Automatic Documentation Generation Tool for Software Integration Phase in NGPF Program Engines

Marycarmen Torres Martínez
Master of Engineering in Computer Engineering
Othoniel Rodríguez, Ph.D.
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— The actual process of Hardware-Software Integration [1] takes too much time for completion due to the required documentation, and that is costly for the customer. Creating this tool help us to reduce significantly the amount the time is taken for the Integration Phase and the delivery to the customer after the start of the Verification Phase, thus lowering the cost of integration. The automatic documentation tool creates documents analyzing the deliverables from the Integration phase after compiling the source code. This tool provides documentation at design level by generating documentation from deliverables and is a way of integration with hand-written documentation. Hand-written documentation is not effective as it raises errors and takes more time. This approach works by reading keywords from a series of templates replacing them with the corresponding data in each document. This approach has been shown to be effective for the customer and reduces the time to deliver new engine software documentation to the informal verification phase.*

Key Terms *— Documentation, Integration, NGPF, Templates.*

INTRODUCTION

A series of studies of program comprehension show that programmers rely on good software documentation [2]. Unfortunately, manually-written documentation is notorious for being incorrect or incomplete, and either way is very time consuming to create. The idea to create the documentation tool was to reduce the time to deliver engine software documentation. Document generator is a programming tool that creates documentation from pre-programmed templates with keywords already identified in the code and replacing them with the corresponding data. This data can be memory addresses, software version, document or repository revision and so on. The advantage of creating a document programming tool is that it offers a valuable opportunity to improve and standardize the quality of the software documentation.

Currently, the software integration phase document is manually generated. The cycle of compiling generation, manual document generation, integration review, quality review and delivery to the client takes a minimum of two and a maximum of three days. The scope of this new tool is to automate the generation of the build software documentation reducing the time significantly and deliver to the client in a minimum of one day and a maximum of two days. Currently, generating manual-written documentation is taking eight hours of time.

This tool process is not expected to affect any other steps of the Integration process. The tool will start generating documents only after the code is frozen, and other deliverables and engine checkout is available.

This proposal is divided into multiple sections, each one explains the purpose of the project.

Definition of Concepts

- *Documentation – are the documents results that the tool will generate. The Documentation contains Software Build Checklist, Software Build Request part B, Pre-delivery Verification Form and Delivery Order.*
- *Integration – Is part of the Software Development Cycle that compiles the code and generates deliverables.*
- *NGPF – Is a commercial engine produced by Pratt and Whitney named Next Generation Program Family*

- *Templates – Is a series of documents that were created based on the documentation that has a series of "keywords". These keywords will be replaced by the data produced by the tool.*

SUPPORTING THEORY

To generate documentation from a system that has executables, memory addresses, code and so on, we need to establish how to extract the data required from the system to a document. In this section we will discuss these approach in detail [3]. In later sections we discuss how we have user this approach.

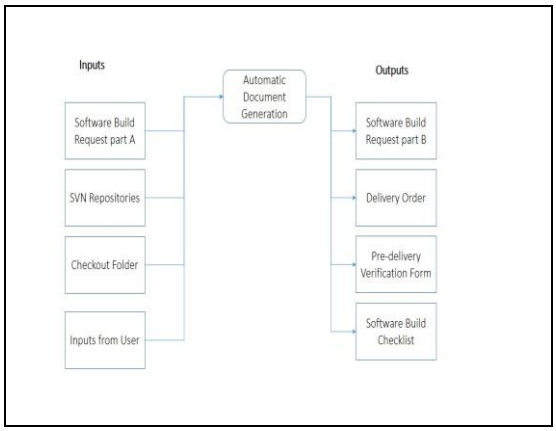

Figure 1
Project Overview

To generate the documentation using the tool, we need to identify which inputs are needed to generate the documentation for the integration phase. In figure 1, four inputs are identified:

- Software Build Request part A (SBR-A) – The SBR-A is a document supplied by the client that have the version to be compiled, the Application Software URL and Revisions, Application Software Interface URL and Revisions, Change Request for the version, reference documents and revisions, OS recommended version, deliverables and plan for testing
- SVN Repositories – The repositories needed to generate the documentation are the variant repository that will have the deliverables, the common repository, documentation repository that will have the SBR-A and metric file.
- Checkout Folder – This folder will have the integration testing after compiling the code with an engine simulation. The memory results are needed to fill the documentation.
- Inputs from User – User needs to provide the following information to start running the tool.

