic about any particular data representation or repository. It has classes and sample applications for persisting information both in XML and in databases, but there is no dependency on any particular format, so it is easy to accommodate whatever existing data format you may have.

I have to mention though that if you are coming from Corgent Diagram than you might have a hard time designing custom nodes as GoDiagram has no graphical style builder. So to get results quickly one solution is to take some sample code and tweak it until it works as expected.

In general I can say that GoDiagram is more flexible but – depending on your projects demands – might well involve more coding. For some things like "Ajaxified"-Dragging of the complete diagram you need to build a workaround to get a similar result. Right now we are doing this by selecting all elements within the diagram and moving them into the mouse drag direction as one object. It is not as responsive as a tile based Ajax-solution but the outcome is nearly identical.

Result:

In the coming months we will continue to add more diagramming features to the product but right know we were able to produce a first version of the web scheduling visualization software in a short timeframe. It renders dynamic Host scheduling data very well.



screenshot of a typical diagram pop-up window

We are very happy with the options that GoDiagram offers us. In the future we can make good use of the nested nodes features as well as interactive data changes triggered by a click action on the diagram. With all the options that GoDiagram offers us we can get very close -feature wise- our customers existing Desktop diagramming applications just that this time it all happens in the browser!

Thanks for the support Northwoods.