CHAPTER 2

Developing Applications

When developing applications with SAS, it is wise to keep in mind some principles and best practices to follow. If you follow these principles, then they will help to avoid many of the common problems and pitfalls that developers are confronted with. These things are much easier to implement at the start of a project rather than part way through and will provide a lot of benefits for the investment of time and effort.

This chapter is aimed at the project manager or architect of a project, as they tend to be the person who thinks about the project as a whole. Sometimes this will be the programmer, especially for small projects. There are advantages to following these principles even with small projects, but the benefits grow as the size of the project grows in size.

Development Models

There are many ways to carry out development. I will briefly outline some of these now.

Freestyle Approach

The freestyle approach is how many untrained people develop things. It basically involves just diving in and starting to code with no planning at all. Of course, you need some idea of what you are trying to build, but that can be a sketch on the back of an envelope or a vague idea in your head. Many great systems have started this way, and often it is a valid way to innovate new solutions. However, typically it's not the best way to develop a big system or application especially when more than one person is involved in the team.

For many years, the most popular model for larger developments was the System Development Life Cycle (SDLC) pictured in Figure 2-1. When you look at this, it makes a lot of logical sense and in fact is pretty much the process any sensible developer would go through if doing it freestyle.

P. Mason, *SAS Stored Processes*, https://doi.org/10.1007/978-1-4842-5925-2_2

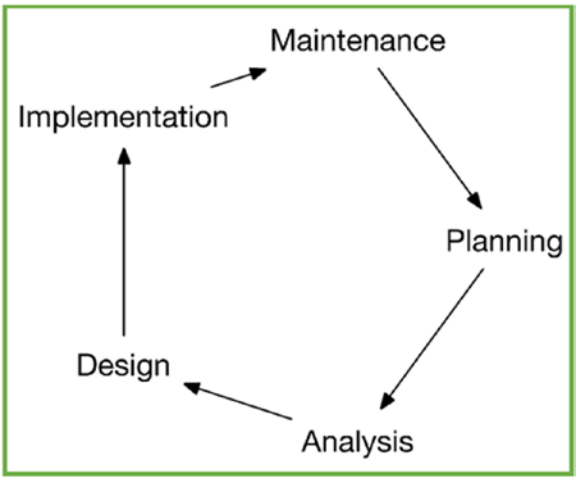

Figure 2-1. *SDLC/Waterfall*

This model can run into some problems when the stages become very prescriptive and lots of rules and guidelines are defined in an attempt to achieve best practice. I have seen companies where there are many long documents that must be delivered at each stage of this process which can mean that a small development that might take a day to write code for ends up taking four weeks to complete once all the documents, meetings, and stages have been done. This model is sometimes called the Waterfall model, as the diagram can be drawn as a waterfall from "Planning" to "Maintenance." Sometimes "Planning" and "Analysis" are replaced by "Requirements," and a "Verification" step is added after "Implementation."

V-Model

The V-model of software development is an extension to the SDLC model. Each phase on the left of Figure 2-2 has a corresponding phase on the right which is for validation. For instance, the "Concepts of Operations" is validated by the "Operation & Maintenance."

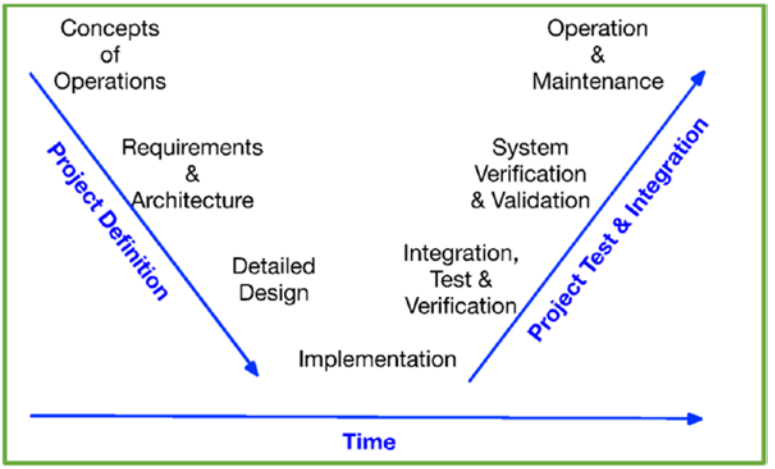

Figure 2-2. *V-model*

One common problem with this model is that it doesn't fit the needs of all people in the project. It's a nice way for a project manager to look at a project, but not the best way for a developer. And it is fairly inflexible although still probably a lot better than SDLC.