SVN credentials:

- SBRA URL – the URL needs to be in .docx format.
- SVN URL Common repository.
- SVN URL Variant Repository.
- System Stack file – Location of Checkout folder.
- ASI Build Compilation CR – The Change Request performed to check in the deliverables.
- ASA Common SVN Revision – Last design revision in the common repository.
- ASA Variant SVN Revision – Last design revision in the variant repository.

There are four documents needs to be automatically generated using a series of documents, repositories and revisions. These documents will become the tool outputs. The documents to be generated are:

- Software Build Checklist – have the content of memory data of compilation.
- Pre-delivery Verification Form – Checklist to make sure every step of the integration was not missed.
- Delivery Order – is the final document, have the repositories, revision and client signatures
- Software Build Request part B (SBR-B) – have the change requests, checkout results and any deviation needed in the compilation.

PROBLEM

Due to the necessity of improvement for the Build Hardware-Software Integration Phase for Next Program Generation Family Engines, this project will be developed. This tool will prevent errors, reduce client cost and deliver all the software releases in less time. The tool will generate the following documents: Software Build

Checklist, Pre-delivery Documentation Form, Software Build Request part B and Delivery Order.

Project Goal

The goal of this project is to complete the Automated Documentation Tool successfully and have client agree on incorporating this tool into the development process. This tool should count as a development process improvement, which increases the product quality and the process robustness for the customer and the company.

METHODOLOGY

Automatic Document Generation Tool is a C# [4] [5] application that will be installed in any local employee computer that will require to do the compiling of code and generate documentation. The tool should be able to work in the employee computer or from a local network drive. It will be installed in a computer with a successfully pre-installed Windows 7 as Operating System, Microsoft Office 2010 and Tortoise SVN 1.8.7 version or higher. The system will help to manage and create documentation of the Integration Build Phase.

The easiest way to create this kind of tool specifically for this engine program is to create templates. Four templates were created with a series of "keywords". The keywords will be what the tool will read. All the keywords have words with "#_ at the beginning and "_#_ at the end.

For example: In the code the variable is "Date" and the template is "#_DATE_#.

The reason of creating the keyword was because is the easiest way to replace a data value and will be as simple as read and replace in the code. The code will read the keywords and replace it with the corresponding data already identified. The tool search in repositories, read from documents and SVN revisions.

The system will have the three following roles.

The first is validate user credentials. The system will be able to read Tortoise SVN credentials, using the class Tortoise SVN. The system will validate user credentials and return the results if user has or not access. Using SVN credentials will validate builder name using builder employee number. For example: SVN Credential: mtorres, Builder: xip0419 that is the employee number and the system with this method will validate my name that is Marycarmen Torres. The system will save and remember the credentials if the user clicks on the credentials checkmark.

The second role is read SBR-A to validate which software version and engine will be worked. The system will be able to export SBR-A to a folder. The SBR-A URL should be a user input. The SBR-A is one of the documents with more information needed to fill the templates. Note that system will read the other information needed from the repositories, SVN revisions and documents already download.

The third role is replacing keywords with the corresponding data to the templates. The tool, after reading keywords and validating information will replace them with the corresponding data. For example: Tool will read "#_DATE_#" and as variable name associated is "DATE" the tool will replace "#_DATE_#" for "May 5, 2016". Tool automatically will fill the templates and place them in a folder.

