Using Data Integration Studio as a Computer Aided Software Engineering (CASE) Tool
David Kiasi, Applications Alternatives

ABSTRACT

With SAS® DI Studio, a SAS developer can now design his/her application, fully documenting it on the front end, in a structured manner; then migrate that design to the application with minimal effort. DI Studio can, if used correctly, create a structured environment in a painless manner. This presentation will discuss the fundamentals of structured analysis and design, show how data flow diagrams and data dictionaries can be created with DI Studio, and show how easy it is to migrate that design to the application.

WHAT IS A SYSTEM LIFE CYCLE?

What is a System Life Cycle? Definition: “The course of a program or system from the inception of the original idea through development, implementation, and maintenance, until it is either replaced or no longer valid.” Essentially, these are the steps we go through to create a final computer application.

HOW SYSTEM LIFE CYCLES ARE DONE

1. Instant Coding. Features of this approach are:
   • No paper trail
   • Documentation only in the code (or worse yet, the code itself.)
   • No one understands code but author.
   • If the specs are interpreted wrong, then re-coding and re-testing must be done
   • Limited testing-no hand off
   • Changes often are made after production has started-introduces errors.
   • No system design-it is an implied part of coding.
   • Files are typically created on the fly-result is that some required fields could easily be left out or formatted wrong.

2. Specs, Then Coding.
   • Specs are clearly defined- they serve as limited documentation
   • Documentation only in the code (or worse yet, the code itself.)
   • No one understands code but author.
   • Limited testing-no hand off.
   • Changes often are made after production has started- however, because specs were clearly defined, less errors are introduced.
   • No system design-it is an implied part of coding.
   • Files are typically created on the fly-result is that some required fields could easily be left out or formatted wrong. However, this is less likely to happen because the clear specs indicate exactly what fields should be the result of the application.
   • A code walkthrough is the only time other project team members get to review any design or coding issues before the code is tested but at least there is a review.

3. Structured System Life Cycle Development
   • Requirements Analysis involves Data Flow Diagrams
   • Specs are clearly defined
   • There is a formal design phase. The design acts as high level documentation
   • Code comes after the design is completed and reviewed by peers.
   • More time consuming
   • No incentive to update design after development is done.

WHAT IS CASE?

What is Case? CASE is Computer-Aided Software Engineering. "A combination of techniques and tools aimed at building and maintaining software systems of all types--large and small, commercial and scientific on-line and read-time. CASE tools provide coverage of the software life cycle by providing auto analysis, design, implementation, and maintenance, as well as project management of software systems.”

There are no popular CASE tools for SAS development.

Immediate advantages of a Case tool are that they:
   • Allow the design to write the code.
   • Combine design and development steps.
   • Allow design documentation to be current-the design is part of the development.

SAS DATA INTEGRATION STUDIO

SAS Data Integration (DI) Studio is a SAS product that is part of the SAS Business Intelligence software. Data Integration Studio is primarily used for taking
data out of a Data Warehouse and editing, transforming, and filtering data for use by end users. It is object oriented. Each step in the process is determined by process icons. Every data store is predefined. Although knowledge of coding SAS is helpful, one doesn't need to know how to code SAS programs to develop applications with DI Studio.

DI Studio has several features that lend itself to structured development. These features force a discipline on the developer that Base SAS code does not. DI Studio does this by:

- For each major step in the development process (DI Studio Job), forcing the user to define input and output data stores before any other development work is done. This is an intricate part of structured development, where data flows are the key component.
- When doing development, forces the user to create process icons within Jobs. These process icons are chosen from a Process Library. Examples are SQL, Extract, Transform, and Load. This forces the user to define each step by its function.

As I will show below, DI Studio can be used as a CASE tool for SAS development.

How can we use DI Studio as a CASE tool? My approach is to do a design with DI Studio first, ignoring the details that are necessary for development. Once the design is completed, one can generate documentation from DI Studio. This allows peers to review the documentation before development begins. Once development starts, it is not done from scratch because many of the objects needed for development have already been completed in the design phase.

**DI STUDIO IN THE DESIGN PHASE**

DI Studio is based on data flows so it enforces good practices for structured development.

1. Designing source data stores.

DI Studio information is stored in Metadata. This Metadata is a part of the SAS BI environment. As good structured analysis and design teaches us, we must determine what data we’re going to start with and how it flows toward the data we want to end up with. So first, we need to design our source data store. In this example, my source data consists of a simple SAS dataset.
d. You click the “Tools” pull down menu. In order to create a source within the Metadata, one clicks “Source Designer.”

e. You must select the source type here. It is the structure of your source data store. In this case, I’ll choose SAS because it is a SAS dataset.

f. DI Studio prompts you to log on again. This is to gain access to the server where SAS metadata exists.

g. Before I started DI Studio, I should have created a Metadata entry for the library that I am using in my personal area. This library, “kiasi_temp”, represents the directory on the UNIX workstation where my target dataset is stored.

h. Within that directory, I am going to choose dataset “TEST”. This is my source dataset.
i. In the Metadata tree, I have to choose a folder where the metadata belongs. I choose, "Ungrouped", since this is a demo and not a permanent entry for our project.

j. I click “Finish” on this window to complete the creation of the Metadata.

k. By right clicking the entry in the Metadata tree for “TEST”, I can pull up this pop-up, and choose, “Properties”, to view my Metadata for this dataset.

l. As you can see below, DI Studio has generated, from my SAS dataset, a well defined display of my data fields and their attributes.