Agile

The Agile approach to development has shorter cycles of development and delivery so we get results quicker which users can see, which in turn affects further development. It is quite similar to another model called Rapid Application Development, which has been around for many more years. Agile welcomes changes in requirements, which other models don't because it means returning to an earlier phase of the process. Agile has many iterations of development, testing, and delivery, so users get things in weeks rather than months. Developers and clients work together closely on an almost daily basis. This model is sometimes preferred by developers and clients but is harder to manage for project managers.

Architectural Concepts

It is wise to keep a range of architectural concepts in mind as you develop your software. This is an incomplete list to remind you of some of the things to consider, which can lead to a better design:

- **Scalability issues –** The biggest scalability issue with stored processes used in web applications is the number of multibridge connections defined. The default is 3 and that is often far too small. You need your SAS administrator to increase this number if it is too small. You may also want to set up your code to run in parallel threads to improve performance. You might want to grid enable the code if you have SAS/Grid. You might want to make use of SPDS if you have that, since it can help improve performance on your tables removing the need for sorting (for example).
- **CRUD issues (Create Read Update Delete) –** This is often a consideration if you are developing web applications and want to create tables, update records, or delete things. It's easy to read data, but there are various issues around these other things. It is more difficult to use SAS tables with CRUD than using some other database systems such as MySQL or Postgres.
- **Browser differences –** If you're developing web applications, then you need to look at how you support different browsers. You can detect the browser you are using and potentially write special code to handle its differences or make use of a JavaScript framework which will handle many of these issues automatically.
- **Complexity of code vs. ability to support it –** You can sometimes write complex code and reduce the number of lines needed. However, you can usually achieve a similar level of performance by using a simpler technique that perhaps has more lines. But if things are easier for a future maintainer of your code to understand, then you are wise to choose the simpler code.

- **Platform differences –** Will your application only ever run on a laptop? What if it runs on a huge monitor with many times the resolution of a laptop? Maybe you want to detect that and change the way you are producing the user interface. What if it runs on a mobile phone or tablet? Perhaps you want to detect that and change things. It is useful to be aware of various development frameworks that handle different platforms, such as Bootstrap. Many of those systems allow setting up things on screen with a grid system and defining different layouts for different sized devices so that you can write one piece of code for mobile phones, laptops, and large screen devices.

Useful Documents to Produce

It's best not to go overboard with the production of documents for your development. However, there are some documents that are usually advisable to create; even if on a small development, you include them all in one:

- **Requirements Documentation –** Identifies what the system should look like and be capable of.
- **Architecture/Design Documentation –** Describes how the software components are designed and should have sufficient information for programmer(s) to develop the programs.
- **Technical Documentation** – Documents the code, algorithms, user interfaces, APIs, and so on. It is a document written by the programmer(s) for other programmer(s) who might come along later and need to understand and maintain the code.
- **End-User Documentation –** Describes how the software works for those people that will use it. That could be end users, system administrators, and support.
- **Marketing/Training Material** – Useful to provide to potential users of the software to show the benefits.

Source Control Systems

Source control systems are used to track changes to files, maintain different versions of the same files, and allow multiple people to collaborate during the development phase. These systems are sometimes called version control systems or revision control systems too. The main idea is that the system manages source code and maintains a number of versions and history.

If you have access to Data Integration Studio, this can handle source control for you – so look no further. You will still need to do some additional configuration beyond the default, such as setting up a SVN, CVS, or Git server. However, if you don't have DI Studio, then read on.

Using source control is extremely beneficial, because as we develop programs, the system maintains previous versions. We can return to an old version if we need to. We can compare the current version of the code to old versions to see what has changed. The system can keep track of who has changed parts of the code. It can manage the code so that only one person at a time is updating it.

Some important features that you should look for when choosing source control systems are

- **Concurrent development** – A source control system should provide tools to allow multiple developers to work on source code at the same time. This might provide a mechanism for merging code together in a controlled way.
- **Tracking changes** – It should provide a mechanism to see what changes have been made by people, even when code is merged back together.
- **Locking or branches** – Locking applies to some source control systems that manage code by locking it for use by one person at a time. Other systems take an alternate approach of keeping multiple copies of code, such as in different branches, which can then be merged together. A system using locking can handle locking of modules and checking code in and out. If a team member wanted to make changes to the code, they could check it out. That would lock the code so that others could not make changes to it until it was finished with and checked back in. The benefits of this grow as the size of a team increases, in that the more people trying to work together on code, the easier a system like this makes it.