This system is intended to be a quick solution to reduce time. The project scope is to provide automation to the process of creating documents providing to employees a Graphical User Interface to work with the input file to be processed and performing data validations at the final delivery file to ensure that the human errors are detected and disallowed.

Relevance and Significance

This kind of tool was never implemented for any of Next Program Generation Family engine programs. At least once a month there is a software release for each engine. Sometimes, the Integration team have 10 to 12 compilation requests each month.

The organization will have some benefits. One of them is less time of Documentation process. The

actual process consists of hand-written documents, review, quality review and delivery to the customer. The new process will have automated document generation, review, quality review and delivery to the customer. Instead of eight hours creating hand-written documents the process will consist of ten minutes or less of automated document generation. This process will be a great improvement, less cost and better quality.

The security of the tool is the best advantage. The program credentials are encrypted in the code not to be shared for security reasons. To access to the tool you must have access to developer or builder of any of the Next Program Generation Family (NPGF) programs. No person outside of Pratt and Whitney or Infotech Aerospace Services can enter the systems.

Assumptions and Dependencies

The system is based on the following assumptions:

- The system is a tool that can be used at any time as required.
- The system is a tool to facilitate and improve the Integration process of the job.
- Users must have computer literacy (expected to know how to use a computer, compile code, and experience generating Build Documentation).
- Users must have a basic Microsoft Office Word 2010 literacy (some experience managing MS Office Word files).
- The only people allowed to use the tool are the one allowed to work in the administrative part of the project.

Development Methods

This tool in essence is a single application that is looking to improve a process that can be automated, because under human standards will take more time that the desire for something that can be worked in less time. This tool basically can be composed by three roles mentioned above, which cover the validation and generation of documents.

For that reason the following method were selected:

- Feature-Oriented design, this method is a type of functional decomposition that assigns features to modules.
- Object-Oriented design, this method assign an object to modules.

From a point of view, with the Feature-Oriented design approach we can put together all the functions related to a feature, which in this case can be the stages of the workflow and with the Object-Oriented design we can start to narrow the near future implementation into a concept of an object to module design.

Architectural Strategies

C sharp as programming language:

This was chosen due to the available of many libraries to this language that will facilitate the composition of the programming, also with the visual studio functionality, the tool can be created in less time due the automatic code generated for UI. Also, with low experience and expertise in this language is easy to work in a short time and generate a product and a prototype as quickly as possible, since with this language we can generate the code necessary for the user interface and the programming to do the expected results.

System Architecture

Automatic Documentation Generation tool is an application that will work a set of keywords pre-determinate. [6] This set is worked by a sequence of task, for that reason the “Distributed” architecture is the best approach. In “Distributed” system [7] computing and storage are in separate system blocks orchestrated separately and connected through networks. The system upgrade is through replacing component blocks. System growth is through adding blocks. The architecture is designed to enable growth and scale out of multiple workloads.

The Automatic Documentation Generation is based on a Feature-Oriented design, with the idea to keep features and function in separated modules.

This was selected this way in to keep the system components in an easy way to maintenance or future enhance the development team.

The implementation of the tool was started following the three main roles. In figure 2 and 3, the class diagram was divided by Presentation Layer, Business layer and Data Layer. The Data Layer have the final results.

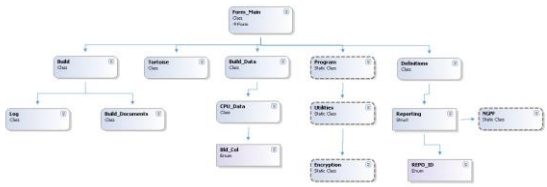

Figure 2
Class Diagram

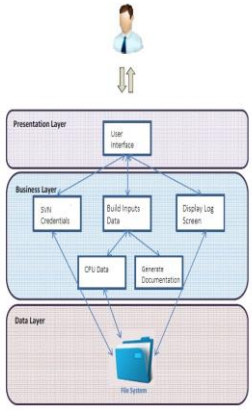

Figure 3
Conceptual Diagram

RESULTS AND DISCUSSION

The tool was generated successfully and is can generate the four documents filled. The tool resolved a real problem in the Integration Phase of the Software for the NGPF program. In figure 4, we can find the final Automatic Documentation tool. After the user writes the username and password, the credentials are validated, and the inputs are written, the user clicks on "Generate Documentation" bottom and the process will start creating the documentation. The display screen will show the process and where the documentation will be created.