2. Next, I will create my Target dataset. DI Studio requires you define your Target dataset in Metadata first. Then, when you populate the Target dataset in Metadata, it creates the actual SAS dataset on the server.
a. First, I click the Tools pull-down menu and click on “Target Designer.”

b. The first window comes up for the Target Designer. I click “Target Table” to create a target dataset. The other option is to create an OLAP Cube.

c. The next window, “Target Table”, comes up. On the Target table window I choose my Metadata data store name. Next to “Name”, I choose, “Cost Sum Table”. Also, I have a chance to enter a description. Here, I make a design decision, by describing what the dataset’s function will be.

d. The “Table Storage Information” window comes up next.

Here, I declare my Metadata library, which is “kiasi_temp”, which I created before. I can name my SAS dataset here. I call it “Cost_Sum_Table” which will be stored as Cost_Sum_Table.sas7bdat on the UNIX server.
e. The “Select Columns” window comes up.

Here, DI Studio gives me the option of creating data fields from other datasets. If the fields are similar, then the amount of work that I have to do, particularly when it comes to field descriptions, can be minimized. In a typical design environment, field information is entered by typing text entries into a document.

f. The “Change Columns/Indexes” window appears.

Here, I can create new fields for my dataset or I can modify the description or field names for the fields I have copied from other datasets’ Metadata.

g. On the “Select Folder” window, one is asked where they want to put the target’s Metadata. These folders are set up in advance by the project team. “Ungrouped” is an exception and is used as a catch all folder.

h. The “Target Table Designer” window, is the last window before the target data store is created. Again, this will not create the dataset on the UNIX server.

Our target is now created in the Metadata.
i. Target Properties Window.

By right clicking the entry in the Metadata tree for this target dataset, we can pull up the “Columns” tab, showing us the fields that make up our target.

3. Now, we want to create our Data Flow Diagram using DI Studio.

a. We first go back to the “Tools” pull down menu and this time we click on “Process Designer”.

b. The “New Job Wizard” window appears.

Here we name our “Job”. A DI Studio Job is a set of one or more Processes that make up a flow of data from one permanent data store to another. We may have interim data stores but these are not saved anywhere. This structure makes it easy to create a design. Natural breaks in the processing are determined by where data is stored.

Also, we can use the Description field to describe the overall function of the “Job”. This description is a part of the design and these will be used by developers to understand how development should proceed.

c. Select Tables Window.

On this window, we define what table we are loading as a result of the Job that we’re creating. I will go through the Metadata tree to find the dataset that I will populate.
d. Select Tables Window, 2.

Selecting the “Cost Sum Table”, the target data store that I’ve just defined, I click it and then click the arrow to bring it over to the Selected Tables list.

e. Select Folder Window.

Next, we determine where the Job should reside once created. One finds a folder in the Metadata tree.

g. The Finish Window.

This window completes the addition of the target data store to the Job.

h. Back to the Process Designer Window.

With this step in the process completed, we can see that we’ve started to create our data flow diagram. DI Studio always generates a Load process before a target is loaded.

Now, we need to continue to create the Job. Since the target is already in the diagram, I will choose to work backward from the target to the source.
Choosing a Process From the Process Library.

With DI Studio, each process icon is created from a list in the Process Library. The Process library contains any number of operations. Most of the ones that developers will use are in the Data Transforms directory.

Displaying the Processes available.

By clicking the folder, we can see the list of processes.

Here’s my problem. First, I want to separate my data into two parts, processing each part separately. Then, I want to do special operations on each set of data. After that, I want to combine the data together again. Working from right to left, I click on the “Append” process and move it over to the dotted box to the left of the Table Loader icon. What results can be seen below.

Next I use a SAS Splitter icon and move the two green temporary dataset icons over the empty boxes of the Append icon. The result is found below.
m. Before the data is split, I want to summarize it by Region and Employee ID. To do this, I click and drag over the Summary Statistics icon. The result is below:

As you can see, DI Studio has allowed me to create a Data Flow diagram. This same Data Flow Diagram will be used by the developer when he/she adds the actual process parameters, mapping, and filtering that will be used in the final application.

Now, we need to embellish our icons with documentation, so that we can tell the developer what to do when it is time for him/her to complete the application.

n. Now I need to move the green temporary table to the dotted box to the right of the SAS Splitter icon. The result is found below:

o. I now realize that I need special processes to occur once the data has been split. For Region One, one kind of operation is needed. For the other regions, another operation is needed. Dragging over two SAS Extract icons, splicing them between the SAS Splitter and the Append icons, I get the following Job. Later on, I will delete the SAS Extract for the non-Region One data because I don’t need to do anything to it.
p. I right click the Summary Statistics icon and get this pop-up window. I click the properties selection.