- **Archive and backup** – A system can also handle archiving and backing up of code. Archiving tends to happen for code that is not currently in use and therefore is copied away to an archive area so that it can be retrieved if it is needed in future. Backups are taken regularly with the aim of being used if code is lost or recent changes lost. It is for current code that is in use but which we want another copy kept as insurance.
- **Release management** – Release management is concerned with releasing versions of code into different environments and can be helped or managed with a source control system. Often an application is made up of a number of programs which many people might be working on together. Many of these pieces of code may form modules within an application. A collection of these are usually bundled up into a release. It might be the entire application code or a collection of modules from the application. When a release happens, you would usually increment the version of your application (e.g., Data_explorer v1.11). Usually you would increment by an integer for a major release and increment decimals for minor releases. A good source control system that manages releases would be able to issue a release, perhaps by packaging up the new bits and passing them to another environment. It could also roll back a release by packaging up a previous release and delivering that to replace a current release that might have problems.

There are many source control systems available with these features and more are coming out year by year. Some of the common ones that have been around for many years are Subversion (SVN), Git, Team Foundation Server (TFS), and Concurrent Versions System (CVS). There are newer systems like GitHub for which SAS now provides some integration. This is a fantastic system to use with SAS development, and I suggest you search the SAS Global Forum proceedings online for papers from users describing how to use this.

You may have a source control system that you have to use because it is the company standard or already in use. If you do get to choose, look for one that has a client for your operating system that supports it and makes it easier to use. For example, GitHub has a desktop version that can be downloaded for Windows or macOS.

Environments for Developing Web Applications

A development environment is the software that allows you to write, test, and edit a program. You can develop in a single environment, but it is advisable to use at least two environments. If you do develop in a single environment and then people use your application from that environment, then when you have to fix a problem and make an enhancement, you are doing it with the live code, which is likely to cause problems to your users. One mistake and the application stops working.

You should develop in one development environment and then deliver the developed code to production in another development environment. This allows you to have the current release of your program running in production while you are changing the development version and getting it ready to become the new production version.

If you are able to have three environments, then that allows you to have a Development, Test, and Production environment. You then develop in Development and, when something is ready, deliver it to Test for testing and user acceptance. Once that is done, it can be moved to Production.

Sometimes people will have a Personal Development Environment (PDEV) as well as a Common Development Environment (CDEV). Then they can do things in PDEV without affecting anyone else, and once they are happy with that code, they can move it to CDEV.

Some larger companies have even more environments:

- Common Development (CDEV)
- Personal Development (PDEV)
- Component Integration Testing (CIT)
- System Integration Testing (SIT)
- User Acceptance Testing (UAT)
- Production (PROD)

This allows different kinds of testing to be done in different environments. Figure 2-3 shows what the flow of development would be in a multiple environment system.

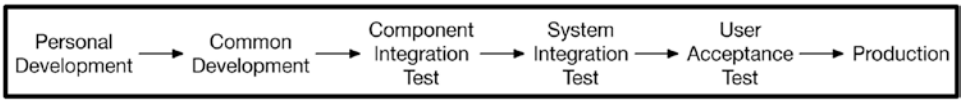

Figure 2-3. *Multiple environment system flow of development*

Ways to Develop with SAS

SAS provides lots of different tools for developing applications. Each has some advantages and disadvantages. It's important to know what release of the software you are using too, as that can make a difference as to what useful features you may or may not have. For instance, at the time of writing, the latest version of Enterprise Guide has a data step debugger built into it, whereas prior versions do not.

Commonly Used SAS Tools

Commonly used SAS tools include the following:

- PC SAS, in which we can write SAS code of all kinds, but if creating Stored Processes, we need to also use SAS Management Console.
- Enterprise Guide, which can create most kinds of SAS code including SAS Stored Processes. It spawns a SAS Workspace server to run SAS code from it.
- SAS Studio, which is similar to Enterprise Guide but only requires a web browser to use. It doesn't allow creation of Stored Processes via any kind of wizard, but you can create them programmatically with standard SAS code. That code would need to make metadata calls in order to create them, perhaps using some open source macros available for that purpose.
- Data Integration Studio provides a controlled way to create SAS programs using a collection of transformations and custom SAS code. It includes the ability to check out and check in code so that teams of people can work on large systems together.
- Office Add-in provides a SAS program window from the add-in toolbar which lets you write SAS code and execute it on the sever. The results are brought back and displayed in the Word, EXCEL, or PowerPoint. You can also view the SAS log.