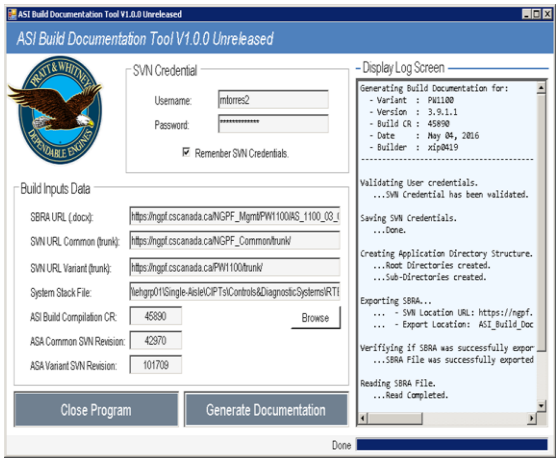

Figure 4
Automatic Documentation Tool User Interface

CONCLUSION

This project was presented for automatically generating documentation for a Next Generation Program Family. Our approach was to look at the easier way to automatic generate documentation and reduce cost.

The tool is simple and straight forward for users. It covers an actual and real problem in the company and it was solved. A better quality of the product will be presented to the client and lower cost for them.

After project conclusion, will going to present the project to the clients for approval and implement the project in other engines.

FUTURE WORK

Plans for enhancing the software application:

The future plan is to improve the process of automation of documents, for that reason the tool is simple, future enhance to the tool will be based in other parts of the engine that are compiled different like Prognostic Health Management Unit (PHMU) and military programs. New improvements will be created after employee/user feedback of the performance of the tool. Also, after client's feedback and suggestions will be determined future enhances to the tool.

ACKNOWLEDGMENT

This material is based upon work supported by, or in part by, the Infotech Aerospace Services. This document export is restricted by the Export Administration Act of 1979, as amended. You may not possess, use, copy or disclose this document or any information in it, for any purpose including without limitation to design, manufacture, or repair parts or obtain FAA or other government approval to do so.

REFERENCES

[1] D. Akka. (2014, May). *6 tips for a successful system integration project* [Online]. Available: http://thenextweb.com/dd/2014/02/09/6-tips-successful-system-integration-project/%23gref/.

[2] P. McBurnel and C. McMillan. (2014, March). *Automatic Documentation Generation via Source Code Summarization of Method* [Online]. Available: https://www3.nd.edu/~cmc/papers/mcburney_icpc_2014.pdf.

[3] H. Halvorsen. (March, 12, 2014). *Introduction to Visual Studio and C sharp* [Online]. Available: http://home.hit.no/~hansha/documents/microsoft.net/tutorials/introduction%20to%20visual%20studio/Introduction%20to%20Visual%20Studio%20and%20CSharp.pdf.

[4] P. Lamas (April 6, 2016) *Sandcastle Help File Builder* [Online]. Available: https://github.com/EWSoftware/SHFB.

[5] C. S. Horstmann, *Object-Oriented Design & Patterns*, 2nd ed., John Wiley and Sons, Inc., Hoboken, NJ. USA, 2006.

[6] S. Apel and C. Kastner. (2007, July). An Overview of Feature-Oriented Software Development Available: http://www.jot.fm/issues/issue_2009_07/column5.pdf.

[7] P. Bilderbeek. (2013, January 15). *Four types of System Architectures* [Online]. Available: http://www.themetisfiles.com/2013/01/the-four-types-of-system-architectures/.