The properties window appears. It defaults to the “General” tab.

r. On the window, I will first change the name of my icon from “Summary Statistics” to “Aggregate by Region and Employee ID”, which is a much more specific way to label the process.

Then, I will use the Description field to enter a brief description of the process so that a developer will have specifications to complete development. This description will be part of the design.

s. Next, I’ll use the same process to rename the SAS Splitter icon to “Separate the Region One Data from the Other Regions.” Under the description field, I describe what the process does.
t. For the SAS Extract icon, I rename it “Multiply data by the Region One factors”. Under the description, I type the factor and how it will be applied to the continuous data variables.

u. For the Append Icon, I rename it “Concatenate the Data” and in the description I describe this process.

v. Finally, I modify the Loader icon so that I describe how the final variables are created from the input variables. I rename the icon, “Load sums of fields into the Cost Sum Table.” In the description, I describe the equation that will be used in the icon to create the new variables.

w. Completed Data Flow Diagram

Now, by looking at the final Data Flow Diagram, you can get a pretty good idea of what the Job is doing.

GENERATING A DESIGN DOCUMENT

Now, we're ready to create our design document from our Job.

The amazing thing about DI Studio is that after moving a bunch of icons together into a diagram, our diagram is now ready to execute. DI Studio automatically generates SAS code that can be executed, even before we've developed the detailed features of the application.
a. So, we go back to the Job window, and right click. The following comes up.

We generate the PROC CONTENTS by using the “View Statistics” option from the pop up window that appears when we right click the target dataset in the Metadata tree or on the Job diagram.

By clicking “Submit”, our Job runs.

b. Below, we can see that the job ran successfully.

d. The PROC CONTENTS appears in a window:

Why did we have to execute the Job? The only way to create the actual SAS dataset on your server is to execute a Job. Only until the target Metadata is actually loaded is the physical dataset created.

c. Once created, we can generate a PROC CONTENTS that can be copied directly into our design document’s Data Dictionary section. We didn’t have to hand create a table with Word.
We can export our Data Flow Diagram by using the “Save to GIF” option. To bring this up, we need to have the Job diagram showing. On the diagram we right click anywhere and the following pop-up appears.

By clicking the “Save to GIF” option, we then bring up the next window.

This window allows us to save the GIF diagram.

Below is the GIF diagram.

Now, you can also export your descriptions. However, an XML file is generated when you do this. In order to use this in a document, you’ll have to write a SAS program to parse the XML file.

To generate the XML file, you right click on the Job name in your Metadata tree. The following pop-up appears.

Below is a part of the XML file that is generated by this feature.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Export Type="Job" Version="9.1.3">
  <AddMetadata>
    <Metadata>
      <FeatureMap IsUserDefined="0" IsActive="1"
        Desc=""
        MetadataCreated="09Aug2007:22:07:42"
        Name="newmap"
        MetadataUpdated="09Aug2007:22:07:42"
        LockedBy=""
        TransformRole="ONETOONE"
        Id="$147"/>

      <SourceCode />
      <PropertySets />
      <SubstitutionVariables />
      <AssociatedClassifierMap>
        <Select ObjRef="$156" />
      </AssociatedClassifierMap>
      <Properties />
      <FeatureSources>
        <Column ObjRef="$64" />
      </FeatureSources>
```

Applications Big and Small
NESUG 2007
The scheduler is executed by the following pull-down window from the main DI Studio window:

Here, you choose “Redeploy Jobs for Scheduling.”

**DEVELOPMENT**

The design phase is over. Now development begins.

The developer can use the design to develop the application. He/she will use the layouts of the datasets, the Data Flow Diagrams, and the descriptions within each process icon to finish the application. He/she will now:

1. Go into each process icon and create the mapping of the fields.
2. Add the necessary filters.
3. Put the code in the windows where fields are modified, combined, summed, within each row of data.
4. Add SQL statements for joins in the menus.
5. Refine the scheduler.

**CONCLUSION**

DI Studio can not only be used for Extract, Transform, and Load applications when building a data warehouse. It can also be used as a structured design tool for any SAS application. The advantage of this approach is that design of applications, which in many shops is ignored entirely, becomes painless. Design and Development are interconnected. What was created for the design can be directly applied to development. This ends up saving time and also allows for a less error prone and well documented
application.

ACKNOWLEDGEMENTS

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.

I would like to acknowledge my client, the DBIA Team of the Decennial Management Division of the U.S. Census Bureau, who provided the SAS Business Intelligence software that this presentation was based on. Also my other clients, Centech, Inc. and Qualex, Inc.

REFERENCES


AUTHOR CONTACT

David Kiasi
Applications Alternatives
P.O. Box 4238
Upper Marlboro, MD, 20775

(301) 350-4752

E-mail: david.kiasi@appalt.com

Web page: www.appalt.com