There are lots of other less common SAS tools that can be used for developing, but I will be focusing on the ones mentioned already.

Most people will create Stored Processes with Enterprise Guide, and it is the way that SAS documentation says to create them. Doing this will use a system account for the stored process though, and it has inherent dangers associated with that. Currently, you must either use Enterprise Guide or Management Console to create Stored Processes as there is no other simple way provided by SAS to do so. There are metadata functions that can be used from Base SAS to create a stored process, and there are even some macros available that make that easy to do. I recommend taking a look at them.[1] One nice thing that Enterprise Guide does for you is to take you through a wizard to help you make the Stored Process. It also will do things like add the stpbegin and stpend macros around your SAS code by default so that ODS will work in the various clients you use your Stored Process with. If you are new to Stored Processes, then use Enterprise Guide to create them until you find a reason to use another method.

Write Your Own Tools in SAS

As we will see in this book, we can write our own tools. You will be able to create bespoke tools that you need with the features you want and without features that are not required. You can leverage the skills you have with SAS to make your tools without needing knowledge of other languages. They can be SAS macros that we can provide parameters to choose what we want them to do. Or we can build Stored Processes which are far more flexible. I have created Stored Process tools like this which run through the web browser and give me functions like

- Scheduling SAS programs to run
- Analyzing directories of SAS and Enterprise Guide projects, producing reports summarizing each of the programs
- Displaying the output and logs from scheduled jobs, allowing them to be viewed
- Displaying the logs or Stored Processes that have run recently or are currently running

[1]You can create a stored process from a SAS program with the code located here: https://github.com/macropeople/macrocore/blob/master/meta/mm_createstp.sas

Simple Techniques for Building Applications with Stored Processes

I have made many a prototype application using one or more Stored Processes in a matter of hours. You can use some simple techniques to do this kind of thing:

- Enterprise Guide can generate a web page automatically via a wizard in older versions. For some reason, this was removed in newer versions of EG.
    - If you have a macro that does something (like produce a report from some parameter choices), then this can be simply turned into a Stored Process. Just make a Stored Process and put the macro invocation into it, along with either the macro code or option to point to an autocall library that has it. You will be able to define a prompt for each macro parameter and use those values to invoke the macro. Run this through the SAS Stored Process Web Application and you have an application based on your macro.
- Stored Processes, which create their own interactive elements, are a great way to build applications. For instance, you can make a Stored Process that produces selection lists, radio buttons, and so on based on SAS data. This lets you build a form with selections you can choose from which can then be defined to call another Stored Process to make use of those selections.
- It's easy to generate HTML with hyperlinks to other HTML from a Stored Process.
- You can implement drill-down by making your Stored Process generate HTML links that link to the Stored Process that produced them, but passed a parameter value in with the call, thereby implementing drill-down. For example, make a graph that has bars with drill-down links that call the same Stored Process but add the info for passing the bar value clicked.
- Make menus by a Stored Process generating forms with HTML, which then call other Stored Processes.

- JavaServer Pages (JSP) can be created in a particular location with the same name as a Stored Process. If you then invoke the Stored Process with a certain _action parameter value, then the JSP will be displayed, rather than the Stored Process being run. This allows a flexible program to be written to prompt the user for parameters before running the actual Stored Process. You can "hack" this process by simply putting an HTML file in the JSP directory, with a JSP file type, and your HTML will then be displayed in the same way.

Useful Tools for Building Web Applications

Here is a range of mostly free tools that can be used to make the process of building web applications with SAS much easier. I will outline some of these tools and describe how they are useful. Tools come and go though, so some things I mention might not be available in future or there may be better tools around. Hopefully, being aware of the kind of tools on offer will enable you to search for others that superseded these ones.

Lint Tools

Lint was originally a tool on UNIX systems that flagged suspicious or non-portable code in C programs. However, people have extended this functionality to other languages and provided more functionality. Some tools will not only look for a range of errors in your code, but also lay out the code in a more standard way. Some tools will uppercase tags and attributes used, highlight unmatched parentheses, wrap long lines, and so on. So, it can take some very hard to read code and make it far easier to make sense of. Here are some useful tools for web application development:

- JavaScript Lint – `www.JavaScriptlint.com/` or `http://www.jslint.com/`
- HTML Tidy – `https://infohound.net/tidy/`
- CSS Lint – `http://csslint.net/`
- JSON Lint – `https://jsonlint.com/`
- CSV Lint – `https://csvlint.io/`

The JavaScript Lint tool will look for these common mistakes, as well as many uncommon ones:

- Missing semi-colons at the end of a line
- Curly braces without an *if*, *for*, *while*, and so on
- Code that is never run because of a *return*, *throw*, *continue*, *or break*
- Case statements in a switch that do not have a *break* statement
- Leading and trailing decimal points on a number
- A leading zero that turns a number into octal (base 8)
- Comments within comments
- Ambiguity whether two adjacent lines are part of the same statement
- Statements that don't do anything

IDE Tools

An IDE is an Interactive Development Environment. These are tools that aid you in developing in one or more particular languages. They provide some or all of these features: a source code editor with code completion, tools to automate building the code, a debugger, compiler, interpreter, version control system, extensive help on the language, and so on. Some IDEs worth looking at include

- NetBeans from Oracle; there are many versions of this and it's best to just download the HTML5/JavaScript version (`https://netbeans.org/`).
- Brackets is an open source code editor with live preview of changes and support for preprocessors (`http://brackets.io/`).
- Atom from GitHub is described as a hackable text editor, which means it can be customized extensively (`https://atom.io/`).
- Visual Studio Code from Microsoft supports debugging, syntax highlighting, code completion, snippets, and more (`https://code.visualstudio.com/`).

- Notepad++ deserves a mention, though it doesn't have fancy tools built into it. It is like a standard text editor on steroids and is my number one choice for editing all kinds of programs whenever possible. It does have syntax highlighting built in and can do great things like edit hundreds of files simultaneously and find text across them all very quickly (https://notepad-plus-plus.org/).

Using a JavaScript IDE

An IDE is an Interactive Development Environment. You can use JavaScript IDEs for developing HTML and JavaScript code. IDEs often have useful features like syntax highlighting, debuggers, preview windows, and so on. Many good ones are free including Notepad++ and Microsoft Visual Studio Code, which both run on Mac, Windows, and Linux.

It is sometimes useful to build some HTML and JavaScript code in an IDE and then look at moving it onto the SAS web server and integrating into a Stored Process.

JavaScript Debuggers

JavaScript debuggers are very useful for running your JavaScript and debugging any errors you have. You can also trace variables reporting their values when they change, which can be helpful in understanding how your JavaScript code runs. You can set breakpoints too, so that the code will run up to a certain point and then pause so you can look at the values of variables. Many web browsers have debuggers built in, so there is no need to buy or download one.

Most developers will have their favorite development tools in their favorite browser. For me, it has changed over time and was Firefox for many years, but more recently, I have found Chrome to be great. These two browsers are great because they work on many operating systems too. Internet Explorer only works on Windows, unless you make use of an emulator such as browserstack.com which lets you test you web page on lots of different browsers and platforms. I find that most browsers have much the same capabilities now, all of which are sufficient for helping to build web applications:

- Microsoft Internet Explorer has built-in developer tools (Figure 2-4) that can be accessed by pressing F12 or using Tools/Developer Tools/Console (https://msdn.microsoft.com/en-us/library/gg589507(v=vs.85).aspx).

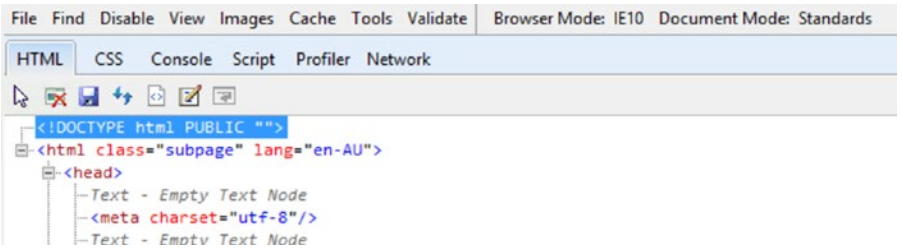

Figure 2-4. *Internet Explorer developer tools*

- Firefox has developer tools too (Figure 2-5) that can be accessed using the Tools/Web Developer menu. There is a debugger, web console, performance tools, and more (`https://developer.mozilla.org/en-US/docs/Tools`).

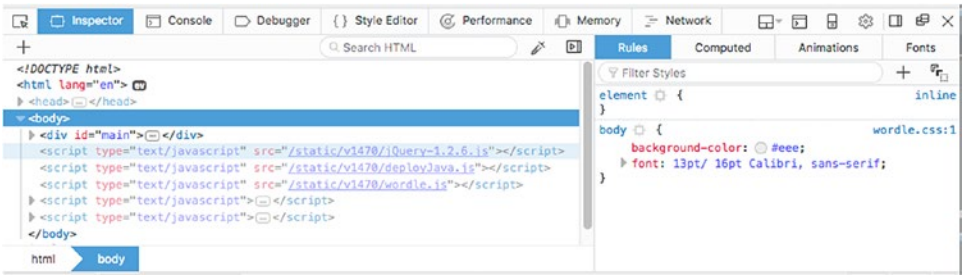

Figure 2-5. *Firefox developer tools*

- Google Chrome has built-in developer tools (Figure 2-6) which are accessed using the Tools/Developer Tools menu. It also has a web console, debugger, and other tools (`https://developer.chrome.com/devtools`).

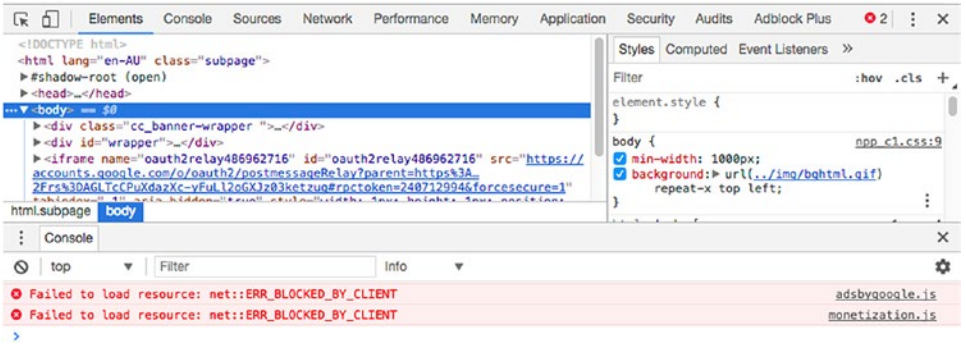

Figure 2-6. *Chrome developer tools*

- Apple Safari has built-in web developer tools (Figure 2-7) such as a Web Inspector, network tools, debugger, and more (https://developer.apple.com/safari/tools/).

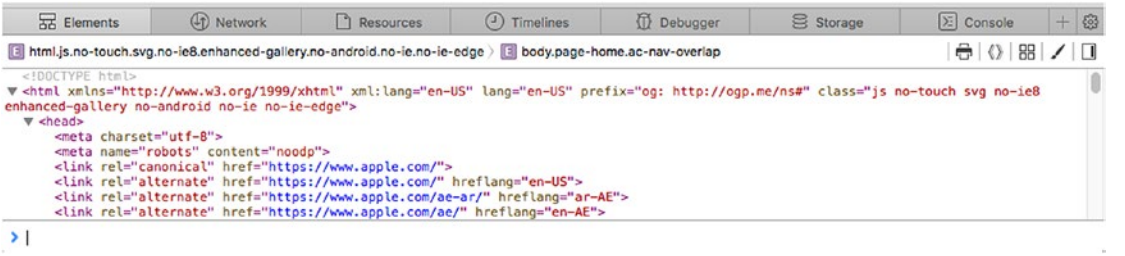

Figure 2-7. *Safari developer tools*

Code Comparison Tools

I have saved many hours of time and done things that were almost impossible to do another way by using code comparison tools such as Beyond Compare (you've got to buy this one if you are using Microsoft Windows) and WinMerge (free and open source). My favorite of all time is Beyond Compare, even though it only runs on Windows and Mac. It will give you a fantastic side-by-side comparison of two directories or files. You can ignore unimportant differences (e.g., different numbers of spaces), show just things that are different, produce reports of the differences, and much more.

Recently, Beyond Compare helped me solve a problem where it showed me that two files were exactly the same, except one was twice the size of the other. Looking at the top of the display, I could see that one file was encoded in ASCII, whereas the other was Unicode. This also meant that when I uploaded one of the files to UNIX and tried to read in the XML it contained, I was getting failures. I was able to bring the encoding into line with what was expected, and my problems were solved.

This tool can produce great reports showing the code in different environments and how it differs. For instance, compare your development code to your production code and see exactly what the differences are. Or compare your current code to the previous version to work out exactly what has changed. It's great!

These tools can compare two files and report the differences:

- Beyond Compare from Scooter Software is the best tool in this category and does everything the others do. As well as comparing files and directories, you can generate reports of differences, use right-click menus through system integration, ignore unimportant differences, and much more (www.scootersoftware.com/).

  In the sample screenshot (Figure 2-8), you can see me comparing two SAS programs. It shows me a map in the top left of where the differences are; I have selected to see the differences in context, and it is very clear what they are; when I select a line, it even shows me the line comparison at the bottom.

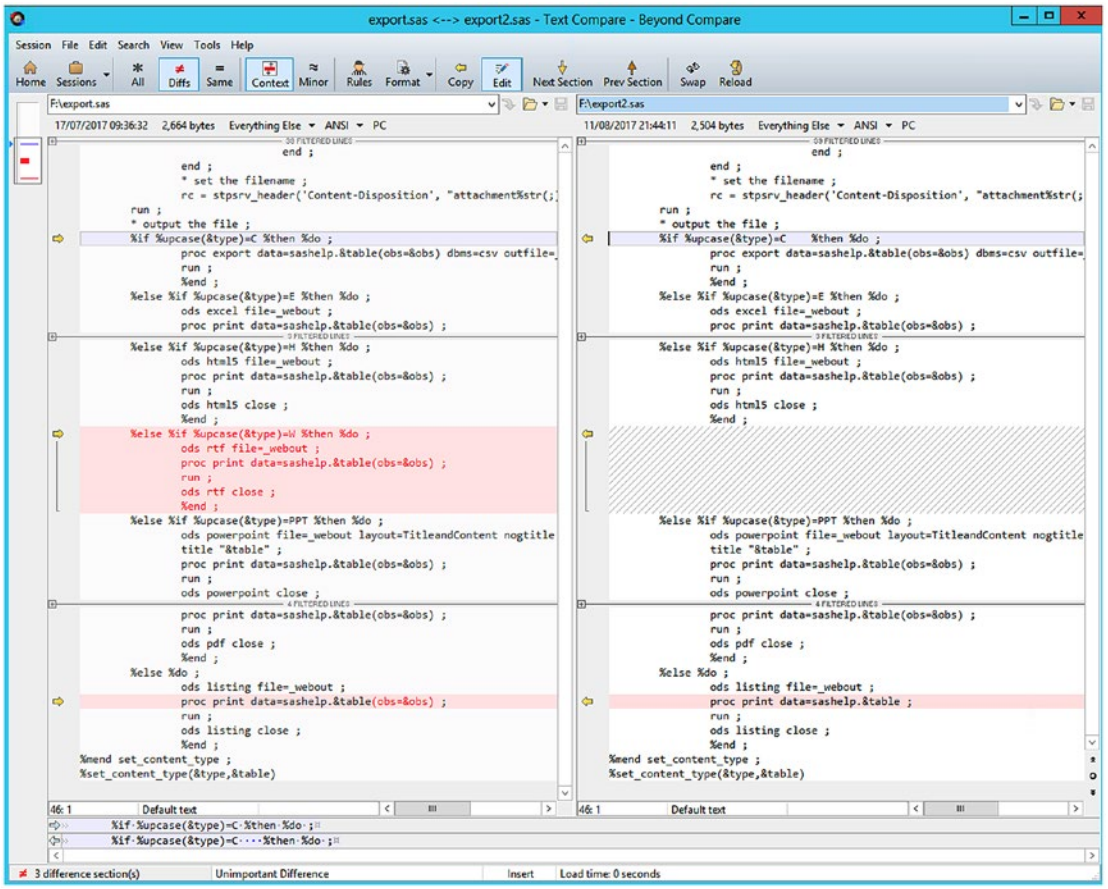

Figure 2-8. *Comparison of two versions of source code using Beyond Compare*

- WinMerge is a free, open source file and directory comparison/synchronization tool. It does much of what Beyond Compare does, and being free may be a better choice for you (http://winmerge.org/?lang=en).

Figure 2-9 shows the same two SAS programs being compared using WinMerge.

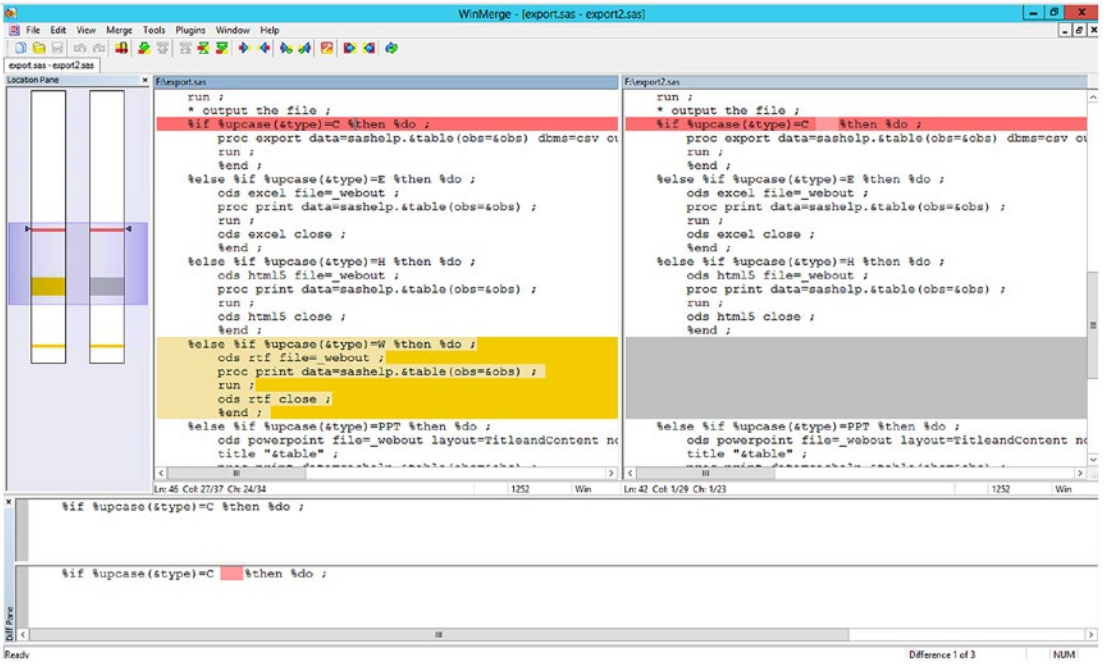

Figure 2-9. *Comparison of two versions of source code using WinMerge*

- FC is a command in the Microsoft operating system which will let you compare files (https://technet.microsoft.com/en-us/library/bb490904.aspx). Figure 2-10 shows the output of the FC command, which is a bit harder to use.

```
F:\>fc export.sas export2.sas
Comparing files export.sas and EXPORT2.SAS
***** export.sas
        * output the file ;
        %if %upcase(&type)=C %then %do ;
                proc export data=sashelp.&table(obs=&obs) dbms=csv outfile=_webo
ut replace ;
***** EXPORT2.SAS
        * output the file ;
        %if %upcase(&type)=C    %then %do ;
                proc export data=sashelp.&table(obs=&obs) dbms=csv outfile=_webo
ut replace ;
*****

***** export.sas
                %end ;
        %else %if %upcase(&type)=W %then %do ;
                ods rtf file=_webout ;
                proc print data=sashelp.&table(obs=&obs) ;
***** EXPORT2.SAS
                %end ;
        %else %if %upcase(&type)=PPT %then %do ;
                ods powerpoint file=_webout layout=TitleandContent nogtitle nogf
ootnote;
                title "&table" ;
                proc print data=sashelp.&table(obs=&obs) ;
*****

***** export.sas
                run ;
                ods rtf close ;
                %end ;
        %else %if %upcase(&type)=PPT %then %do ;
                ods powerpoint file=_webout layout=TitleandContent nogtitle nogf
ootnote;
                title "&table" ;
***** EXPORT2.SAS
                run ;
                ods powerpoint close ;
                %end ;
        %else %if %upcase(&type)=PDF %then %do ;
                ods pdf file=_webout ;
                title "&table" ;
*****

***** export.sas
                run ;
                ods powerpoint close ;
                %end ;
        %else %if %upcase(&type)=PDF %then %do ;
                ods pdf file=_webout ;
                title "&table" ;
                proc print data=sashelp.&table(obs=&obs) ;
```

Figure 2-10. *Comparing two files using the FC command*

- DIFF and DIFF3 are utilities built into most UNIX operating systems which let you compare files. Their output is similar to the Microsoft FC command, however perhaps a bit easier to use. You can read more about them here: www.computerhope.com/unix/udiff.htm.

Summary

In this chapter, we have learned some more general concepts about developing applications, which are very useful when developing SAS Stored Process–based applications:

- Freestyle approach to development has many disadvantages which other approaches overcome.
- SDLC and the Waterfall model are commonly used methodologies which are quite useful.
- The V-model is a development of the Waterfall model and has some advantages over it.
- Agile development has become very popular in software development in recent years, and I would recommend this.
- When planning for development, you should consider the architectural aspects, especially from a SAS architecture standpoint.
- Make sure you have the most useful documents required for a development project.
- Consider your toolkit such as IDE, source control, debuggers, code comparison, automation tools, deployment tools, and so on.